

# **ID-ONE COSMO V9 ESSENTIAL**

Public Security Target



## About IDEMIA

OT-Morpho is now IDEMIA, the global leader in trusted identities for an increasingly digital world, with the ambition to empower citizens and consumers alike to interact, pay, connect, travel and vote in ways that are now possible in a connected environment.

Securing our identity has become mission critical in the world we live in today. By standing for Augmented Identity, we reinvent the way we think, produce, use and protect this asset, whether for individuals or for objects. We ensure privacy and trust as well as guarantee secure, authenticated and verifiable transactions for international clients from Financial, Telecom, Identity, Security and IoT sectors.

With close to €3bn in revenues, IDEMIA is the result of the merger between OT (Oberthur Technologies) and Safran Identity & Security (Morpho). This new company counts 14,000 employees of more than 80 nationalities and serves clients in 180 countries.

| For more information, visit [www.idemia.com](http://www.idemia.com) / Follow @IdemiaGroup on Twitter

© IDEMIA. All rights reserved.

Specifications and information are subject to change without notice.

The products described in this document are subject to continuous development and improvement.

All trademarks and service marks referred to herein, whether registered or not in specific countries, are the properties of their respective owners.

- Printed versions of this document are uncontrolled -



**DOCUMENT MANAGEMENT**

<b>Business Unit – Department</b>	<b>CI – R&amp;D</b>
<b>Document type</b>	<b>FQR</b>
<b>Document Title</b>	<b>ID-ONE COSMO V9 ESSENTIAL – Public Security Target</b>
<b>FQR No</b>	<b>110 8959</b>
<b>FQR Issue</b>	<b>3</b>

**DOCUMENT REVISION**

<b>Date</b>	<b>Revision</b>	<b>Modification</b>
<b>06/11/2018</b>	1.0	Creation
<b>08/11/2018</b>	2.0	Update table of contents
<b>10/12/2018</b>	3.0	Correction of date for “Essential Application Loading Protection Guidance” in references section



## TABLE OF CONTENT

---

<b>1</b>	<b>INTRODUCTION.....</b>	<b>11</b>
1.1	<b>TOE Reference .....</b>	<b>11</b>
1.1.1	Card Identification Data (tag 'DF52').....	11
1.1.2	IC Card Manufacturing Data (tag 'DF50') .....	13
1.1.3	TOE Reference .....	14
1.2	<b>Security Target Overview .....</b>	<b>15</b>
1.3	<b>REFERENCES.....</b>	<b>15</b>
1.4	<b>ACRONYMS and NOTATIONS.....</b>	<b>18</b>
1.4.1	Abbreviations .....	18
1.4.2	Definitions.....	19
1.5	<b>TOE OVERVIEW .....</b>	<b>20</b>
1.5.1	TOE Type.....	20
1.5.2	TOE usage .....	21
1.6	<b>PRODUCT ARCHITECTURE .....</b>	<b>22</b>
1.6.1	Logical scope of the TOE .....	22
1.6.2	Physical scope of the TOE.....	23
1.6.3	TOE Guidance .....	24
1.6.4	Platform isolation.....	25
1.6.5	Applications .....	25
1.7	<b>TOE Description .....</b>	<b>26</b>
1.7.1	Defensive Java Card Platform .....	26
1.7.2	Global Platform.....	27
1.7.1	Integrated Circuit (IC), IFX Secure Smart Card Controller.....	28
1.7.1.1	IFX Secure Smart Card Controller .....	28
1.7.2	Operating System (OS).....	28
1.7.2.1	BIOS.....	29
1.7.2.2	Cryptographic features .....	29
1.7.2.3	Biometric feature .....	29
1.7.2.4	Virtual Machine .....	29
1.7.2.5	The Java Card Runtime Environment.....	30
1.7.2.6	APIs.....	30
1.7.2.7	Open and isolating Platform.....	30



1.7.2.8 Resident Application .....	30
1.7.2.9 Applets .....	30
<b>1.8 LIFE-CYCLE .....</b>	<b>31</b>
1.8.1 Phases .....	32
1.8.2 Phase 1: Security IC Embedded Software development .....	33
1.8.3 Phase 2: Security IC Development .....	34
1.8.4 Phase 3 and phase 4: Security IC Manufacturing and packaging .....	34
1.8.5 Phase 5: Composite Product Integration .....	34
1.8.6 Phase 6: Composite Product Personalisation .....	35
1.8.7 Phase 7: Operational Usage .....	35
1.8.8 Application loading .....	35
<b>1.9 Software Components Life Cycle .....</b>	<b>35</b>
1.9.1 Card Life Cycle .....	35
1.9.2 Security Domain Life Cycle States .....	38
1.9.3 Load File Life Cycle .....	39
1.9.4 Application Life Cycle .....	40
<b>1.10 Major Security feature of the TOE .....</b>	<b>43</b>
<b>1.11 NON-TOE HW/SW/FW AVAILABLE TO THE TOE .....</b>	<b>46</b>
<b>2 CONFORMANCE CLAIM .....</b>	<b>48</b>
<b>2.1 Common Criteria Conformance Claim .....</b>	<b>48</b>
<b>2.2 Protection Profile Claim .....</b>	<b>48</b>
<b>2.3 Conformance rationale .....</b>	<b>49</b>
2.3.1 TOE SAR conformance .....	49
2.3.2 TOE Type conformance .....	49
2.3.3 SPD Statement Consistency .....	49
2.3.3.1 Assets .....	49
2.3.3.2 Threats .....	50
2.3.3.3 OSPs .....	50
2.3.3.4 Assumptions .....	50
2.3.3.5 Objectives .....	50
2.3.3.6 Security Objectives for the TOE .....	50
2.3.3.7 Security Objectives for the Operational Environment .....	51
2.3.3.8 SFR and SARs Statements consistency .....	51
2.3.3.9 SFRs Consistency .....	51
2.3.4 Rational for SAR .....	54



<b>3</b>	<b>SECURITY ASPECTS .....</b>	<b>55</b>
3.1	Confidentiality .....	55
3.2	Integrity .....	55
3.3	Unauthorized executions .....	56
3.4	Bytecode verification .....	56
3.4.1	CAP file verification .....	57
3.4.2	Integrity and authentication .....	57
3.4.3	Linking and authentication .....	57
3.5	Card management .....	57
3.6	Services .....	58
<b>4</b>	<b>SECURITY PROBLEM DEFINITION .....</b>	<b>60</b>
4.1	Assets.....	60
4.1.1	User data .....	60
4.1.2	TSF data .....	61
4.1.3	Additional assets.....	61
4.2	Users / Subjects .....	62
4.2.1	Additional Users / Subjects.....	62
4.2.2	Miscellaneous.....	62
4.3	Threats.....	63
4.3.1	CONFIDENTIALITY .....	63
4.3.2	INTEGRITY .....	64
4.3.3	IDENTITY USURPATION .....	65
4.3.4	UNAUTHORIZED EXECUTION .....	65
4.3.5	DENIAL OF SERVICE .....	65
4.3.6	CARD MANAGEMENT .....	66
4.3.7	SERVICES.....	66
4.3.8	MISCELLANEOUS.....	66
4.3.9	Additional threats .....	66
4.4	Organisational Security Policies .....	67
4.5	Assumptions .....	67
<b>5</b>	<b>SECURITY OBJECTIVES.....</b>	<b>68</b>
5.1	Security Objectives for the TOE.....	68
5.1.1	IDENTIFICATION.....	68
5.1.2	EXECUTION .....	68



5.1.3	SERVICES.....	69
5.1.4	OBJECT DELETION .....	69
5.1.5	APPLET MANAGEMENT.....	69
5.1.6	Additional security objectives for the TOE.....	70
<b>5.2</b>	<b>Security objectives for the Operational Environment.....</b>	<b>72</b>
<b>5.3</b>	<b>Security Objectives Rationale .....</b>	<b>73</b>
5.3.1	Threats .....	73
5.3.1.1	Additional threats .....	77
5.3.2	Organisational Security Policies .....	78
5.3.3	Assumptions.....	78
5.3.4	SPD and Security Objectives .....	78
<b>6</b>	<b>EXTENDED REQUIREMENTS.....</b>	<b>85</b>
<b>6.1</b>	<b>Extended Families .....</b>	<b>85</b>
6.1.1	Extended Family FCS_RNG - FCS_RNG: Random Number Generation .....	85
6.1.1.1	Description .....	85
	Extended Component FCS_RNG.1 .....	85
<b>7</b>	<b>SECURITY REQUIREMENTS .....</b>	<b>86</b>
<b>7.1</b>	<b>Security Functional Requirements.....</b>	<b>86</b>
7.1.1	CoreG_LC Security Functional Requirements.....	92
7.1.1.1	Firewall Policy .....	92
7.1.1.2	Application Programming Interface.....	99
7.1.1.3	Card Security Management .....	107
7.1.1.4	AID Management .....	109
7.1.2	InstG Security Functional Requirements .....	111
7.1.3	ADELG Security Functional Requirements .....	114
7.1.4	ODELG Security Functional Requirements.....	117
7.1.5	CarG Security Functional Requirements .....	118
7.1.5.1	Additional Security Functional Requirements for CM .....	122
7.1.5.2	Additional Security Functional Requirements for Resident application .....	125
7.1.5.3	Additional Security Functional Requirements for SmartCard Platform.....	132
7.1.5.4	Additional Security Functional Requirements for the applets .....	133
7.1.5.5	Additional Security Functional Requirements for Runtime Verification .....	134
<b>7.2</b>	<b>Security Assurance Requirements.....</b>	<b>140</b>
<b>7.3</b>	<b>Security Requirements Rationale .....</b>	<b>140</b>
7.3.1	Objectives .....	140



7.3.1.1	Security Objectives for the TOE.....	140
7.3.2	Rationale tables of Security Objectives and SFRs.....	148
7.3.3	Dependencies.....	158
7.3.3.1	SFRs Dependencies .....	158
7.3.4	Rationale for the Security Assurance Requirements .....	167
7.3.5	AVA_VAN.5 Advanced methodical vulnerability analysis.....	167
7.3.6	ALC_DVS.2 Sufficiency of security measures.....	167
<b>8</b>	<b>TOE SUMMARY SPECIFICATION .....</b>	<b>168</b>
<b>8.1</b>	<b>TOE Summary Specification.....</b>	<b>168</b>
<b>8.2</b>	<b>SFRs and TSS .....</b>	<b>175</b>
8.2.1	SFRs and TSS - Rationale .....	175
8.2.2	Association tables of SFRs and TSS .....	188

## LIST OF TABLES

---

Table 1 : Card identification data .....	13
Table 2 : IC Card Manufacturing data.....	14
Table 3: Guidance references.....	24
Table 4: TOE Life Cycle - Summary .....	32
Table 5: CC rationale.....	48
Table 6: Threats and Security Objectives – Coverage .....	80
Table 7: Security Objectives and Threats – Coverage.....	82
Table 8: OSPs and Security Objectives – Coverage.....	82
Table 9: Security Objectives and OSPs – Coverage.....	83
Table 10: Assumptions and Security Objectives for the Operational Environment – Coverage.....	84
Table 11: Security Objectives for the Operational Environment and Assumptions – Coverage.....	84
Table 12: Security Objectives and SFRs - Coverage .....	152
Table 13: SFRs and Security Objectives .....	157
Table 14: SFRs Dependencies .....	164
Table 15: SARs Dependencies .....	167
Table 16: SFRs and TSS – Coverage.....	193
Table 17: TSS and SFRs - Coverage .....	196



## 1 INTRODUCTION

This Public Security Target aims to satisfy the requirements of Common Criteria level EAL5+, augmented with AVA\_VAN.5, ALC\_DVS.2 in defining the security enforcing functions of the Target Of Evaluation and describing the environment in which it operates.

The basis for this composite evaluation is the composite evaluation of Platform and the hardware plus the cryptographic library.

NB: the IFX component provides the Mifare features. However these functionalities are out of the scope of the current ST.

### Application note

This TOE claims an assurance level EAL5+, i.e. augmented with ALC\_DVS.2 and AVA\_VAN.5. AVA\_VAN.5 implies that the TOE is resistant to attacks performed by an attacker possessing "High Attack Potential".

Not all key sizes specified in this security target have sufficient cryptographic strength for satisfying the AVA\_VAN.5 "High Attack Potential". In order to be protected against attackers with a "High Attack Potential", sufficiently large cryptographic key sizes SHALL be configured for this TOE. Please refer to national and international document standards for more and up-to-date details.

### 1.1 TOE Reference

<b>Product Commercial Names</b>	ID-One COSMO V9 Essential Platform
<b>Product Technical Name</b>	ID-One Cosmo V9 Essential on SLC32
<b>IC</b>	IFX_CCI_000005, IFX_CCI_000008 and IFX_CCI_000014 For more details see Table 2 : IC Card Manufacturing data
<b>TOE Name</b>	ID-ONE COSMO V9 ESSENTIAL
<b>TOE version</b>	3
<b>TOE Documentations</b>	Guidance 1.6.3 TOE Guidance
<b>MOC library</b>	ID3MOC V5.2

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

The TOE is identified by the Tag identity ('DF52', see below 1.1.1 Card Identification Data (tag 'DF52')) and CPLC data. These data are available by executing a dedicated command. Information and values to identified TOE are described of [AGD-OPE] document.

The tag identity provides information on the product and allow to identify each product configuration in term of features included or not in each specific product configuration.

#### 1.1.1 Card Identification Data (tag 'DF52')

Get Data command is used to retrieve the card identification as the mask identification, locks identification and patches identification.



Tag	Length	Value			
'01'	'01'	Hardware platform identifier			
		Length	Content	Value	Meaning
		1	Component number	'54'	Version of the component
'02'	'03'	Hardware platform version			
		Length	Content	Value	Meaning
		1	Chip interface	'00'	TCL / contact / mifare are available
				'04'	TCL / contact is available Mifare disable
				'05'	Only TCL is available
	'06'			Only contact is available	
	1	NVM Size	Maximum NVM Size		
			Mifare 1K No Loader	'70'	
			Mifare 1K Loader	'68'	
			Mifare 4K No Loader	'60'	
			Mifare 4K Loader	'58'	
			No Mifare No Loader	'82'	
			No Mifare Loader	'7A'	
1			NVM Configuration	Mifare 1K No Loader	'40'
	Mifare 1K Loader	'42'			
	Mifare 4K No Loader	'46'			
	Mifare 4K Loader	'48'			
	No Mifare No Loader	'44'			
	No Mifare Loader	'24'			
'03'	'02'	Mask identification			
		Length	Content	Value	Meaning
		1	Mask Number	'5B'	"ID-One Cosmo V9.0 Essential"
1	Mask version	'03'			
'04'		Jpatch identification			
		Length	Value	Content	
		'00'	0	-	No JPATCH
	'xx'	'xx'	'xx...xx'	Optional codes available if any	
'05'	'00'	FIPS Mode			
'06'	'10'	Locks identification			
		Length	Content		
		1	Checksum lock		
		1	FIPS lock		
		4	Post lock		
1	CVM lock				

		1	TCL Management lock	
		1	EC RSA lock	
		1	Security lock	
		1	Biometry lock	
		1	Rotation lock	
		1	CLA B02FE lock	
		2	Erase lock	
		1	Downsizing Info Lock	
'07'	'01'	Card state (life cycle)		
'08'	'xx'	TCL Contactless Historical bytes (only through TCL interface)		
'09'	'07'	TCL Contactless Protocol Bytes		
'0A'	'09'	KCV ('A0 MASTER MSK KCV 80 MSK KCV 81 LSK KCV')		
'0B'	'06'	Commercial version & Extended Commercial version (09455353 0000')		
'0C'	'04'	SAAAAR Code	<b>OS</b>	Item number encoded in BCD ('F' byte used for not signifying digits)
			<b>Value</b>	
'0D'	'01'	Mifare identification		
		<b>Value</b>		<b>Meaning</b>
		'1X'		Mifare Classic 1K available (Card Image depends on the prepersonalization phase)

Table 1 : Card identification data

### 1.1.2 IC Card Manufacturing Data (tag 'DF50')

The 37 bytes of the IC Card Manufacturing are structured as following:



Name		Format	Length
Chip Serial Number	SPLIT_LOT_NUMBER	'XX'	1 byte
	WAFER_NUMBER_IN_LOT	'XX'	1 byte
	COORDINATE_X	'XX XX'	2 bytes
	COORDINATE_Y	'XX XX'	2 bytes
Batch Number	PROD_SITE	'XX'	1 byte
	PROD_BUSINESS_YEAR	'XX'	1 byte
	LOT_NUMBER	'XX XX'	2 bytes
Device Coding (DC)	SLC32GDL400G3	'0013 002B 0000'	6 bytes
	SLC32GDA400G3	'0013 002A 0000'	
	SLC32GDA348G3	'0013 002E 0000'	
	SLC32GDL348G3	'0013 002F 0000'	
	SLC32PDL400	'0012 0054 0000'	
ROM Code Number (RCN)		'5B 03 00'	3 bytes
Mask Number		'5B'	1 byte
Mask Version		'03'	1 byte
Component Number		'54'	1 byte
IC Type	Bytes number 2 and four of Device Coding (DC) field (see above)		2 bytes

Table 2 : IC Card Manufacturing data

### 1.1.3 TOE Reference

TOE Name	ID-One COSMO V9 Essential Platform
Platform code identification	089233
TOE version	3
TOE Label	FINAL_089233
IC reference	IFX SLC32xDy SLC32xDy (x = G or P and y = L or A)
IC ST identification	[R26]
IC EAL	EAL6 augmented by ALC.FLR.1
IC certificate	BSI-DSZ-CC-0945-V2-2018
Date of IC certification	20 April 2018

## 1.2 Security Target Overview

The main objectives of this Security Target are to:

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

## 1.3 REFERENCES

Ref	Document details
[R1]	"Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", April 2017, Version 3.1 revision 5.
[R2]	"Common Criteria for information Technology Security Evaluation, Part 2: Security Functional component", April 2017, Version 3.1 revision 5.
[R3]	"Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance components", April 2017, Version 3.1 revision 5.
[R4]	"Composite product evaluation for Smart Cards and similar devices" August 2015, Version 1.4
[R5]	Java Card Protection Profile – Open Configuration Version 3.0 May 2012 ANSSI-CC-PP-2010/03_M01
[R6]	"Java Card 3.0.5 Classic - API" Application Programming Interfaces, Version 3.0.5, 2015, Oracle Technology Network
[R7]	"Java Card – JCRE" Runtime Environment Specification, Classic Edition Version 3.0.5, 2015, Oracle Technology Network
[R8]	"Java Card 3.0.5 - Virtual Machine Specifications" Virtual Machine Java Card™ Platform, Version 3.0.5, 2015, Oracle Technology Network
[R9]	"GlobalPlatform Card Specification" Version 2.3 Public Release - October 2015 Document Reference: GPC_SPE_034
[R10]	GlobalPlatform Card Technology Card Specification – ISO Framework" Version 0.9.0.18 - Public Review July 2013



	Document Reference: GPC_SPE_055
<b>[R11]</b>	GlobalPlatform Card - Common Implementation Configuration” Version 2.0 Member Release - November 2015 Document Reference: GPC_GUI_080
<b>[R12]</b>	GlobalPlatform Card Technology - Secure Channel Protocol '03', Card Specification v2.2 – Amendment D” Version 1.1.1 - Public Release July 2014 Document Reference: GPC_SPE_014
<b>[R13]</b>	GlobalPlatform Card Technology - Security Upgrade for Card Content Management, Card Specification v2.2 - Amendment E” Version 1.0 – Public Release November 2011 Document Reference: GPC_SPE_042
<b>[R14]</b>	"Identification cards - Integrated Circuit(s) Cards with contacts, Part 6: Interindustry data elements for interchange" ISO/IEC 7816-6 (2004)
<b>[R15]</b>	"Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)" ANSI X9.31-1998, American Bankers Association
<b>[R16]</b>	"FIPS PUB 46-3, Data Encryption Standard" October 25, 1999 (ANSI X3.92), National Institute of Standards and Technology
<b>[R17]</b>	"FIPS PUB 81, DES Modes of Operation" April 17, 1995, National Institute of Standards and Technology
<b>[R18]</b>	"FIPS PUB 180-3, Secure Hash Standard" October 2008 , National Institute of Standards and Technology
<b>[R19]</b>	"FIPS PUB 186-3" June 2009, Digital Signature Standard (DSS)
<b>[R20]</b>	"Public Key Cryptography using RSA for the financial services industry" ISO/IEC 9796-1, annex A, section A.4 and A.5, and annex C (1995)
<b>[R21]</b>	"Information technology – Security techniques: Data integrity mechanism using a cryptographic check function employing a block cipher algorithm" ISO/IEC 9797-1 (1999) , International Organization for Standardization
<b>[R22]</b>	"FIPS PUB 140-2, Security requirements for cryptographic modules" Mars 2002 , National Institute of Standards and Technology
<b>[R23]</b>	PKCS#1 The public Key Cryptography standards RSA Data Security Inc. 1993



<b>[R24]</b>	Security IC Platform Protection Profile with Augmentation Packages Version 1.0, 13 January 2014, BSI-CC-PP-0084-2014
<b>[R26]</b>	<a href="#">Public Security Target</a> , BSI-DSZ-CC-0945-V2-2018, Version 0.7, 06.11.2017, “Public Security Target IFX_CCI_000003h, IFX_CCI_000005h, IFX_CCI_000008h, IFX_CCI_00000Ch, IFX_CCI_000013h, IFX_CCI_000014h, IFX_CCI_000015h, IFX_CCI_00001Ch, IFX_CCI_00001Dh, IFX_CCI_000021h, IFX_CCI_000022h design step H13”, Infineon Technologies AG (sanitised public document)
<b>[R27]</b>	IEEE Std 1363a-2004 Standard Specification of Public-Key Cryptography
<b>[R28]</b>	FIPS PUB 197, The Advanced Encryption Standard (AES) U.S. DoC/NIST, November 26, 2001.
<b>[R29]</b>	Certification of « open » smart card products, Version 1.1 (for trial use), 4 February 2013.
<b>[R30]</b>	The NIST SP 800-90 Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revise) March 2007
<b>[R31]</b>	Référentiel général de sécurité Processus de qualification d'un produit de sécurité - niveau renforcé – version 1.0
<b>[R34]</b>	ANSI x9.62-2005 Public Key Cryptography for the Financial Services Industry – The Elliptic Curve Digital Signature Algorithm (ECDSA)
<b>[R35]</b>	ANSI x9.63-2001 Public Key Cryptography for the Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography
<b>[R36]</b>	JIL-Guidance-for-smartcard-evaluation-v2-0
<b>[R37]</b>	ID-One COSMO V9 Essential Security Recommendations FQR 110 8794 Ed4 – 29/10/2018
<b>[R38]</b>	ID-One COSMO V9 Essential Reference Guide FQR 110 8823 Ed5 – 22/10/2018
<b>[R39]</b>	ID-One COSMO V9 Essential Pre-Perso Guide FQR 110 8797 Ed5 – 22/10/2018
<b>[R40]</b>	ID-One COSMO V9 Essential Application Loading Protection Guidance FQR 110 8798 Ed2 – 24/09/2018
<b>[R41]</b>	The Java Virtual Machine Specification. Lindholm, Yellin ISBN 0-201-43294-3
<b>[R42]</b>	Java Card 3 Platform Off-card Verification Tool Specification, Classic Edition, Version 1.0. Published by Oracle
<b>[R43]</b>	Java Card System Standard 2.2 Configuration Protection Profile – PP/0305



	Version 1.0b – August 2003
<b>[R48]</b>	FQR 110 8827 Ed1- 23/04/2018 - Java Card API on ID-One Cosmo V9 platform
<b>[R49]</b>	FQR 110 8873 Ed0.2 – 12/07/2018 - Cosmo flash image generation
<b>[R50]</b>	FQR 110 8779 Ed2 – 10/12/2018 - ID-One COSMO V9 Essential – Security Target
<b>[R51]</b>	FQR 110 8921 Ed1 – 24/09/2018 - Secure acceptance and delivery of sensitive elements

## 1.4 ACRONYMS and NOTATIONS

### 1.4.1 Abbreviations

AES	Advanced Encryption Standard
AID	Applet Identifier
APDU	Application Protocol Data Unit
API	Application Programmer Interface
APSD	Application Provider Security Domain
BIOS	Basic Input/Output System
CASD	Controlling Authority Security Domain
CC	Common Criteria
CM	Card Manager
CPLC	Card Production Life Cycle
DAP	Data Authentication Pattern
DES	Cryptographic module "Data Encryption Standard"
EAL	Evaluation Assurance Level
EC	Elliptic Curves
EEPROM	Electrically Erasable and Programmable Read Only MemorES Embedded Software
FAT	File Allocation Table
GP	Global Platform
IC	Integrated Circuit
ISD	Issuer Security Domain
IT	Information Technology
JCP	Java Card Platform
JCRE	Java Card Runtime Environment
MSK	Master Secret Key
LSK	Load Secret Key
MOC	Match-On-Card
OSP	Organizational Security Policy



PP	Protection Profile
RNG	Random Number Generation
ROM	Read Only Memory
RSA	Cryptographic module "Rivest, Shamir, Adleman"
SF	Security Function
SFP	Security Function Policy
SHA-1	Cryptographic module "Secure hash standard"
ST	Security Target
TOE	Target of Evaluation.
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSP	TOE Security Policy
VASD	Validation Authority Security Domain
VM	Virtual Machine

#### 1.4.2 Definitions

**Applet** Application which can be loaded and executed with the environment of the Java Card platform

#### Applet developer

The one who is in charge of the Applet development intended to be loaded on the platform. He/she is the recipient of the [R37] in order to respect recommendations, if any, identified by the platform evaluation. These recommendations shall be followed by the applet developer and shall be checked before loading the application on the platform.

**Card Issuer** Entity that owns the card and is ultimately responsible for the behaviour of the card

**Card Manager** Main entity which represents the issuer and supervises the whole services available on the card. The Card Manager entity encompasses the Open and the Issuer Security domain.

**DAP** Part of the Load File used for ensuring authenticity of the Load File. The DAP is the signature of the Load File Data Block Hash and is provided during the loading.

#### Issuer Security Domain

The primary on-card entity providing support for the control, security, and communication requirements of the Card Issuer.

#### Load File Data Block Hash

The Load File Data Block Hash provides integrity of the Load File Data Block following receipt of the complete Load File Data Block.



**OPEN** Part of the Card Manager entity which has the responsibilities to provide an API to applications, command dispatch, Application selection, logical channel management, and Card Content management. The OPEN also manages the installation of applications loaded to the card. The OPEN is responsible for enforcing the security policy defined for Card Content management.

**Security Domain** On-card entity providing support for the control, security, and communication requirements of an off-card entity (e.g. the Card Issuer, an Application Provider or a Controlling Authority).

**Match On Card** The match on card (MOC) technology consists in registering the fingerprints (or their template) on a smart card or a USB key, the card being unlocked using the finger that functions as code

**MSK** Master Secret Key for authentication used to authenticate the card Manufacturer. This key has a given value (i.e. MSK value) and its try counter, i.e. MSK Key counter (as for a PIN). Once the try limit counter is reached, the authentication is no more possible with this key.

**LSK** Secret Key used by the OS developer (the TOE developer) to encrypt locks, patches and ISK keys.

This Security target addresses a family ID-ONE COSMO V9 Essential based on IFX component SLC32XDY (X = G or P and Y = L for Mifare or A without Mifare) with NVM size 348 or 400 (348 Kbytes is commercial derivative product)

Family Name	Product Name	IDEMIA reference code	Chip Reference	Chip Reference/Chip configuration In the public ST lite
ID-One COSMO V9 Essential	ID-One COSMO V9 Essential Platform	083623	SLC32 family SLC32XDY See above	IFX_CCI_000005 IFX_CCI_000008 IFX_CCI_000014 See [R26]

The “ID-One COSMO V9 Essential Platform” is dual.

The present security target addresses only the platform, not the applet that may be loaded on, (the Card Manager applet is in the scope of the evaluation as it is an entire part of the platform)

## 1.5 TOE OVERVIEW

### 1.5.1 TOE Type

The ID-One COSMO V9 Essential Platform on Infineon (IFX) chip is a dual Java Card platform based, compatible with multi-application ID-One Cosmo product family.

The functional level of the OS is based on a Java™ based multi-application™ platform, compliant with Java Card 3.0.5 Classic Edition and Global Platform 2.3 specifications.

This ID-ONE COSMO V9 Essential platform is able to receive and manage different types of applications, Basic and Sensitive ones.



All the platform code including GP Java application called card manager are loaded in the FLASH memory.

The product is open at use phase. However, the Issuer can forbid the loading of applets before or after the issuance of the card.

The basic or sensitive applications can be loaded on the flash memory, at prepersonalisation, personalisation or use phase. These applications are out of the scope of the present evaluation.

#### 1.5.2 TOE usage

This Platform is an open and isolating platform that is compliant with the ANSSI Application Note 10 that deals with open and isolating platforms.

Smart cards are used as data carriers that are secure against forgery and tampering as well as personal, highly reliable, small size devices capable of replacing paper transactions by electronic data processing. Data processing is performed by a piece of software embedded in the smart card chip, called an application.

The Java Card System is intended to transform a smart card into a platform capable of executing applications written in a subset of the Java programming language. The intended use of a Java Card platform is to provide a framework for implementing IC independent applications conceived to safely coexist and interact with other applications into a single smart card.

Applications installed on a Java Card platform can be selected for execution when the card communicates with a card reader.

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

So far, the most typical applications are:

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



This platform provides a highly secure technology for smartcards applications. The Match-On-Card technology is an entire part of the product, and enables the Authentication by way of digital prints.

## 1.6 PRODUCT ARCHITECTURE

### 1.6.1 Logical scope of the TOE

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

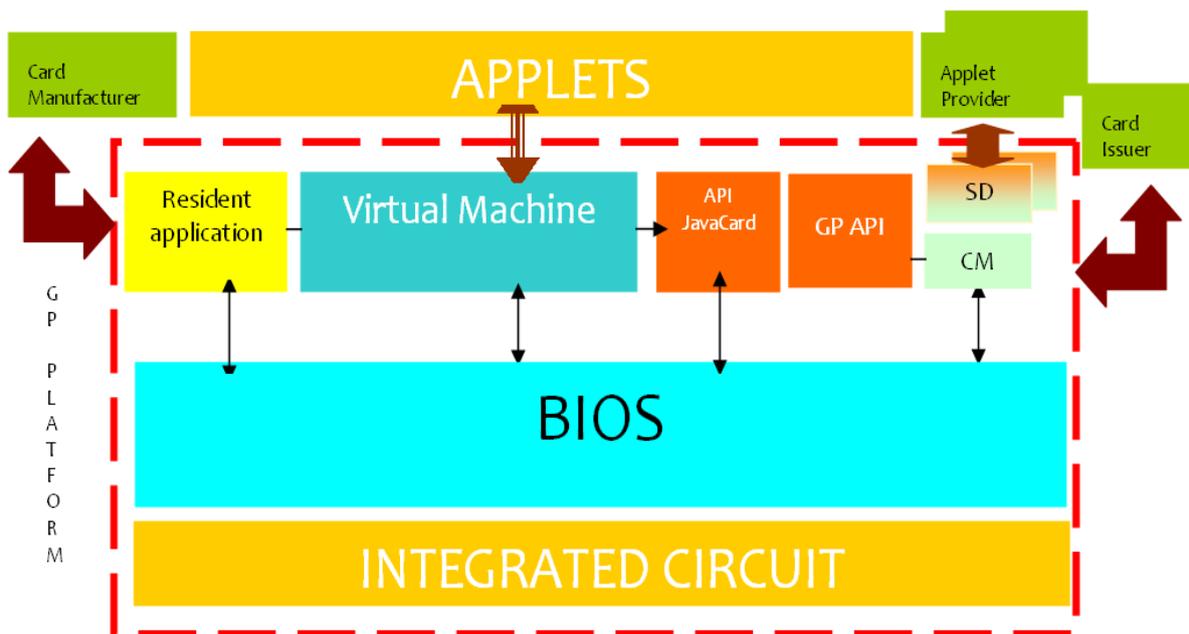


Figure 1: Java Platform Architecture

The TOE includes the BIOS, the Virtual Machine, the APIs, the Global Platform application, the Resident application and the IC component. Details of components are presented in the TOE description.

## 1.6.2 Physical scope of the TOE

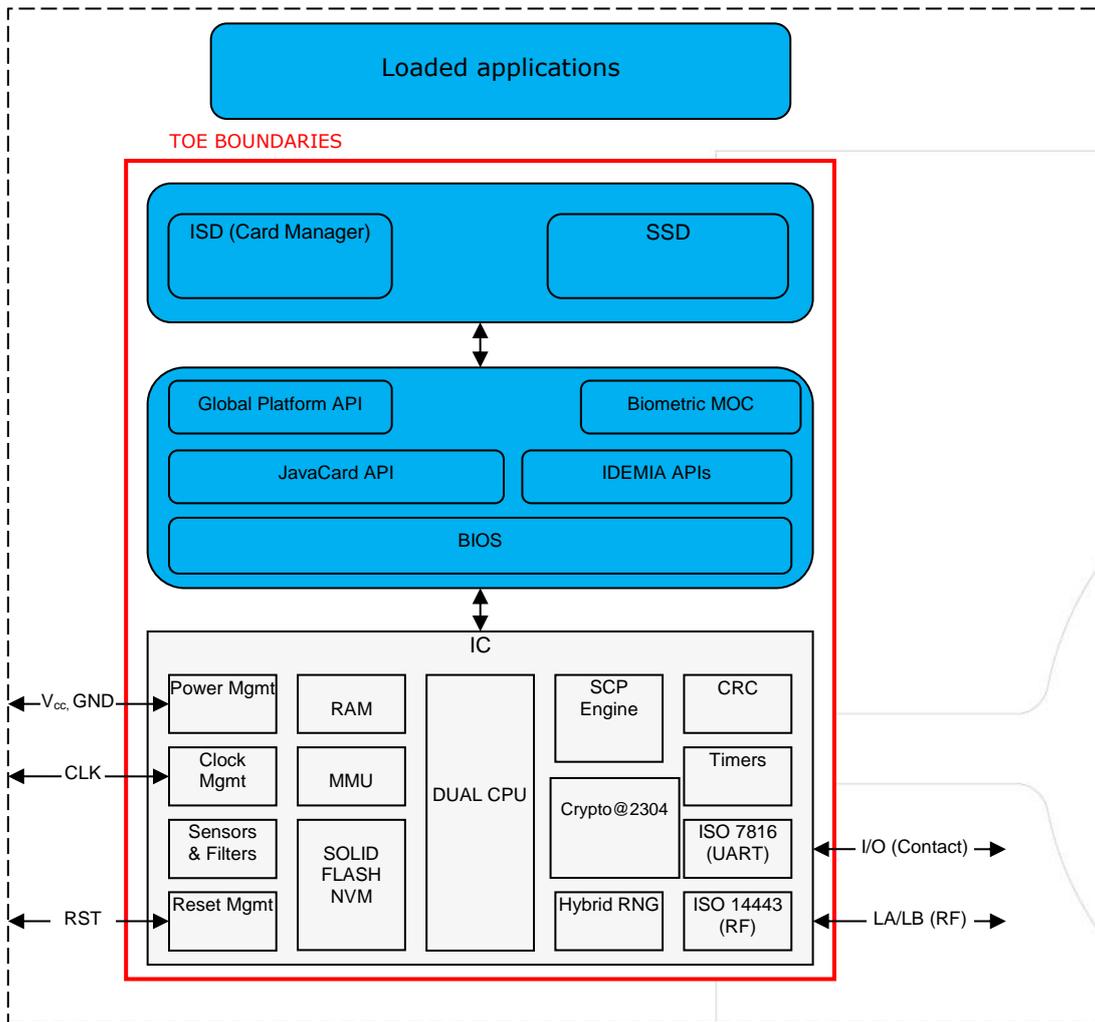


Figure 2: TOE Boundaries

IC libraries:

Object	Version
HSL	V01.22.4346-SLCx2_C65.lib
MCS (Mifare lib)	V02.03.3446

IC Identifiers:

Common Criteria IDs
IFX_CCI_000005
IFX_CCI_000008
IFX_CCI_000014

Ref to [Public Security Target](#) for more details

The ID-One COSMO V9 Essential on IFX is a dual Java Card platform based, compatible with multi-application ID-One Cosmo product family.

The TOE is defined by:

- The underlying IC with its dedicated software,



- the OS based on a Java™ based multi-application platform, compliant with **Java Card 3.0.5 Classic Edition** and **Global Platform 2.3** specifications.
- The ability to receive and manage different types of applications, Basic and Sensitive ones.
- All the platform code including GP Java application called card manager are loaded in the FLASH memory.
- The product is open at use phase.
- The basic or sensitive applications can be loaded on the flash memory, at prepersonalisation, personalisation or use phase. These applications are out of the scope of the present evaluation.

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

Java Card RMI is not implemented in the TOE.

### 1.6.3 TOE Guidance

The ID-ONE COSMO V9 ESSENTIAL is evaluated with its guidance. The Public ST derived from this restricted ST is also a guidance to all users of the TOE.

The guidance's of the Platform are listed hereafter:

Audience	Ref	Title	Form factor of delivery
Developer of sensitive applications	[R37]	ID-ONE COSMO V9 ESSENTIAL Security Recommendations	Electronic version
Guidance for application developer	[R38]	ID-ONE COSMO V9 ESSENTIAL Reference Guide	
	[R48]	Java Card API on ID-One Cosmo V9 platform	
For prepersonalisation	[R39]	ID-ONE COSMO V9 ESSENTIAL Pre-Perso Guide	
Issuer of the platform that aims to load applications	[R40]	ID-ONE COSMO V9 ESSENTIAL Application Loading Protection Guidance	
Chip Manufacturer Third party Production sites	[R51]	Secure acceptance and delivery of sensitive elements	

Table 3: Guidance references

Some guidance are mandatory, they shall be used by its users. Some guidance are not mandatory, they constitute a help to users of the TOE or developer of applet to load on the TOE.



#### 1.6.4 Platform isolation

To ensure the platform isolation, see objective OE1 [29] the following verifications must be done:

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

2: The byte code verification (required for any package intended to be loaded on the platform) must be done using export files provided by IDEMIA.

Those verifications shall be done for all application intended to be loaded on the platform.

#### 1.6.5 Applications

For sensitive application, the recommendations listed in [R37] are mandatory. The evaluator of the sensitive application, checks that the guidance is followed by the sensitive application developer.

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

This check shall be ensured by the organisational measures or by security mechanisms.

The platform is evaluated without applications.

#### [R37]

If the applet needs to have a security certification, the applet must follow recommendations listed in the document.

If the applet is a basic application, and does not need security certification with the platform, the certificate of the Platform is still valid if the applet go through the verifier before the loading of this applet (the security function of the platform are still available).

This guide is provided to the Developer and evaluator of a sensitive application to be certified.

#### [R38]

This document describes the ID-ONE COSMO V9 Essential smart card usage. It describes how to use the card from an APDU commands point of view and gets onto topics such as common platform APDU commands, secure channels and security domains.

This document also describes the available JavaCard and proprietary APIs for applet developers.

This guide is provided to the Developer of JavaCard applications to be certified or not. It does not mandate any requirement for the developer; it constitute a help.

#### [R39]

This document describes the prepersonalisation steps that should be followed to correctly initialize the ID-ONE COSMO V9 Essential platforms. The TOE is finalized once it is prepersonalised.

This guide is provided to the user (in phases 4-5).

#### [R40]



This document describes the loading procedure, in compliance with ANSSI Note 10 and the Java Card Open Platform protection profile.

The [R40] is provided to the Loading Authority, who is in charge of loading an application.

[R48]

This document summarizes the ID proprietary API (packages, classes, methods and fields) available on the Javacard Identity Platforms.

This guide is provided to the Developer of JavaCard applications to be certified or not. It does not mandate any requirement for the developer; it constitute a help.

[R51]

This document specifies the secure acceptance of the components that comprise the TOE.

## 1.7 TOE Description

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

### 1.7.1 Defensive Java Card Platform

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

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

The TOE is compliant with the version of the Java Card 3.0.5 classic edition, specified by three documents related to Java Card API, Java Card Runtime Environment and Java Card Virtual Machine Specifications, defined respectively in [R6], [R7] and [R8]. The next paragraph introduces those three elements.

As the terminology is sometimes confusing, the term “Java Card System” has been introduced in that defines the set constituted by the Java Card RE, the Java Card VM and the Java Card API.

The Java Card System provides an intermediate layer between the operating system of the card and the applications. This layer allows applications written for one smart card platform enabled with Java Card technology to run on any other such platform.

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

Applet isolation is achieved through the Java Card Firewall mechanism defined in [R7]. This mechanism confines an applet to its own designated memory area. Thus, each applet is prevented from accessing fields and operations related to objects owned by other applets, unless those applets provide a specific interface (shareable interface) for that purpose. This access control policy is enforced at runtime by the Java Card VM.

However, applet isolation cannot be entirely granted by the firewall mechanism if certain well-formedness conditions are not satisfied by loaded applications.



Therefore, a bytecode verifier (BCV) formally verifies those conditions. The BCV is out of the scope of the Java Card System defined in [R5].

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

And as this security target claims a demonstrable conformance to [R5]. The off card verifier is also used. All applications are verified by the latest Oracle off card verifier.

Possible Verifier	Type	When?
Off-card bytecode verifier	Static	Once, outside of the card
Runtime verifier	Dynamic	Every time, during execution

The Java Card API (JCAPI) provides classes and interfaces for the core functionality of a Java Card application. It defines the calling conventions by which an applet may access the JCRE and services such as, among others, I/O management functions, PIN and cryptographic specific management and the exceptions mechanism. The JCAPI is compatible with formal international standards, such as ISO/IEC 7816 and industry specific standards.

#### 1.7.2 Global Platform

The TOE is compliant with the Global Platform 2.3 (GP) standard [R13] which provides a set of APIs and technologies to perform in a secure way, the operations involved in the management of the applications hosted by the card. Using GP maximizes the compatibility and the opportunities of communication as it becomes the current card management standard.

The main features addressed by GP are:

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

These operations are addressed by a set of APIs used by the applications hosted on the card in order to communicate with the external world on a standard basis.

The version considered in this document is version 2.3 of the GP Card specification. The following GP functionalities, at least, are present within the TOE:

- Card content loading
- Extradition
- Asymmetric keys
- DAP support, Mandated DAP support
- DAP calculation with asymmetric cryptography
- Logical channels
- SCP02 support
- SCP03 support [R12]



- Support for contact and contactless cards different implicit selection on different interfaces and channels
- Support for Supplementary Security Domains
- Trusted path privileges
- Post-issuance personalisation of Security Domain [R12]
- Application personalisation [R12]
- Crypto algorithms as detailed in 1.7.2.2 Cryptographic features

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

The platform is designed on IFX components:

#### 1.7.1.1 IFX Secure Smart Card Controller

The configuration used is the IFX SLC32xDy SLC32xDy (x = G or P and y = L or A) with and without MIFARE implementation.

X: G or P.

L: mean with Mifare

A: means without Mifare

Regarding NVM size, 348KBytes is a derivate from 400 Kbytes chip.

FLASH = 384/400 KB

RAM = 9KB up to 12KB

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

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

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

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

More information regarding the components is available in the public security target of the chip [R26].

### 1.7.2 Operating System (OS)

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

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



### 1.7.2.1 BIOS

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

- APDU management, using T=0, T=1 and T=CL protocols (Type A and type B)
- Timer management
- Exceptions management
- Transaction management
- Flash memory access

### 1.7.2.2 Cryptographic features

The following crypto services are included in the OS:

Cryptographic Services	
RSA from 1024 to 2048-bits by step of 256-bits	References are standard ones
ECC with 160, 192, 256, 384, 512 and 521-bits key sizes	
TDES with 56, 112 and 168-bits key sizes	
AES with 128, 192, 256 key sizes	
SHA-1, SHA 224, 256, 384 and 512 (for data integrity only does not provide confidentiality)	
RSA, ECC Key generation	
CRC 16 and CRC 32 (for data integrity only does not provide confidentiality)	
RNG CTR_DRBG SP800-90	
RSA signature/verification	Based on supported RSA key sizes
ECDSA signature/verification	Based on supported ECC key sizes
ECDH	Based on supported ECC key sizes
AES secure messaging	References are standard ones
TDES secure messaging	
HMAC (64 bits up to 1016 bits)	

### 1.7.2.3 Biometric feature

ID-ONE COSMO V9 ESSENTIAL embeds the MOC algorithm.

The biometric feature allows matching a CANDIDATE Template with REFERENCE Templates (up to 10). However this feature is out of the scope of this ST.

### 1.7.2.4 Virtual Machine

The Virtual Machine, which is compliant with the Java Card 3.0.5 classic edition, interprets the byte code of Java Card applets.

The Virtual Machine supports logical channels; this means that it allows an applet to be selected on a channel, while a different applet is selected on another channel.

It also supports secure execution of applets loaded and stored in ROM.

The Virtual Machine is activated upon the selection of an applet.



#### 1.7.2.5 The Java Card Runtime Environment

The Java Card Runtime Environment (JCRC) contains the Java Card Virtual Machine (VM), the Java Card Application Programming Interface (API) classes and industry-specific extensions, and support services. For details please refer to reference [R7].

#### 1.7.2.6 APIs

The APIs, compliant with the Java Card 3.0.5 classic edition, support key generation, Key Agreement, signature, ciphering of messages and proprietary IDEMIA API.

Proprietary APIs [R48] have been developed like ISOSecureMessaging to assure the data are exchanged in confidentiality and integrity; utilBER\_Reader to read BER-TLV; SecureStore to store integrity sensitive information, SAC (to perform the PACE access control) API for SAC computation that are used to compute generic or integrated mapping.

#### 1.7.2.7 Open and isolating Platform

This security target claims conformance to the Application Note 10 on Open and Isolating platform, issued by ANSSI [R29].

An “open platform” can host new applications:

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

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

#### 1.7.2.8 Resident Application

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

It also deals with the Card Manufacturer authentication.

The dispatcher is always activated. commands for administration are only available during prepersonalisation phase.

#### 1.7.2.9 Applets

Applets bytecodes shall go thru the latest Oracle and IDEMIA off card verifiers before the loading.

The platform evaluation shall identify, if any, recommendations in order to maintain isolation properties. These recommendations then shall be followed by the applet developer and shall be checked before loading.



**1.8 LIFE-CYCLE**

The development and manufacturing processes of the Composite Product is separated into seven distinct phases to be in accordance with the Java Card™ System Protection Profile [R24] (section 2.4, figure 3). Each phase is under the control of one (or several) administrator(s) and protected by an environment. Each phase is covered by the assurance components.

Phase	Phase name	Covered by	Comment
1	Security IC Embedded Software development	ALC[PLT]	The Platform code development shall take place in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of all these elements.
2	Security IC Development	ALC[IC]	Correspond to "IC Development" of the PP [R24]
3	Composite Product Integration (with Option a)	ALC[IC]	Option a) Loading platform code on the IC in secured environment only (chip manufacturer sites). IDEMIA code is send following the IFX procedure with "secureX" refer to [R49]

Composite product delivery with option a)

4	Security IC Packaging	ALC[IC]	Integration in an appropriate form factor (wafers, card, token ...).
5	Composite Product Integration (2 Options b) + Prepersonalisation	AGD_PRE for case b-1) Or ALC[PLT] For case b-2)	<p>The loading of the platform (and applets if required) based on script generated that contains generic data (non-individual ones). A custom image is prepared at and by IDEMIA R&amp;D [R49].</p> <p>If the loading is not realized at previous phase, 2 cases are available: b-1) the Loading is done thanks to IC Package 2 at IDEMIA sites or partners/integrator sites. b-2) the loading is done thanks to IC Package 1 at IDEMIA Audited Sites.</p> <p>The delivery of the TOE occurs at the end of this phase: Once the OS with the potential applications are loaded on the IC</p>

Composite product delivery with options b)



6	Composite Product Personalisation	AGD_OPE Or ALC[PLT]	Correspond to “Product Personalisation” of the PP [R24], PREPERSONALISATION AND PERSONALISATION of the TOE.  Reference from [R29] for loading application : [ISO_VERIF]: <ul style="list-style-type: none"> <li>- [ORG_LOAD] if the loading is done at IDEMIA audited sites</li> <li>- or [TECH_LOAD] if the application loading is done outside Idemia audited sites</li> </ul>
7	Operational Usage	AGD_OPE	Reference from [R29] for loading application : [ISO_VERIF] [TECH_LOAD]

Table 4: TOE Life Cycle - Summary

Note:

ALC[PLT] refers to IDEMIA audited sites

ALC[IC] refers to the IFX audited sites

[ISO\_VERIF]: is related to chapter 1.6.4 Platform isolation.

[ORG\_LOAD] is related organisational measure to ensure the integrity and authenticity of the application after the verifications defined in chapter 1.6.4 Platform isolation. This guidance is defined in [R40]

[TECH\_LOAD] is related to the mandatory use of the mandated DAP specified in the present security target. The guidance is defined in [R40]

1.8.1 Phases



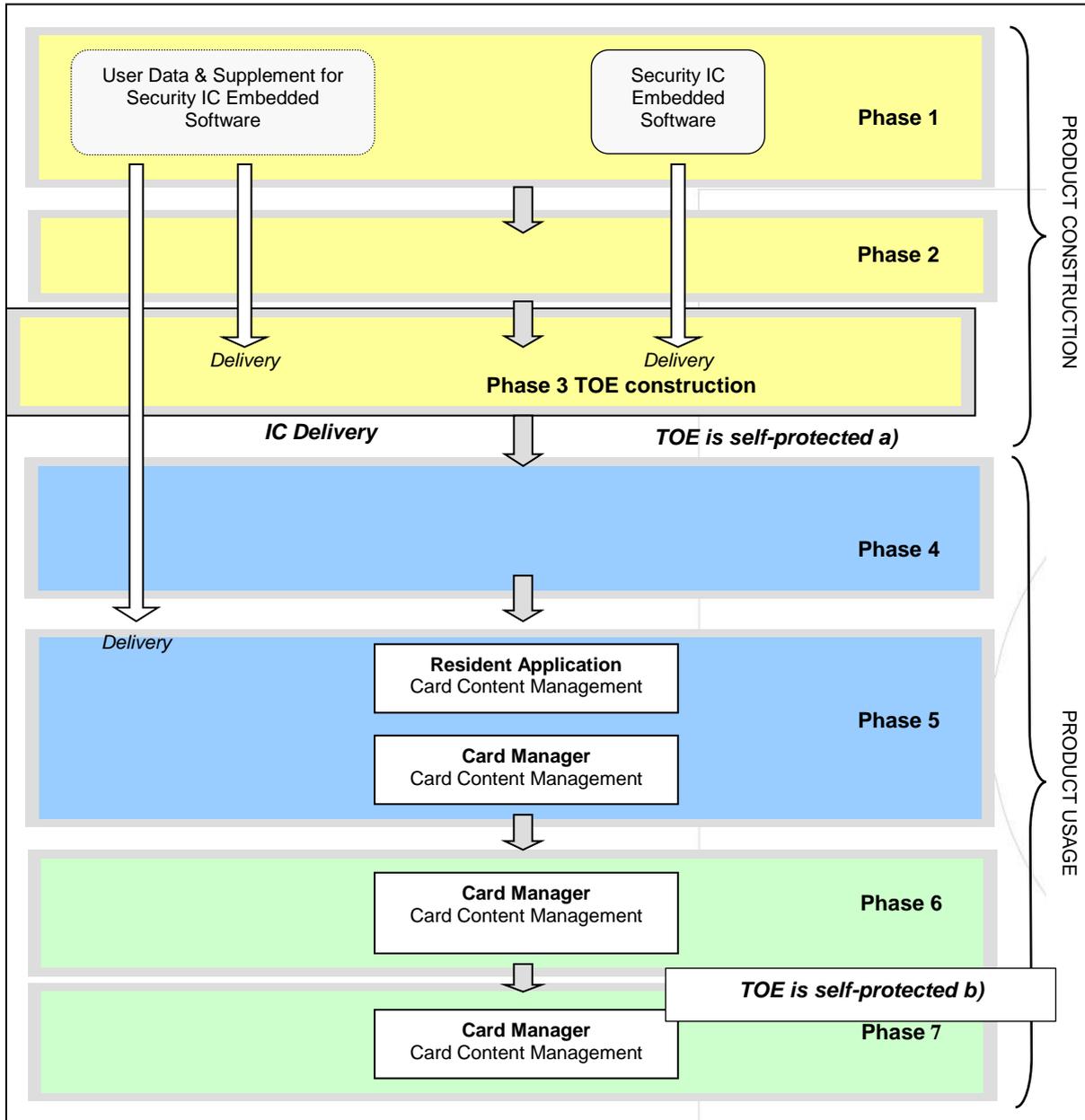


Figure 3: TOE Life Cycle – Overviews

### 1.8.2 Phase 1: Security IC Embedded Software development

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

The code and the associated data are sent  
To the IC manufacturer for loading on the IC, option a).



To IDEMIA audited sites, option b-1  
To IDEMIA or third party sites, option b-2

### 1.8.3 Phase 2: Security IC Development

The Composite Product life cycle covers Security IC development which is described in the IC ST identification (see [R26] ).

### 1.8.4 Phase 3 and phase 4: Security IC Manufacturing and packaging

The Phase 3 of the Composite Product life cycle covers the IC production and when required for option a) the loading of the platform code on the flash memory. This loading is done thanks to the IC security functions.

For option a, when the code is loaded by IFX, the TOE delivery is at the end of phase 3 at any form factor of the chip (on wafer, modules, inlay, cards PVC/PETF or on die...).

### 1.8.5 Phase 5: Composite Product Integration

Three cases are covered in this phase:

Case a) For the case, where the IC is directly delivered without the OS, the loading takes place at this phase.

Case b-1) At IDEMIA audited sites, the loading is done thanks to loading package 1. Following IFX guidances.

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

### **The delivery is done at this step.**

From this phase a) or b), the Javacard Platform Embedded Software is active. The life cycle is defined in the following sections.

The Life Cycle of the TOE is defined by the Card Manager Life cycle. The states in this phase are Pre\_production, OP\_READY and INITIALIZED.

This Card Life Cycle begins in **Pre\_production** state. During this state, the Resident Application manages the Javacard Platform Embedded Software and ensures the security. This state supports:

- the Javacard Platform Embedded Software configuration (by loading User Data in clear, User Data ciphered, Supplement for Javacard Platform Embedded Software ciphered by LSK),
- the Card Content Management (activating Load File from persistent memory, loading, installation, and deletion of Load Files and applet instances),
- Card Manager configuration (first key-set ...).

By switching from the **Pre\_production** state up to **OP\_READY**, the Card Manager replaces the Resident Application. From this state, the Card Manager ensures the Card Content Management (loading, installation, and deletion of Load Files and applet instances) and provides security in place of the Resident Application.

At the end of this phase, the state is the Card Life Cycle state is **INITIALIZED**.



#### 1.8.6 Phase 6: Composite Product Personalisation

The Card Life Cycle states defined in this phase are INITIALIZED and then SECURED.

These states allow the Card Content Management.

The applets already instantiated may be personalized. Additional Load Files may be loaded.

At the end of this phase:

Card Life Cycle state is SECURED the Security Domain Mandated DAP is fully operational (PERSONALIZED) and protects the Load File loading during Phase 7.

#### 1.8.7 Phase 7: Operational Usage

In this phase, the Card Life Cycle state is SECURED.

This state allows the Card Content Management. The applets may be instantiated and personalized from Load File already present or from additional Load File loaded during this phase. The applets already instantiated may be personalized.

Additional Load File may be loaded and requires a DAP Verification to ensure the Load File integrity. This guidance shall be applied [R40].

#### 1.8.8 Application loading

The applets can be loaded at phases 5, 6 and 7 :

At IDEMIA audited sites, in phases 5 and 6, the DAP mechanism is not enforced; applet loading is performed in IDEMIA site, using IDEMIA process. Before loading each applet, the .cap file is checked by IDEMIA verifier, mandatory in this ST.

For the other cases, after platform code loading, in phase 5, 6 and 7 the DAP mechanism is mandatory.

### 1.9 Software Components Life Cycle

#### 1.9.1 Card Life Cycle



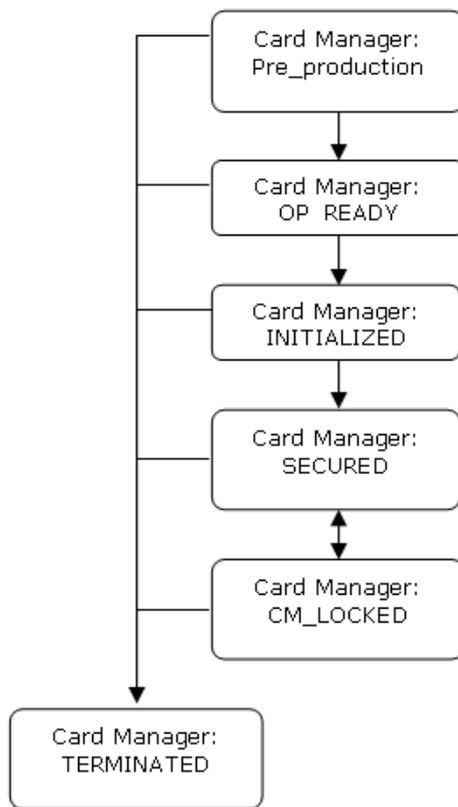


Figure 4: Card Life Cycle

### Pre\_production

This initial life state of the Card allows managing the prepersonalisation of the Javacard Platform Embedded Software up to the Card Manager Life Cycle OP\_READY.

During this state, the Resident Application provides a set of APDU commands which allows:

- Writing User Data for configuring the Javacard Platform Embedded Software. This configuration (by using lock mechanism) is only carried out during this state.
- Writing Supplement for Javacard Platform Embedded Software (patch code). It is developed at IDEMIA premises (Phase 1), delivered and loaded securely in volatile memory (Flash/) during the Composite Product Integration (Phase 5). The security of this loading is fully enforced by technical measures provided by the TOE, and evaluated by the ITSEF. This task is only carried out during this state.
- Activating Load Files from immutable persistent memory (ROM). This task is only carried out during this state.
- Loading Load Files from mutable persistent memory (Flash).
- Instantiating the Issuer Security Domain (Card Manager). Only one ISD is available by card.
- Populating with initialization key (ISK) and Chip CPLC. The Card Life Cycle switches automatically in OP\_READY state when the initialization key (ISK) is populated in the ISD.

The APDU commands depend on the mutual authentication carries out (by using MSK).

The next states possible are OP\_READY or TERMINATED. The transition is irreversible.



## OP\_READY

During this life cycle state, all the basic functionalities of the runtime environment are available and the Card Manager is ready to receive, execute and respond to APDU commands. During this state, a new keyset have to be loaded before switching to INITIALIZED life state.

The card is assumed to have the following functionalities in the OP\_READY state:

- The runtime environment is ready for execution.
- An Initialization key is available within the Card Manager.
- Card Content Management operations are supported.
- Post-issuance personalisation of applets belonging to the Card Issuer can be carried out via the Card Manager.

The next states possible are INITIALIZED or TERMINATED. The transition is irreversible.

## INITIALIZED

This life state is an administrative card production state. Most of the personalisation of the Card Manager is performed when entering in this state.

The card is assumed to have the following functionalities in the INITIALIZED state:

- The runtime environment is ready for execution.
- A keyset is available within the Card Manager.
- Card Content Management operations are supported.
- Post-issuance personalisation of applets belonging to the Card Issuer can be carried out via the Card Manager.

The next states possible are SECURED or TERMINATED. The transition is irreversible.

## SECURED

The Card life cycle state SECURED is the normal operating life cycle state of the card after issuance. This state is the indicator for the Card Manager to enforce the Card Issuer's security policies related to post-issuance card behaviour such as applet loading and activation.

The card is assumed to have the following functionality in the state SECURED:

- The Card Manager contains all necessary key sets and security elements for full functionality.
- Card Issuer initiated card content changes can be carried out through the Card Manager.
- Card Content Management operations are supported.
- Post-issuance personalisation of applets belonging to the Card Issuer can be carried out via the Card Manager.

The next states possible are CM\_LOCKED or TERMINATED.

The transition in the TERMINATED state is irreversible.

## CM\_LOCKED

The state CM\_LOCKED is used to instruct the Card Manager to temporarily disable all applets on the card except for the Card Manager. This state is created to give the Card Issuer the ability to temporarily disable functionality of the card on detection of security threats (either internal or external to the card).

Setting the Card Manager to this state implies that the card will no longer work, except via the Card Manager which is controlled by the Card Issuer. No Card Content Management operation is possible.

The next states possible are SECURED or TERMINATED.



The transition in the TERMINATED state is irreversible.

### TERMINATED

The Card Manager is set to the life cycle state TERMINATED to permanently disable all card functionalities including the functionality of the Card Manager itself. This state is created as a mechanism for the Card Issuer to logically 'destroy' the card for such reasons as the detection of a severe security threat or upon expiration of the card.

Only GET DATA (CPLC) command is available. No Card Content Management operation is possible.

The Card Manager state TERMINATED is irreversible and signals the end of the card's life cycle.

### 1.9.2 Security Domain Life Cycle States

The Security Domain Life Cycle begins when a Security Domain is instantiated in the card. The Security Domain Life Cycle States defined by Global Platform are INSTALLED, SELECTABLE, PERSONALIZED and LOCKED. There are no proprietary Security Domain Life Cycle States.

Figure 5: Security Domain Life Cycle illustrates the state transition diagram for the Security Domain Life Cycle. This can typically be viewed as a sequential process with certain possibilities for reversing a state transition or skipping states.

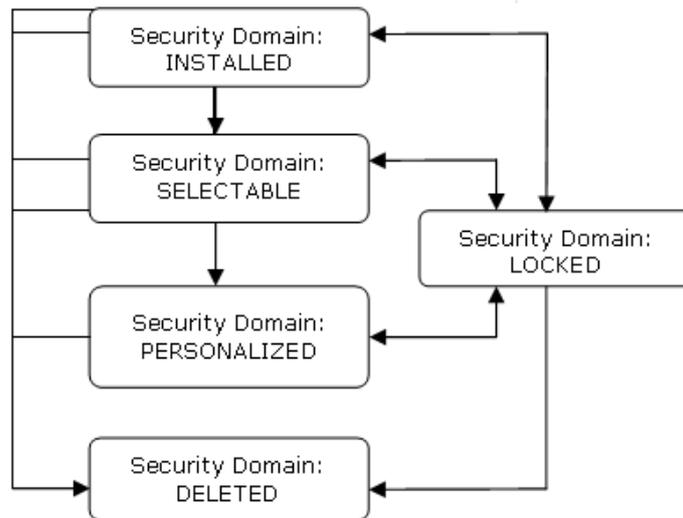


Figure 5: Security Domain Life Cycle

### INSTALLED

The state INSTALLED means that the Security Domain becomes an entry in the Global Platform Registry and this entry is accessible to off-card entities authenticated by the associated Security Domain. The Security Domain is not yet available for selection. It cannot be associated with Executable Load Files or Applications yet and therefore its Security Domain services are not available to Applications.

### SELECTABLE



The state **SELECTABLE** means that the Security Domain is able to receive commands (specifically personalisation commands) from off-card entities. As they still do not have keys, the Security Domains cannot be associated with Executable Load Files or Applications and therefore their services are not available to Applications when they are in this state. The state transition from **INSTALLED** to **SELECTABLE** is irreversible. The transition to **SELECTABLE** may be combined with the Security Domain installation process.

### PERSONALIZED

The definition of what is required for a Security Domain to transition to the state **PERSONALIZED** is Security Domain dependent but is intended to indicate that the Security Domain has all the necessary personalisation data and keys for full runtime functionality (i.e. usable in its intended environment). The transition from **SELECTABLE** to **PERSONALIZED** is irreversible.

In the state **PERSONALIZED**, the Security Domain may be associated with Applications and its services become available to these associated Applications.

### LOCKED

The **OPEN**, the Security Domain itself, the Security Domain's associated Security Domain (if any), an Application with the Global Lock privilege or a Security Domain with the Global Lock privilege uses the state **LOCKED** as a security management control to prevent the selection of the Security Domain.

If the **OPEN** detects a threat from within the card and determines that the threat is associated with a particular Security Domain, that Security Domain may be prevented from further selection by the **OPEN** setting the Security Domain's Life Cycle State to **LOCKED**.

Alternatively, the off-card entity may determine that a particular Security Domain on the card needs to be locked for a business or security reason and may initiate the state transition via the **OPEN**.

Locking a Security Domain prevents this Security Domain from being associated with new Executable Load Files or Applications. In this state DAP verification, extradition and access to that Security Domain's services shall fail. In summary, if a Security Domain is in the lifecycle state **LOCKED**, it shall reject all received commands.

Once the Life Cycle State is **LOCKED**, only the Security Domain's associated Security Domain (if any), an Application with Global Lock privilege or a Security Domain with Global Lock privilege is allowed to unlock the Security Domain. The **OPEN** shall ensure that the Security Domain's Life Cycle returns to its previous state.

### DELETED

At any point in the Security Domain Life Cycle, the **OPEN** may receive a request to delete a Security Domain.

The space previously used to store a physically deleted Security Domain is reclaimed and may be reused. The entry within the Global Platform Registry shall no longer be available, and the **OPEN** is not required to maintain a record of the deleted Security Domain's previous existence.

#### 1.9.3 Load File Life Cycle

The Load Files Life Cycle begins when a Load File is activated from immutable persistent memory (ROM) or loaded in mutable persistent memory (Flash).

Figure 6: Load File Life Cycle illustrates the state transition diagram for the Load File Life Cycle. This can typically be viewed as a sequential process.



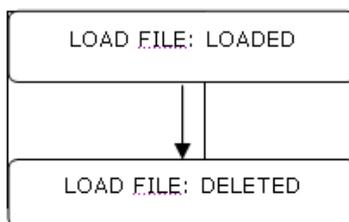


Figure 6: Load File Life Cycle

The Load Files activated (Phase 5) or loaded (Phase 5 and/or 6) must satisfy a process using the following tools:

- Compiler: software that generates machine-independent code (bytecode)
- Converter: software that preprocesses all of the Java programming language class files that make up a package, and converts the package to a standard file format for the binary compatibility of the Java Card platform (CAP file). The Converter also produces an export file.
- Loader: software that transfers the Load File.

During the Phase 7, the TOE must prevent the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification. The loading process requires adding the proof of the origin of the Load File (computed by off-card entity) and verifying it by a Security Domain with Mandated DAP privilege. The following tools are used:

- Compiler: software that generates machine-independent code (bytecode)
- Converter: software that preprocesses all of the Java programming language class files that make up a package, and converts the package to a standard file format for the binary compatibility of the Java Card platform (CAP file). The Converter also produces an export file.
- Verifier: software that performs static checks on the bytecodes of the methods of a CAP file and generates a signature <DAPBlock>.
- Loader: software that transfers the Load File (including the <DAPBlock>).

The bytecode, CAP file, DAP Block can be generated from any software.

### LOADED

The state LOADED is the initial life state just after it has been activated (from the Resident Application) or loaded (from the Resident Application or the Card Manager).

This state is independent of the visibility of the Load File (Get Status command of the Card Manager) and just depends on the presence in the Global Platform registry.

### DELETED.OS

The OPEN may receive a request to delete a Load File. For Load Files in Flash, the space previously used to store a physically deleted Load File is reclaimed and may be reused. For Load Files in ROM, a flag definitely prohibits further use. The entry within the Global Platform Registry is also removed, and the OPEN is not required to maintain a record of the deleted Load File's previous existence.

#### 1.9.4 Application Life Cycle

The Application Life Cycle begins when an applet is instantiated in the card. This instantiation may occur directly after loading transaction or alternatively from a Load File which is present on the card. The Application Life Cycle States defined by Global Platform are INSTALLED, SELECTABLE or LOCKED.



Figure 6: Load File Life Cycle, illustrates the state transition diagram for the Application Life Cycle. This can typically be viewed as a sequential process with certain possibilities for reversing a state transition or skipping states.

In addition to these states, the Application may define its own Application dependent states. Once the Application reaches the SELECTABLE state, it is responsible for managing the next steps of its own Life Cycle. It may use any Application specific states as long as these do not conflict with the states already defined by Global Platform. The OPEN may not perform these transitions without instruction from the Application and the Application is responsible for defining state transitions and ensuring that these transitioning rules are respected.

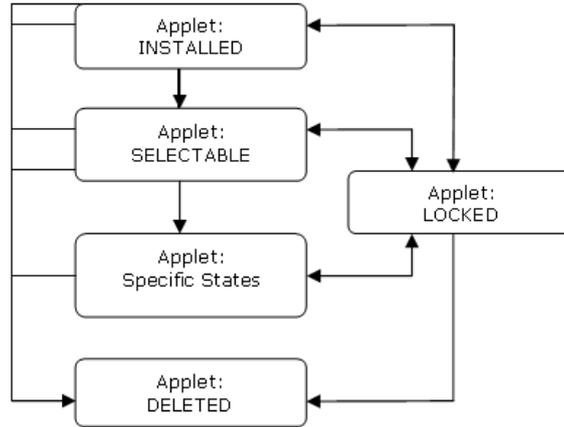


Figure 7: Application Life Cycle

### INSTALLED

The INSTALLED state means that the Application executable code has been properly linked and that any necessary memory allocation has taken place. The Application becomes an entry in the Global Platform Registry and this entry is accessible to authenticated off-card entities. The Application is not yet selectable. The installation process is not intended to incorporate personalisation of the Application, which may occur as a separate step.

### SELECTABLE

The SELECTABLE state implies that the applet is able to receive commands from off-card entities. The state transition from INSTALLED to SELECTABLE is irreversible. The Application shall be properly installed and functional before it may be set to the state. The transition to SELECTABLE may be combined with the Application installation process. The behaviour of the Application in the SELECTABLE state is beyond the scope of this Specification.

### LOCKED

The OPEN or the off-card entity authenticated by the Issuer Security Domain uses the state LOCKED as a security management control to prevent the selection, and therefore the execution, of the Application. If the OPEN detects a threat from within the card and determines that the threat is associated to a particular Application, this Application may be prevented from further selection by the OPEN setting its state to LOCKED. Alternatively, the off-card entity authenticated by the Issuer Security Domain may determine that a particular Application on the card needs to be locked for a business or security reason and may initiate the Application Life Cycle transition via the OPEN. Once the state is LOCKED, only the Issuer Security Domain is allowed to unlock the Application. The OPEN shall ensure that the Application Life Cycle returns to its previous state.



**DELETED**

At any point in the Application Life Cycle, the OPEN may receive a request to delete an Application. The space previously used to store a physically deleted Application is reclaimed and may be reused. The entry within the Global Platform Registry is also removed, and the OPEN is not required to maintain a record of the deleted Application's previous existence.

**Application Specific Life Cycle States**

These states are Application specific. The behaviour of the Applet during these states is determined by the Applet itself and is beyond the scope of this document. The OPEN does not enforce any control on Application specific Life Cycle State transitions.



### 1.10 Major Security feature of the TOE

The main goal of the TOE is to provide a sound and secure execution environment to critical assets that need to be protected against unauthorized disclosure and/or modification.

The TOE with its security function has to protect itself and protect applets from bypassing, abuse or tampering of its services that could compromise the security of all sensitive data. Even if the applets are not in the scope of this evaluation.

#### **Atomic Transactions**

The TOE shall provide a transaction mechanism. It shall execute a sequence of modifications and allocations on the persistent memory so that either all of them are completed, or the TOE behaves as if none of them had been attempted.

The transaction mechanism shall permit to update internal TSF data as well as to perform different functions of the TOE, like installing a new package on the card.

This mechanism shall be available for applet instances

The TOE shall perform the necessary actions to roll back to a safe state upon interruption.

#### **Card Content Management**

The TOE shall control the loading, installation, and deletion of packages and applet instances.

To remove the code of a package from the card, or to definitely deactivate an applet instance, so that it becomes no longer selectable; it shall perform physical removal of those packages and applet data stored in memories (except applet in ROM memory that shall only be logically removed).

#### **Card Management Environment**

This function shall initialize and manage the internal data structure of the Card Manager. During the initialization phase of the card, it creates the Installer and the Applet Deletion Manager and initializes their internal data structures. The internal data structure of the Card Manager includes the Package and Applet Registries, which respectively contains the currently loaded packages and the currently installed applet instances, together with their associated AIDs.

This function shall also be in charge of dispatching the APDU commands to the applet instances installed on the card and keeping trace of the currently active ones.

It therefore handles sensitive TSF data of other security functions, like the Firewall.

#### **Cardholder Verification**

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

#### **Clearing of sensitive information**

The TOE shall ensure that no residual information is available from memories, and shall protect sensitive information that is no longer used. The Platform has to securely clear and destroy this information. It concerns PINs, keys, sensitive data and buffer APDU.



This function is also available to applet.

**DAP Verification**

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

**Data coherency**

As coherency of data should be maintained, and as power is provided by the CAD and might be stopped at all moment (by tearing or attacks), a transaction mechanism need to be implemented.

When updating data, before writing the new ones, the old ones are saved in a specific memory area. If a failure appears, at the next start-up, if old data are valid in the transaction area, the system restores them for staying in a coherent state.

**Data integrity**

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

**Encryption and Decryption**

The TOE provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.

Ciphering operations are implemented to resist environmental stress and glitches and include measures for preventing information leakage through covert channels.

**Entity authentication/secure Channel**

Off-card entity authentication is achieved through the process of initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity.

If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated).

The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s).

**Exception**

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

**Firewall**

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

**GP\_Dispatcher**

While a Security Domain or Card Manager is selected, the TOE shall test for every



command if Security Domain Owner authentication is required. If a secure channel is opened, the TOE tests according to the Security Domain state and the Card state for every command if secure messaging is required.

**Hardware operating**

The TOE shall boot after the IC has successfully powered-up. The TOE boot operations shall ensure the correct initialization of the TOE functionalities and the integrity of the code and data.

The TOE shall monitor IC detectors (e.g. out-of-range voltage, temperature, frequency, active shield, memory aging) and shall provide automatic answers to potential security violations through interruption routines that leave the device in a secure state.

**Key Access**

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

**Key Agreement**

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

**Key destruction**

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

**Key Distribution**

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

**Key Generation**

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

**Key management**

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

**Manufacturer Authentication**

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

**Memory failure**

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

**Message Digest**

The TOE shall provide the applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. This value can be used as a short representative of the information contained in the whole byte array.

For Hashing algorithms that do not pad the messages, the TSF checks that the information is block aligned before computing its hash value.

**Prepersonalisation**

This function shall permit to pre-initialize the internal data structures, to load the configuration of the card and to load patch code if needed and locks.

The TOE shall allow loading of TOE sensitive data: configuration data. Configuration data can contain patches. The TOE shall check the integrity of the incoming data. Unless stated otherwise, the origin of the incoming data shall be ensured by organisational means. The TOE shall ensure that TOE code and patches installed after delivery cannot be bypassed. The loading functionality of patches shall be disabled before entering the final usage phase. The TOE identification shall take into account the patches installed after delivery.

#### **Random Number**

This TOE functionality provides the card manager, the resident application and the applets a mechanism for generating challenges and key values.

The Number Generator is a combination of hardware and software RNG. The RNG is compliant with **[R30]**.

#### **Resident Application dispatcher**

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

#### **Runtime Verifier**

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

#### **Security functions of the IC**

This TOE functionality ensures the correct execution of the IC functionalities.

#### **Signature**

This TSF shall provide the applet instances with a mechanism for generating an electronic signature of the contents of a byte array and verifying an electronic signature contained in a byte array.

An electronic signature is made of a hash value of the information to be signed, encrypted with a secret key. The verification of the electronic signature includes decrypting the hash value and checking that it actually corresponds to the block of signed bytes. Signature operations shall be implemented to resist environmental stress and glitches and include measures for preventing information leakage through covert channels.

#### **Unobservability**

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

#### **CRC 16**

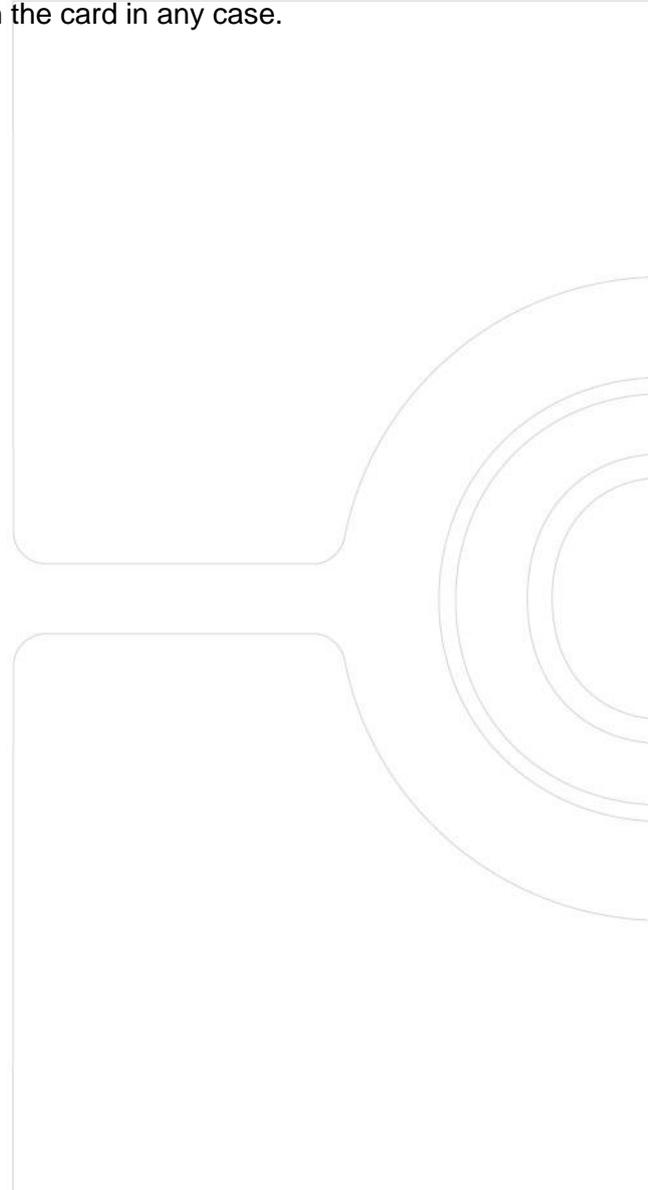
The TOE provides this security function to guarantee the integrity of the sensitive objects (such as keys or PINs) store in the card.

#### **1.11 NON-TOE HW/SW/FW AVAILABLE TO THE TOE**

The only non-TOE component required on the product is the bytecode verifier. The bytecode verifier is a program that performs static checks on the bytecodes of the methods of a CAP file.



Bytecode verification is a **key** component of security: applet isolation, for instance, depends on the file satisfying the properties a verifier checks to hold. A method of a CAP file that has been verified shall not contain, for instance, an instruction that allows forging a memory address or an instruction that makes improper use of a return address as if it were an object reference. In other words, bytecodes are verified to hold up to the intended use to which they are defined. This TOE considers static bytecode verification; it has to be performed on the host at off-card verification and prior to the loading of the file on the card in any case.



## 2 CONFORMANCE CLAIM

---

### 2.1 Common Criteria Conformance Claim

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

- [1] "Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", April 2017, Version 3.1 revision 5.
- [2] "Common Criteria for information Technology Security Evaluation, Part 2: Security Functional component", April 2017, Version 3.1 revision 5.
- [3] "Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance components", April 2017, Version 3.1 revision 5.

The Conformance to the Common Criteria is claims as follows:

Common Criteria	Conformance rationale
Part 1	Conformance
Part 2	Conformance to the extended part. FCS_RNG.1: "Random number generation"
Part 3	Compliant to EAL5 +, augmented with <ul style="list-style-type: none"> <li>- ALC_DVS.2: "Sufficiency of security measures" (highest component)</li> <li>- AVA_VAN.5: "Advanced methodical vulnerability analysis" (highest component)</li> </ul>

Table 5: CC rationale

### 2.2 Protection Profile Claim

This security target claims a demonstrable conformance to: **[R5]**

This security target is a composite security target, including the IC security target.

However, the security problem definition, the objectives, and the SFR of the IC are not described in this document.

As the IC is included in the TOE, OE.CARD-MANAGEMENT, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC are changed into the following Objectives on the TOE: O.CARD-MANAGEMENT, O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC.

As the SCP is included in the TOE, OE.NATIVE, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC are changed into the following Objectives on the TOE: O.NATIVE, O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC.



There are extra TOE objectives to provide additional services to applications. But such extension has no impact on PP coverage.

There are extra Threats, OSP, Assumptions, TOE objectives and SFR without conflict with [R5].

The RMI is not supported by the TOE, all related Threats, OSP, Assumptions, objectives and SFRs are then removed.

Finally as no other modification was done, we can conclude that the conformance is demonstrated

The product is in conformance with the minimum assurance level EAL5+ augmented with ALC\_DVS.2 and AVA\_VAN.5 described in paragraph 3.2 of the Protection Profile by claiming an evaluation level EAL5+ augmented with ALC\_DVS.2 and AVA\_VAN.5.

### 2.3 Conformance rationale

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

#### 2.3.1 TOE SAR conformance

The PP requires EAL4+ (AVA.VAN.5, ALC.DVS.2) level, the ST claims EAL5+ level; augmented with AVA\_VAN.5, ALC\_DVS.2

#### 2.3.2 TOE Type conformance

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

#### 2.3.3 SPD Statement Consistency

##### 2.3.3.1 Assets

All assets from the protection profile are included in the security target.

The following assets have been added:

Assets	Rationale
D.CONFIG	This asset defines the elements of configuration during the prepersonalisation phase. It includes Locks, keys and patch if any.
D.SENSITIVE_DATA	This asset describes the set of sensitive data to be protected
D.ARRAY	This asset describes the applets sensitive data
D.JCS_KEYS	This asset describes two cryptographic keys used during the loading of a file in the card



### 2.3.3.2 Threats

All threats from the protection profile are included in the security target. Three additional threats have been added in the security target:

**T.CONFIGURATION** This threat is an additional threat to cover unauthorized modifications and read access of the configuration of the TOE at prepersonalisation phase. It is an addition to the threats defined in the PP [5].

It does not contradict existing SPD, as its scope is limited to the prepersonalisation phase.

#### **T.CONF\_DATA\_APPLET**

The attacker executes an application to disclose sensitive data manipulated by an applet during in the comparison between two byte arrays. This threat doesn't contradict existing threats as it concerns additional data belonging to applications to be loaded on the TOE. It concerns service available to the applications that require secure comparison of byte arrays.

Directly threatened asset(s): D.APP\_C\_DATA

**T.PATCH\_LOADING** This threat is directly linked to patch loading. It is added to cover unauthorized loading and modification and disclosure of the patch at loading phase and after the loading. This threat doesn't contradict any existing threat as it concerns the loading of patches at prepersonalisation and/ or personalization phases.

### 2.3.3.3 OSPs

All the OSP from the protection profile is included in the security target, no additional OSP have been added.

### 2.3.3.4 Assumptions

All the assumptions from the protection profile have been added in the security target, except A.DELETION.

A.DELETION has been removed from the security target because the deletion of applets is in the scope of the evaluation, as O.CARD\_MANAGEMENT is an objective in this security target.

### 2.3.3.5 Objectives

### 2.3.3.6 Security Objectives for the TOE

All the security objectives for the TOE from the protection profile are included in the security target.

The following security objectives have been added:

Security objectives for the	Rationale
	}
	}
	}
	}

TOE	
O.SCP.SUPPORT	This security objective comes from a security objective for the operational environment
O.SCP.IC	This security objective comes from a security objective for the operational environment
O.SCP.RECOVERY	This security objective comes from a security objective for the operational environment
O.RESIDENT_APPLICATION	This security objective deals with the security of the resident application
O.CARD_MANAGEMENT	This security objective comes from a security objective for the operational environment
O.SECURE_COMPARE	This security objective is linked to D.ARRAY. This Objective concerns the new service available to the applications that need secure comparison.
O.PATCH_LOADING	This security objective is related to patch loading. This objective ensure the secure loading of patch in prepersonalisation phase.

#### 2.3.3.7 Security Objectives for the Operational Environment

All the security objectives for the operational environment are included in the security target.

Some security objectives for the operational environment has been transformed in security objectives for the TOE, the rationale is presented in the previous chapter.

#### 2.3.3.8 SFR and SARs Statements consistency

#### 2.3.3.9 SFRs Consistency

All the SFR from the protection profile have been added in the security target.

The following SFR have been added in the security target:

#### **Additional SFR for the Card Manager**

SFR	Rationale
FPT_TST.1	Initial startup test in case of future requirement
FCO_NRO.2/CM_DAP	Refinement of the requirements in terms of non-repudiation of the origin to the Card Manager during the DAP process
FIA_AFL.1/CM	Concerns applets composition evaluation



FIA_UAU.1/CM	Concerns smartcard product and composition
FIA_UAU.4/CardIssuer	Prevents from Card Issuer authentication reuse
FIA_UAU.7/CardIssuer	Defines the authentication process
FPR_UNO.1/Key_CM	Prevents from observation of import key operation
FPT_TDC.1/CM	Technical requirement for communication with another trusted IT product
FMT_SMR.2/CM	Defines several roles
FCS_COP.1/CM	Defines the Cryptographic algorithm available to the CM for the Card Issuer authentication

**Additional SFR for the resident application**

SFR	Rationale
FDP_ACC.2/PP	Access control policy during prepersonalisation
FDP_ACF.1/PP	Access control functions during prepersonalisation
FDP_UCT.1/PP	Precision of the prepersonalisation access control regarding inter-TSF user data confidentiality transfer protection
FDP_ITC.1/PP	Precision of the import of user data during prepersonalisation
FIA_AFL.1/PP	Precision of the authentication failures during prepersonalisation
FIA_UAU.1/PP	Precision of the user accessible functions before user authentication during prepersonalisation
FIA_UID.1/PP	Precision of the user accessible functions before user identification during prepersonalisation
FMT_MSA.1/PP	Precision of the management of the security attributes during prepersonalisation
FMT_SMF.1/PP	Precision of the specification of the management functions during prepersonalisation
FIA_UAU.4/CardManu	Prevents from Card Manufacturer authentication reuse during prepersonalisation
FIA_UAU.7/CardManu	Defines the authentication process of the Card Manufacturer during prepersonalisation
FMT_MOF.1/PP	Management of functions of the TSF during prepersonalisation, especially for the resident application
FMT_SMR.2/PP	Restrictions on security roles during prepersonalisation
FMT_MSA.3/PP	Precision of the security attribute initialization during prepersonalisation
FCS_COP.1/PP	Cryptographic operation available during prepersonalisation



FCS_CKM.4/PP	Cryptographic key destruction during prepersonalisation
FDP_UIT.1/PP	Ensures the integrity of the patch loaded
FCS_CKM.1/PP	Provides the MSK diversification
FAU_STG.2	Provides the patch identification evidence

**Additional SFR for the SmartCard Platform**

SFR	Rationale
FPT_PHP.3/SCP	Additional security features are added in the product, using security features of the IC
FPT_RCV.4/SCP	Additional SFR regarding operational objective on the operational environment transformed in security objectives
FRU_FLT.1/SCP	Additional SFR regarding operational objective on the operational environment transformed in security objectives
FCS_RNG.1/SCP	Additional SFR for RNG management
FPR_UNO.1/USE_KEY	Additional SFR for the unobservability of keys
FIA_AFL.1/PIN	Precision of the authentication failures for the PIN
FMT_MTD.2/GP_PIN	Additional SFR for the management of limits on TSF data regarding the GP PIN
FDP_SDI.2/Applet	Additional SFR for the unobservability of array comparison by applets, regarding D.ARRAY
FMT_MTD.1/PIN	Additional SFR for the management of TSF data regarding the PIN
FIA_AFL.1/GP_PIN	Precision of the authentication failures for the GP PIN

**Additional SFR for the stack control**

SFR	Rationale
FDP_ACC.2/RV_Stack	Access control policy for stack control
FDP_ACF.1/RV_Stack	Access control functions for stack control
FMT_MSA.1/RV_Stack	Precision of the Stack access control SFP
FMT_MSA.2/RV_Stack	Precision of the secure security attributes for stack control
FMT_MSA.3/RV_Stack	Precision of the static attribute initialization for stack control
FMT_SMF.1/RV_Stack	Specification of management functions for stack control

**Additional SFR for the heap access**

SFR	Rationale
FDP_ACC.2/RV_Heap	Access control policy for heap access



FDP_ACF.1/RV_Heap	Access control functions for heap access
FMT_MSA.1/RV_Heap	Precision of the heap access control SFP
FMT_MSA.2/RV_Heap	Precision of the secure security attributes for heap control
FMT_MSA.3/RV_Heap	Precision of the static attribute initialization for heap control
FMT_SMF.1/RV_Heap	Specification of management functions for heap control

**Additional SFR for the transient control**

<b>SFR</b>	<b>Rationale</b>
FDP_ACC.2/RV_Transient	Access control policy for transient control
FDP_ACF.1/RV_Transient	Access control functions for transient control
FMT_MSA.1/RV_Transient	Precision of the transient access control SFP
FMT_MSA.2/RV_Transient	Precision of the secure security attributes for transient control
FMT_MSA.3/RV_Transient	Precision of the static attribute initialization for transient control
FMT_SMF.1/RV_Transient	Specification of management functions for transient control

## 2.3.4 Rational for SAR

The PP require EAL4+



## 3 SECURITY ASPECTS

---

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

For instance, we will define hereafter the following aspect:

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

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

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

### 3.1 Confidentiality

#### **#.CONFID-APPLI-DATA:**

Application data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain read access to other application’s data.

#### **#.CONFID-JCS-CODE:**

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

#### **#.CONFID-JCS-DATA:**

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

### 3.2 Integrity

#### **#.INTEG-APPLI-CODE:**

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



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

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

**#.INTEG-JCS-CODE:**

Java Card System code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to executable code.

**#.INTEG-JCS-DATA:**

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

**3.3 Unauthorized executions****#.EXE-APPLI-CODE:**

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

**#.EXE-JCS-CODE:**

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

**#.FIREWALL:**

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

**#.NATIVE:**

Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside those TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

**3.4 Bytecode verification****#.VERIFICATION**

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



### 3.4.1 CAP file verification

Bytecode verification includes checking at least the following properties: (3) bytecode instructions represent a legal set of instructions used on the Java Card platform; (4) adequacy of bytecode operands to bytecode semantics; (5) absence of operand stack overflow/underflow; (6) control flow confinement to the current method (that is, no control jumps to outside the method); (7) absence of illegal data conversion and reference forging; (8) enforcement of the private/public access modifiers for class and class members; (9) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (10) enforcement of rules for binary compatibility (full details are given in [R8], [R41], [R42]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [R8] on the bytecodes and the correctness of the CAP files’ format.

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

Another important aspect to be considered about bytecode verification and application downloading is, first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [R8] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

### 3.4.2 Integrity and authentication

Verification off-card is useless if the application package is modified afterwards. The usage of cryptographic certifications coupled with the verifier in a secure module is a simple means to prevent any attempt of modification between package verification and package installation.

Once a verification authority has verified the package, it signs it and sends it to the card. Prior to the installation of the package, the card verifies the signature of the package, which authenticates the fact that it has been successfully verified. In addition to this, a secured communication channel is used to communicate into 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.

### 3.4.3 Linking and authentication

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

## 3.5 Card management

### #.CARD\_MANAGEMENT:

|            }            }            }            }

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

**#.INSTALL:**

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

**#.SID:**

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

**#OBJ-DELETION:**

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

**#DELETION:**

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

The deletion procedure and its characteristics (whether deletion is either physical or logical, what happens if the deleted application was the default applet, the order to be observed on the deletion steps) are implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage).

Deletion of a single applet instance and deletion of a whole package are functionally different operations and may obey different security rules. For instance, specific packages can be declared to be undeletable (for instance, the Java Card API packages), or the dependency between installed packages may forbid the deletion (like a package using super classes or super interfaces declared in another package).

### 3.6 Services

**#.ALARM:**

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



**#.OPERATE:**

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

**#.RESOURCES:**

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

**#.CIPHER:**

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

**#.KEY-MNGT:**

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

**#.PIN-MNGT:**

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

**#.SCP:**

The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection). (6) It safely transmits low-level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well-defined set of policies and standards (for instance, those specified in [R24]), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

**#.TRANSACTION:**

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



## 4 SECURITY PROBLEM DEFINITION

---

### 4.1 Assets

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

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

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

#### 4.1.1 User data

##### **D.APP\_CODE**

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

To be protected from unauthorized modification.

##### **D.APP\_C\_DATA**

Confidential sensitive data of the applications, like the data contained in an object, a static field of a package, a local variable of the currently executed method, or a position of the operand stack.

To be protected from unauthorized disclosure.

##### **D.APP\_I\_DATA**

Integrity sensitive data of the applications, like the data contained in an object, a static field of a package, a local variable of the currently executed method, or a position of the operand stack.

To be protected from unauthorized modification.

##### **D.APP\_KEYS**

Cryptographic keys owned by the applets.

To be protected from unauthorized disclosure and modification.

##### **D.PIN**

Any end-user's PIN.

To be protected from unauthorized disclosure and modification.



#### 4.1.2 TSF data

##### **D.API\_DATA**

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

To be protected from unauthorized disclosure and modification.

##### **D.CRYPTO**

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

To be protected from unauthorized disclosure and modification.

##### **D.JCS\_CODE**

The code of the Java Card System.

To be protected from unauthorized disclosure and modification.

##### **D.JCS\_DATA**

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

To be protected from unauthorized disclosure or modification.

##### **D.SEC\_DATA**

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

To be protected from unauthorized disclosure and modification.

#### 4.1.3 Additional assets

##### **D.CONFIG**

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

##### **D.SENSITIVE\_DATA**

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

- o D.NB\_AUTHENTIC: Number of authentications. This number is specified in the SFR
- o D.NB\_REMAINTRYOWN: Number of remaining tries for owner PIN. This number is specified in the SFR
- o D.NB\_REMAINTRYGLB: Number of remaining tries for a global PIN. This number is specified in the SFR
- o ASG.CARDREG: Card registry (AS.APID: Applet Identifier (AID), AS.CMID: Card Manager ID (AID))



- o ASG.APPRIV: Applet privileges group (Card Manager lock privilege, Card terminate privilege, Default selected privilege, PIN change privilege, Security Domain privilege, Security Domain with DAP verification privilege, Security Domain with Mandated DAP verification privilege)
- o AS.AUTH\_MSK\_STATUS: Authentication MSK Status, this Security Attribute verifies if the authentication with the MSK key is performed successfully or not.
- o AS.CMLIFECYC: this Security Attribute represents the Card life cycle state. It can be either: Prepersonalisation, Personalisation and use phases of the card.

**D.ARRAY**

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

**D.JCS\_KEYS**

AS.KEYSET\_VERSION and AS.KEYSET\_Value Cryptographic keys used when loading a file into the card. To be protected from unauthorized disclosure and modification.

**4.2 Users / Subjects**

## 4.2.1 Additional Users / Subjects

**S.RESIDENT\_APPLICATION**

The resident application

**R.personaliser**

Card Issuer or card Manufacturer

**R.Prepersonaliser**

Card manufacturer

**U.Card\_Issuer**

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

**U.Card\_Manufacturer**

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

## 4.2.2 Miscellaneous

**S.ADEL**

The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([R7], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §7.1.3.1.



**S.APPLET**

Any applet instance.

**S.BCV**

The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy

**S.CAD**

The CAD represents the actor that requests, by issuing commands to the card. It also plays the role of the off-card entity that communicates with the S.INSTALLER.

**S.INSTALLER**

The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.

**S.JCRE**

The runtime environment under which Java programs in a smart card are executed.

**S.JCVM**

The bytecode interpreter that enforces the firewall at runtime.

**S.LOCAL**

Operands stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.

**S.MEMBER**

Any object's field, static field or array position.

**S.PACKAGE**

A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.

**S.TOE**

Source code.

**4.3 Threats**

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

**4.3.1 CONFIDENTIALITY****T.CONFID-APPLI-DATA**

The attacker executes an application to disclose data belonging to another application. See #.CONFID-APPLI-DATA for details.

Directly threatened asset(s): D.APP\_C\_DATA, D.PIN and D.APP\_KEYS.

#### **T.CONFID-JCS-CODE**

The attacker executes an application to disclose the Java Card System code. See #.CONFID-JCS-CODE for details.

Directly threatened asset(s): D.JCS\_CODE.

#### **T.CONFID-JCS-DATA**

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

Directly threatened asset(s): D.API\_DATA, D.SEC\_DATA, D.JCS\_DATA, D.CRYPTO and D.JCS\_KEYS.

### 4.3.2 INTEGRITY

#### **T.INTEG-APPLI-CODE**

The attacker executes an application to alter (part of) its own code or another application's code. See #.INTEG-APPLI-CODE for details.

Directly threatened asset(s): D.APP\_CODE.

#### **T.INTEG-APPLI-CODE.LOAD**

The attacker modifies (part of) its own or another application code when an application package is transmitted to the card for installation. See #.INTEG-APPLI-CODE for details.

Directly threatened asset(s): D.APP\_CODE.

#### **T.INTEG-APPLI-DATA**

The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA for details.

Directly threatened asset(s): D.APP\_I\_DATA, D.PIN and D.APP\_KEYS.

#### **T.INTEG-APPLI-DATA.LOAD**

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

Directly threatened asset(s): D.APP\_I\_DATA and D\_APP\_KEY.

#### **T.INTEG-JCS-CODE**

The attacker executes an application to alter (part of) the Java Card System code. See #.INTEG-JCS-CODE for details.

Directly threatened asset(s): D.JCS\_CODE.

#### **T.INTEG-JCS-DATA**

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



Directly threatened asset(s): D.API\_DATA, D.SEC\_DATA, D.JCS\_DATA, D.JCS\_KEYS and D.CRYPTO.

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

#### 4.3.3 IDENTITY USURPATION

##### **T.SID.1**

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

Directly threatened asset(s): D.SEC\_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN, D.JCS\_KEYS and D.APP\_KEYS and D.SENSITIVE\_DATA.

##### **T.SID.2**

The attacker modifies the TOE's attribution of a privileged role (e.g. default applet and currently selected applet), which allows illegal impersonation of this role. See #.SID for further details.

Directly threatened asset(s): D.SEC\_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged) and D.SENSITIVE\_DATA.

#### 4.3.4 UNAUTHORIZED EXECUTION

##### **T.EXE-CODE.1**

An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.

Directly threatened asset(s): D.APP\_CODE.

##### **T.EXE-CODE.2**

An applet performs an execution of a method fragment or arbitrary data. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.

Directly threatened asset(s): D.APP\_CODE.

##### **T.NATIVE**

An applet executes a native method to bypass a TOE Security Function such as the firewall. See #.NATIVE for details.

Directly threatened asset(s): D.JCS\_DATA.

#### 4.3.5 DENIAL OF SERVICE

##### **T.RESOURCES**

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

Directly threatened asset(s): D.JCS\_DATA.



## 4.3.6 CARD MANAGEMENT

**T.DELETION**

The attacker deletes an applet or a package already in use on the card, or uses the deletion functions to pave the way for further attacks (putting the TOE in an insecure state). See #.DELETION for details).

Directly threatened asset(s): D.SEC\_DATA, D.APP\_CODE and D.SENSITIVE\_DATA.

**T.INSTALL**

The attacker fraudulently installs post-issuance of an applet on the card. This concerns either the installation of an unverified applet or an attempt to induce a malfunction in the TOE through the installation process. See #.INSTALL for details.

Directly threatened asset(s): D.SEC\_DATA (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application) and D.SENSITIVE\_DATA.

## 4.3.7 SERVICES

**T.OBJ-DELETION**

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

Directly threatened asset(s): D.APP\_C\_DATA, D.APP\_I\_DATA and D.APP\_KEYS.

## 4.3.8 MISCELLANEOUS

**T.PHYSICAL**

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

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

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

## 4.3.9 Additional threats

**T.CONFIGURATION**

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



Directly threatened asset(s): D.CONFIG

#### **T.CONF\_DATA\_APPLET**

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

Directly threatened asset(s): D.ARRAY

#### **T.PATCH\_LOADING**

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

- altering a patch (during loading or once loaded),
- exploiting the patch loading mechanism to load unauthenticated code on the TOE

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

Directly threatened asset(s): D.CONFIG

### **4.4 Organisational Security Policies**

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

### **4.5 Assumptions**

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

Due to the Protection Profile and Security Target definition, T.DELETION replaces A.DELETION as O.CARD\_MANAGEMENT replaces OE.CARD\_MANAGEMENT.

#### **A.APPLET**

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([R8], §3.3) outside the API.

#### **A.VERIFICATION**

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



## 5 SECURITY OBJECTIVES

---

### 5.1 Security Objectives for the TOE

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

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

#### 5.1.1 IDENTIFICATION

##### **O.SID**

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

#### 5.1.2 EXECUTION

##### **O.FIREWALL**

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

##### **O.GLOBAL\_ARRAYS\_CONFID**

The TOE shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection.

The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

##### **O.GLOBAL\_ARRAYS\_INTEG**

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

##### **O.NATIVE**

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

##### **O.OPERATE**

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

##### **O.REALLOCATION**

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

##### **O.RESOURCES**

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



for details.

### 5.1.3 SERVICES

#### **O.ALARM**

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

#### **O.CIPHER**

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

#### **O.KEY-MNGT**

The TOE shall provide a means to securely manage cryptographic keys (D.APP\_KEYS, D.JCS\_KEYS and D.CRYPTO). This concerns the correct generation, distribution, access and destruction of cryptographic keys. See #.KEY-MNGT.

#### **O.PIN-MNGT**

The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT for details. This concerns at least the correct authentication of the cardholder and the PIN before having access to protected operations; the observability of the comparison between presented PIN and stored PIN.

*Application Note:*

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

#### **O.TRANSACTION**

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

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

### 5.1.4 OBJECT DELETION

#### **O.OBJ-DELETION**

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

### 5.1.5 APPLLET MANAGEMENT

#### **O.DELETION**

The TOE shall ensure that both applet and package deletion perform as expected. See



#.DELETION for details.

**O.LOAD**

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

*Application Note:*

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

**O.INSTALL**

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

#### 5.1.6 Additional security objectives for the TOE

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

- OE.SCP.IC
- OE.SCP.SUPPORT
- OE.SCP.RECOVERY
- OE.CARD\_MANAGEMENT

**O.SCP.SUPPORT**

The TOE shall support the following functionalities:

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

**O.SCP.IC**

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

**O.SCP.RECOVERY**

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

**O.RESIDENT\_APPLICATION**

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

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

**O.CARD\_MANAGEMENT**

The card manager shall control the access to card management functions such as the installation, update or deletion of applets. It shall also implement the card issuer's policy on the card.

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

**O.SECURE\_COMPARE**

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

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

**O.PATCH\_LOADING**

The TOE shall provide a secure patch code loading mechanism. The data to be loaded are encrypted. Once these data loaded, the integrity (SHA256) of the modified code is update



and compared to the provided one in the patch package.

## 5.2 Security objectives for the Operational Environment

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

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

- OE.SCP.SUPPORT
- OE.SCP.IC
- OE.SCP.RECOVERY
- OE.CARD\_MANAGEMENT

### OE.APPLET

No applet loaded post-issuance shall contain native methods.

### OE.VERIFICATION

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

#### *Application Note:*

Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

### OE.CODE-EVIDENCE

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

#### *Application Note:*

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



### 5.3 Security Objectives Rationale

#### 5.3.1 Threats

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

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

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

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

As applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys, PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.

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

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

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

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

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



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

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

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

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

**T.INTEG-APPLI-CODE** This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can run to modify a piece of code.

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

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

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that integrity and authenticity evidences exist for the application code loaded into the platform.

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

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



integrity and authenticity. The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter. Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned. Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the objective O.GLOBAL\_ARRAYS\_INTEG. Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

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

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

**T.INTEG-JCS-DATA** This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken. The objectives O.CARD\_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively. The 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.SID.1** As impersonation is usually the result of successfully disclosing and modifying some assets, this threat is mainly countered by the objectives concerning the isolation of application data (like PINs), ensured by the (O.FIREWALL). Uniqueness of subject-identity (O.SID) also participates to face this threat. It should be noticed that the AIDs, which are used for applet identification, are TSF data.

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

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

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

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

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

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

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

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

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

**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.

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

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

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

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

**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.

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

#### 5.3.1.1 Additional threats

**T.CONFIGURATION** This threat is covered by O.RESIDENT\_APPLICATION.

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

**T.CONF\_DATA\_APPLET** This threat is covered by the O.SECURE\_COMPARE security objective.

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

**T.PATCH\_LOADING** This threat is covered by O.PATCH\_LOADING security objective.

If an attacker tries to avoid the loading of a patch or alter a patch (during loading or once loaded), O.PATCH\_LOADING ensures trustable identification and authentication (static signature) data of the loaded patch are returned by the TOE. This information enables to check the presence of the genuine patch. Moreover, O.PATCH\_LOADING, ensures authentication of the entity loading the patch before the patch is loaded in the TOE. This



objective ensures patch loading can only be performed during a limited moment in the TOE life cycle (before phase 6) and once the TOE has reached phase 6, this feature is not available anymore.

### 5.3.2 Organisational 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.

### 5.3.3 Assumptions

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

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

### 5.3.4 SPD and Security Objectives



Threats	Security Objectives	Rationale
<a href="#">T.CONFID-APPLI-DATA</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.SID</a> , <a href="#">O.OPERATE</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.ALARM</a> , <a href="#">O.TRANSACTION</a> , <a href="#">O.CIPHER</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.SCP.RECOVERY</a> , <a href="#">O.SCP.SUPPORT</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.CONFID-JCS-CODE</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.NATIVE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.CONFID-JCS-DATA</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.SID</a> , <a href="#">O.OPERATE</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.ALARM</a> , <a href="#">O.SCP.RECOVERY</a> , <a href="#">O.SCP.SUPPORT</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INTEG-APPLI-CODE</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.NATIVE</a> , <a href="#">OE.CODE-EVIDENCE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INTEG-APPLI-CODE.LOAD</a>	<a href="#">O.LOAD</a> , <a href="#">OE.CODE-EVIDENCE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INTEG-APPLI-DATA</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.SID</a> , <a href="#">O.OPERATE</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.GLOBAL_ARRAYS_INTEG</a> , <a href="#">O.ALARM</a> , <a href="#">O.TRANSACTION</a> , <a href="#">O.CIPHER</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.SCP.RECOVERY</a> , <a href="#">O.SCP.SUPPORT</a> , <a href="#">OE.CODE-EVIDENCE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INTEG-APPLI-DATA.LOAD</a>	<a href="#">O.LOAD</a> , <a href="#">OE.CODE-EVIDENCE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INTEG-JCS-CODE</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.NATIVE</a> , <a href="#">OE.CODE-EVIDENCE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INTEG-JCS-DATA</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.SID</a> , <a href="#">O.OPERATE</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.ALARM</a> , <a href="#">O.SCP.RECOVERY</a> , <a href="#">O.SCP.SUPPORT</a> , <a href="#">OE.CODE-EVIDENCE</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.SID.1</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.GLOBAL_ARRAYS_INTEG</a> , <a href="#">O.INSTALL</a> , <a href="#">O.SID</a> , <a href="#">O.CARD_MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.SID.2</a>	<a href="#">O.SID</a> , <a href="#">O.OPERATE</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.INSTALL</a> , <a href="#">O.SCP.RECOVERY</a> , <a href="#">O.SCP.SUPPORT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.EXE-CODE.1</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">O.FIREWALL</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.EXE-CODE.2</a>	<a href="#">OE.VERIFICATION</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.NATIVE</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">OE.APPLET</a> , <a href="#">O.NATIVE</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.RESOURCES</a>	<a href="#">O.INSTALL</a> , <a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.SCP.RECOVERY</a> , <a href="#">O.SCP.SUPPORT</a>	<a href="#">Section 6.3.1</a>



Threats	Security Objectives	Rationale
<a href="#">T.DELETION</a>	<a href="#">O.DELETION</a> , <a href="#">O.CARD MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.INSTALL</a>	<a href="#">O.INSTALL</a> , <a href="#">O.LOAD</a> , <a href="#">O.CARD MANAGEMENT</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.OBJ-DELETION</a>	<a href="#">O.OBJ-DELETION</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.PHYSICAL</a>	<a href="#">O.SCP.IC</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.CONFIGURATION</a>	<a href="#">O.RESIDENT_APPLICATION</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.CONF_DATA_APPLET</a>	<a href="#">O.SECURE_COMPARE</a>	<a href="#">Section 6.3.1</a>
<a href="#">T.PATCH_LOADING</a>	<a href="#">O.PATCH_LOADING</a>	<a href="#">Section 6.3.1</a>

Table 6: Threats and Security Objectives – Coverage



Security Objectives	Threats	Rationale
<a href="#">O.SID</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.SID.1</a> , <a href="#">T.SID.2</a>	
<a href="#">O.FIREWALL</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.SID.1</a> , <a href="#">T.SID.2</a> , <a href="#">T.EXE-CODE.1</a>	
<a href="#">O.GLOBAL_ARRAYS_CONFID</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.SID.1</a>	
<a href="#">O.GLOBAL_ARRAYS_INTEG</a>	<a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.SID.1</a>	
<a href="#">O.NATIVE</a>	<a href="#">T.CONFID-JCS-CODE</a> , <a href="#">T.INTEG-APPLI-CODE</a> , <a href="#">T.INTEG-JCS-CODE</a> , <a href="#">T.NATIVE</a>	
<a href="#">O.OPERATE</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.SID.2</a> , <a href="#">T.RESOURCES</a>	
<a href="#">O.REALLOCATION</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a>	
<a href="#">O.RESOURCES</a>	<a href="#">T.RESOURCES</a>	
<a href="#">O.ALARM</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-DATA</a>	
<a href="#">O.CIPHER</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a>	
<a href="#">O.KEY-MNGT</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a>	
<a href="#">O.PIN-MNGT</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a>	
<a href="#">O.TRANSACTION</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a>	
<a href="#">O.OBJ-DELETION</a>	<a href="#">T.OBJ-DELETION</a>	
<a href="#">O.DELETION</a>	<a href="#">T.DELETION</a>	
<a href="#">O.LOAD</a>	<a href="#">T.INTEG-APPLI-CODE.LOAD</a> , <a href="#">T.INTEG-APPLI-DATA.LOAD</a> , <a href="#">T.INSTALL</a>	
<a href="#">O.INSTALL</a>	<a href="#">T.SID.1</a> , <a href="#">T.SID.2</a> , <a href="#">T.RESOURCES</a> , <a href="#">T.INSTALL</a>	
<a href="#">O.SCP.SUPPORT</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.SID.2</a> , <a href="#">T.RESOURCES</a>	
<a href="#">O.SCP.IC</a>	<a href="#">T.PHYSICAL</a>	
<a href="#">O.PATCH_LOADING</a>	<a href="#">T.PATCH_LOADING</a>	

|                    }                    }                    }                    }

Security Objectives	Threats	Rationale
<a href="#">O.SCP.RECOVERY</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.SID.2</a> , <a href="#">T.RESOURCES</a>	
<a href="#">O.RESIDENT_APPLICATION</a>	<a href="#">T.CONFIGURATION</a>	
<a href="#">O.CARD_MANAGEMENT</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-CODE</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-CODE</a> , <a href="#">T.INTEG-APPLI-CODE.LOAD</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA.LOAD</a> , <a href="#">T.INTEG-JCS-CODE</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.SID.1</a> , <a href="#">T.DELETION</a> , <a href="#">T.INSTALL</a>	
<a href="#">O.SECURE_COMPARE</a>	<a href="#">T.CONF_DATA_APPLET</a>	
<a href="#">OE.APPLET</a>	<a href="#">T.NATIVE</a>	
<a href="#">OE.VERIFICATION</a>	<a href="#">T.CONFID-APPLI-DATA</a> , <a href="#">T.CONFID-JCS-CODE</a> , <a href="#">T.CONFID-JCS-DATA</a> , <a href="#">T.INTEG-APPLI-CODE</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-JCS-CODE</a> , <a href="#">T.INTEG-JCS-DATA</a> , <a href="#">T.EXE-CODE.1</a> , <a href="#">T.EXE-CODE.2</a> , <a href="#">T.NATIVE</a>	
<a href="#">OE.CODE-EVIDENCE</a>	<a href="#">T.INTEG-APPLI-CODE</a> , <a href="#">T.INTEG-APPLI-CODE.LOAD</a> , <a href="#">T.INTEG-APPLI-DATA</a> , <a href="#">T.INTEG-APPLI-DATA.LOAD</a> , <a href="#">T.INTEG-JCS-CODE</a> , <a href="#">T.INTEG-JCS-DATA</a>	

Table 7: Security Objectives and Threats – Coverage

Organisational Security Policies	Security Objectives	Rationale
<a href="#">OSP.VERIFICATION</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">OE.CODE-EVIDENCE</a>	<a href="#">Section 6.3.2</a>

Table 8: OSPs and Security Objectives – Coverage



Security Objectives	Organisational Security Policies	Rationale
<a href="#">O.SID</a>		
<a href="#">O.FIREWALL</a>		
<a href="#">O.GLOBAL_ARRAYS_CONFID</a>		
<a href="#">O.GLOBAL_ARRAYS_INTEG</a>		
<a href="#">O.NATIVE</a>		
<a href="#">O.OPERATE</a>		
<a href="#">O.REALLOCATION</a>		
<a href="#">O.RESOURCES</a>		
<a href="#">O.ALARM</a>		
<a href="#">O.CIPHER</a>		
<a href="#">O.KEY-MNGT</a>		
<a href="#">O.PIN-MNGT</a>		
<a href="#">O.TRANSACTION</a>		
<a href="#">O.OBJ-DELETION</a>		
<a href="#">O.DELETION</a>		
<a href="#">O.LOAD</a>		
<a href="#">O.INSTALL</a>		
<a href="#">O.SCP.SUPPORT</a>		
<a href="#">O.SCP.IC</a>		
<a href="#">O.SCP.RECOVERY</a>		
<a href="#">O.RESIDENT_APPLICATION</a>		
<a href="#">O.CARD_MANAGEMENT</a>		
<a href="#">O.SECURE_COMPARE</a>		
<a href="#">OE.APPLET</a>		
<a href="#">OE.VERIFICATION</a>	<a href="#">OSP.VERIFICATION</a>	
<a href="#">OE.CODE-EVIDENCE</a>	<a href="#">OSP.VERIFICATION</a>	

Table 9: Security Objectives and OSPs – Coverage



Assumptions	Security Objectives for the Operational Environment	Rationale
<a href="#">A.APPLET</a>	<a href="#">OE.APPLET</a>	<a href="#">Section 6.3.3</a>
<a href="#">A.VERIFICATION</a>	<a href="#">OE.VERIFICATION</a> , <a href="#">OE.CODE-EVIDENCE</a>	<a href="#">Section 6.3.3</a>

Table 10: Assumptions and Security Objectives for the Operational Environment – Coverage

Security Objectives for the Operational Environment	Assumptions	Rationale
<a href="#">OE.APPLET</a>	<a href="#">A.APPLET</a>	
<a href="#">OE.VERIFICATION</a>	<a href="#">A.VERIFICATION</a>	
<a href="#">OE.CODE-EVIDENCE</a>	<a href="#">A.VERIFICATION</a>	

Table 11: Security Objectives for the Operational Environment and Assumptions – Coverage

## 6 EXTENDED REQUIREMENTS

---

### 6.1 Extended Families

#### 6.1.1 Extended Family FCS\_RNG - FCS\_RNG: Random Number Generation

##### 6.1.1.1 Description

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

Extended Component FCS\_RNG.1

#### **Description**

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

#### **Family behaviour:**

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

#### **Component levelling:**

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

**Management:** There are no management activities foreseen

**Audit:** There are no actions defined to be auditable

**Hierarchical to:** No other components

#### **Definition**

<b>FCS_RNG.1 Random Number Generation</b>
---

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

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

Dependencies: No dependencies.



## 7 SECURITY REQUIREMENTS

### 7.1 Security Functional Requirements

This section states the security functional requirements for the Java Card System - Open configuration. For readability and for compatibility with the original Java Card System Protection Profile Collection - Standard 2.2 Configuration [R43], requirements are arranged into groups. All the groups defined in the table below apply to this Security Target.

Group	Description
Core with Logical Channels ( <i>CoreG_LC</i> )	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 ( <i>CoreG</i> ) and the Logical channels ( <i>LCG</i> ) groups defined in [R43] (cf. Java Card System Protection Profile Collection [R43]).
Installation ( <i>InstG</i> )	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 ( <i>ADELG</i> )	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 ( <i>ODELG</i> )	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 ( <i>CarG</i> )	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.

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

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

|                    }                    }                    }                    }

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

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

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

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



Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected".
Applet's version number	The version number of an applet (package) indicated in the export file.
Class	Identifies the implementation class of the remote object.
Context	Package AID or "Java Card RE".
COD Context attribute	Delimits the space occupied in volatile memory by the data of the CLEAR_ON_DESELECT transient arrays of a package
COR Context attribute	Delimits the space occupied in volatile memory by the data of the CLEAR_ON_RESET transient arrays of a package
Current Frame Context	The lower and upper Boundary of the local variables area on the stack frame for a method and the lower and upper Boundary of the operand stack area on the stack frame for a method
Currently Active Context	Package AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([R8], §4.5.2).
ExportedInfo	Boolean (indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies the remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*) or CLEAR_ON_RESET
Object Boundary	Delimits the space occupied by an object in the heap
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file.
Package Boundary	Delimits the space occupied by the code and the static fields of a package



Security attribute	Description/Value
Program Counter	Position of the next Bytecode to execute
Registered Applets	The set of AID of the applet instances registered on the card.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, Java Card RE entry point or global array.
Stack Pointer	Position of the next free slot in the stack
Static Fields	Static fields of a package
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

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

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



Operation	Description
OP.ARRAY_ACCESS (O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE (Sharing, LifeTime) (* )	Creation of an object (new or makeTransient call).
OP.DELETE_APPLET (O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG ( O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET (O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.
OP.FLOW (O.CODE_PKG)	Any operation that modify the execution flow
OP.IMPORT_KEY	Import of the keys
OP.INSTANCE_FIELD (O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.
OP.INVK_INTERFACE (O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.INVK_VIRTUAL (O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.JAVA (...)	Any access in the sense of [R7], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.LOCAL_STACK_ACCESS (...)	Any operation that read or write the local stack
OP.OPERAND_STACK_ACCESS (...)	Any operation that push or pop items on the operand stack
OP.PUT (S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.STATIC_FIELD (O.CODE_PKG, field)	Read/Write a static field of a class in the JAVA programming language
OP.THROW (O.JAVAOBJECT)	Throwing of an object (athrow, see [R7], §6.2.8.7).
OP.TYPE_ACCESS (O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).
Cardholder Authentication	Authentication of the cardholder
U.Card_Issuer authentication	Authentication of U.Card_Issuer



(\*) For this operation, there is no accessed object. This rule enforces that shareable transient objects are not allowed, except some objects, such as COR. For more information refer to the Java Doc [R6]. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

### 7.1.1 CoreG\_LC Security Functional Requirements

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

#### 7.1.1.1 Firewall Policy

#### **FDP\_ACC.2/FIREWALL Complete access control**

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

*Refinement:*

The operations involved in the policy are:

- o OP.CREATE,
- o OP.INVK\_INTERFACE,
- o OP.INVK\_VIRTUAL,
- o OP.JAVA,
- o OP.THROW,
- o OP.TYPE\_ACCESS.

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

*Application Note:*

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

#### **FDP\_ACF.1/FIREWALL Security attribute based access control**

**FDP\_ACF.1.1/FIREWALL** The TSF shall enforce the **FIREWALL access control SFP** to



objects based on the following:

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

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

- **R.JAVA.1 ([R7], §6.2.8): S.PACKAGE may freely perform OP.ARRAY\_ACCESS, OP.INSTANCE\_FIELD, OP.INVK\_VIRTUAL, OP.INVK\_INTERFACE, OP.THROW or OP.TYPE\_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**
- **R.JAVA.2 ([R7], §6.2.8): S.PACKAGE may freely perform OP.ARRAY\_ACCESS, OP.INSTANCE\_FIELD, OP.INVK\_VIRTUAL, OP.INVK\_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**
- **R.JAVA.3 ([R7], §6.2.8.10): S.PACKAGE may perform OP.TYPE\_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**
- **R.JAVA.4 ([R7], §6.2.8.6): S.PACKAGE may perform OP.INVK\_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following conditions applies:**
  - **a) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",**
  - **b) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute Active Applets.**
- **R.JAVA.5: S.PACKAGE may perform OP.CREATE only if the value of the Sharing parameter is "Standard".**

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

- **1) The subject S.JCRE can freely perform OP.JAVA("") and OP.CREATE, with the exception given in FDP\_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**



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

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

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

Application Note:

FDP\_ACF.1.4/FIREWALL:

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

In the case of an array type, fields are components of the array ([R41], §2.14, §2.7.7), as well as the length; the only methods of an array object are those inherited from the Object class.

The Sharing attribute defines four categories of objects:

- Standard ones, whose both fields and methods are under the firewall policy,
- Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
- JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,
- Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.

When a new object is created, it is associated with the Currently Active Context. But the object is owned by the applet instance within the Currently Active Context when the object is instantiated ([R7], §6.1.3). An object is owned by an applet instance, by the JCRE or by the package library where it has been defined (these latter objects can only be arrays that initialize static fields of packages).

([R7], Glossary) Selected Applet Context. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.

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

([R7], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).



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

It should be noticed that the Java Card platform, version 2.2.x and version 3 Classic Edition, introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([R8], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([R7], §4).

**FDP\_IFC.1/JCVM Subset information flow control**

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

*Application Note:*

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

**FDP\_IFF.1/JCVM Simple security attributes**

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

Subjects	Security attributes
S.JCVM	Currently Active Context

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

- o **An operation OP.PUT(S1, S.MEMBER, I.DATA) is allowed if and only if the Currently Active Context is "Java Card RE";**



- o other OP.PUT operations are allowed regardless of the Currently Active Context's value.

**FDP\_IFF.1.3/JCVM** The TSF shall enforce the **none**.

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

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

*Application Note:*

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([R7], §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**.

*Application Note:*

The semantics of the Java programming language requires for any object field and array position to be initialized with default values when the resource is allocated [R41], §2.5.1.

#### **FMT\_MSA.1/JCRE Management of security attributes**

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

*Application Note:*

The modification of the Selected Applet Context should be performed in accordance with the



rules given in [R7], §4 and [R8], §3.4.

**FMT\_MSA.1/JCVM Management of security attributes**

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

*Application Note:*

The modification of the Currently Active Context should be performed in accordance with the rules given in [R7], §4 and [R8], §3.4.

**FMT\_MSA.2/FIREWALL\_JCVM Secure security attributes**

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

*Application Note:*

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

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

**FMT\_MSA.3/FIREWALL Static attribute initialisation**

**FMT\_MSA.3.1/FIREWALL** The TSF shall enforce the **FIREWALL access control SFP** to



provide **restrictive** default values for security attributes that are used to enforce the SFP.

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

*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 ([R7], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".
- The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT\_MSA.3.2/FIREWALL

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

#### FMT\_MSA.3/JCVM Static attribute initialisation

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

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

#### FMT\_SMF.1 Specification of Management Functions

**FMT\_SMF.1.1** The TSF shall be capable of performing the following management functions:

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

#### FMT\_SMR.1 Security roles



**FMT\_SMR.1.1** The TSF shall maintain the roles:

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

**FMT\_SMR.1.2** The TSF shall be able to associate users with roles.

7.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.

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

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

**FCS\_CKM.1 Cryptographic key generation**

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

Cryptographic key generation algorithm	Cryptographic key size	List of standards
TDES	112 bits or 168 bits	FIPS PUB 46-3 (ANSI X3.92), FIPS PUB 81
ECKeyP	from 160 to 521 bits	IEEE Std 1363a-2004 [R27]
RSA	from 512 to 2048 bits with a step of 256-bits	ANSI X9.31
AES	from 128 to 256 bits with a step of 64 bits	FIPS PUB 197
GP Keys - TDES (ECB)	112 bits	GP 2.3
GP Keys – AES (ECB)	128, 192, 256 bits	GP 2.3
GP Keys – AES (ECB)	128 bits	GP 2.3

*Application Note:*

- The keys can be generated and diversified in accordance with [R6] specification in classes KeyBuilder and KeyPair (at least Session key generation).



- This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms [R6].
- This component shall be instantiated according to the version of the Global Platform GP 2.3 [R12] and [R13].

**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]**.

Algorithm	Distribution method	Standards	Proprietary
RSA	setKey()	Java Card API	None
AES	setKey() setEncKey() setMacKey	Java Card API	ISOSecureMessaging APIs: "com.oberthurcs.javacard.utilSM"
DES/TDES	setKey() setMacKey(); setEncKey();	Java Card API	ISOSecureMessaging API: "com.oberthurcs.javacard.utilSM"
ECkeyP	setKey()	Java Card API	SAC API: com.oberthurcs.javacard.sac

*Application Note:*

- SetKey meets [R6] specification.
- This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms ([R6]).

**FCS\_CKM.3 Cryptographic key access**


**FCS\_CKM.3.1** The TSF shall perform **the following types of cryptographic key access** in accordance with a specified cryptographic key access method **see refinement below** that meets the following:

- o **"Java Card API" specification [R6].**

*Refinement:*

Type of cryptographic key access Cryptographic key access methods:

**DES/AES**

The following JC API key access method: DES.getKey(), AES.getKey

**RSA**

The following JC API key access methods: RSAPrivateCRTKey.getP, RSAPrivateCRTKey.getQ, RSAPrivateCRTKey.getPQ, RSAPrivateCRTKey.getDP1, RSAPrivateCRTKey.getDQ1, RSAPrivateKey.getModulus, RSAPrivateKey.getExponent, RSAPublicKey.getModulus, RSAPublicKey.getExponent,

**ECkeyP**

The following JC API key access methods:

ECPrivateKey.getField, ECPrivateKey.getA, ECPrivateKey.getB, ECPrivateKey.getG, ECPrivateKey.getR, ECPrivateKey.getK, , ECPrivateKey.getS, ECPublicKey.setA, , ECPublicKey.getField, ECPublicKey.getA, ECPublicKey.getB, ECPublicKey.getG, ECPublicKey.getR, ECPublicKey.getK, ECPublicKey.getW,

*Application Note:*

- The keys can be accessed as specified in [R6] Key class.
- This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([R6]).
- All \*get\*key() methods are concerned by this SFR.

**FCS\_CKM.4 Cryptographic key destruction**

**FCS\_CKM.4.1** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **The keys are reset with the method clearKey()** that meets the following: **"Java Card API" specification [R6]. The keys used in class ISOSecureMessaging (Package "com.oberthurcs.javacard.utilSM") [R48] are classes Key that meets the following: "Java Card API" specification [R6]. The methods 'reset' and 'setKeyFormat' call the method key.clearKey() for clearing the value of each key.**

*Application Note:*

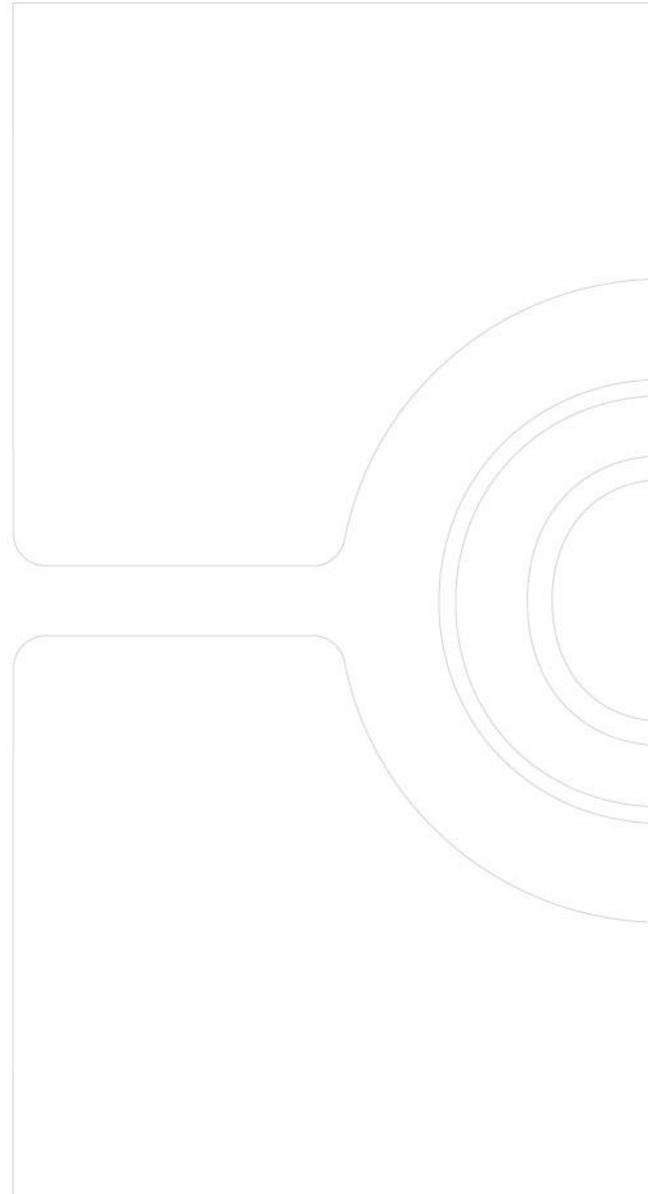
- The keys are reset as specified in [R6] Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing shall throw an exception.

**FCS\_COP.1 Cryptographic operation**

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



below:



Cryptographic operation	Cryptographic algorithm	TOE supported size	"Standardized" size	List of standards
signature, signature's verification, encryption and decryption	DES – TDES with Modes CBC, ECB, and CMAC	56, 112 or 168 bits	TDES 168	FIPS PUB 46-3, ANSI X3.92, FIPS PUB 81, ISO/IEC 9797(1999), Data integrity mechanism [R17]
signature, signature's verification, encryption and decryption	AES with Modes CBC, ECB, and CMAC	from 128 to 256 bits with a step of 64 bits	greater than 128 bits	FIPS PUB 197 SP800-38B (CMAC)
signature, signature's verification, encryption and decryption	RSA CRT, private keys. RSA SFM (Straightforward) public and private keys.	from 1024 to 2048 bits with a step of 256-bits	2048 bits	ANSI X9.31, ISO/IEC 9796-1, annex A, section A.4 and A.5, and annex C, PKCS#1
signature	HMAC	64 bits up to 1016 bits Based on SHA-256, SHA-384 and SHA-512	SHA-2 with key size > 128 bits	FIPS 198 The Keyed-Hash Message Authentication Code (HMAC)
Hash functions	SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512	SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512	SHA-224, SHA-256, SHA-384 and SHA-512	Secure Hash Standard, FIPS PUB 180-3
signature, signature's verification	ECDSA	160, 192, 256, 384, 512 and 521bits	Greater than 200 bits	ANSI X9.62-1998



Cryptographic operation	Cryptographic algorithm	TOE supported size	"Standardized" size	List of standards
Key agreement	ECDH	160 to 521 bits	Greater than 200 bits	IEEE P1363
Checksum	16-bit using the hardware co-processor	16 bits		ISO3309
	32-bit in software implementation	32 bits		
SAC dedicated APIs from com.oberthurcs.javacard.sac.sac	AES	128 up to 256 bits	greater than 128 bits	
	TDES	112 bits	TDES 168 bits	
	ECDH	160 to 521 bits	Greater than 200 bits	
Java Package tools for applets "com.oberthurcs.javacard.utilSM"	TDES	112 or 168 bits	TDES 168	
	AES	128 up to 256 bits	Greater than 128 bits	

*Refinement:*

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

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

SHA /IDEMIA has developed the algorithm/Hash function/SHA-224/No cryptographic



key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

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

SHA /IDEMIA has developed the algorithm/Hash function/SHA-384/No cryptographic key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

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

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

RSA without CRT /IDEMIA has developed the algorithm using HW PK accelerator/Data Encryption and Decryption/RSA Without CRT Data /Between 1024 bits to 2048 bits/PKCS#1 V2.0; 1st October, 1998

RSA with CRT /IDEMIA has developed the algorithm using HW PK accelerator/Data Encryption and Decryption/RSA With CRT Data /Between 1024 bits and 2048 bits/PKCS#1 V2.0; 1st October, 1998 RNG/IDEMIA has developed the algorithm using HW RNG as seed/Random generator//No cryptographic key/FIPS SP800-90, 2007, March

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

*Application Note:*

- The TOE shall provide a subset of cryptographic operations defined in [R6] (see javacardx.crypto.Cipher and javacardx.security packages).
- This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([R6]).

**FDP\_RIP.1/ABORT Subset residual information protection**

**FDP\_RIP.1/ABORT** The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any reference to an object instance created during an aborted transaction.**

*Application Note:*

The events that provoke the de-allocation of a transient object are described in [R7], §5.1.

**FDP\_RIP.1/APDU Subset residual information protection**

**FDP\_RIP.1/APDU** The TSF shall ensure that any previous information content of a resource



is made unavailable upon the **allocation of the resource** to the following objects: **the APDU buffer**.

**FDP\_RIP.1/bArray Subset residual information protection**

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

*Application Note:*

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

**FDP\_RIP.1/KEYS Subset residual information protection**

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

*Application Note:*

- The javacard.security & javacardx.crypto packages do provide secure interfaces to the cryptographic buffer in a transparent way. See javacard.security.KeyBuilder and Key interface of [R6].

**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 [R7], §5.1.
- The clearing of CLEAR\_ON\_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable applet instances, CLEAR\_ON\_DESELECT memory segments may be attached to applets that are active in different logical channels. Multiselectable applet instances within a same package must share the transient memory segment if they are concurrently active ([R7], §4.2).



**FDP\_ROL.1/FIREWALL Basic rollback**

**FDP\_ROL.1.1/FIREWALL** The TSF shall enforce the **FIREWALL access control SFP** and the **JCVM information flow control SFP** to permit the rollback of the operations **OP.JAVA** and **OP.CREATE** on the object **O.JAVAOBJECT**.

**FDP\_ROL.1.2/FIREWALL** The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [R7], §7.7, within the bounds of the Commit Capacity ([R7], §7.8), and those described in [R6].**

*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 [R6] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

## 7.1.1.3 Card Security Management

**FAU\_ARP.1 Security alarms**

**FAU\_ARP.1.1** The TSF shall take **one of the following actions:**

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

upon detection of a potential security violation.

*Refinement:*

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

- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context,
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow



**FDP\_SDI.2 Stored data integrity monitoring and action**

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

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

*Application Note:*

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

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

**FPR\_UNO.1 Unobservability**

**FPR\_UNO.1.1** The TSF shall ensure that **any user** is unable to observe the operation **Cardholder authentication** on **D.PIN** by **no user and no subject**.

*Application Note:*

Although it is not required in [R7] specifications, the non-observability of operations on sensitive information such as keys appears as impossible to circumvent in the smart card world. The precise list of operations and objects is left unspecified, but should at least concern secret keys and PIN codes when they exist on the card, as well as the cryptographic operations and comparisons performed on them.

**FPT\_FLS.1 Failure with preservation of secure state**

**FPT\_FLS.1.1** The TSF shall preserve a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU\_ARP.1**.

*Application Note:*

The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([R7], §6.2.3) or after a proximity card (PICC) activation sequence ([R7]). Behaviour of the TOE on power loss and reset is described in [R7], §3.6 and §7.1. Behaviour of the TOE on RF signal loss is described in [R7], §3.6.1.



**FPT\_TDC.1 Inter-TSF basic TSF data consistency**

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

**FPT\_TDC.1.2** The TSF shall use

- o **the rules defined in [R8] specification,**
- o **the API tokens defined in the export files of reference implementation** when interpreting the TSF data from another trusted IT product.

*Application Note:*

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

## 7.1.1.4 AID Management

**FIA\_ATD.1/AID User attribute definition**

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

- o **Package AID,**
- o **Applet's version number,**
- o **Registered applet AID,**
- o **Applet Selection Status ([R8], §6.5).**

*Refinement:*

"Individual users" stand for applets.

**FIA\_UID.2/AID User identification before any action**

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

*Application Note:*

- By users here it must be understood the ones associated to the packages (or applets) that act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected applet or the



package that is the subject's owner. Means of identification are provided during the loading procedure of the package and the registration of applet instances.

- The role Java Card RE defined in FMT\_SMR.1 is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself to the TOE, but it is part of it.

#### FIA\_USB.1/AID User-subject binding

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

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

**FIA\_USB.1.3/AID** The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **The initially assigned Package AID is unchangeable**.

*Application Note:*

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

#### FMT\_MTD.1/JCRE Management of TSF data

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

*Application Note:*

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

#### FMT\_MTD.3/JCRE Secure TSF data

**FMT\_MTD.3.1/JCRE** The TSF shall ensure that only secure values are accepted for **the**



registered applets' AIDs.

#### 7.1.2 InstG Security Functional Requirements

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

#### FDP\_ITC.2/Installer Import of user data with security attributes

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

**FDP\_ITC.2.2/Installer** The TSF shall use the security attributes associated with the imported user data.

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

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

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

**Package loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the dependent package is lesser than or equal to the major (minor) Version attribute associated to the resident package ([R8], §4.5.2)..**

*Application Note:*

FDP\_ITC.2.1/Installer:

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

FDP\_ITC.2.3/Installer:



- The format of the CAP file is precisely defined in [R8] specifications; it contains the user data (like applet's code and data) and the security attributes altogether. Therefore there is no association to be carried out elsewhere.

FDP\_ITC.2.4/Installer:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers ([R8], §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 ([R8], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatible. However, package files do have "package Version Numbers" ([R8]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

FDP\_ITC.2.5/Installer:

- A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs.
- The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([R8], §4.4).
- The installation (the invocation of an applet's install method by the installer) is implementation dependent ([R7], §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, [R7], §11).

#### FMT\_SMR.1/Installer Security roles

**FMT\_SMR.1.1/Installer** The TSF shall maintain the roles: **S.INSTALLER**.

**FMT\_SMR.1.2/Installer** The TSF shall be able to associate users with roles.

#### FPT\_FLS.1/Installer Failure with preservation of secure state

**FPT\_FLS.1.1/Installer** The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [R7] §11.1.4.**

*Application Note:*

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



**FPT\_RCV.3/Installer Automated recovery without undue loss**

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

Application note:

There is no maintenance mode on the TOE.

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

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

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

*Application Note:*

FPT\_RCV.3.1/Installer:

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

FPT\_RCV.3.2/Installer:

- Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [R7], §11.1.5 for possible scenarios. Precise behaviour is left to implementers. This component shall include among the listed failures the deletion of a package/applet. See ([R7], 11.3.4) for possible scenarios. Precise behaviour is left to implementers.
- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [R24]) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT\_FLS.1.1, FDP\_RIP.1/TRANSIENT, FDP\_RIP.1/ABORT and FDP\_ROL.1/FIREWALL.

FPT\_RCV.3.3/Installer:



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

### 7.1.3 ADELG Security Functional Requirements

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

#### **FDP\_ACC.2/ADEL Complete access control**

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

*Refinement:*

The operations involved in the policy are:

- o OP.DELETE\_APPLET,
- o OP.DELETE\_PCKG,
- o OP.DELETE\_PCKG\_APPLET.

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

#### **FDP\_ACF.1/ADEL Security attribute based access control**

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

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



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

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

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

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

- R.JAVA.14 ([R7], §11.3.4.1, Applet Instance Deletion): S.ADEL may perform OP.DELETE\_APPLET upon an O.APPLET only if,
  - (1) S.ADEL is currently selected,
  - (2) there is no instance in the context of O.APPLET that is active in any logical channel and
  - (3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([R7], §8.5) O.JAVAOBJECT is remote reachable.
- R.JAVA.15 ([R7], §11.3.4.1, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE\_APPLET upon several O.APPLET only if,
  - (1) S.ADEL is currently selected,
  - (2) there is no instance of any of the O.APPLET being deleted that is active in any logical channel and
  - (3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([R7], §8.5) O.JAVAOBJECT is remote reachable.
- R.JAVA.16 ([R7], §11.3.4.2, Applet/Library Package Deletion): S.ADEL may perform OP.DELETE\_PCKG upon an O.CODE\_PKG only if,
  - (1) S.ADEL is currently selected,
  - (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE\_PKG that is an instance of a class that belongs to O.CODE\_PKG, exists on the card and
  - (3) there is no resident package on the card that depends on O.CODE\_PKG.
- R.JAVA.17 ([R7], §11.3.4.3, Applet Package and Contained Instances Deletion): S.ADEL may perform OP.DELETE\_PCKG\_APPLET upon an O.CODE\_PKG only if,
  - (1) S.ADEL is currently selected,
  - (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE\_PKG, which is an instance of a class that belongs to O.CODE\_PKG exists on the card,



- (3) there is no package loaded on the card that depends on O.CODE\_PKG, and
- (4) for every O.APPLET of those being deleted it holds that: (i) there is no instance in the context of O.APPLET that is active in any logical channel and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([R7], §8.5) O.JAVAOBJECT is remote reachable.

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

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

*Application Note:*

FDP\_ACF.1.2/ADEL:

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

#### **FDP\_RIP.1/ADEL Subset residual information protection**

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

*Application Note:*

Deleted freed resources (both code and data) may be reused, depending on the way they were deleted (logically or physically). Requirements on de-allocation during applet/package deletion are described in [R7], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

#### **FMT\_MSA.1/ADEL Management of security attributes**

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



**FMT\_MSA.3/ADEL Static attribute initialisation**

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

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

**FMT\_SMF.1/ADEL Specification of Management Functions**

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

*Application Note:*

The modification of the Active Applets security attribute should be performed in accordance with the rules given in [R7], §4.

**FMT\_SMR.1/ADEL Security roles**

**FMT\_SMR.1.1/ADEL** The TSF shall maintain the roles: **applet deletion manager**.

**FMT\_SMR.1.2/ADEL** The TSF shall be able to associate users with roles.

**FPT\_FLS.1/ADEL Failure with preservation of secure state**

**FPT\_FLS.1.1/ADEL** The TSF shall preserve a secure state when the following types of failures occur: **the applet deletion manager fails to delete a package/applet as described in [R7], §11.3.4**.

*Application Note:*

- The TOE may provide additional feedback information to the card manager in case of a potential security violation (see FAU\_ARP.1).
- The Package/applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with Java Card specification ([R7], §11.3.4.)

## 7.1.4 ODELG Security Functional Requirements

The following requirements concern the object deletion mechanism. This mechanism is

| } } } }

triggered by the applet that owns the deleted objects by invoking a specific API method.

**FDP\_RIP.1/ODEL Subset residual information protection**

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

*Application Note:*

- Freed data resources resulting from the invocation of the method `javacard.framework.JCSystem.requestObjectDeletion()` may be reused. Requirements on de-allocation after the invocation of the method are described in [R6].
- There is no conflict with FDP\_ROL.1 here because of the bounds on the rollback mechanism: the execution of `requestObjectDeletion()` is not in the scope of the rollback because it must be performed in between APDU command processing, and therefore no transaction can be in progress.

**FPT\_FLS.1/ODEL Failure with preservation of secure state**

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

*Application Note:*

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

#### 7.1.5 CarG Security Functional Requirements

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

**FCO\_NRO.2/CM Enforced proof of origin**

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

**FCO\_NRO.2.2/CM [Editorially Refined]** The TSF shall be able to relate the **identity** of the originator of the information, and the **application package contained in** the information to



which the evidence applies.

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

*Application Note:*

FCO\_NRO.2.1/CM:

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

FCO\_NRO.2.3/CM:

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

#### **FDP\_IFC.2/CM Complete information flow control**

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

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

*Application Note:*

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

#### **FDP\_IFF.1/CM Simple security attributes**



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

**FDP\_IFF.1.2/CM** The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **the rules describing the communication protocol used by the CAD and the card for transmitting a new package, see chapter 9.3.9 [R9].**

**FDP\_IFF.1.3/CM** The TSF shall enforce the **none.**

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

**FDP\_IFF.1.5/CM** The TSF shall explicitly deny an information flow based on the following rules: **the rules describing the communication protocol used by the CAD and the card for transmitting a new package, see chapter 9.3.9 [R9].**

*Application Note:*

FDP\_IFF.1.1/CM:

- The security attributes used to enforce the **PACKAGE LOADING SFP** are implementation dependent. More precisely, they depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that can be used are: (1) the keys used by the subjects to encrypt/decrypt their messages; (2) the number of pieces the application package has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the package, etc. See for example Appendix D of [R12].

FDP\_IFF.1.2/CM:

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

#### **FDP\_UIT.1/CM Data exchange integrity**

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

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

*Application Note:*



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

**FIA\_UID.1/CM Timing of identification**

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

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

*Application Note:*

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

**FMT\_MSA.1/CM Management of security attributes**

**FMT\_MSA.1.1/CM** The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to restrict the ability to **modify** the security attributes **AS.KEYSET\_VERSION, AS.KEYSET\_VALUE, Default SELECTED Privileges, AS.CMLIFECYC** to **R.Card\_Manager**.

**FMT\_MSA.3/CM Static attribute initialisation**

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

**FMT\_MSA.3.2/CM** The TSF shall allow the **Card manager** to specify alternative initial values to override the default values when an object or information is created.

**FMT\_SMF.1/CM Specification of Management Functions**

**FMT\_SMF.1.1/CM** The TSF shall be capable of performing the following management functions: **Modify the following security attributes: AS.KEYSET\_VERSION, AS.KEYSET\_VALUE, Default SELECTED Privileges, AS.CMLIFECYC.**



**FMT\_SMR.1/CM Security roles**

**FMT\_SMR.1.1/CM** The TSF shall maintain the roles **Card manager**.

**FMT\_SMR.1.2/CM** The TSF shall be able to associate users with roles.

**FTP\_ITC.1/CM Inter-TSF trusted channel**

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

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

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

*Application Note:*

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

7.1.5.1 Additional Security Functional Requirements for CM

**FPT\_TST.1 TSF testing**

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

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

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

*Application Note:*

Namely, “stored TSF executable code” encompasses the patch and java packages. During startup, the TOE checks the integrity of the patch/java packages. To do so, the related bits should have been set accordingly at the prepersonalisation phase, c.f. AGD\_PRE, [R39] section Lock POST (DO ‘DF41’).

Other self-tests are described in AGD\_PRE, [R39] section Lock **POST (DO ‘DF41’)**. Namely, According to the protocol used Known Answer Test (or POST for Power On Self Tests) checks SHA, RSA, ECDSA either in startup or during 1<sup>st</sup> use. Those latter tests are configurable.



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

**FCO\_NRO.2/CM\_DAP Enforced proof of origin**

**FCO\_NRO.2.1/CM\_DAP** The TSF shall enforce the generation of evidence of origin for transmitted **Loadfile** at all times.

**FCO\_NRO.2.2/CM\_DAP** The TSF shall be able to relate the **AS.KEYSET\_VALUE** of the originator of the information, and the **CAP file components** of the information to which the evidence applies.

**FCO\_NRO.2.3/CM\_DAP** The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **during CAP file loading**.

Application Note:

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

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

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

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

**FIA\_AFL.1/CM Authentication failure handling**

**FIA\_AFL.1.1/CM** The TSF shall detect when **1** unsuccessful authentication attempts occur related to **U.Card\_Issuer authentication**.

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

**FIA\_UAU.1/CM Timing of authentication**

**FIA\_UAU.1.1/CM** The TSF shall allow **Get Data, Initialize Update, Select** on behalf of the



user to be performed before the user is authenticated.

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

#### **FIA\_UAU.4/CardIssuer Single-use authentication mechanisms**

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

**Application note:**

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

#### **FIA\_UAU.7/CardIssuer Protected authentication feedback**

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

#### **FPR\_UNO.1/Key\_CM Unobservability**

**FPR\_UNO.1.1/Key\_CM** The TSF shall ensure that **all subjects** are unable to observe the operation **OP.IMPORT\_KEY** on **Key** by **D.JCS\_KEYS**.

#### **FPT\_TDC.1/CM Inter-TSF basic TSF data consistency**

**FPT\_TDC.1.1/CM** The TSF shall provide the capability to consistently interpret **AS.KEYSET\_VALUE** when shared between the TSF and another trusted IT product.

**FPT\_TDC.1.2/CM** The TSF shall use **the rules defined in the GP [R9] section 11.8** when interpreting the TSF data from another trusted IT product.

#### **FMT\_SMR.2/CM Restrictions on security roles**



**FMT\_SMR.2.1/CM** The TSF shall maintain the roles: **see below**.

**FMT\_SMR.2.2/CM** The TSF shall be able to associate users with roles.

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

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

#### **FCS\_COP.1/CM Cryptographic operation**

**FCS\_COP.1.1/CM** The TSF shall perform **see table below** in accordance with a specified cryptographic algorithm **see table below** and cryptographic key sizes **see table below** that meet the following:

Cryptographic operation	Algorithm	Key length	Standard
TOE authentication key ISK/KMC	SCP02	112 bits	GP 2.3
TOE authentication key ISK/KMC	SCP03	128/192/256 bits	GP 2.3
SCP02 - signature, verification of signature, encryption and decryption (KEK (key encryption key) for sensitive objects such as PIN, keys ... is mandatory)	TDES	112 bits	SCP02 – GP 2.3
SCP03 - signature, verification of signature, encryption and decryption	AES	128/192/256 bits	SCP03 – GP 2.3

#### 7.1.5.2 Additional Security Functional Requirements for Resident application



**FDP\_ACC.2/PP Complete access control**

**FDP\_ACC.2.1/PP** The TSF shall enforce the **Access Control** on **See below** and all operations among subjects and objects covered by the SFP

Access Control	
Prepersonalisation Access Control	S.Resident application and for all objects
Patch & Locks Loading Access Control	S.TOE and for all objects

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

*Application note:*

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

**FDP\_ACF.1/PP Security attribute based access control**

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

Subject	Attribute
R.Prepersonaliser	AS_AUTH_MSK_STATUS

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

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

**FDP\_ACF.1.3/PP** The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

**FDP\_ACF.1.4/PP** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application note:* the integrity of the memory area to be modified is verified, signature SHA256 before and after patch is loaded.

**FDP\_UCT.1/PP Basic data exchange confidentiality**

**FDP\_UCT.1.1/PP** The TSF shall enforce the **Prepersonalisation access control and Patch and Locks loading access control** to **receive** user data in a manner protected from unauthorised disclosure.

*Application note:*



For the Prepersonalisation access control, the MSK is used to cipher the data transmitted (ISK). For the Patch and Locks loading access control, the LSK is used to cipher the data transmitted.

**FDP\_ITC.1/PP Import of user data without security attributes**

**FDP\_ITC.1.1/PP** The TSF shall enforce the **Prepersonalisation access control and Patch and Locks loading access control** when importing user data, controlled under the SFP, from outside of the TOE.

**FDP\_ITC.1.2/PP** The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

**FDP\_ITC.1.3/PP** The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **none**.

**FIA\_AFL.1/PP Authentication failure handling**

**FIA\_AFL.1.1/PP** The TSF shall detect when **3** unsuccessful authentication attempts occur related to **U.Card\_manufacturer authentication**.

**FIA\_AFL.1.2/PP** When the defined number of unsuccessful authentication attempts has been met, the TSF shall **always return an error**.

**FIA\_UAU.1/PP Timing of authentication**

**FIA\_UAU.1.1/PP** The TSF shall allow **INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, SELECT APPLET** on behalf of the user to be performed before the user is authenticated.

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

**FIA\_UID.1/PP Timing of identification**

**FIA\_UID.1.1/PP** The TSF shall allow **INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, and SELECT APPLET** on behalf of the user to be performed before the user is identified.

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

**FMT\_MSA.1/PP Management of security attributes**

**FMT\_MSA.1.1/PP** The TSF shall enforce the **Prepersonalisation access control** to restrict the ability to **modify** the security attributes **AS.AUTH\_MSK\_STATUS**, **MSK Key value**, **MSK Key counter** to **R.Prepersonaliser**.

**FMT\_SMF.1/PP Specification of Management Functions**

**FMT\_SMF.1.1/PP** The TSF shall be capable of performing the following management functions: **modify the MSK keys of the Card Manufacturer in Prepersonalisation phase after a successful authentication with the MSK.**

**FIA\_UAU.4/CardManu Single-use authentication mechanisms**

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

**FIA\_UAU.7/CardManu Protected authentication feedback**

**FIA\_UAU.7.1/CardManu** The TSF shall provide only **the result of the authentication (NOK) and the random** to the user while the authentication is in progress.

**FMT\_MOF.1/PP Management of security functions behaviour**

**FMT\_MOF.1.1/PP** The TSF shall restrict the ability to **see below** the functions **see below** to:

	Functions	Role
Disable	INITIALIZE AUTHENTICATION PROCESS, EXTERNAL AUTHENTICATE, INSTALL, UPDATE SECURE, LOAD APPLET, GET DATA	R.Prepersonaliser
Modify the behaviour of	Self-tests described in FPT_TST.1	R.Prepersonaliser

*Application note:*

The first operation ensures the irreversible locking of the patch and locks loading features once in OP\_READY, after pre production state. Once in OP\_READY state, those APDU cannot be used. The second operation described the product configuration regarding self tests, as described in AGD\_PRE, chapter 8 [R39].

The last operation permits the loading of patch and locks during phase 5.



**FMT\_SMR.2/PP Restrictions on security roles**

**FMT\_SMR.2.1/PP** The TSF shall maintain the roles: **R.Prepersonaliser**.

**FMT\_SMR.2.2/PP** The TSF shall be able to associate users with roles.

**FMT\_SMR.2.3/PP** The TSF shall ensure that the conditions **see refinement below** are satisfied.

*Refinement:*

Roles	Condition for this role
R.Prepersonaliser	Successful authentication (of Card Manufacturer) using MSK and card still in prepersonalisation state, in phase 4-5.

**FMT\_MSA.3/PP Static attribute initialisation**

**FMT\_MSA.3.1/PP** The TSF shall enforce the **Prepersonalisation access control** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

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

**FCS\_COP.1/PP Cryptographic operation**

**FCS\_COP.1.1/PP** The TSF shall perform **see table below** in accordance with a specified cryptographic algorithm **see table below** and cryptographic key sizes **see table below** that



meet the following:

Cryptographic operation	Algorithm	Key length	Standard
Decryption (MSK)	DES	112 bits	FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797, Data integrity mechanism
Card Manufacturer authentication (MSK)	DES	112 bits	FIPS PUB 197
Card Manufacturer authentication (MSK)	AES	128, 192 and 256 bits	FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797, Data integrity mechanism
Decryption (of patch and locks ciphered with LSK)	AES	128 bits	FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797
ISK MAC verification	DES	112 bits	FIPS PUB 197

#### FCS\_CKM.4/PP Cryptographic key destruction

**FCS\_CKM.4.1/PP** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **Key is set to NULL** that meets the following: **no**.

*Application Note:*

In phase 5, reaching OP\_READY state, this SFR ensures the secure erasing of the LSK and MSK keys.

In phases 3-4, MSK is diversified during the first command, and then replaced by the new value generated by FCS\_CKM.1/PP.

#### FDP\_UIT.1/PP Data exchange integrity

**FDP\_UIT.1.1/PP** The TSF shall enforce the **Patch and Prepersonalisation loading access control SFP to receive** user data in a manner protected from **modification** errors.

**FDP\_UIT.1.2/PP [Editorially Refined]** The TSF shall be able to determine on receipt of user data, whether **modification of some of the pieces of the application sent by the TOE developer and Card Manufacturer** has occurred.

*Application Note:*

Modification errors should be understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the patch or the locks to be installed on the card to be different from the one sent by the TOE Developer. The locks loading is performed by the TOE Prepersonaliser via the command UPDATE SECURE described in



FCS\_COP.1/PP. The ISK loading is performed by the Card Manufacturer via the command PUT KEY, its integrity is ensured by a MAC, described in FCS\_COP.1/PP.

**FCS\_CKM.1/PP Cryptographic key generation**

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

Cryptographic key generation algorithm	Cryptographic key size	List of standards
TOE's MSK derived from the MSK loaded in phase 1, using SHA-256	16, 24 and 32 bytes	None

*Application Note:*

Key derivation algorithm is detailed in AGD\_PRE, §5 [R39].

**FAU\_STG.2 Guarantees of audit data availability**

**FAU\_STG.2.1** The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

**FAU\_STG.2.2** The TSF shall be able to **prevent** unauthorized modifications to the stored audit records in the audit trail.

**FAU\_STG.2.3** The TSF shall ensure that **Patch code identification** stored audit records will be maintained when the following conditions occur: **audit storage exhaustion, failure and attack**.

*Application Note:*

Patch loading steps are:

1. Loading CAP file containing the code in EEPROM (install and load commands)
2. Decryption with LSK key
3. Check the signature (SHA256) of the area to be patched if signature is OK
4. Write the new code (patch itself)
5. Computation of the new SHA signature
6. Deletion of the CAP file stored temporary in EEPROM (DELETE command)

Information on the Patch code (unique identifier) is directly retrieved by GET DATA command.



## 7.1.5.3 Additional Security Functional Requirements for SmartCard Platform

**FPT\_PHP.3/SCP Resistance to physical attack**

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

*Application Note:*

The physical manipulation and physical probing include: changing operational conditions every times: the frequency of the external clock, power supply, and temperature

**FPT\_RCV.4/SCP Function recovery**

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

**FRU\_FLT.1/SCP Degraded fault tolerance**

**FRU\_FLT.1.1/SCP** The TSF shall ensure the operation of **Fault tolerance** when the following failures occur: **failure of flash**.

*Application Note:*

The TOE implements a mechanism to detect a problem of Flash. During the life of the TOE, the Transaction area reduces its size to skip damaged FLASH bytes. During the writing or erasing operations, up to 3 maximum attempts to get successful programming are done. Otherwise the EXCEPTION\_EEPROM\_ERROR is raised.

**FPR\_UNO.1/USE\_KEY Unobservability**

**FPR\_UNO.1.1/USE\_KEY** The TSF shall ensure that **all subjects** are unable to observe the operation **use on key** by **D.JCS\_KEYS** and **APP\_KEYS**.

**FCS\_RNG.1/SCP Random Number Generation**

**FCS\_RNG.1.1/SCP** The TSF shall provide a **deterministic hybrid** random number generator that implements: **CTR\_DRBG as defined in NIST SP800-90**.

**FCS\_RNG.1.2/SCP** The TSF shall provide random numbers that meet **The average Shannon**



**entropy per internal random bit exceeds 0.999.**

7.1.5.4 Additional Security Functional Requirements for the applets

**FIA\_AFL.1/PIN Authentication failure handling**

**FIA\_AFL.1.1/PIN** The TSF shall detect when **an administrator configurable positive integer within from 1 to 127 for OwnerPIN** unsuccessful authentication attempts occur related to **any user authentication using a PIN.**

**FIA\_AFL.1.2/PIN** When the defined number of unsuccessful authentication attempts has been met, the TSF shall **block the PIN.**

**FMT\_MTD.2/GP\_PIN Management of limits on TSF data**

**FMT\_MTD.2.1/GP\_PIN** The TSF shall restrict the specification of the limits for **D.NB\_REMAINTRYGLB, GlobalPIN to R.Card\_Manager.**

**FMT\_MTD.2.2/GP\_PIN** The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: **block D.PIN.**

R.Card_Manager	Entity after Successful authentication (of Card Issuer) using its key set, with CM life cycle phase from OP_READY to SECURED
----------------	--

**FMT\_MTD.1/PIN Management of TSF data**

**FMT\_MTD.1.1/PIN** The TSF shall restrict the ability to **change\_default, query and modify** the **OwnerPIN to applet itself.**

**FIA\_AFL.1/GP\_PIN Authentication failure handling**

**FIA\_AFL.1.1/GP\_PIN** The TSF shall detect when **an administrator configurable positive integer within 3 to 15** unsuccessful authentication attempts occur related to **any user authentication using a Global PIN.**

**FIA\_AFL.1.2/GP\_PIN** When the defined number of unsuccessful authentication attempts has been met, the TSF shall **block the Global PIN.**



## 7.1.5.5 Additional Security Functional Requirements for Runtime Verification

## Stack Control

**FDP\_ACC.2/RV\_Stack Complete access control**

**FDP\_ACC.2.1/RV\_Stack** The TSF shall enforce the **Stack Access Control SFP** on **S.STACK** and all operations among subjects and objects covered by the SFP.

*Refinement:*

The operations involved in the policy are:

- o OP.OPERAND\_STACK\_ACCESS
- o OP.LOCAL\_STACK\_ACCESS

**FDP\_ACC.2.2/RV\_Stack** The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

**FDP\_ACF.1/RV\_Stack Security attribute based access control**

**FDP\_ACF.1.1/RV\_Stack** The TSF shall enforce the **Stack Access Control** to objects based on the following:

Subject/Object	Security attributes
S.APPLLET	Active Applets, Applet Selection Status
S.STACK	Stack Pointer
S.JCVM	Current Frame Context

**FDP\_ACF.1.2/RV\_Stack** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- o **An Active Applet selected may freely perform OP.LOCAL\_STACK\_ACCESS upon stack pointer only if the index of the local variable accessed matches the Current Frame Context attribute**
- o **An Active Applet selected may freely perform OP.OPERAND\_STACK\_ACCESS upon Stack Pointer only if the attribute Stack Pointer matches the attribute Current Frame Context of S.JCVM.**

**FDP\_ACF.1.3/RV\_Stack** The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

**FDP\_ACF.1.4/RV\_Stack** The TSF shall explicitly deny access of subjects to objects based on



the following additional rules: **none**.

*Application Note:*

Any bytecode accessing a local variable has an index in parameter (byte or short). The first rule aims at verifying that this index is always positive and inferior to the numbers of local variables defined for this stack frame. Then the local variable slot is accessed using the index that is relative to the base of local variables for this stack frame.

Any bytecode accessing the operand stack for push or pop operations is under the control of rule 2. The second rule aims at verifying that the stack pointer is always in the range defined by the base-of-stack and top-of-stack values defined for this stack frame.

The frame context attribute is made of the following elements:

- number-of-local variables and base-of-local-variable
- base-of-stack and top-of-stack

The policies defined in this SFR are enforced dynamically, each time an operation is performed. Nevertheless, those verifications may be redundant with the ones made statically by the off-card verifier, during the applet verification stage.

#### FMT\_MSA.1/RV\_Stack Management of security attributes

**FMT\_MSA.1.1/RV\_Stack** The TSF shall enforce the **Stack Access Control SFP** to restrict the ability to **modify** the security attributes **Current Frame Context and Stack Pointer** to the Java Card VM (S.JCVM).

#### FMT\_MSA.2/RV\_Stack Secure security attributes

**FMT\_MSA.2.1/RV\_Stack** The TSF shall ensure that only secure values are accepted for **Current Frame Context and Stack Pointer**.

#### FMT\_MSA.3/RV\_Stack Static attribute initialisation

**FMT\_MSA.3.1/RV\_Stack** The TSF shall enforce the **Stack Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

**FMT\_MSA.3.2/RV\_Stack** The TSF shall allow the **any role** to specify alternative initial values to override the default values when an object or information is created.

#### FMT\_SMF.1/RV\_Stack Specification of Management Functions



**FMT\_SMF.1.1/RV\_Stack** The TSF shall be capable of performing the following management functions: **Modify the Current Frame Context and modify the Stack Pointer.**

*Application Note:*

The frame context attribute is modified on method invocation. In that case, the previous context attribute is saved on the stack. It will be restored on return of the invoked method.

Heap Access

**FDP\_ACC.2/RV\_Heap Complete access control**

**FDP\_ACC.2.1/RV\_Heap** The TSF shall enforce the **Heap Access Control SFP** on **O.CODE\_PKG, O.JAVAOBJECT, S.JCVM, S.APPLET** and all operations among subjects and objects covered by the SFP.

*Refinement:*

The operations involved in the policy are:

- o OP.ARRAY\_ACCESS
- o OP.INSTANCE\_FIELD
- o OP.STATIC\_FIELD
- o OP.FLOW

**FDP\_ACC.2.2/RV\_Heap** The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

**FDP\_ACF.1/RV\_Heap Security attribute based access control**

**FDP\_ACF.1.1/RV\_Heap** The TSF shall enforce the **Heap Access Control SFP** to objects based on the following:

Subject/Object	Security attributes
O.CODE_PKG	Package Boundary
O.JAVAOBJECT	Object Boundary
S.JCVM	Program Counter
S.APPLET	Active Applets, Applet Selection Status

**FDP\_ACF.1.2/RV\_Heap** The TSF shall enforce the following rules to determine if an operation



among controlled subjects and controlled objects is allowed:

- **S.APPLET** may freely perform **OP.ARRAY\_ACCESS** and **OP.INSTANCE\_FIELD** upon any **O.JAVAOBJECT** if the array cell index or the instance field index match the object boundary attribute of **O.JAVAOBJECT**
- **S.APPLET** may freely perform **OP.STATIC\_FIELD** upon any **O.CODE\_PKG** if the static field index matches the Package Boundary attribute of **O.CODE\_PKG**.
- **S.APPLET** may freely perform **OP.FLOW** upon **O.CODE\_PKG** if the Program Counter attribute of **S.JCVM** matches the Package Boundary attribute of **O.CODE\_PKG**.

**FDP\_ACF.1.3/RV\_Heap** The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

**FDP\_ACF.1.4/RV\_Heap** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application Note:*

The upper and lower boundaries of any object allocated on the heap are registered (Object Boundary Attribute). Each time an object is accessed, the first rule verifies that the accessed NVM location is comprised between those two boundaries.

The second rule aims at verifying that when a static field is accessed, the index of this field is positive and inferior to the number of static fields of this package (part of Package Boundary attribute).

The third rule aims at verifying that when a change of execution flow occurs, the computed value for the newly computed value for the Program Counter is comprised within the boundaries defined for this package (part of Package Boundary Attribute). This rule does not concern invocation bytecode.

The policies defined in this SFR are enforced dynamically, each time an operation is performed. Nevertheless, those verifications may be redundant with the ones made statically by the off-card verifier, during the applet verification stage.

#### **FMT\_MSA.1/RV\_Heap Management of security attributes**

**FMT\_MSA.1.1/RV\_Heap** The TSF shall enforce the **Heap Access Control SFP** to restrict the ability to **modify** the security attributes **Package Boundary, Object Boundary and Program Counter** to **S.JCVM**.

#### **FMT\_MSA.2/RV\_Heap Secure security attributes**

**FMT\_MSA.2.1/RV\_Heap** The TSF shall ensure that only secure values are accepted for

|            |            |            |            |

Package Boundary, Object Boundary and Program Counter.

**FMT\_MSA.3/RV\_Heap Static attribute initialisation**

**FMT\_MSA.3.1/RV\_Heap** The TSF shall enforce the **Heap Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

**FMT\_MSA.3.2/RV\_Heap** The TSF shall allow the **no role** to specify alternative initial values to override the default values when an object or information is created.

**FMT\_SMF.1/RV\_Heap Specification of Management Functions**

**FMT\_SMF.1.1/RV\_Heap** The TSF shall be capable of performing the following management functions: **to modify the Program Counter attribute.**

Transient Control

**FDP\_ACC.2/RV\_Transient Complete access control**

**FDP\_ACC.2.1/RV\_Transient** The TSF shall enforce the **Transient Access Control SFP** on **S.APPLET, S.JCVM and O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

*Refinement:*

The operation involved in the policy is:

- o OP.ARRAY\_ACCESS

**FDP\_ACC.2.2/RV\_Transient** The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

**FDP\_ACF.1/RV\_Transient Security attribute based access control**

**FDP\_ACF.1.1/RV\_Transient** The TSF shall enforce the **Transient Access Control SFP** to objects based on the following:

Subject/Object	Security Attributes
S.APPLET	Active Applets, Applet Selection Status
S.JCVM	COR Context, COD Context
O.JAVAOBJECT	LifeTime



**FDP\_ACF.1.2/RV\_Transient** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **S.APPLLET** may freely perform **OP.ARRAY\_ACCESS** on **O.JAVAOBJECT** whose **LifeTime** attribute has value "**CLEAR\_ON\_RESET**" only if the targeted volatile memory space matches the **COR Context** attribute of **S.JCVM**
- **S.APPLLET** may freely perform **OP.ARRAY\_ACCESS** on **O.JAVAOBJECT** whose **LifeTime** attribute has value "**CLEAR\_ON\_DESELECT**" only if the targeted volatile memory space matches the **COD Context** attribute of **S.JCVM**.

**FDP\_ACF.1.3/RV\_Transient** The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

**FDP\_ACF.1.4/RV\_Transient** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application Note:*

Each time an applet accesses a Clear On Reset (resp. Clear On Deselect) transient, these rules verify that the accessed RAM area is in the range of the Clear On Reset transients space (resp. Clear On Deselect) allocated for all the transients created by the applets of this package.

The **COR** context attribute represents the lower and upper limits for the Clear On Reset transient space of the active applet package. The **COD** context attribute represents the lower and upper limits for the Clear On Deselect transient space of the currently selected applet package.

The policies defined in this SFR are enforced dynamically, each time an operation is performed. Nevertheless, those verifications may be redundant with the ones made statically by the off-card verifier, during the applet verification stage.

**FMT\_MSA.1/RV\_Transient Management of security attributes**

**FMT\_MSA.1.1/RV\_Transient** The TSF shall enforce the **Transient Access Control SFP** to restrict the ability to **modify** the security attributes **the security attributes COR Context and COD Context to Java Card VM (S.JCVM)**.

**FMT\_MSA.2/RV\_Transient Secure security attributes**

**FMT\_MSA.2.1/RV\_Transient** The TSF shall ensure that only secure values are accepted for **COR Context and COD Context Security attributes of the Transient Access Control SFP**.

**FMT\_MSA.3/RV\_Transient Static attribute initialisation**

| } } } }

**FMT\_MSA.3.1/RV\_Transient** The TSF shall enforce the **Transient Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

**FMT\_MSA.3.2/RV\_Transient** The TSF shall allow the **no role** to specify alternative initial values to override the default values when an object or information is created.

**FMT\_SMF.1/RV\_Transient Specification of Management Functions**

**FMT\_SMF.1.1/RV\_Transient** The TSF shall be capable of performing the following management functions: **modify the COR Context and COD Context Security Attributes.**

**7.2 Security Assurance Requirements**

The Evaluation Assurance Level is EAL5 augmented with AVA\_VAN.5 and ALC\_DVS.2.

**7.3 Security Requirements Rationale**

7.3.1 Objectives

7.3.1.1 Security Objectives for the TOE

**IDENTIFICATION**

**O.SID** Subjects' identity is AID-based (applets, packages), and is met by the following SFRs: FDP\_ITC.2/Installer, FIA\_ATD.1/AID, FMT\_MSA.1/JCRE, FMT\_MSA.1/JCVM, FMT\_MSA.1/REM\_REFS,, FMT\_MSA.1/ADEL, FMT\_MSA.1/CM, FMT\_MSA.3/ADEL, FMT\_MSA.3/FIREWALL, FMT\_MSA.3/JCVM, FMT\_MSA.3/CM, FMT\_SMF.1/CM, FMT\_SMF.1/ADEL, FMT\_MTD.1/JCRE and FMT\_MTD.3/JCRE.

Lastly, installation procedures ensure protection against forgery (the AID of an applet is under the control of the TSFs) or re-use of identities (FIA\_UID.2/AID, FIA\_USB.1/AID).

**EXECUTION**

**O.FIREWALL** This objective is met by the FIREWALL access control policy FDP\_ACC.2/FIREWALL and FDP\_ACF.1/FIREWALL, the JCVM information flow control policy (FDP\_IFF.1/JCVM, FDP\_IFC.1/JCVM), the functional requirement FDP\_ITC.2/Installer. The functional requirements of the class FMT (FMT\_MTD.1/JCRE, FMT\_MTD.3/JCRE, FMT\_SMR.1/Installer, FMT\_SMR.1, FMT\_SMF.1, FMT\_SMR.1/ADEL, FMT\_SMF.1/ADEL, FMT\_SMF.1/CM, FMT\_MSA.1/CM, FMT\_MSA.3/CM, FMT\_SMR.1/CM, FMT\_MSA.2/FIREWALL\_JCVM, FMT\_MSA.3/FIREWALL, FMT\_MSA.3/JCVM, FMT\_MSA.1/ADEL, FMT\_MSA.3/ADEL, S, FMT\_MSA.1/JCRE, FMT\_MSA.1/JCVM,) also indirectly contribute to meet this objective.

This objective is also covered by the following additional SFRs:

- Stack control (\* /RV\_Stack): FDP\_ACC.2/RV\_Stack, FDP\_ACF.1/RV\_Stack, FMT\_MSA.1/RV\_Stack, FMT\_MSA.2/RV\_Stack, FMT\_MSA.3/RV\_Stack,



FMT\_SMF.1/RV\_Stack

- Heap control (\*RV\_Heap): FDP\_ACC.2/RV\_Heap, FDP\_ACF.1/RV\_Heap, FMT\_MSA.1/RV\_Heap, FMT\_MSA.2/RV\_Heap, FMT\_MSA.3/RV\_Heap, FMT\_SMF.1/RV\_Heap

- Transient control (\*RV\_Transient): FDP\_ACC.2/RV\_Transient, FDP\_ACF.1/RV\_Transient, FMT\_MSA.1/RV\_Transient, FMT\_MSA.2/RV\_Transient, FMT\_MSA.3/RV\_Transient, FMT\_SMF.1/RV\_Transient

For each of those control, the SFR define the access control (FDP\_ACC and FDP\_ACF), the operation (FMT\_MSA) and the role (FMT\_SMF).

The Stack control enforces O.FIREWALL by defining additional rules, such as the control of the stack is more precise. Information is provided in the application note.

The Heap control enforces O.FIREWALL by defining additional rules, such as the heap usage is improved. Information is provided in the application note.

The Transient enforces O.FIREWALL by defining additional rules, such as the heap usage is improved. Information is provided in the application note.

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

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

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

**O.OPERATE** The TOE is protected in various ways against applets' actions (FPT\_TDC.1), the FIREWALL access control policy FDP\_ACC.2/FIREWALL and FDP\_ACF.1/FIREWALL, and is able to detect and block various failures or security violations during usual working (FPT\_FLS.1/ADEL, FPT\_FLS.1, FPT\_FLS.1/ODEL, FPT\_FLS.1/Installer, FAU\_ARP.1). Its security-critical parts and procedures are also protected: safe recovery from failure is ensured (FPT\_RCV.3/Installer), applets' installation may be cleanly aborted (FDP\_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FDP\_ITC.2/Installer, FIA\_ATD.1/AID, FIA\_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).

The FPT\_RCV.4/SCP contributes to the objective O.OPERATE as it ensures that when reading or writing operations are interrupted by power loss, the operation either is completed or recovers in a consistent and secure state.



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

*Application note:* Startup of the TOE (TSF-testing) can be covered by FPT\_TST.1. This SFR component is not mandatory in [R7], but appears in most of security requirements documents for masked applications. Testing could also occur randomly. Self-tests may become mandatory in order to comply with FIPS certification [FIPS 140-2].

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

**O.RESOURCES** The TSFs detects stack/memory overflows during execution of applications (FAU\_ARP.1, FPT\_FLS.1/ADEL, FPT\_FLS.1, FPT\_FLS.1/ODEL, FPT\_FLS.1/Installer). Failed installations are not to create memory leaks (FDP\_ROL.1/FIREWALL, FPT\_RCV.3/Installer) as well. Memory management is controlled by the TSF (FMT\_MTD.1/JCRE, FMT\_MTD.3/JCRE, FMT\_SMR.1/Installer, FMT\_SMR.1, FMT\_SMF.1 FMT\_SMR.1/ADEL, FMT\_SMF.1/ADEL, FMT\_SMF.1/CM and FMT\_SMR.1/CM).

## SERVICES

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

**O.CIPHER** This security objective is directly covered by FCS\_CKM.1, FCS\_CKM.2, FCS\_CKM.3, FCS\_CKM.4, FCS\_COP.1 and FCS\_COP.1/PP. FPR\_UNO.1 and FPR\_UNO.1/USE\_KEY contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys.

**O.KEY-MNGT** This relies on the same security functional requirements as O.CIPHER, plus FDP\_RIP.1 and FDP\_SDI.2 as well. Precisely it is met by the following components: FCS\_CKM.1, FCS\_CKM.2, FCS\_CKM.3, FCS\_CKM.4, FCS\_COP.1, FCS\_COP.1/PP, FPR\_UNO.1, FPR\_UNO.1/USE\_KEY, FDP\_RIP.1/ODEL, FDP\_RIP.1/OBJECTS, FDP\_RIP.1/APDU, FDP\_RIP.1/bArray, FDP\_RIP.1/ABORT, FDP\_RIP.1/KEYS, FDP\_RIP.1/ADEL and FDP\_RIP.1/TRANSIENT.

**O.PIN-MNGT** This security objective is ensured by FDP\_RIP.1/ODEL, FDP\_RIP.1/OBJECTS, FDP\_RIP.1/APDU, FDP\_RIP.1/bArray, FDP\_RIP.1/ABORT, FDP\_RIP.1/KEYS, FDP\_RIP.1/ADEL, FDP\_RIP.1/TRANSIENT, FPR\_UNO.1, FDP\_ROL.1/FIREWALL and FDP\_SDI.2 security functional requirements. The TSFs behind these are implemented by API classes. The firewall security functions FDP\_ACC.2/FIREWALL and FDP\_ACF.1/FIREWALL shall protect the access to private and internal data of the objects. FIA\_AFL1.1/CM, FIA\_AFL.1/PIN and FIA\_AFL.1/GP\_PIN ensure the objective regarding authentications failures. FMT\_MTD.1/PIN ensures the objective regarding the management



of the TSF data.

**O.TRANSACTION** Directly met by FDP\_ROL.1/FIREWALL, FDP\_RIP.1/ABORT, FDP\_RIP.1/ODEL, FDP\_RIP.1/APDU, FDP\_RIP.1/bArray, FDP\_RIP.1/KEYS, FDP\_RIP.1/ADEL, FDP\_RIP.1/TRANSIENT and FDP\_RIP.1/OBJECTS (more precisely, by the element FDP\_RIP.1.1/ABORT).

#### OBJECT DELETION

**O.OBJ-DELETION** This security objective specifies that deletion of objects is secure. The security objective is met by the security functional requirements FDP\_RIP.1/ODEL and FPT\_FLS.1/ODEL.

#### APPLET MANAGEMENT

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

**O.LOAD** This security objective specifies that the loading of a package into the card must be secure. Evidence of the origin of the package is enforced (FCO\_NRO.2/CM) and the integrity of the corresponding data is under the control of the PACKAGE LOADING information flow policy (FDP\_IFC.2/CM, FDP\_IFF.1/CM) and FDP\_UIT.1/CM. Appropriate identification (FIA\_UID.1/CM) and transmission mechanisms are also enforced (FTP\_ITC.1/CM).

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

#### Additional security objectives for the TOE

**O.SCP.SUPPORT** The components FPT\_RCV.4/SCP (SCP stands for smart card platform) are used to support the objective O.SCP.SUPPORT to assist the TOE to recover in the event of a power failure. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be interrupted leaving the TOE in an inconsistent state.

All the Crypto SFRS support this objective as they provide secure low-level cryptographic processing to the Java Card System and Global Platform:

- FCS\_CKM.1, FCS\_CKM.2, FCS\_CKM.3, FCS\_CKM.4, FCS\_COP.1,
- FCS\_COP.1/CM,
- FCS\_CKM.1/PP, FCS\_COP.1/PP, FCS\_CKM.4/PP



All the FSRs related to the Firewall contribute to the realization of the objective.

The FDP\_ROL.1 Firewall ensures the rollback of some operations within the specified scope as defined in the ROL.1.2/Firewall.

Application Note: all SFRs related to O.OPERATE and O.ALARM support the O.SCP.SUPPORT

**O.SCP.IC** This objective is met by the component FPT\_PHP.3/SCP.

**O.SCP.RECOVERY** The component FPT\_RCV.4/SCP is used to support the objective O.SCP.RECOVERY to assist the TOE to recover in the event of a power failure. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be interrupted leaving the TOE in an inconsistent state. This objective is met by the components FPT\_FLS.1, FAU\_ARP.1 and FRU\_FLT.1/SCP.

**O.RESIDENT\_APPLICATION** This objective is covered by the following set of SFR:

- o Access control: FDP\_ACC.2/PP, FDP\_ACF.1/PP, FDP\_UCT.1/PP and FDP\_ITC.1/PP
- o Rules for authentication: FIA\_AFL.1/PP, FIA\_UAU.1/PP
- o Security Management: FMT\_MSA.1/PP, FMT\_SMF.1/PP, FMT\_MOF.1/PP, FMT\_SMR.2/PP, FMT\_MSA.3/PP and FMT\_SMR.2/CM
- o Once the ISK is loaded, the RA ensures MSK and LSK are no more available thanks to Cryptographic Key Destruction: FCS\_CKM.4/PP.

This objective is also covered by Card Manufacturer authentication: Rules for authentication: FIA\_UAU.7/CardIssuer, FIA\_UAU.4/CardIssuer, FIA\_UAU.4/CardManu, FIA\_UAU.7/CardManu.

**O.CARD\_MANAGEMENT** This objective is fulfilled by the following set of SFR:

The FDP\_ACC.2/ADEL and FDP\_ACF.1/ADEL contribute to meet the ADEL access control policy that ensures the non-introduction of security holes.

The FDP\_RIP.1/ADEL ensure that the deleted information is not accessible.

The FMT\_MSA.1/ADEL ensures restrict the ability to modify the secure attributes the FMT\_MSA.3/ADEL ensures the assignment of restrictive values.

The FMT\_SMR.1/ADEL maintains the role of the applet deletion manager.

The FPT\_FLS.1/ADEL contributes to the objective by protecting the TSFs against possible failures of the deletion procedure.

The 2 SFRs FPT\_RCV.3/Installer and FPT\_FLS.1/Installer contributes to meet the objective by protecting the TSFs from failures of the deletion procedure.

The SFR FDP\_UIT.1/CM contributes by enforcing the Secure Channel Protocol Information flow control policy and the Security Domain access control policy which control the integrity of the corresponding data. The SFR FIA\_UID.1/CM testes if the Secure Channel is open to allow card management operations.

The SFR FDP\_IFF.1/CM ensures the access control policy for the loaded data (as packages).

The FCO\_NRO.2/CM this SFR ensures the origin of the load file. It verifies the identity of the



origin of the load file before start the loading.

FCO\_NRO.2/CM\_DAP this SFR generates an evidence of the origin of the transmitted load file during CAP File loading.

The FDP\_IFC.2/CM, this SFR ensures that loading commands are issued in the Secure Channel session.

The SFR FDP\_ROL.1/FIREWALL ensures that the card management operations are cleaned aborted.

The SFR FDP\_ITC.2/Installer enforces the Firewall access control policy and flow control policy when importing card management data.

The SFR FPT\_FLS.1/ODEL ensures the preservation of secure state when failures occur.  
The SFR FMT\_MSA.1/CM ensures the management of the security attributes to the card manager, for the modification of the life cycle of the card, the keyset version and value,...

The SFR FMT\_MSA.3/CM, this SFR ensures that the security attributes can only be changed by the card manager.

The SFR FMT\_SMF.1/CM, Only the card manger is able to modify the security attributes of the management functions. The security role is specified in the FMT\_SMR.1/CM and FMT\_SMR.2/CM.

The SFR FTP\_ITC.1/CM ensures the trusted Channel Communications.  
FIA\_AFL.1/CM, FIA\_UAU.1/CM, FIA\_UAU.4/CardIssuer and FIA\_UAU.7/CardIssuer, ensure the authentication of the card issuer before gaining access to management operations.  
The FPR\_UNO.1/Key\_CM ensures the un-observability of the CM key when imported..

**The FPT\_TST.1**This TSF contributes to ensure the correct operation of the card management functions as it tests the integrity of the TSF functions during initial start-up.

**O.SECURE\_COMPARE** This objective is fulfilled by FDP\_SDI.2/Applet. It ensures that comparison is confidential.

## O.PATCH\_LOADING

### Authentication of the entity loading the patch by the TOE

FDP\_ACC.2/PP, FDP\_ACF.1/PP, FIA\_UAU.1/PP and FIA\_UID.1/PP provide access control for patch loading. The subject entitled to load the patch – the card manufacturer - is authenticated by the TOE thanks to FCS\_COP.1/PP. Wrong authentication of the Card manufacturer agent are detected thanks to FIA\_AFL.1/PP

### Authentication of the TOE

To avoid impersonation of the TOE by a fake chip, the TOE authenticates itself ; from phase 6 (after patch loading) with FTP\_ITC.1/CM and FCS\_COP.1/CM thanks to the TOE authentication key (ISK/KMC). From phase 6, the TOE authentication is required prior to any trusted channel establishment with FTP\_ITC.1/CM (data sent by the TOE must be



decrypted to carry on the authentication).

The TOE authentication key (ISK/KMC) is securely loaded in phase 4/5 protected in confidentiality with FDP\_UCT.1/PP and integrity with FDP\_UIT.1/PP through the trusted channel established by the Card Manufacturer with FDP\_ITC.1/PP. The trusted channel and the TOE authentication key (ISK/KMC) encryption is supported by FCS\_COP.1/PP that relies on the TOE's MSK which is the first key present in the TOE.

#### **Diversification of keys**

The TOE's MSK used to authenticate the Card manufacturer is derived from the MSK thanks to FCS\_CKM.1/PP before the first use. The MSK is loaded in the TOE in phase 1 (covered by [ALC]).

#### **Integrity, confidentiality and authenticity of the patch during loading**

Patch loading is performed in a confidential manner with FDP\_UCT.1/PP and protected in integrity and confidentiality with FDP\_UIT.1/PP. Confidentiality, integrity and authenticity of the patch loading is supported by cryptographic mechanisms supported by FCS\_COP.1/PP. Patch data to be written in the TOE have been prior encrypted by the TOE developer using LSK key. Once these data loaded, the integrity (SHA256) of the modified code is update and compared to the provided one in the patch package.

#### **Irreversible locking of the patch loading features**

The patch can be loaded in phase 4 and 5 of the TOE's life cycle. At the end of phase 5, FMT\_MOF.1/PP ensures the irreversible locking once in OP\_READY, after pre-production state. Once in OP\_READY state, those APDU cannot be used.

#### **Erasure of the key used**

FCS\_CKM.4/PP ensures the secure destruction of the keys involved in the patch loading mechanism (LSK and MSK) at the end of phase 5.

#### **Identification of the patch after loading**

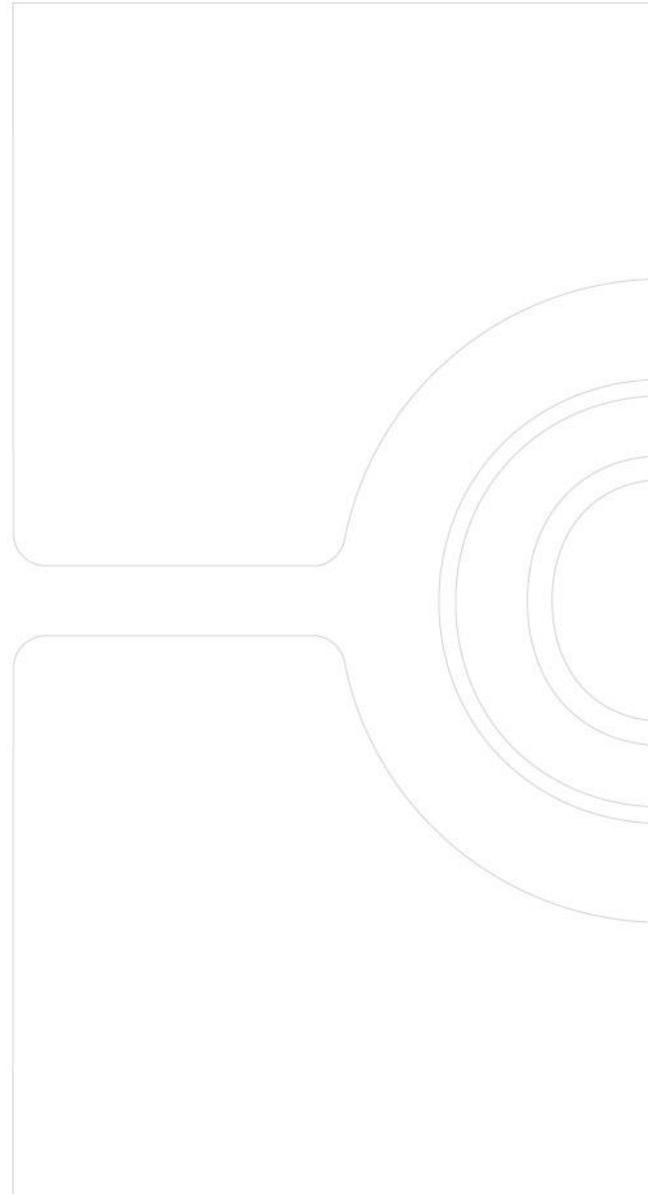
Once loaded and during the rest of the TOE life cycle, the identification and authentication (unique identifier of the patch) of the patch, being a part of the TOE is provided by FAU\_STG.2. When requested, the identification and authentication data (of entire code, including patch) are dynamically retrieved from the patch code stored in the non-volatile memory of the TOE.

#### **Integrity check before usage of the patch**

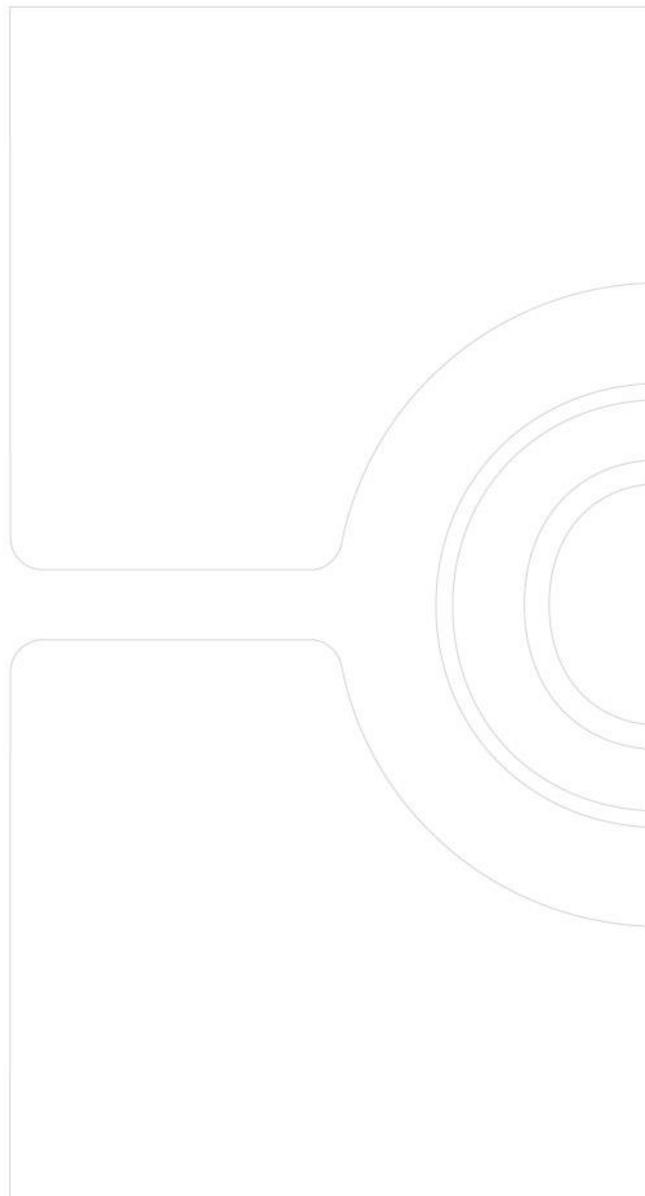
At start up, the integrity of the entire code, patch included, is checked by the TOE through self-tests provided by FPT\_TST.1. In case the computed signature differs from the one



stored in NVM, an integrity error is detected and a killcard is raised.



## 7.3.2 Rationale tables of Security Objectives and SFRs



Security Objectives	Security Functional Requirements	Rationale
<a href="#">O.SID</a>	<a href="#">FIA_ATD.1/AID</a> , <a href="#">FIA_UID.2/AID</a> , <a href="#">FMT_MSA.1/JCRE</a> , , <a href="#">FMT_MSA.1/REM_REFS</a> , <a href="#">FMT_MSA.1/ADEL</a> , <a href="#">FMT_MSA.3/ADEL</a> , <a href="#">FMT_MSA.3/FIREWALL</a> , <a href="#">FMT_MSA.1/CM</a> , <a href="#">FMT_MSA.3/CM</a> , <a href="#">FDP_ITC.2/Installer</a> , <a href="#">FMT_SMF.1/CM</a> , <a href="#">FMT_SMF.1/ADEL</a> , , <a href="#">FMT_MTD.1/JCRE</a> , <a href="#">FMT_MTD.3/JCRE</a> , <a href="#">FIA_USB.1/AID</a> , <a href="#">FMT_MSA.1/JCVM</a> , <a href="#">FMT_MSA.3/JCVM</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.FIREWALL</a>	<a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FDP_IFF.1/JCVM</a> , <a href="#">FMT_SMR.1/Installer</a> , <a href="#">FMT_MSA.1/CM</a> , <a href="#">FMT_MSA.3/CM</a> , <a href="#">FMT_SMR.1/CM</a> , <a href="#">FMT_MSA.3/FIREWALL</a> , <a href="#">FMT_SMR.1</a> , <a href="#">FMT_MSA.1/ADEL</a> , <a href="#">FMT_MSA.3/ADEL</a> , <a href="#">FMT_SMR.1/ADEL</a> , <a href="#">FMT_MSA.1/REM_REFS</a> , , , , <a href="#">FMT_MSA.1/JCRE</a> , <a href="#">FDP_ITC.2/Installer</a> , , , <a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_ACF.1/FIREWALL</a> , <a href="#">FMT_SMF.1/ADEL</a> , , <a href="#">FMT_SMF.1/CM</a> , <a href="#">FMT_SMF.1</a> , <a href="#">FMT_MSA.2/FIREWALL_JCVM</a> , <a href="#">FMT_MTD.1/JCRE</a> , <a href="#">FMT_MTD.3/JCRE</a> , <a href="#">FMT_MSA.1/JCVM</a> , <a href="#">FMT_MSA.3/JCVM</a> , <a href="#">FDP_ACC.2/RV_Stack</a> , <a href="#">FDP_ACF.1/RV_Stack</a> , <a href="#">FMT_MSA.1/RV_Stack</a> , <a href="#">FMT_MSA.2/RV_Stack</a> , <a href="#">FMT_MSA.3/RV_Stack</a> , <a href="#">FMT_SMF.1/RV_Stack</a> , <a href="#">FDP_ACC.2/RV_Heap</a> , <a href="#">FDP_ACF.1/RV_Heap</a> , <a href="#">FMT_MSA.1/RV_Heap</a> , <a href="#">FMT_MSA.2/RV_Heap</a> , <a href="#">FMT_MSA.3/RV_Heap</a> , <a href="#">FMT_SMF.1/RV_Heap</a> , <a href="#">FDP_ACC.2/RV_Transient</a> , <a href="#">FDP_ACF.1/RV_Transient</a> , <a href="#">FMT_MSA.1/RV_Transient</a> , <a href="#">FMT_MSA.2/RV_Transient</a> , <a href="#">FMT_MSA.3/RV_Transient</a> , <a href="#">FMT_SMF.1/RV_Transient</a>	<a href="#">Section 4.3.1</a>



Security Objectives	Security Functional Requirements	Rationale
<a href="#">O.GLOBAL_ARRAYS_CONFID</a>	<a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FDP_IFF.1/JCVM</a> , <a href="#">FDP_RIP.1/bArray</a> , <a href="#">FDP_RIP.1/APDU</a> , <a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FDP_RIP.1/OBJECTS</a> , <a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_RIP.1/KEYS</a> , <a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FDP_RIP.1/TRANSIENT</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.GLOBAL_ARRAYS_INTEG</a>	<a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FDP_IFF.1/JCVM</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.NATIVE</a>	<a href="#">FDP_ACF.1/FIREWALL</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.OPERATE</a>	<a href="#">FPT_RCV.4/SCP</a> , <a href="#">FAU_ARP.1</a> , <a href="#">FDP_ROL.1/FIREWALL</a> , <a href="#">FIA_ATD.1/AID</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FPT_FLS.1/ODEL</a> , <a href="#">FPT_FLS.1/Installer</a> , <a href="#">FDP_ITC.2/Installer</a> , <a href="#">FPT_RCV.3/Installer</a> , <a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_ACF.1/FIREWALL</a> , <a href="#">FPT_TDC.1</a> , <a href="#">FIA_USB.1/AID</a> , <a href="#">FPT_TST.1</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.REALLOCATION</a>	<a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_RIP.1/APDU</a> , <a href="#">FDP_RIP.1/bArray</a> , <a href="#">FDP_RIP.1/KEYS</a> , <a href="#">FDP_RIP.1/TRANSIENT</a> , <a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FDP_RIP.1/OBJECTS</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.RESOURCES</a>	<a href="#">FAU_ARP.1</a> , <a href="#">FDP_ROL.1/FIREWALL</a> , <a href="#">FMT_SMR.1/Installer</a> , <a href="#">FMT_SMR.1</a> , <a href="#">FMT_SMR.1/ADEL</a> , , <a href="#">FPT_FLS.1/Installer</a> , <a href="#">FPT_FLS.1/ODEL</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FPT_RCV.3/Installer</a> , <a href="#">FMT_SMR.1/CM</a> , <a href="#">FMT_SMF.1/ADEL</a> , , <a href="#">FMT_SMF.1/CM</a> , <a href="#">FMT_SMF.1</a> , <a href="#">FMT_MTD.1/JCRE</a> , <a href="#">FMT_MTD.3/JCRE</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.ALARM</a>	<a href="#">FPT_FLS.1/Installer</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FPT_FLS.1/ODEL</a> , <a href="#">FAU_ARP.1</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.CIPHER</a>	<a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.2</a> , <a href="#">FCS_CKM.3</a> , <a href="#">FCS_CKM.4</a> , <a href="#">FCS_COP.1</a> , <a href="#">FPR_UNO.1</a> , <a href="#">FPR_UNO.1/USE_KEY</a> , <a href="#">FCS_COP.1/PP</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.KEY-MNGT</a>	<a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.2</a> , <a href="#">FCS_CKM.3</a> , <a href="#">FCS_CKM.4</a> , <a href="#">FCS_COP.1</a> , <a href="#">FPR_UNO.1</a> , <a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FDP_RIP.1/OBJECTS</a> , <a href="#">FDP_RIP.1/APDU</a> , <a href="#">FDP_RIP.1/bArray</a> , <a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_RIP.1/KEYS</a> , <a href="#">FDP_SDI.2</a> , <a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FDP_RIP.1/TRANSIENT</a> , <a href="#">FPR_UNO.1/USE_KEY</a> , <a href="#">FCS_COP.1/PP</a>	<a href="#">Section 4.3.1</a>



Security Objectives	Security Functional Requirements	Rationale
<a href="#">O.PIN-MNGT</a>	<a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FDP_RIP.1/OBJECTS</a> , <a href="#">FDP_RIP.1/APDU</a> , <a href="#">FDP_RIP.1/bArray</a> , <a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_RIP.1/KEYS</a> , <a href="#">FPR_UNO.1</a> , <a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FDP_RIP.1/TRANSIENT</a> , <a href="#">FDP_ROL.1/FIREWALL</a> , <a href="#">FDP_SDI.2</a> , <a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_ACF.1/FIREWALL</a> , <a href="#">FIA_AFL.1/PIN</a> , <a href="#">FMT_MTD.2/GP_PIN</a> , <a href="#">FMT_MTD.1/PIN</a> , <a href="#">FIA_AFL.1/GP_PIN</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.TRANSACTION</a>	<a href="#">FDP_ROL.1/FIREWALL</a> , <a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FDP_RIP.1/APDU</a> , <a href="#">FDP_RIP.1/bArray</a> , <a href="#">FDP_RIP.1/KEYS</a> , <a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FDP_RIP.1/TRANSIENT</a> , <a href="#">FDP_RIP.1/OBJECTS</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.OBJ-DELETION</a>	<a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FPT_FLS.1/ODEL</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.DELETION</a>	<a href="#">FDP_ACC.2/ADEL</a> , <a href="#">FDP_ACF.1/ADEL</a> , <a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FPT_RCV.3/Installer</a> , <a href="#">FMT_MSA.1/ADEL</a> , <a href="#">FMT_MSA.3/ADEL</a> , <a href="#">FMT_SMR.1/ADEL</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.LOAD</a>	<a href="#">FCO_NRO.2/CM</a> , <a href="#">FDP_IFC.2/CM</a> , <a href="#">FDP_IFF.1/CM</a> , <a href="#">FDP_UTI.1/CM</a> , <a href="#">FIA_UID.1/CM</a> , <a href="#">FTP_ITC.1/CM</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.INSTALL</a>	<a href="#">FDP_ITC.2/Installer</a> , <a href="#">FPT_RCV.3/Installer</a> , <a href="#">FPT_FLS.1/Installer</a> ,	<a href="#">Section 4.3.1</a>
<a href="#">O.SCP.SUPPORT</a>	<a href="#">FPT_RCV.4/SCP</a> , <a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.2</a> , <a href="#">FCS_CKM.3</a> , <a href="#">FCS_CKM.4</a> , <a href="#">FCS_COP.1</a> , <a href="#">FCS_COP.1/CM</a> , <a href="#">FCS_CKM.1/PP</a> , <a href="#">FCS_COP.1/PP</a> , <a href="#">FCS_CKM.4/PP</a> , <a href="#">FDP_ROL.1</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.SCP.IC</a>	<a href="#">FPT_PHP.3/SCP</a>	<a href="#">Section 4.3.1</a>
<a href="#">O.SCP.RECOVERY</a>	<a href="#">FRU_FLT.1/SCP</a> , <a href="#">FAU_ARP.1</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FPT_RCV.4/SCP</a>	<a href="#">Section 4.3.1</a>

Security Objectives	Security Functional Requirements	Rationale
<u>O.RESIDENT APPLICATION</u>	<u>FDP_ACC.2/PP, FDP_ACF.1/PP, FDP_UCT.1/PP, FDP_ITC.1/PP, FIA_AFL.1/PP, FIA_UAU.1/PP, FMT_MSA.1/PP, FMT_SMF.1/PP, FMT_MOF.1/PP, FMT_SMR.2/PP, FMT_MSA.3/PP, FCS_CKM.4/PP, FMT_SMR.2/CM, FIA_UAU.1/CM, FIA_UAU.4/CardManu, FIA_UAU.7/CardManu</u>	<u>Section 4.3.1</u>
<u>O.CARD MANAGEMENT</u>	<u>FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FPT_FLS.1/Installer, FDP_UIT.1/CM, FDP_ROL.1/FIREWALL, FDP_ITC.2/Installer, FPT_FLS.1/ODEL, FIA_AFL.1/CM, FPT_TST.1, FIA_UID.1/CM, FDP_IFF.1/CM, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.2/PP, FMT_SMF.1/PP, FTP_ITC.1/CM, FMT_SMR.2/CM, FDP_IFC.2/CM, FCO_NRO.2/CM_DAP, FIA_UAU.7/CardIssuer, FPR_UNO.1/Key_CM, FIA_UAU.4/CardIssuer, FPT_TDC.1/CM, FIA_UAU.4/CardManu, FIA_UAU.7/CardManu, FMT_SMF.1/CM, FMT_SMR.1/CM, FIA_UAU.1/CM</u>	<u>Section 4.3.1</u>
<u>O.SECURE COMPARE</u>	<u>FDP_SDI.2/Applet</u>	<u>Section 4.3.1</u>
<u>O.PATCH LOADING</u>	<u>FDP_ACC.2/PP, FDP_ACF.1/PP, FIA_UAU.1/PP, FIA_UID.1/PP, FCS_COP.1/PP, FTP_ITC.1/CM, FCS_COP.1/CM, FDP_UIT.1/PP, FDP_ITC.1/PP, FCS_CKM.1/PP, FDP_UCT.1/PP, FMT_MOF.1/PP, FCS_CKM.4/PP, FAU_STG.2, FIA_AFL.1/PP, FPT_TST.1</u>	<u>Section 4.3.1</u>

Table 12: Security Objectives and SFRs - Coverage



Security Functional Requirements	Security Objectives	Rationale
<a href="#">FDP_ACC.2/FIREWALL</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.OPERATE</a> , <a href="#">O.PIN-MNGT</a>	
<a href="#">FDP_ACF.1/FIREWALL</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.NATIVE</a> , <a href="#">O.OPERATE</a> , <a href="#">O.PIN-MNGT</a>	
<a href="#">FDP_IFC.1/JCVM</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.GLOBAL_ARRAYS_INTEG</a>	
<a href="#">FDP_IFF.1/JCVM</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.GLOBAL_ARRAYS_INTEG</a>	
<a href="#">FDP_RIP.1/OBJECTS</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.TRANSACTION</a>	
<a href="#">FMT_MSA.1/JCRE</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.1/JCVM</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.2/FIREWALL_JCVM</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.3/FIREWALL</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.3/JCVM</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a>	
<a href="#">FMT_SMF.1</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a>	
<a href="#">FMT_SMR.1</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a>	
<a href="#">FCS_CKM.1</a>	<a href="#">O.CIPHER</a> , <a href="#">O.KEY-MNGT</a>	
<a href="#">FCS_CKM.2</a>	<a href="#">O.CIPHER</a> , <a href="#">O.KEY-MNGT</a>	
<a href="#">FCS_CKM.3</a>	<a href="#">O.CIPHER</a> , <a href="#">O.KEY-MNGT</a>	
<a href="#">FCS_CKM.4</a>	<a href="#">O.CIPHER</a> , <a href="#">O.KEY-MNGT</a>	
<a href="#">FCS_COP.1</a>	<a href="#">O.CIPHER</a> , <a href="#">O.KEY-MNGT</a>	
<a href="#">FDP_RIP.1/ABORT</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.TRANSACTION</a>	
<a href="#">FDP_RIP.1/APDU</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.TRANSACTION</a>	
<a href="#">FDP_RIP.1/bArray</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , , <a href="#">O.TRANSACTION</a>	
<a href="#">FDP_RIP.1/KEYS</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.TRANSACTION</a>	



Security Functional Requirements	Security Objectives	Rationale
<a href="#">FDP_RIP.1/TRANSIENT</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , , <a href="#">O.TRANSACTION</a>	
<a href="#">FDP_ROL.1/FIREWALL</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.TRANSACTION</a> , <a href="#">O.SCP.SUPPORT</a> , <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FAU_ARP.1</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.ALARM</a> , <a href="#">O.SCP.RECOVERY</a>	
<a href="#">FDP_SDI.2</a>	<a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-MNGT</a>	
<a href="#">FPR_UNO.1</a>	<a href="#">O.CIPHER</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-MNGT</a>	
<a href="#">FPT_FLS.1</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.ALARM</a> , <a href="#">O.SCP.RECOVERY</a>	
<a href="#">FPT_TDC.1</a>	<a href="#">O.OPERATE</a>	
<a href="#">FIA_ATD.1/AID</a>	<a href="#">O.SID</a> , <a href="#">O.OPERATE</a>	
<a href="#">FIA_UID.2/AID</a>	<a href="#">O.SID</a>	
<a href="#">FIA_USB.1/AID</a>	<a href="#">O.SID</a> , <a href="#">O.OPERATE</a>	
<a href="#">FMT_MTD.1/JCRE</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a>	
<a href="#">FMT_MTD.3/JCRE</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a>	
<a href="#">FDP_ITC.2/Installer</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.OPERATE</a> , <a href="#">O.INSTALL</a> , <a href="#">O.CARD_MANAGEMENT</a> ,	
<a href="#">FMT_SMR.1/Installer</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a>	
<a href="#">FPT_FLS.1/Installer</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.ALARM</a> , <a href="#">O.INSTALL</a> , <a href="#">O.CARD_MANAGEMENT</a> ,	
<a href="#">FPT_RCV.3/Installer</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.DELETION</a> , <a href="#">O.INSTALL</a> , <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FDP_ACC.2/ADEL</a>	<a href="#">O.CARD_MANAGEMENT</a> , <a href="#">O.DELETION</a>	
<a href="#">FDP_ACF.1/ADEL</a>	<a href="#">O.CARD_MANAGEMENT</a> , <a href="#">O.DELETION</a>	
<a href="#">FDP_RIP.1/ADEL</a>	<a href="#">O.GLOBAL_ARRAYS_CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-</a> <a href="#">MNGT</a> , <a href="#">O.TRANSACTION</a> , <a href="#">O.DELETION</a> , <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FMT_MSA.1/ADEL</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.DELETION</a> , <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FMT_MSA.3/ADEL</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.DELETION</a> , <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FMT_SMF.1/ADEL</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a>	

| } } } }

Security Functional Requirements	Security Objectives	Rationale
<a href="#">FMT_SMR.1/ADEL</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.DELETION</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FPT_FLS.1/ADEL</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.ALARM</a> , <a href="#">O.DELETION</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FDP_RIP.1/ODEL</a>	<a href="#">O.GLOBAL ARRAYS CONFID</a> , <a href="#">O.REALLOCATION</a> , <a href="#">O.KEY-MNGT</a> , <a href="#">O.PIN-MNGT</a> , <a href="#">O.TRANSACTION</a> , <a href="#">O.OBJ-DELETION</a>	
<a href="#">FPT_FLS.1/ODEL</a>	<a href="#">O.OPERATE</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.ALARM</a> , <a href="#">O.OBJ-DELETION</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FCO_NRO.2/CM</a>	<a href="#">O.LOAD</a> , <a href="#">O.INSTALL</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FDP_IFC.2/CM</a>	<a href="#">O.LOAD</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FDP_IFF.1/CM</a>	<a href="#">O.LOAD</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FDP_UIT.1/CM</a>	<a href="#">O.LOAD</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FIA_UID.1/CM</a>	<a href="#">O.LOAD</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FMT_MSA.1/CM</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FMT_MSA.3/CM</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FMT_SMF.1/CM</a>	<a href="#">O.SID</a> , <a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FMT_SMR.1/CM</a>	<a href="#">O.FIREWALL</a> , <a href="#">O.RESOURCES</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FTP_ITC.1/CM</a>	<a href="#">O.LOAD</a> , <a href="#">O.INSTALL</a> , <a href="#">O.CARD MANAGEMENT</a> , <a href="#">O.PATCH LOADING</a>	
<a href="#">FPT_TST.1</a>	<a href="#">O.OPERATE</a> , <a href="#">O.PATCH LOADING</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FCO_NRO.2/CM_DAP</a>	<a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FIA_AFL.1/CM</a>	<a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FIA_UAU.1/CM</a>	<a href="#">O.RESIDENT APPLICATION</a> , <a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FIA_UAU.4/CardIssuer</a>	<a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FIA_UAU.7/CardIssuer</a>	<a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FPR_UNO.1/Key_CM</a>	<a href="#">O.CARD MANAGEMENT</a>	
<a href="#">FPT_TDC.1/CM</a>	<a href="#">O.CARD MANAGEMENT</a>	

| } } } }

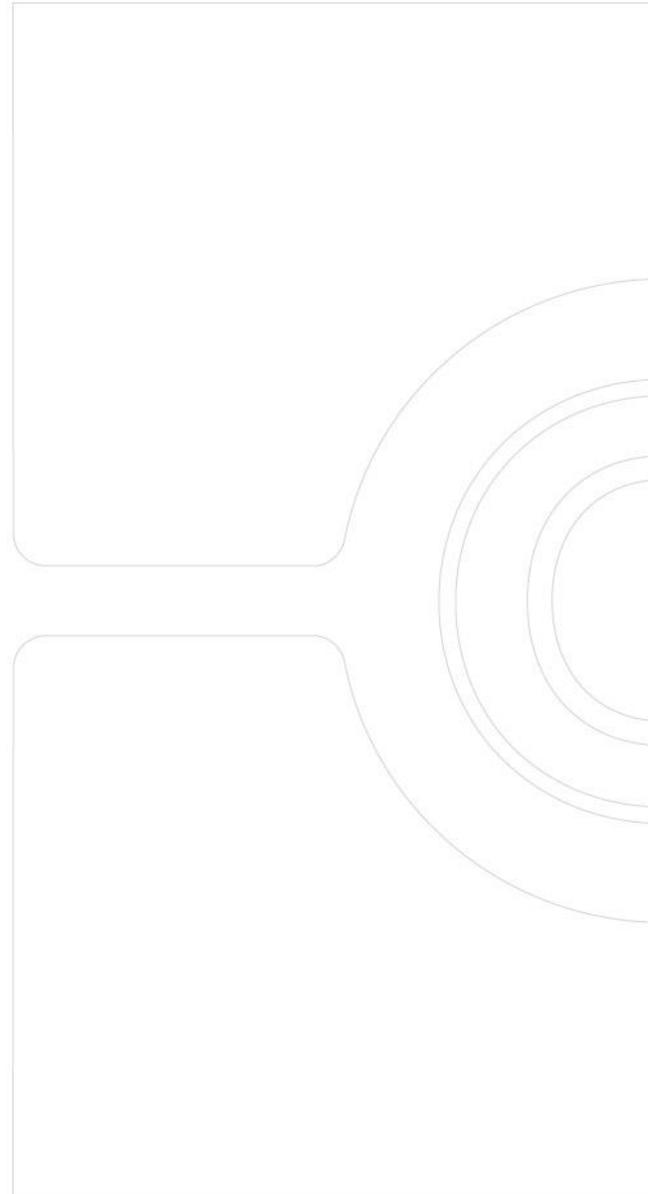
Security Functional Requirements	Security Objectives	Rationale
<a href="#">FMT_SMR.2/CM</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FDP_ACC.2/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FDP_ACF.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FDP_UCT.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FDP_ITC.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FIA_AFL.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FIA_UAU.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FIA_UID.1/PP</a>	<a href="#">O.PATCH_LOADING</a>	
<a href="#">FMT_MSA.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION</a>	
<a href="#">FMT_SMF.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FIA_UAU.4/CardManu</a>	<a href="#">O.CARD_MANAGEMENT,</a> <a href="#">O.RESIDENT_APPLICATION</a>	
<a href="#">FIA_UAU.7/CardManu</a>	<a href="#">O.CARD_MANAGEMENT,</a> <a href="#">O.RESIDENT_APPLICATION</a>	
<a href="#">FMT_MOF.1/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FMT_SMR.2/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.CARD_MANAGEMENT</a>	
<a href="#">FMT_MSA.3/PP</a>	<a href="#">O.RESIDENT_APPLICATION</a>	
<a href="#">FCS_COP.1/PP</a>	<a href="#">O.CIPHER, O.KEY-MNGT,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FCS_CKM.4/PP</a>	<a href="#">O.RESIDENT_APPLICATION,</a> <a href="#">O.PATCH_LOADING</a>	
<a href="#">FPT_PHP.3/SCP</a>	<a href="#">O.SCP.IC</a>	
<a href="#">FPT_RCV.4/SCP</a>	<a href="#">O.SCP.SUPPORT, O.OPERATE</a>	
<a href="#">FRU_FLT.1/SCP</a>	<a href="#">O.SCP.RECOVERY</a>	
<a href="#">FPR_UNO.1/USE_KEY</a>	<a href="#">O.CIPHER, O.KEY-MNGT</a>	

| } } } }

Security Functional Requirements	Security Objectives	Rationale
<a href="#">FCS_RNG.1/SCP</a>	<a href="#">O.SCP.IC</a>	
<a href="#">FIA_AFL.1/PIN</a>	<a href="#">O.PIN-MNGT</a>	
<a href="#">FMT_MTD.2/GP_PIN</a>	<a href="#">O.PIN-MNGT</a>	
<a href="#">FDP_SDI.2/Applet</a>	<a href="#">O.SECURE_COMPARE</a>	
<a href="#">FMT_MTD.1/PIN</a>	<a href="#">O.PIN-MNGT</a>	
<a href="#">FIA_AFL.1/GP_PIN</a>	<a href="#">O.PIN-MNGT</a>	
<a href="#">FDP_ACC.2/RV_Stack</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FDP_ACF.1/RV_Stack</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.1/RV_Stack</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.2/RV_Stack</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.3/RV_Stack</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_SMF.1/RV_Stack</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FDP_ACC.2/RV_Heap</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FDP_ACF.1/RV_Heap</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.1/RV_Heap</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.2/RV_Heap</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.3/RV_Heap</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_SMF.1/RV_Heap</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FDP_ACC.2/RV_Transient</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FDP_ACF.1/RV_Transient</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.1/RV_Transient</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.2/RV_Transient</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_MSA.3/RV_Transient</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FMT_SMF.1/RV_Transient</a>	<a href="#">O.FIREWALL</a>	
<a href="#">FCS_COP.1/CM</a>	<a href="#">O.PATCH_LOADING</a>	
<a href="#">FDP_UIT.1/PP</a>	<a href="#">O.PATCH_LOADING</a>	
<a href="#">FCS_CKM.1/PP</a>	<a href="#">O.PATCH_LOADING</a>	
<a href="#">FAU_STG.2</a>	<a href="#">O.PATCH_LOADING</a>	

Table 13: SFRs and Security Objectives



**7.3.3 Dependencies****7.3.3.1 SFRs Dependencies**

Requirements	CC Dependencies	Satisfied Dependencies
<a href="#">FDP_ITC.2/Installer</a>	(FDP_ACC.1 or FDP_IFC.1) and (FPT_TDC.1) and (FTP_ITC.1 or FTP_TRP.1)	<a href="#">FDP_IFC.2/CM</a> , <a href="#">FTP_ITC.1/CM</a> , <a href="#">FPT_TDC.1</a>
<a href="#">FMT_SMR.1/Installer</a>	(FIA_UID.1)	
<a href="#">FPT_FLS.1/Installer</a>	No Dependencies	
<a href="#">FPT_RCV.3/Installer</a>	(AGD_OPE.1)	<a href="#">AGD_OPE.1</a>
<a href="#">FDP_ACC.2/ADEL</a>	(FDP_ACF.1)	<a href="#">FDP_ACF.1/ADEL</a>
<a href="#">FDP_ACF.1/ADEL</a>	(FDP_ACC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/ADEL</a> , <a href="#">FMT_MSA.3/ADEL</a>
<a href="#">FDP_RIP.1/ADEL</a>	No Dependencies	
<a href="#">FMT_MSA.1/ADEL</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/ADEL</a> , <a href="#">FMT_SMF.1/ADEL</a> , <a href="#">FMT_SMR.1/ADEL</a>
<a href="#">FMT_MSA.3/ADEL</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_MSA.1/ADEL</a> , <a href="#">FMT_SMR.1/ADEL</a>
<a href="#">FMT_SMF.1/ADEL</a>	No Dependencies	
<a href="#">FMT_SMR.1/ADEL</a>	(FIA_UID.1)	
<a href="#">FPT_FLS.1/ADEL</a>	No Dependencies	
<a href="#">FDP_RIP.1/ODEL</a>	No Dependencies	
<a href="#">FPT_FLS.1/ODEL</a>	No Dependencies	
<a href="#">FCO_NRO.2/CM</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/CM</a>
<a href="#">FDP_IFC.2/CM</a>	(FDP_IFF.1)	<a href="#">FDP_IFF.1/CM</a>
<a href="#">FDP_IFF.1/CM</a>	(FDP_IFC.1) and (FMT_MSA.3)	<a href="#">FDP_IFC.2/CM</a> , <a href="#">FMT_MSA.3/CM</a>
<a href="#">FDP_UIT.1/CM</a>	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	<a href="#">FDP_IFC.2/CM</a> , <a href="#">FTP_ITC.1/CM</a>
<a href="#">FIA_UID.1/CM</a>	No Dependencies	
<a href="#">FMT_MSA.1/CM</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FDP_IFC.2/CM</a> , <a href="#">FMT_SMF.1/CM</a> , <a href="#">FMT_SMR.1/CM</a>



Requirements	CC Dependencies	Satisfied Dependencies
<a href="#">FMT_MSA.3/CM</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_MSA.1/CM</a> , <a href="#">FMT_SMR.1/CM</a>
<a href="#">FMT_SMF.1/CM</a>	No Dependencies	
<a href="#">FMT_SMR.1/CM</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/CM</a>
<a href="#">FTP_ITC.1/CM</a>	No Dependencies	
<a href="#">FDP_ACC.2/FIREWALL</a>	(FDP_ACF.1)	<a href="#">FDP_ACF.1/FIREWALL</a>
<a href="#">FDP_ACF.1/FIREWALL</a>	(FDP_ACC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FMT_MSA.3/FIREWALL</a>
<a href="#">FDP_IFC.1/JCVM</a>	(FDP_IFF.1)	<a href="#">FDP_IFF.1/JCVM</a>
<a href="#">FDP_IFF.1/JCVM</a>	(FDP_IFC.1) and (FMT_MSA.3)	<a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FMT_MSA.3/JCVM</a>
<a href="#">FDP_RIP.1/OBJECTS</a>	No Dependencies	
<a href="#">FMT_MSA.1/JCRE</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_MSA.1/JCVM</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FMT_SMF.1</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_MSA.2/FIREWALL JCVM</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FMT_MSA.1/JCRE</a> , <a href="#">FMT_MSA.1/JCVM</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_MSA.3/FIREWALL</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_MSA.1/JCRE</a> , <a href="#">FMT_MSA.1/JCVM</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_MSA.3/JCVM</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_MSA.1/JCVM</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_SMF.1</a>	No Dependencies	
<a href="#">FMT_SMR.1</a>	(FIA_UID.1)	<a href="#">FIA_UID.2/AID</a>
<a href="#">FCS_CKM.1</a>	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	<a href="#">FCS_CKM.2</a> , <a href="#">FCS_CKM.4</a>
<a href="#">FCS_CKM.2</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	<a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.4</a>



Requirements	CC Dependencies	Satisfied Dependencies
<a href="#">FCS_CKM.3</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	<a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.4</a>
<a href="#">FCS_CKM.4</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	<a href="#">FCS_CKM.1</a>
<a href="#">FCS_COP.1</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	<a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.4</a>
<a href="#">FDP_RIP.1/ABORT</a>	No Dependencies	
<a href="#">FDP_RIP.1/APDU</a>	No Dependencies	
<a href="#">FDP_RIP.1/bArray</a>	No Dependencies	
<a href="#">FDP_RIP.1/KEYS</a>	No Dependencies	
<a href="#">FDP_RIP.1/TRANSIENT</a>	No Dependencies	
<a href="#">FDP_ROL.1/FIREWALL</a>	(FDP_ACC.1 or FDP_IFC.1)	<a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_IFC.1/JCVM</a>
<a href="#">FAU_ARP.1</a>	(FAU_SAA.1)	
<a href="#">FDP_SDI.2</a>	No Dependencies	
<a href="#">FPR_UNO.1</a>	No Dependencies	
<a href="#">FPT_FLS.1</a>	No Dependencies	
<a href="#">FPT_TDC.1</a>	No Dependencies	
<a href="#">FIA_ATD.1/AID</a>	No Dependencies	
<a href="#">FIA_UID.2/AID</a>	No Dependencies	
<a href="#">FIA_USB.1/AID</a>	(FIA_ATD.1)	<a href="#">FIA_ATD.1/AID</a>
<a href="#">FMT_MTD.1/JCRE</a>	(FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FMT_SMF.1</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_MTD.3/JCRE</a>	(FMT_MTD.1)	<a href="#">FMT_MTD.1/JCRE</a>
<a href="#">FPT_TST.1</a>	No Dependencies	
<a href="#">FCO_NRO.2/CM_DAP</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/PP</a>
<a href="#">FIA_AFL.1/CM</a>	(FIA_UAU.1)	<a href="#">FIA_UAU.1/CM</a>
<a href="#">FIA_UAU.1/CM</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/CM</a>
<a href="#">FIA_UAU.4/CardIssuer</a>	No Dependencies	



Requirements	CC Dependencies	Satisfied Dependencies
<a href="#">FIA_UAU.7/CardIssuer</a>	(FIA_UAU.1)	<a href="#">FIA_UAU.1/CM</a>
<a href="#">FPR_UNO.1/Key_CM</a>	No Dependencies	
<a href="#">FPT_TDC.1/CM</a>	No Dependencies	
<a href="#">FMT_SMR.2/CM</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/PP</a>
<a href="#">FDP_ACC.2/PP</a>	(FDP_ACF.1)	<a href="#">FDP_ACF.1/PP</a>
<a href="#">FDP_ACF.1/PP</a>	(FDP_ACC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/PP</a> , <a href="#">FMT_MSA.3/PP</a>
<a href="#">FDP_UCT.1/PP</a>	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	<a href="#">FDP_ACC.2/PP</a>
<a href="#">FDP_ITC.1/PP</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/PP</a> , <a href="#">FMT_MSA.3/PP</a>
<a href="#">FIA_AFL.1/PP</a>	(FIA_UAU.1)	<a href="#">FIA_UAU.1/PP</a>
<a href="#">FIA_UAU.1/PP</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/PP</a>
<a href="#">FIA_UID.1/PP</a>	No Dependencies	
<a href="#">FMT_MSA.1/PP</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/PP</a> , <a href="#">FMT_SMF.1/PP</a> , <a href="#">FMT_SMR.2/PP</a>
<a href="#">FMT_SMF.1/PP</a>	No Dependencies	
<a href="#">FIA_UAU.4/CardManu</a>	No Dependencies	
<a href="#">FIA_UAU.7/CardManu</a>	(FIA_UAU.1)	<a href="#">FIA_UAU.1/PP</a>
<a href="#">FMT_MOF.1/PP</a>	(FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FMT_SMF.1/PP</a> , <a href="#">FMT_SMR.2/PP</a>
<a href="#">FMT_SMR.2/PP</a>	(FIA_UID.1)	<a href="#">FIA_UID.1/PP</a>
<a href="#">FMT_MSA.3/PP</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_MSA.1/PP</a> , <a href="#">FMT_SMR.2/PP</a>
<a href="#">FCS_COP.1/PP</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	<a href="#">FDP_ITC.1/PP</a> , <a href="#">FCS_CKM.4/PP</a>



Requirements	CC Dependencies	Satisfied Dependencies
<a href="#">FCS_CKM.4/PP</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	<a href="#">FDP_ITC.1/PP</a>
<a href="#">FPT_PHP.3/SCP</a>	No Dependencies	
<a href="#">FPT_RCV.4/SCP</a>	No Dependencies	
<a href="#">FRU_FLT.1/SCP</a>	(FPT_FLS.1)	
<a href="#">FPR_UNO.1/USE_KEY</a>	No Dependencies	
<a href="#">FCS_RNG.1/SCP</a>	No Dependencies	
<a href="#">FIA_AFL.1/PIN</a>	(FIA_UAU.1)	<a href="#">See justification</a>
<a href="#">FMT_MTD.2/GP_PIN</a>	(FMT_MTD.1) and (FMT_SMR.1)	<a href="#">See justification</a>
<a href="#">FDP_SDI.2/Applet</a>	No Dependencies	
<a href="#">FMT_MTD.1/PIN</a>	(FMT_SMF.1) and (FMT_SMR.1)	<a href="#">See justification</a>
<a href="#">FIA_AFL.1/GP_PIN</a>	(FIA_UAU.1)	<a href="#">See justification</a>
<a href="#">FDP_ACC.2/RV_Stack</a>	(FDP_ACF.1)	<a href="#">FDP_ACF.1/RV_Stack</a>
<a href="#">FDP_ACF.1/RV_Stack</a>	(FDP_ACC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/RV_Stack,</a> <a href="#">FMT_MSA.3/RV_Stack</a>
<a href="#">FMT_MSA.1/RV_Stack</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FMT_SMR.1,</a> <a href="#">FDP_ACC.2/RV_Stack,</a> <a href="#">FMT_SMF.1/RV_Stack</a>
<a href="#">FMT_MSA.2/RV_Stack</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_SMR.1,</a> <a href="#">FDP_ACC.2/RV_Stack,</a> <a href="#">FMT_MSA.1/RV_Stack</a>
<a href="#">FMT_MSA.3/RV_Stack</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_SMR.1,</a> <a href="#">FMT_MSA.1/RV_Stack</a>
<a href="#">FMT_SMF.1/RV_Stack</a>	No Dependencies	
<a href="#">FDP_ACC.2/RV_Heap</a>	(FDP_ACF.1)	<a href="#">FDP_ACF.1/RV_Heap</a>
<a href="#">FDP_ACF.1/RV_Heap</a>	(FDP_ACC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/RV_Heap,</a> <a href="#">FMT_MSA.3/RV_Heap</a>
<a href="#">FMT_MSA.1/RV_Heap</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FMT_SMR.1,</a> <a href="#">FDP_ACC.2/RV_Heap,</a> <a href="#">FMT_SMF.1/RV_Heap</a>



Requirements	CC Dependencies	Satisfied Dependencies
<a href="#">FMT_MSA.2/RV_Heap</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_SMR.1</a> , <a href="#">FDP_ACC.2/RV_Heap</a> , <a href="#">FMT_MSA.1/RV_Heap</a>
<a href="#">FMT_MSA.3/RV_Heap</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_SMR.1</a> , <a href="#">FMT_MSA.1/RV_Heap</a>
<a href="#">FMT_SMF.1/RV_Heap</a>	No Dependencies	
<a href="#">FDP_ACC.2/RV_Transient</a>	(FDP_ACF.1)	<a href="#">FDP_ACF.1/RV_Transient</a>
<a href="#">FDP_ACF.1/RV_Transient</a>	(FDP_ACC.1) and (FMT_MSA.3)	<a href="#">FDP_ACC.2/RV_Transient</a> , <a href="#">FMT_MSA.3/RV_Transient</a>
<a href="#">FMT_MSA.1/RV_Transient</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/RV_Transient</a> , <a href="#">FMT_SMR.1</a> , <a href="#">FMT_SMF.1/RV_Transient</a>
<a href="#">FMT_MSA.2/RV_Transient</a>	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FDP_ACC.2/RV_Transient</a> , <a href="#">FMT_SMR.1</a> , <a href="#">FMT_MSA.1/RV_Transient</a>
<a href="#">FMT_MSA.3/RV_Transient</a>	(FMT_MSA.1) and (FMT_SMR.1)	<a href="#">FMT_MSA.1/RV_Transient</a> , <a href="#">FMT_SMR.1</a>
<a href="#">FMT_SMF.1/RV_Transient</a>	No Dependencies	
<a href="#">FCS_COP.1/CM</a>	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
<a href="#">FDP_UIT.1/PP</a>	(FDP_ACC.1 or FDP_IFC.1) and (FDP_ITC.1 or FDP_TRP.1)	<a href="#">FDP_ACC.2/PP</a>
<a href="#">FCS_CKM.1/PP</a>	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	<a href="#">FCS_COP.1/PP</a> , <a href="#">FCS_CKM.4/PP</a>
<a href="#">FAU_STG.2</a>	(FAU_GEN.1)	See justification

Table 14: SFRs Dependencies



## Rationale for the exclusion of Dependencies

**The dependency FIA\_UID.1 of FMT\_SMR.1/Installer is discarded.** This PP does not require the identification of the "installer" since it can be considered as part of the TSF.

**The dependency FIA\_UID.1 of FMT\_SMR.1/ADEL is discarded.** This PP does not require the identification of the "deletion manager" since it can be considered as part of the TSF.

**The dependency FMT\_SMF.1 of FMT\_MSA.1/JCRE is discarded.** The dependency between FMT\_MSA.1/JCRE and FMT\_SMF.1 is not satisfied because no management functions are required for the Java Card RE.

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

**The dependency FAU\_GEN.1 of FAU\_STG.2 is discarded.**

The FAU\_STG.2 is related to the patch. When the identification of the patch is incorrect, the TOE rise a kill Card exception. The FAU\_GEN is then discarded as the card returns only the ATR. There is need to store any audit function.

**The dependency with FIA\_UAU.1 of FIA\_AFL.1/PIN, FIA\_AFL.1/GP\_PIN is not applicable.** The TOE implements the firewall access control SFP, based on which access to the object implementing FIA\_AFL.1/PIN is organized.

**The dependency with FMT\_SMF.1 and FMT\_SMR.1 of FMT\_MTD.1/PIN is not applicable.** The TOE implements the firewall access control of applications based on the AIDs.

SARs Dependencies



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



Table 15: SARs Dependencies

#### 7.3.4 Rationale for the Security Assurance Requirements

The ID-ONE COSMO V9 ESSENTIAL product claims a conformance to the Common Criteria level EAL5, augmented with the component ALC\_DVS.2 (sufficiency of security measures), AVA\_VAN.5 (advanced methodical vulnerability analysis).

#### 7.3.5 AVA\_VAN.5 Advanced methodical vulnerability analysis

The TOE is intended to operate in hostile environments. AVA\_VAN.5 "Advanced methodical vulnerability analysis" is considered as the expected level for Java Card technology-based products hosting sensitive applications, in particular in payment and identity areas. AVA\_VAN.5 has dependencies on ADV\_ARC.1, ADV\_FSP.4, ADV\_TDS.3, ADV\_IMP.1, AGD\_PRE.1, AGD\_OPE.1 and ATE\_DPT.1. All of them are satisfied by EAL5.

#### 7.3.6 ALC\_DVS.2 Sufficiency of security measures

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



## 8 TOE SUMMARY SPECIFICATION

---

### 8.1 TOE Summary Specification

#### SF\_ATOMIC\_TRANSACTION

This TSF provides means to execute a sequence of modifications and allocations on the persistent memory so that either all of them are completed, or the TOE behaves as if none of them had been attempted. The transaction mechanism is used for updating internal TSF data as well as for performing different functions of the TOE, like installing a new package on the card. This TSF is also available for applet instances through the `javacard.framework.JCSystem`, `javacard.framework.Util` and `javacardx.framework.util.ArrayLogic` classes. The first class provides the applet instances with methods for starting, aborting and committing a sequence of modifications of the persistent memory. The other classes provide methods for atomically copying arrays. This TSF ensures that the following data is never updated conditionally:

- o The validated flag of the PINs
- o The reason code of the `CardException` and `CardRuntimeException`
- o Transient objects
- o Global arrays, like the APDU buffer and the buffer that the applet instances use to store installation data
- o Any intermediate result state in the implementation instance of the `Checksum`, `Signature`, `Cipher`, and `Message Digest` classes of the JavaCard API.

This TSF is in charge of setting back the state of the persistent memory as it was before they were started, when the following operations specified are not completed:

- o Loading and linking of a package
- o Installing a new applet instance
- o Deleting a package
- o Deleting an applet instance
- o Collecting unreachable objects
- o Reading from and writing to a static field, instance field or array position
- o Populating, updating or clearing a cryptographic key
- o Modifying a PIN value

Upon deallocation of a resource from any reference to an object instance created during an aborted transaction, any previous information content of the resource is made unavailable.

Finally, this TSF ensures that no transaction is in progress when a method of an applet instance is invoked for installing, deselecting, selecting or processing an APDU sent to the applet instance. Concerning memory limitations on the transaction journal, this TSF guarantees that an exception is thrown when the maximal capacity is reached. The TSF preserves a secure state when such limit is reached. Atomic Transactions are detailed in the chapter `Atomicity and Transactions` of the [R7] and in the documentation associated to the `JCSystem` class in the [R6].

#### SF\_CARD\_CONTENT\_MANAGEMENT

This TSF ensures the following functionalities:

- o Loading (Section 9.3.5 of [R12]): This function allows the addition of code to mutable persistent memory in the card. During card content loading, this TSF checks that the required packages are already installed on the card. If one of the

required packages does not exist, or if the version installed on the card is not binary compatible with the version required, then the loading of the package is rejected. Loading is also rejected if the version of the CAP format of the package is newer than the one supported by the TOE. If any of those checks fails, a suitable error message is returned to the CAD.

- o Installation (Section 9.3.6 of [R12]): This function allows the Installer to create an instance of a previously loaded Applet subclass and make it selectable. In order to do this, the `install()` method of the Applet subclass is invoked using the context of that new instance as the currently active context. If this method returns with an exception, the exception is trapped and the smart card rolls back to the state before starting the installation procedure.
- o Deletion (Section 9.5 of [R12]): This function allows the Applet Deletion Manager to remove the code of a package from the card, or to definitely deactivate an applet instance, so that it becomes no longer selectable. This TSF performs physical removal of those packages and applet data stored in NVRAM, while only logical removal is performed for packages in ROM. This TSF checks that the package or applet actually exists, and that no other package or applet depends on it for its execution. In this case, the entry of the package or applet is removed from the registry, and all the objects on which they depend are garbage collected. Otherwise, a suitable error is returned to the CAD. The deletion of the Applet Deletion Manager, the Installer or any of the packages required for implementing the Java Card platform Application Programming Interface (Java Card API) is not allowed.
- o Extradition (Section 9.4.1 of): This function allows the Installer to associate load files or applet instances to a Security Domain different than their currently associated Security Domain. It is also used to associate a Security Domain to another Security Domain or to itself thus creating Security Domains hierarchies. If this method returns with an exception, the exception is trapped and the smart card rolls back to the state before starting the extradition procedure.
- o Registry update (Section 9.4.2 of): This function allows the Installer to populate, modify or delete elements of the Registry entry of applet instances. If this method returns with an exception, the exception is trapped and the smart card rolls back to the state before starting the extradition procedure.

#### **SF\_CARD\_MANAGEMENT\_ENVIRONMENT**

This TSF is in charge of initializing and managing the internal data structures of the Card Manager. During the initialization phase of the card, this TSF creates the Installer and the Applet Deletion Manager and initializes their internal data structures. The internal data structures of the Card Manager includes the Package and Applet Registries, which respectively contains the currently loaded packages and the currently installed applet instances, together with their associated AIDs. This TSF is also in charge of dispatching the APDU commands to the applets instances installed on the card and keeping traces of which are the currently active ones. It therefore handles sensitive TSF data of other security functions, like the Firewall.

#### **SF\_CARDHOLDER\_VERIFICATION**

This TSF enables applet instances to authenticate the sender of a request as the true cardholder. Applet instances have access to these services through the OwnerPIN class. Cardholder authentication is performed using the following security attributes:

- o A secret enabling to authenticate the cardholder
- o The maximum number of consecutive unsuccessful comparison attempts that are admitted

- o A counter of the number of consecutive unsuccessful comparison attempts that have been performed so far
- o The current life cycle state of the secret (reference value). This state is always updated, even if the modification is in the scope of an open transaction. Each time an attempt is made to compare a value to the reference value, and prior to the comparison being actually performed, if the reference is blocked, then the comparison fails and the reference value is not accessed. Otherwise, the try counter is decremented by one. This operation is always performed, even if it is in the scope of an open transaction. If the comparison is successful, then the try counter is reset to the try limit. When the try counter reaches zero, the reference enters into a blocked state, and cannot be used until it is unblocked. Cardholder Verification Method services are implemented to resist to environmental stress and glitches and include measures for preventing information leakage through covert channels. In particular, unsuccessful authentication attempts consume the same power and execution time than successful ones. The Cardmanager uses the class OwnerPin to provide the services to the Applet that want benefit of the Shared GP\_PIN. The **SF\_CARDHOLDER\_VERIFICATION implements all Pin verifications: D.PIN, GP.PIN.**

#### **SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION**

This TSF clears all the data containers that hold sensitive information when that information is no longer used or upon the allocation of the resource. This includes:

- o The contents of the memory blocks allocated for storing class instances, arrays, static field images and local variables, before allocating a fresh block
- o The objects reclaimed by the Java Card VM garbage collector
- o The code of the deleted packages
- o The objects accessible from a deleted applet instance
- o The content of the bArray argument of the Applet.install method after a new applet instance is installed
- o The content of CLEAR ON DESELECT transient objects owned by an applet instance that has been deselected when no other applets from the same package are active on the card
- o The content of all transient objects after a card reset
- o The contents of the cryptographic buffer after performing cryptographic operations
- o The Reference to an object instance created during an aborted transaction

#### *Application Note:*

This function is in charge of clearing the information contained in the objects that are no longer accessible from the installed packages and applet instances. Clearing is performed on demand of an applet instance through the JCSYSTEM.requestObjectDeletion() method.

#### **SF\_DAP\_VERIFICATION**

An Application Provider may require that its Application code to be loaded on the card is checked for integrity and authenticity. The DAP Verification privilege of the Application Provider's Security Domain detailed in Section 9.2.1 of provides this service on behalf of an Application Provider. A Controlling Authority may require that all Application code to be loaded onto the card shall be checked for integrity and authenticity. The Mandated DAP Verification privilege of the Controlling Authority's Security Domain detailed in Section 9.2.1

of provides this service on behalf of the Controlling Authority. The keys and algorithms to be used for DAP Verification or Mandated DAP Verification are implicitly known by the corresponding Security Domain.

**SF\_DATA\_COHERENCY**

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

**SF\_DATA\_INTEGRITY**

Some of the data in non volatile memory can be protected. Keys, PIN package and patch code are protected with integrity value. When reading and writing operation, the integrity value is checked and maintained valid. In case of incoherency, an exception is raised to prevent the bad use of the data. SecureStore is a mean for protecting JavaCard data in integrity.

**SF\_ENCRYPTION\_AND\_DECRYPTION**

This TSF provides the applet instances with mechanisms for encrypting and decrypting the contents of a byte array.

The ciphering algorithms are available to the applets through the Cipher class of the Java Card API, ISOSecureMessaging class and SecureChannel class. The length of the key to be used for the ciphering operation is defined by the applet instance when the key is generated. Before encrypting or decrypting the byte array, the TSF verifies that the specified key has been previously initialized, and that is in accordance with the specified ciphering algorithm (DES, RSA, etc). The TSF also checks that it has been provided with all the information necessary for the encryption/decryption operation. Once the ciphering operation is performed, the internal TSF data used for the operation like the ICV is cleared. Ciphering operations are implemented to resist to environmental stress and glitches and include measures for preventing information leakage through covert channels.

Mechanisms of encrypting and decrypting for Secure Messaging are available to the applets through the SecureChannel (Global Platform Card 2.2" specification) and ISOSecureMessaging (Proprietary API **[R48]**) classes.

**SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL**

Off-card entity authentication is achieved by initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity. If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated). The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s).

**SF\_EXCEPTION**

In case of abnormal event: data unavailable on an allocation, illegal access to a data, the system owns an internal mechanism that allows to stop the code execution and raise an exception.

**SF\_FIREWALL**

This TSF enforces the Firewall security policy and the information flow control policy at

runtime. The former policy controls object sharing between different applet instances, and between applet instances and the Java Card RE. The latter policy controls the access to global data containers shared by all applet instances. This TSF is enforced by the Java Card platform Virtual Machine (Java Card VM). During the execution of an applet, the Java Card VM keeps track of the applet instance that is currently performing an action. This information is known as the currently active context. Two kinds of contexts are considered: applet instances contexts and the Java Card RE context, which has special privileges for accessing objects. The TSF makes no difference between instances of applets defined in the same package: all of them belong to the same active context. On the contrary, instances of applets defined in different packages belong to different contexts. Each object belongs to the context that was active when the object was allocated. Initially, when the Java Card VM is launched, the context corresponding to the applet instance selected for execution becomes the first active context. Each time an instance method is invoked on an object, a context switch is performed, and the owner of the object becomes the new active context. On the contrary, the invocation of a static method does not entail a context switch. Before executing a bytecode that accesses an object, the object's owner is checked against the currently active context in order to determine if access is allowed. Access is determined by the Firewall access control rules specified in the chapter Applet Isolation and Object Sharing of the [R7]. Those rules enable controlled sharing of objects through interface methods that the object's owner explicitly exports to other applet instances, and provided that the object's owner explicitly accepts to share it upon request of the method's invoker.

**SF\_GP\_DISPATCHER**

While a Security Domain is selected, this function tests for every command, according to the Security Domain life cycle state and the Card life cycle state, if security requirements are needed (if a Secure Channel is required).

**SF\_HARDWARE\_OPERATING**

When needed, at each start up or before first use, a self test of each hardware functional module is done, i.e.: DES, RSA, RNG implements a know calculus and checks if the result is correct. When executing, external hardware event can be triggered to prevent attacks or bad use. Temperature, frequency, voltage, light, glitch are considered as abnormal environmental conditions and put the card in frozen state. The TOE shall monitor IC detectors (e.g. out-of-range voltage, temperature, frequency, active shield, memory aging) and shall provide automatic answers to potential security violations through interruption routines that leave the device in a secure state.

The TOE with the IC has detectors of operational conditions. It shall resist to attackers with high-attack potential according to [R36] characterisation, in particular, to leakage attacks, intrusive (e.g. probing, fault injection) and non-intrusive (e.g. SPA, DPA, EMA) attacks, operational conditions manipulation (voltage, clock, temperature, etc) and physical attacks aiming at modification of the IC content or behaviour. To be compliant to related SUN Protection Profile [R5], the off-card verifier is mandatory in this ST; however, this TOE runs some additional verification at execution time. These verifications ensure that: 1. No read accesses are made to Java Card System code, data belonging to another application, data belonging to the Java Card System, 2. No write accesses are made to another application's code, Java Card System code, another application's data Java Card System or API data, 3. No execution of code is done from a method or from a method fragment belonging to another package (including execution on arbitrary data).

**SF\_KEY\_ACCESS**

This TSF enforces secure access to all cryptographic keys of the card: RSA keys, DES keys, EC keys, AES keys

**SF\_KEY\_AGREEMENT**

This TSF provides the applet instances with a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman [IEEE P1363].

**SF\_KEY\_DESTRUCTION**

This TSF disables the use of a key both logically and physically. When a key is cleared, the internal life cycle of the key container is moved to a state in which no operation is allowed. Applet instances may invoke this TSF through the interfaces declared in the javacard.security package of the Java Card API.

**SF\_KEY\_DISTRIBUTION**

This TSF enforces the distribution of all the cryptographic keys of the card using the method specified in that SFR.

**SF\_KEY\_GENERATION**

This TSF enforces the creation and/or the oncard generation of all the cryptographic keys of the card using the method specified in that SFR.

**SF\_KEY\_MANAGEMENT**

This function enables key sets management (PIN). It allows creating updating and deleting key sets. It is used to load keys to the card. It also implements verification of Key sets attributes: key lengths, key types... and enforces the loaded keys integrity

**SF\_MANUFACTURER\_AUTHENTICATION**

At prepersonalisation phase, manufacturer authentication at the beginning of a communication session is mandatory prior to any relevant data being transferred to the TOE. The manufacturer uses the random number returned in the INITIALIZE AUTHENTICATION PROCESS Command APDU. EXTERNAL AUTHENTICATE Command APDU is used to verify the cryptogram computed from the challenge by the manufacturer.

The max number of unsuccessful authentication attempts (basically 3) is described on the related FIA\_AFL.1/PP.

**SF\_MESSAGE\_DIGEST**

This TSF provides the applet instances with a mechanism for generating an (almost) unique value for a byte array content. That value can be used as a short representative of the information contained in the whole byte array. The hashing algorithms are available to the applets through the MessageDigest class of the Java Card API. Before generating the hash value, the TSF verifies that it has been provided with all the information necessary for the hashing operation. For those algorithms that do not pad the messages, the TSF checks that the information is block aligned before computing its hash value.

**SF\_MEMORY\_FAILURE**

When using the non volatile memory, in case of a bad writing, internal mechanisms are implemented to prevent an incoherency of the written data. In case of an impossible writing, an exception is raised.

**SF\_PREPERSONALISATION**

This function is in charge of pre-initializing the internal data structures, loading the configuration of the card and loading patch code, if needed. The patch contains its

identification elements that are used, during audit, to uniquely identify loaded code.

**SF\_RANDOM\_NUMBER**

This TSF provides to card manager, resident application, applets a mechanism for generating challenges and key values. Random number generators are available to the applets through the RandomData class of the Java Card API. Off-card entity authentication is achieved through the process of initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity. If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated). The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s).

**SF\_RESIDENT\_APPLICATION\_DISPATCHER**

During prepersonalisation phase, this function tests for every command if manufacturer authentication is required.

**SF\_RUNTIME\_VERIFIER**

This security functionality ensures the secure processing of information by ensuring the following elements:

- o Stack Control
- o Heap Control
- o Transient Control

Information on the processing is described on the related FDP\_ACF.1.

**SF\_SECURITY\_FUNCTIONS\_OF\_THE\_IC**

The TOE uses the security functions of the IC. The list of the security function is presented in the ST lite of the IC component.

**SF\_SIGNATURE**

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

The signature algorithms are available to the applets through the javacard.Signature class of the Java Card API, ISOSecureMessaging class and SecureChannel class. The length of the key to be used for the signature is defined by the applet instance when the key is created. Before generating the signature, the TSF verifies that the specified key is suitable for the operation (secret keys for signature generation), that it has been previously initialized, and that is in accordance with the specified signature algorithm (DES, RSA, etc). The TSF also checks that it has been provided with all the information necessary for the signature operation. For those algorithms that do not pad the messages, the TSF checks that the information to be signed is block aligned before performing the signature operation. Once the signature operation is performed, the internal TSF data used for the operation like the ICV is cleared. Signature operations are implemented to resist to environmental stress and glitches and include measures for preventing information leakage through covert channels.

Mechanisms of signature for Secure Messaging are available to the applets through the SecureChannel (Global Platform Card 2.2" specification) and ISOSecureMessaging

(Proprietary API) classes. The signature is included in Data Objects.

## SF\_UNOBSERVABILITY

This function assures that processing based on secure elements of the TOE does not reveal any information on those elements. For example, observation of a PIN verification cannot reveal the PIN value, observation a cryptographic computation cannot give information on the key.

## 8.2 SFRs and TSS

### 8.2.1 SFRs and TSS - Rationale

#### CoreG\_LC Security Functional Requirements

##### *Firewall Policy*

**FDP\_ACC.2/FIREWALL** The access control policy is ensured by SF\_FIREWALL, it controls whether an instance of an applet class declared in a package (subject) may read, write or execute an instance method (operations) of an object (object).

**FDP\_ACF.1/FIREWALL** FIREWALL Security attribute based access control -which security attributes is attached to which subject/object of the policy- is specified in the SF\_FIREWALL.

**FDP\_IFC.1/JCVM** This requirement is fulfilled by SF\_FIREWALL, this TSF enforces the information flow control rules of Firewall security policy. It controls whether an applet instance or javacard RE (subject) may store into persistent memory a reference of a global shared data container (objects).

**FDP\_IFF.1/JCVM** This requirement is fulfilled by SF\_FIREWALL. This TSF controls operations, based on current active context implemented in SF\_FIREWALL.

**FDP\_RIP.1/OBJECTS** SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION. The TSF clears the contents of the freshly allocated objects before releasing the object to the applet. On the TSF, memory is cleared when the object is removed during Garbage Collection. All this TSFI lead to Garbage Collection

**FMT\_MSA.1/JCRE** SF\_FIREWALL When an instance method is applied to an object, this TSF is in charge of performing a context switch to the context of the object's owner. The TSF is also in charge of dispatching the APDU commands to the applets instances installed on the card and keeping trace of which are the currently active ones.

**FMT\_MSA.1/JCVM** SF\_FIREWALL When an instance method is applied to an object, this TSF is in charge of performing a context switch to the context of the object's owner. The TSF is also in charge of dispatching the APDU commands to the applets instances installed on the card and keeping traces of which are the currently active ones.

**FMT\_MSA.2/FIREWALL\_JCVM** SF\_FIREWALL When an applet instance is selected for execution, this TSF initializes the currently active context with (the context of) that instance. Applet selection includes the verification that the instance actually exists on the card. Then, during the execution of the Java Card VM, this TSF propagates that secure value the other

security attributes involved in the Firewall policy (object's owner).

**FMT\_MSA.3/FIREWALL SF\_FIREWALL** The TSF initializes the security attributes of the Firewall and Java Card VM security policies when an applet instance is selected for execution, when an instance method is invoked and when an object is allocated. This TSF does not provide means for a subject to override those initial values.

**FMT\_MSA.3/JCVM SF\_FIREWALL.** The TSF initializes the security attributes of the Firewall and Java Card VM security policies when an applet instance is selected for execution, when an instance method is invoked and when an object is allocated. This TSF does not provide means for a subject to override those initial values.

**FMT\_SMF.1** This SFR is fulfilled by SF\_CARD\_CONTENT\_MANAGEMENT, when an instance method is applied to an object; this TSF is in charge of performing a context switch to the context of the object's owner.

**FMT\_SMR.1** This requirement is full filled by SF\_FIREWALL, this TSF uses a special value for the currently active context that identifies the Java Card RE (JCRE) and Java Card VM (JCVM).

*Application Programming Interface*

**FCS\_CKM.1** This requirement is fulfilled by SF\_KEY\_GENERATION. It enforces the creation and/or the oncard generation of all the cryptographic keys of the card.

**FCS\_CKM.2** This SFR is fulfilled by SF\_KEY\_DISTRIBUTION. It enforces the distribution of all the cryptographic keys of the card using the method specified in that SFR.

**FCS\_CKM.3** This SFR is implemented by SF\_KEY\_ACCESS. It enforces the access of all the cryptographic keys of the card using the method specified in that SFR

**FCS\_CKM.4** SF\_KEY\_DESTRUCTION fulfils this SFR, it enforces the destruction of all the cryptographic keys of the card using the method specified in that SFR.

**FCS\_COP.1** This SFR is verified by the following set of Security functionalities:

- o All signature and verification operation by RSA, TDES and AES are fulfilled by SF\_SIGNATURE, also fulfilled by SF\_KEY\_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms such EC Diffie-Hellman [R27].
- o This requirement by using SF\_ENCRYPTION\_AND\_DECRYPTION provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.
- o SF\_SIGNATURE permits to hash functions with SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512. It is also fulfilled by SF\_MESSAGE\_DIGEST by providing applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. Also fulfilled by SF\_KEY\_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman [R27].

**FDP\_RIP.1/ABORT** Any reference to an object instance created during an aborted transaction- see SF\_ATOMIC\_TRANSACTIONS- is cleaned by using

SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION.

**FDP\_RIP.1/APDU** The TSF SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION enforces the clearing of the previous contents of the APDU buffer before processing a new APDU.

**FDP\_RIP.1/bArray** The TSF SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION enforces the clearing of the previous contents of the buffer containing the installation data of an applet instance before installing a new one.

**FDP\_RIP.1/KEYS** In order to perform a cryptographic operation, the key involved in the operation has to be copied out of its secure container into the cryptographic buffer of the IC co-processor. This function is in charge of ensuring that such buffer is cleared immediately after completing the operation, the clearing is done by SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION.

**FDP\_RIP.1/TRANSIENT** This function is in charge of clearing the information contained in the transient objects when a clearing event arrives (deselection or card reset), invoked by SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION.

**FDP\_ROL.1/FIREWALL** SF\_ATOMIC\_TRANSACTION, when the operations specified are not completed, this TSF is in charge of setting back the state of the persistent memory as it was before they were started. As required in chapter 7 of the [R7] and the [R6], this TSF does not undo those modifications performed on the RAM, like the modification of the APDU buffer, the installation buffer, the transient objects, the try counters of the PINs and the reason code of the card exceptions. If the commit capacity is reached, this TSF prevents any further modification of the persistent memory.

**FDP\_ROL.1.1/FIREWALL** The TSF is implemented in the SF\_ATOMIC\_TRANSACTION it permits the rollback of the some **operations** on the java Card objects within the scope of the commands **select()**, **deselect()**, **process()**, **install()** or **uninstall()** call, **notwithstanding the restrictions given in [R7], §7.7, within the bounds of the Commit Capacity ([R7], §7.8), and those described in [R6].**

#### *Card Security Management*

**FAU\_ARP.1** The SF\_FIREWALL throws an instance of the SecurityException class when an attempt to violate a security policy rule is detected.

**FDP\_SDI.2** The TSF SF\_DATA\_INTEGRITY ensures integrity of PIN, Keys and application code (package)(CRC 16). A loss of integrity increases killcard counter.

**FPR\_UNO.1** The TSF SF\_UNOBSERVABILITY ensures no user is able to observe PIN values when authentication of the cardholder.

**FPT\_FLS.1** This SFR is enforced by the following TSF:

- o SF\_ATOMIC\_TRANSACTIONS: card tearing and power failures and abortion of a transaction in an unexpected context
- o SF\_FIREWALL: violations of the Firewall access control rules,
- o SF\_CARD\_CONTENT\_MANAGEMENT: insufficient resources to install a package and CAP file inconsistency errors.

- o **SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION:** ensures the erase of previous information stored, like the flags of pin or reason code contained in the CardException or CardRuntimeException.

**FPT\_TDC.1** This SFR is fulfilled by **F\_CARD\_CONTENT\_MANAGEMENT\_ENVIRONMENT**. It interprets cap files: bytes code and data arguments.

#### *AID Management*

**FIA\_ATD.1/AID** This SFR is fulfilled by **SF\_ARC\_CONTENT\_MANAGEMENT**: It controls the addition of new entries in the Applet Registry. Each time a new entry is added, the TSF controls that it contains the information specified in that SFR. This is done on package loading and applet installation.

**FIA\_UID.2/AID** The TSF **SF\_FIREWALL** identifies the applet instance requesting access to objects through the currently active context. Retrieving the currently active context always precedes the execution of the bytecodes under the control of the Firewall, as this information is required for checking the premises of its access control rules.

**FIA\_USB.1/AID** The TSF **SF\_FIREWALL** uses the security attribute introduced in the SFR to check whether an applet instance (subject) representing an Application Provider (user) may access an object through the firewall.

**FMT\_MTD.1/JCRE** **SF\_CARD\_CONTENT\_MANAGEMENT** fulfils this SFR, it controls the creation of new applet instances on the card. Each time an applet instance is created, the Installer adds an entry for it in the Applet Registry

**FMT\_MTD.3/JCRE** This SFR is fulfilled by **SF\_CARD\_CONTENT\_MANAGEMENT**: it controls that only secure values are assigned as attributes of an applet instance. Invalid AIDs for the applet instances, like an AID that is already in use, are also rejected

#### *InstG Security Functional Requirements*

**FDP\_ITC.2/Installer** This SFR is implemented by **SF\_CARD\_CONTENT\_MANAGEMENT**: The SF ensures safe package loading and applet installation process. It modifies the CAP files to produce the TOE intern representation of the loaded package. It also performs coherency checks on the CAP files and verifies the export references.

**FMT\_SMR.1/Installer** This SFR is implemented by **SF\_CARD\_CONTENT\_MANAGEMENT**: The TSF is in charge of creating the applet instance that plays the role of the Applet Installation Manager.

**FPT\_FLS.1/Installer** This SFR is fulfilled by the following SF:

- o The **SF\_CARD\_CONTENT\_MANAGEMENT**: is in charge of checking that all the conditions for safely installing a package or an applet instance are fulfilled during the installation procedure. If conditions cannot be verified the installation is deemed unsuccessful and either an exception is thrown or the card is frozen, depending of the failure severity. Card tearing or reset also cause an installation failure.
- o **SF\_ATOMIC\_TRANSACTIONS** is in charge of rolling back to a secure state when the installation of a package or an applet instance is aborted



- o This function is in charge of clearing the information contained in the packages that is not necessary for the execution of the code of the applet invoked by SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION.

**FPT\_RCV.3/Installer** This SFR is fulfilled by the following SF:

- o SF\_CARD\_CONTENT\_MANAGEMENT: In case of severe failure during package or applet installation, the card is frozen (KillCard). Such failures (for example the loading of a CAP file with an invalid format) are considered as security problems. The maintenance mode is represented by the frozen state of the card. The secure state is then reached on next card reset where Garbage Collector is launch to retrieve lost memory and where the transaction mechanism allows retrieving the initial state.
- o SF\_ATOMIC\_TRANSACTION: The TSF is in charge of rolling back to a secure state when the installation of a package or an applet instance is aborted.

#### ADELG Security Functional Requirements

**FDP\_ACC.2/ADEL** The access control policy for deletion is made by SF\_CARD\_CONTENT\_MANAGEMENT, it controls whether the Applet Deletion Manager (subject) may delete (operation) a package or an applet instance (object).

**FDP\_ACF.1/ADEL** The access control policy for deletion is made by SF\_CARD\_CONTENT\_MANAGEMENT, it controls whether the Applet Deletion Manager (subject) may delete (operation) a package or an applet instance (object).

**FDP\_RIP.1/ADEL** The TSF SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION renders inaccessible the code of a deleted package and the class instances and arrays allocated by a deleted applet instance.

**FMT\_MSA.1/ADEL** The ADEL access policy is implemented in SF\_CARD\_CONTENT\_MANAGEMENT, this TSF keeps track of which applet instances are currently active on which logical channels. Only the Card Manager (which in [R5] is identified with the Java Card RE role) is allowed to associate or remove the association between an applet instance and a logical channel. These actions are performed as part of command dispatching

**FMT\_MSA.3/ADEL** The SF\_CARD\_CONTENT\_MANAGEMENT enforces the assignment of restrictive values for the security attributes of the Applet Deletion policy.

**FMT\_SMF.1/ADEL** Modifying the active applet security context is done by SF\_CARD\_MANAGEMENT\_ENVIRONMENT, it's allowed to card manager.

**FMT\_SMR.1/ADEL** This SFR is fulfilled by SF\_CARD\_MANAGEMENT\_ENVIRONMENT: it keeps track of which applet instances are currently active on which logical channels. Only the Card Manager is allowed to associate or remove the association between an applet instance and a logical channel.

**FPT\_FLS.1/ADEL** This SFR is ensured by the following TSF:

- o SF\_CARD\_CONTENT\_MANAGEMENT is in charge of checking that all the conditions for safely deleting a package or an applet instance are fulfilled before starting the deletion procedure.

- o **SF\_ATOMIC\_TRANSACTION**: This TSF is in charge of rolling back to a secure state when the deletion of a package or an applet instance is aborted.
- o **SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION**: is in charge of checking that all the conditions for safely deleting a package or an applet instance are fulfilled before starting the deletion procedure.

#### ODELG Security Functional Requirements

**FDP\_RIP.1/ODEL** This SFR is met by:

- o **SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION**: This TSF renders inaccessible the code of a deleted package and the class instances and arrays allocated by a deleted applet instance.

**FPT\_FLS.1/ODEL** The TSF **SF\_CLEARING\_OF\_SENSITIVE\_INFORMATION** is in charge of checking that all the conditions for safely deleting a package or an applet instance are fulfilled before starting the deletion procedure.

#### CarG Security Functional Requirements

**FCO\_NRO.2/CM** During the loading phase, the **SF\_CARD\_CONTENT\_MANAGEMENT**: controls card content loading, it verifies the proof of the origin of the Load File. Before to start the loading, the open checks that the user is authenticated, checks the presence of the < DAPBlock > in the < LoadFile >, requires the Security Domain Verifier to verify it.

**FDP\_IFC.2/CM** The rule of the package loading flow control policy is specified by **SF\_CARD\_CONTENT\_MANAGEMENT**: it verifies that all the loading commands are issued in the Secure Channel session. It compares the Load File Data Block Hash present in the command install for load against the received. It also requires the Dap verification of all entities committed in the loading phase, ensured by **SF\_DAP\_VERIFICATION**.

**FDP\_IFF.1/CM** This SFR is implemented by **SF\_DAP\_VERIFICATION**, it controls the communication protocol used by the CAD and the card for transmitting packages. The SFR is also implemented in the **SF\_CARD\_CONTENT\_MANAGEMENT** to ensure the access control policy for the loading of the packages.

**FDP\_UIT.1/CM** This SFR is implemented by **SF\_DAP\_VERIFICATION**, it controls imported data from modification, deletion, insertion, replay of some of the pieces of the application sent by the CAD. The verification is made by using: Encryption and decryption operations by **SF\_ENCRYPTION\_AND\_DECRYPTION** function.

**FIA\_UID.1/CM** The Security Functionality **SF\_GP\_DISPATCHER** met this SFR: While the Card manager (ISD) or Supplementary Security domain is selected, these functions test for every command if the secure channel is open. When the secure channel is not open then only these commands are available: Get data and Initialize Update. The initialize Update returns to the user the key set version, Secure Channel identifier and the card random and the card cryptogram.

**FMT\_MSA.1/CM** This SFR is implemented by two security functions:  
**SF\_KEY\_MANAGEMENT**: The TSF controls that only the CM can modify its key set and can change the card life cycle and set the default application  
**SF\_CARD\_CONTENT\_MANAGEMENT**: This TSF controls whether the active entity has

the privilege and the pre-authorization for make the Card Content Management operations, and that operation still available on the card. Its controls also that the card state allows the operations.

**FMT\_MSA.3/CM** The TSF SF\_CARD\_CONTENT\_MANAGEMENT provides the way to lock the Security Domain with Authorized Management privilege in order to restrict its card content management ability. This TSF provides also to disable permanently the Card Content Management operations for all entities on the card.

**FMT\_SMF.1/CM** The TSF SF\_CARD\_CONTENT\_MANAGEMENT controls whether the active entity has the privilege and the pre-authorization for making the Card Content Management operations - modify security attributes. -, and that operation still available on the card. Its controls also that the card state allows the operations.

**FMT\_SMR.1/CM** The TSF SF\_CARD\_CONTENT\_MANAGEMENT verifies that authentication is successful and the active entity has loading privilege (Authorized Management privilege) before processes any Card Content management command. The successful authentication proves the user identity and role.

**FTP\_ITC.1/CM** Installing a new package is verified by SF\_CARD\_CONTENT\_MANAGEMENT: the SF\_GP\_DISPATCHER tests if secure channel is required, and verification is made by SF\_DAP\_VERIFICATION.

#### Additional Security Functional Requirements

##### *Additional Security Functional Requirements for CM*

**FCO\_NRO.2/CM\_DAP** During the loading phase, SF\_DAP\_VERIFICATION verifies the proof of the origin of the Load File. Before to start the loading, the open checks that the user is authenticated, checks the presence of the < DAPBlock > in the < LoadFile >, requires the Security Domain Verifier to verify it.

**FIA\_AFL.1/CM** This Requirement is fulfilled by SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL. It tests the result of authentication. By default the authentication result is assumed unsuccessful, so the authentication failure is recorded, the associated counter and the slowdown counter are incremented. If the authentication is successful, the authentication failure counter is decremented and the slowdown counter is reset.

**FIA\_UAU.1/CM** This SFR is implemented by the following TSF:

- o SF\_GP\_DISPATCHER: While the Card manager (ISD) or Supplementary Security domain is selected, these functions test for every command by SF\_GP\_DISPATCHER if the secure channel is open.
- o SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL: When the secure channel is not open then only the command available are Get Data, Initialize Update, Select.

**FIA\_UAU.4/CardIssuer** Present the use of CardIssuer authentication, function implemented in SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL, is given by using a RNG defined



in SF\_RANDOM\_NUMBER.

**FIA\_UAU.7/CardIssuer** This SFR is implemented by the following TSF:

- o SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL: It uses the key set version, Secure channel identifier and the card random and the card cryptogram for authentication.
- SF\_RANDOM\_NUMBER: It permits CardIssuer Protected authentication feedback, no other information is given while the authentication is in progress.

**FPR\_UNO.1/Key\_CM** Import of keys are not observable by all subjects. This requirement is ensured by SF\_KEY\_MANAGEMENT and SF\_UNOBSERVABILITY.

**FPT\_TDC.1/CM** Key set and packages when imported are consistently interpreted by implementation of SF\_KEY\_MANAGEMENT.

**FMT\_SMR.2/CM** The TSF SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL verifies that authentication is successful and the active entity has loading privilege before processes any Card Content management command. The successful authentication proves the user identity and role.

**FCS\_COP.1/CM** The TSF SF\_ENTITY\_AUTHENTICATION/SECURE\_CHANNEL covers this SFR. It requires the cryptographic operations for the creation and management of secure channel. The TSF SF\_ENCRYPTION\_AND\_DECRYPTION provides a mechanism for encrypting and decrypting the contents of a byte array.

*Additional Security Functional Requirements for Resident application*

**FDP\_ACC.2/PP** The SFR is implemented by the following TSF:

- o SF\_RESIDENT\_APPLICATION\_DISPATCHER: This TSF implements Access control policy for the resident application,
- o SF\_MANUFACTURER\_AUTHENTICATION: This TSF is in charge of the card manufacturer authentication.

These TSF controls all access to all objects and all operations.

**FPT\_TST.1** This SFR is supported by the following TSF:

- o SF\_HARDWARE\_OPERATING: At each start up, security function SF\_Hardware\_Operating is done. Random, DES, and CRC functional modules systematically tested: a known calculus is implemented and the result is checked. SHA, RSA, AES and ECC functional modules are tested at each start up or at first use, using the same method.

SF\_DATA\_INTEGRITY: At each start up, the entire NVM integrity, so executable code, is checked. The NVM integrity is updated after patch loading so the next startup does not rise a kill card exception.

**FDP\_ACF.1/PP** This SFR is met by the following TSF:

- o SF\_MANUFACTURER\_AUTHENTICATION: The rules granting access rights are made by SF\_MANUFACTURER\_AUTHENTICATION, it guarantees once the Prepersonalisation is authenticated, the card verifies for each action that the authentication is successful

**FDP\_UCT.1/PP** This SFR is met by the following TSF:

- SF\_MANUFACTURER\_AUTHENTICATION: To transmit the InitKey (ISK) to Card Manager, user or subject must be successfully authenticated with the MSK. The transmitted InitKey is protected from disclosure by being ciphered by the MSK
- SF\_PREPERSONALISATION: This TSF is in charge of the prepersonalisation stage, this includes the patch loading.

**FDP\_ITC.1/PP** The SFR is implemented by the following TSF:

- SF\_MANUFACTURER\_AUTHENTICATION enables trusted channel establishment thanks to authentication with the MSK and allows the prepersonalisation.
- SF\_PREPERSONALISATION enables to load patches and locks.
- SF\_RESIDENT\_APPLICATION\_DISPATCHER During prepersonalisation phase, this function tests for every command if manufacturer authentication is required

**FIA\_AFL.1/PP** The SFR is implemented by the following TSF:

- SF\_MANUFACTURER\_AUTHENTICATION: After 3 consecutive unsuccessful authentications a status error is always returned by the card.

**FIA\_UAU.1/PP** The SFR is met by the following SF:

- SF\_RESIDENT\_APPLICATION\_DISPATCHER: The set of command (INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, SELECT APPLLET) of the resident application can be performed without authentication.

**FIA\_UID.1/PP** This SFR is implemented by the following TSF:

- SF\_RESIDENT\_APPLICATION\_DISPATCHER: The set of command (INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, SELECT APPLLET) of the resident application can be performed without authentication.

**FMT\_MSA.1/PP** This SFR is implemented by the following TSF:

- SF\_MANUFACTURER\_AUTHENTICATION: The MSK keys of the Card Manufacturer can be modified in Prepersonalisation phase after a successful authentication with the MSK.

**FMT\_SMF.1/PP** This SFR is implemented by the following TSF:

- SF\_MANUFACTURER\_AUTHENTICATION: The MSK keys of the Card Manufacturer can be modified in Prepersonalisation phase after a successful authentication with the MSK.

**FIA\_UAU.4/CardManu** This SFR is implemented by the following TSF:

- SF\_MANUFACTURER\_AUTHENTICATION: Prevents from Card Manufacturer authentication reuse during prepersonalisation

**FIA\_UAU.7/CardManu** This SFR is implemented by the following SFR:

- SF\_MANUFACTURER\_AUTHENTICATION: During authentication, the command "INITIALIZE AUTHENTICATION PROCESS" is used to provide a random number implemented by the SF\_RANDOM\_NUMBER.
- Only Random number and NOK as result of authentication are provided to the user.

**FMT\_MOF.1/PP** This SFR is implemented by the following TSF:

- o **SF\_CARD\_MANAGEMENT\_ENVIRONMENT**: Some commands and features of the resident application are active only during the Prepersonalisation Phase, such as the patch and lock loading. As soon as the Card Manager status is set to OP\_READY this phase stops, and the commands and features are irreversibly disabled.
- o **SF\_PREPERSONALISATION**: This function permits on Prepersonalisation phase to load patch code on the card.
- o **SF\_RESIDENT\_APPLICATION\_DISPATCHER** : During prepersonalisation phase, this function tests for every command if manufacturer authentication is required as commands used to load patches on the card.

**FMT\_SMR.2/PP** This SFR is implemented by the following TSF:

- o **SF\_PREPERSONALISATION**: This function permits on Prepersonalisation phase to load patch code on the card.
- o **SF\_MANUFACTURER\_AUTHENTICATION**: After a successful authentication (of Card Manufacturer) using MSK, the TSF and card stay still in Prepersonalisation state.
- o **SF\_RESIDENT\_APPLICATION\_DISPATCHER**: During prepersonalisation phase, this function tests for every command according to the life cycle of the resident application

**FMT\_MSA.3/PP** This SFR is implemented by the following TSF:

- o **SF\_MANUFACTURER\_AUTHENTICATION**: This TSF implements Access control policy in prepersonalisations phase.
- o **SF\_PREPERSONALISATION**: This TSF, once authenticated, implements prepersonalisations operations.

**FCS\_COP.1/PP** At prepersonalisation phase, authentication cryptogram (signature computation and verification) are used by SF\_MANUFACTURER\_AUTHENTICATION. Data decryption (of patch, locks or keys), integrity pattern verification (signature/MAC) are used by SF\_PREPERSONALISATION. Encrypted and decrypted data in bytes arrays are manipulated using SF\_ENCRYPTION\_AND\_DECRYPTION. These functions call Cryptographic ones defined in previous FCS\_COP operation

**SF\_MANUFACTURER\_AUTHENTICATION**: This TSF implements Access control policy in prepersonalisations phase.

**FCS\_CKM.4/PP** This SFR is implemented by the following TSF:

- o **SF\_KEY\_DESTRUCTION**: As soon as the Card Manager status is set to OP\_READY, the MSK and LSK key is set to null (as the checksum is also to null) the key is not useful. The MSK after the first use is diversified, according to AGD\_PRE [R39]. The new version of the key replaces the previous one.
- o



**FDP\_UIT.1/PP** The SFR is implemented by the following TSF:

- o **SF\_PREPERSONALISATION:** The data (patch or locks) sent to the TOE are protected in integrity thanks to a signature computed by the TOE developer with the LSK (PUT KEY command). The ISK is loaded ciphered with the MSK, and requires an authentication with the MSK.

**FCS\_CKM.1/PP** The SFR is implemented by the following TSF:

- o **SF\_KEY\_GENERATION:** During first command, the individual MSK of the TOE is generated from the master MSK.

**FAU\_STG.2** The SFR is implemented by the following TSF:

- o **SF\_PREPERSONALISATION:** Upon request, the identification of the patch is returned.

*Additional Security Functional Requirements for SmartCard Platform*

**FPT\_PHP.3/SCP** When executing, **SF\_HARDWARE\_OPERATING** shall resist changing operational conditions every times, external hardware event can be triggered to prevent attacks or bad use. Temperature, frequency, voltage, light, glitch are considered as abnormal environmental conditions and put the card in frozen state.

**FPT\_RCV.4/SCP** The TSF **SF\_DATA\_COHERENCY** shall ensure that reading from and writing to static and objects' fields interrupted by power loss have the property that the function either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

**FRU\_FLT.1/SCP** When there is a failure of FLASH, in case of a bad writing, the **SF\_MEMORY\_FAILURE** implements internal mechanisms to prevent an incoherency of the written data. In case of an impossible writing, an exception is raised.

**FPR\_UNO.1/USE\_KEY** This SFR is implemented by the following TSF:

- o **SF\_UNOBSERVABILITY:** No user are able to observe keys whether the keys are in use.

**FCS\_RNG.1/SCP** This SFR is implemented by the following TSF:

- o **SF\_SECURITY\_FUNCTIONS\_OF\_THE\_IC:** This TSF ensures that the security functionalities from the chip are provided to the software, and in particular RNG based on AIS31.
- o **SF\_RANDOM\_NUMBER:** This TSF is in charge of providing random numbers.

*Additional Security Functional Requirements for the applets*

**FIA\_AFL.1/PIN** This SFR is implemented by the following TSF:

**SF\_CARDHOLDER\_VERIFICATION:** The TSF detects that the number of PIN presentations exceeds the maximum value previously configured. It blocks the PIN in this case. The entire OwnerPin class is involved in the process since it is used to set the PIN size and the maximum of successful tries, to verify the PIN, to reset the validation flag.



**FMT\_MTD.2/GP\_PIN** This SFR is implemented by the following TSF:

- o **SF\_CARDHOLDER\_VERIFICATION**: The TOE ensures that the GlobalPin is blocked when its associated PIN try counter at reach the PIN try limit value.

**FDP\_SDI.2/Applet** This SFR is implemented by the following SFR:

- o **SF\_UNOBSERVABILITY**: Unobservability of Comparison on two byte arrays is a service provided to the applet.

**FMT\_MTD.1/PIN** The SFR is implemented by the following TSF:

- o **SF\_CARDHOLDER\_VERIFICATION** provides a complete PIN mechanism: change\_default, query and modify to applets.

**FIA\_AFL.1/GP\_PIN** This SFR is implemented by the following TSF:

- o **SF\_CARDHOLDER\_VERIFICATION**: The TOE checks that only the Card Manager and privileged application can change the pin try limit and Update the global Pin.

**FDP\_ACC.2/RV\_Stack** This SFR is implemented by the following TSF:

**SF\_RUNTIME\_VERIFIER**: The SF implements a complete access control on the Stack operations.

**FDP\_ACF.1/RV\_Stack** This SFR is implemented by the following TSF:

- o **SF\_RUNTIME\_VERIFIER**: This SFR enforces the access conditions which guarantee the protection of the Stack.

**FMT\_MSA.1/RV\_Stack** This SFR is implemented by the following TSF:

- o **SF\_RUNTIME\_VERIFIER**: This TSF implements the management of the security attributes.

**FMT\_MSA.2/RV\_Stack** This SFR is implemented by the following TSF:

**SF\_RUNTIME\_VERIFIER**: This TSF ensures that only secure values for the attributes are accepted **FMT\_MSA.3/RV\_Stack** This SFR is implemented by the following TSF:

- o **SF\_RUNTIME\_VERIFIER**: This TSF implements the initialisation of the attributes of the access control policy.

**FMT\_SMF.1/RV\_Stack** This SFR is implemented by the following TSF:

- o **SF\_RUNTIME\_VERIFIER**: The TSF specify the management function to modify the stack pointer. It controls the Stack and is able to change the associated parameter.

**FDP\_ACC.2/RV\_Heap** This SFR is implemented by the following TSF:

- o **SF\_RUNTIME\_VERIFIER**: it implements the access control to the Heap. ,

**FDP\_ACF.1/RV\_Heap** This SFR is implemented by the following TSF:

- o **SF\_RUNTIME\_VERIFIER**: it ensures the access conditions to the Heap.

**FMT\_MSA.1/RV\_Heap** This SFR is implemented by the following TSF:

**SF\_RUNTIME\_VERIFIER**: This TSF implements the management of the modification of the security attributes of the access control to the Heap.

**FMT\_MSA.2/RV\_Heap** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: This TSF implements the control that only security values are accepted for the security attributes..

**FMT\_MSA.3/RV\_Heap** This SFR is implemented by the following TSF:

SF\_RUNTIME\_VERIFIER: This TSF implements initialisation of the security attributes.

**FMT\_SMF.1/RV\_Heap** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: The TSF directly controls the Heap and is able to change the associated parameter.

**FDP\_ACC.2/RV\_Transient** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: it implements the access control to guarantee the protection of Transient objects.

**FDP\_ACF.1/RV\_Transient** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: it implements access conditions and defines the security rules which guarantee the protection of Transient objects.

**FMT\_MSA.1/RV\_Transient** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: This TSF implements the management of the security attributes for the access control to the transient..

**FMT\_MSA.2/RV\_Transient** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: This TSF implements the condition that only secure attributes are accepted for the access control policy to the Transient.

**FMT\_MSA.3/RV\_Transient** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: This TSF controls that only restrictive values are accepted for security attributes used to enforce the transient access control policy.

**FMT\_SMF.1/RV\_Transient** This SFR is implemented by the following TSF:

- SF\_RUNTIME\_VERIFIER: The TSF directly controls the Transient and is able to change the associated parameter.



## 8.2.2 Association tables of SFRs and TSS

| } } } }

Security Functional Requirements	TOE Summary Specification
<a href="#">FDP_ACC.2/FIREWALL</a>	<a href="#">SF FIREWALL</a>
<a href="#">FDP_ACF.1/FIREWALL</a>	<a href="#">SF FIREWALL</a>
<a href="#">FDP_IFC.1/JCVM</a>	<a href="#">SF FIREWALL</a>
<a href="#">FDP_IFF.1/JCVM</a>	<a href="#">SF FIREWALL</a>
<a href="#">FDP_RIP.1/OBJECTS</a>	<a href="#">SF CLEARING OF SENSITIVE INFORMATION</a>
<a href="#">FMT_MSA.1/JCRE</a>	<a href="#">SF FIREWALL</a>
<a href="#">FMT_MSA.1/JCVM</a>	<a href="#">SF FIREWALL</a>
<a href="#">FMT_MSA.2/FIREWALL_JCVM</a>	<a href="#">SF FIREWALL</a>
<a href="#">FMT_MSA.3/FIREWALL</a>	<a href="#">SF FIREWALL</a>
<a href="#">FMT_MSA.3/JCVM</a>	<a href="#">SF FIREWALL</a>
<a href="#">FMT_SMF.1</a>	<a href="#">SF CARD CONTENT MANAGEMENT</a>
<a href="#">FMT_SMR.1</a>	<a href="#">SF FIREWALL</a>
<a href="#">FCS_CKM.1</a>	<a href="#">SF KEY GENERATION</a>
<a href="#">FCS_CKM.2</a>	<a href="#">SF KEY DISTRIBUTION</a>
<a href="#">FCS_CKM.3</a>	<a href="#">SF KEY ACCESS</a>
<a href="#">FCS_CKM.4</a>	<a href="#">SF KEY DESTRUCTION</a>
<a href="#">FCS_COP.1</a>	<a href="#">SF KEY AGREEMENT, SF MESSAGE DIGEST, SF ENCRYPTION AND DECRYPTION, SF SIGNATURE</a>
<a href="#">FDP_RIP.1/ABORT</a>	<a href="#">SF CLEARING OF SENSITIVE INFORMATION, SF ATOMIC TRANSACTION,</a>
<a href="#">FDP_RIP.1/APDU</a>	<a href="#">SF CLEARING OF SENSITIVE INFORMATION</a>
<a href="#">FDP_RIP.1/bArray</a>	<a href="#">SF CLEARING OF SENSITIVE INFORMATION</a>
<a href="#">FDP_RIP.1/KEYS</a>	<a href="#">SF CLEARING OF SENSITIVE INFORMATION</a>
<a href="#">FDP_RIP.1/TRANSIENT</a>	<a href="#">SF CLEARING OF SENSITIVE INFORMATION</a>
<a href="#">FDP_ROL.1/FIREWALL</a>	<a href="#">SF ATOMIC TRANSACTION</a>
<a href="#">FAU_ARP.1</a>	<a href="#">SF FIREWALL</a>
<a href="#">FDP_SDI.2</a>	<a href="#">SF DATA INTEGRITY</a>
<a href="#">FPR_UNO.1</a>	<a href="#">SF UNOBSERVABILITY</a>
<a href="#">FPT_FLS.1</a>	<a href="#">SF FIREWALL, SF ATOMIC TRANSACTION, SF CARD CONTENT MANAGEMENT, SF CLEARING OF SENSITIVE INFORMATION</a>
<a href="#">FPT_TDC.1</a>	<a href="#">SF CARD MANAGEMENT ENVIRONMENT</a>
<a href="#">FIA_ATD.1/AID</a>	<a href="#">SF CARD CONTENT MANAGEMENT</a>
<a href="#">FIA_UID.2/AID</a>	<a href="#">SF FIREWALL</a>

| } } } }

Security Functional Requirements	TOE Summary Specification
<a href="#">FIA_USB.1/AID</a>	<a href="#">SF_FIREWALL</a>
<a href="#">FMT_MTD.1/JCRE</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FMT_MTD.3/JCRE</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FDP_ITC.2/Installer</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FMT_SMR.1/Installer</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FPT_FLS.1/Installer</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT,</a> <a href="#">SF_ATOMIC_TRANSACTION,</a> <a href="#">SF_CLEARING_OF_SENSITIVE_INFORMATION</a>
<a href="#">FPT_RCV.3/Installer</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT,</a> <a href="#">SF_ATOMIC_TRANSACTION</a>
<a href="#">FDP_ACC.2/ADEL</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FDP_ACF.1/ADEL</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FDP_RIP.1/ADEL</a>	<a href="#">SF_CLEARING_OF_SENSITIVE_INFORMATION</a>
<a href="#">FMT_MSA.1/ADEL</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FMT_MSA.3/ADEL</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FMT_SMF.1/ADEL</a>	<a href="#">SF_CARD_MANAGEMENT_ENVIRONMENT</a>
<a href="#">FMT_SMR.1/ADEL</a>	<a href="#">SF_CARD_MANAGEMENT_ENVIRONMENT</a>
<a href="#">FPT_FLS.1/ADEL</a>	<a href="#">SF_CLEARING_OF_SENSITIVE_INFORMATION,</a> <a href="#">SF_ATOMIC_TRANSACTION,</a> <a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FDP_RIP.1/ODEL</a>	<a href="#">SF_CLEARING_OF_SENSITIVE_INFORMATION</a>
<a href="#">FPT_FLS.1/ODEL</a>	<a href="#">SF_CLEARING_OF_SENSITIVE_INFORMATION</a>
<a href="#">FCO_NRO.2/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FDP_IFC.2/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT,</a> <a href="#">SF_DAP_VERIFICATION</a>
<a href="#">FDP_IFT.1/CM</a>	<a href="#">SF_DAP_VERIFICATION,</a> <a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FDP_UIT.1/CM</a>	<a href="#">SF_DAP_VERIFICATION,</a> <a href="#">SF_ENCRYPTION_AND_DECRYPTION</a>
<a href="#">FIA_UID.1/CM</a>	<a href="#">SF_GP_DISPATCHER</a>
<a href="#">FMT_MSA.1/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT,</a> <a href="#">SF_KEY_MANAGEMENT</a>
<a href="#">FMT_MSA.3/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FMT_SMF.1/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>
<a href="#">FMT_SMR.1/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a>



Security Functional Requirements	TOE Summary Specification
<a href="#">FTP_ITC.1/CM</a>	<a href="#">SF_CARD_CONTENT_MANAGEMENT</a> , <a href="#">SF_DAP_VERIFICATION</a> , <a href="#">SF_GP_DISPATCHER</a> ,
<a href="#">FPT_TST.1</a>	<a href="#">SF_HARDWARE_OPERATING</a> , <a href="#">SF_DATA_INTEGRITY</a>
<a href="#">FCO_NRO.2/CM_DAP</a>	<a href="#">SF_DAP_VERIFICATION</a>
<a href="#">FIA_AFL.1/CM</a>	<a href="#">SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL</a>
<a href="#">FIA_UAU.1/CM</a>	<a href="#">SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL</a> , <a href="#">SF_GP_DISPATCHER</a>
<a href="#">FIA_UAU.4/CardIssuer</a>	<a href="#">SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL</a> , <a href="#">SF_RANDOM_NUMBER</a>
<a href="#">FIA_UAU.7/CardIssuer</a>	<a href="#">SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL</a> , <a href="#">SF_RANDOM_NUMBER</a>
<a href="#">FPR_UNO.1/Key_CM</a>	<a href="#">SF_KEY_MANAGEMENT</a> , <a href="#">SF_UNOBSERVABILITY</a>
<a href="#">FPT_TDC.1/CM</a>	<a href="#">SF_KEY_MANAGEMENT</a>
<a href="#">FMT_SMR.2/CM</a>	<a href="#">SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL</a>
<a href="#">FDP_ACC.2/PP</a>	<a href="#">SF_RESIDENT_APPLICATION_DISPATCHER</a> , <a href="#">SF_MANUFACTURER_AUTHENTICATION</a> ,
<a href="#">FDP_ACF.1/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a>
<a href="#">FDP_UCT.1/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a> , <a href="#">SF_PREPERSONALISATION</a> , <a href="#">SF_RESIDENT_APPLICATION_DISPATCHER</a>
<a href="#">FDP_ITC.1/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a> , <a href="#">SF_RESIDENT_APPLICATION_DISPATCHER</a> <a href="#">SF_PREPERSONALISATION</a>
<a href="#">FIA_AFL.1/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a>
<a href="#">FIA_UAU.1/PP</a>	<a href="#">SF_RESIDENT_APPLICATION_DISPATCHER</a>
<a href="#">FIA_UID.1/PP</a>	<a href="#">SF_RESIDENT_APPLICATION_DISPATCHER</a>
<a href="#">FMT_MSA.1/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a>
<a href="#">FMT_SMF.1/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a>
<a href="#">FIA_UAU.4/CardManu</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a>
<a href="#">FIA_UAU.7/CardManu</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a>
<a href="#">FMT_MOF.1/PP</a>	<a href="#">SF_CARD_MANAGEMENT_ENVIRONMENT</a> , <a href="#">SF_PREPERSONALISATION</a>
<a href="#">FMT_SMR.2/PP</a>	<a href="#">SF_MANUFACTURER_AUTHENTICATION</a> , <a href="#">SF_RESIDENT_APPLICATION_DISPATCHER</a> , <a href="#">SF_PREPERSONALISATION</a>

|                    }                    }                    }                    }

Security Functional Requirements	TOE Summary Specification
<a href="#">FMT_MSA.3/PP</a>	<a href="#">SF MANUFACTURER AUTHENTICATION,</a> <a href="#">SF PREPERSONALISATION</a>
<a href="#">FCS_COP.1/PP</a>	<a href="#">SF PREPERSONALISATION,</a> <a href="#">SF MANUFACTURER AUTHENTICATION,</a> <a href="#">SF ENCRYPTION AND DECRYPTION</a>
<a href="#">FCS_CKM.4/PP</a>	<a href="#">SF KEY DESTRUCTION</a>
<a href="#">FPT_PHP.3/SCP</a>	<a href="#">SF HARDWARE OPERATING</a>
<a href="#">FPT_RCV.4/SCP</a>	<a href="#">SF DATA COHERENCY</a>
<a href="#">FRU_FLT.1/SCP</a>	<a href="#">SF MEMORY FAILURE</a>
<a href="#">FPR_UNO.1/USE_KEY</a>	<a href="#">SF UNOBSERVABILITY</a>
<a href="#">FCS_RNG.1/SCP</a>	<a href="#">SF RANDOM NUMBER,</a> <a href="#">SF SECURITY FUNCTIONS OF THE IC</a>
<a href="#">FIA_AFL.1/PIN</a>	<a href="#">SF CARDHOLDER VERIFICATION</a>
<a href="#">FMT_MTD.2/GP_PIN</a>	<a href="#">SF CARDHOLDER VERIFICATION</a>
<a href="#">FDP_SDI.2/Applet</a>	<a href="#">SF UNOBSERVABILITY</a>
<a href="#">FMT_MTD.1/PIN</a>	<a href="#">SF CARDHOLDER VERIFICATION</a>
<a href="#">FIA_AFL.1/GP_PIN</a>	<a href="#">SF CARDHOLDER VERIFICATION</a>
<a href="#">FDP_ACC.2/RV Stack</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FDP_ACF.1/RV Stack</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.1/RV Stack</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.2/RV Stack</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.3/RV Stack</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_SMF.1/RV Stack</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FDP_ACC.2/RV Heap</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FDP_ACF.1/RV Heap</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.1/RV Heap</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.2/RV Heap</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.3/RV Heap</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_SMF.1/RV Heap</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FDP_ACC.2/RV Transient</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FDP_ACF.1/RV Transient</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.1/RV Transient</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.2/RV Transient</a>	<a href="#">SF RUNTIME VERIFIER</a>
<a href="#">FMT_MSA.3/RV Transient</a>	<a href="#">SF RUNTIME VERIFIER</a>

| } } } }

Security Functional Requirements	TOE Summary Specification
<a href="#">FMT_SMF.1/RV_Transient</a>	<a href="#">SF_RUNTIME_VERIFIER</a>
<a href="#">FAU_STG.2</a>	<a href="#">SF_PREPERSONALISATION</a>
<a href="#">FCS_CKM.1/PP</a>	<a href="#">SF_KEY_GENERATION</a>
<a href="#">FDP_UIT.1/PP</a>	<a href="#">SF_PREPERSONALISATION</a>
<a href="#">FCS_COP.1/CM</a>	<a href="#">SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, SF_ENCRYPTION_AND_DECRYPTION</a>

Table 16: SFRs and TSS – Coverage

TOE Summary Specification	Security Functional Requirements
<a href="#">SF ATOMIC TRANSACTION</a>	<a href="#">FPT_FLS.1/Installer</a> , <a href="#">FPT_RCV.3/Installer</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_ROL.1/FIREWALL</a> , <a href="#">FPT_FLS.1</a>
<a href="#">SF CARD CONTENT MANAGEMENT</a>	<a href="#">FDP_ITC.2/Installer</a> , <a href="#">FMT_SMR.1/Installer</a> , <a href="#">FPT_FLS.1/Installer</a> , <a href="#">FPT_RCV.3/Installer</a> , <a href="#">FDP_ACC.2/ADEL</a> , <a href="#">FDP_ACF.1/ADEL</a> , <a href="#">FMT_MSA.3/ADEL</a> , <a href="#">FCO_NRO.2/CM</a> , <a href="#">FDP_IFC.2/CM</a> , <a href="#">FMT_MSA.1/CM</a> , <a href="#">FMT_MSA.3/CM</a> , <a href="#">FMT_SMF.1/CM</a> , <a href="#">FMT_SMR.1/CM</a> , <a href="#">FTP_ITC.1/CM</a> , <a href="#">FMT_SMF.1</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FIA_ATD.1/AID</a> , <a href="#">FMT_MTD.1/JCRE</a> , <a href="#">FMT_MTD.3/JCRE</a> <a href="#">FDP_IFF.1/CM</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FMT_MSA.1/ADEL</a>
<a href="#">SF CARD MANAGEMENT ENVIRONMENT</a>	<a href="#">FMT_MSA.1/ADEL</a> , <a href="#">FMT_SMF.1/ADEL</a> , <a href="#">FMT_SMR.1/ADEL</a> , <a href="#">FPT_TDC.1</a> , <a href="#">FMT_MOF.1/PP</a>
<a href="#">SF CARDHOLDER VERIFICATION</a>	<a href="#">FIA_AFL.1/PIN</a> , <a href="#">FMT_MTD.2/GP_PIN</a> , <a href="#">FIA_AFL.1/GP_PIN</a>
<a href="#">SF CLEARING OF SENSITIVE INFORMATION</a>	<a href="#">FDP_RIP.1/ADEL</a> , <a href="#">FPT_FLS.1/ADEL</a> , <a href="#">FDP_RIP.1/ODEL</a> , <a href="#">FPT_FLS.1/ODEL</a> , <a href="#">FDP_RIP.1/OBJECTS</a> , <a href="#">FDP_RIP.1/ABORT</a> , <a href="#">FDP_RIP.1/APDU</a> , <a href="#">FDP_RIP.1/bArray</a> , <a href="#">FDP_RIP.1/KEYS</a> , <a href="#">FDP_RIP.1/TRANSIENT</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FPT_FLS.1/Installer</a>
<a href="#">SF DAP VERIFICATION</a>	<a href="#">FDP_IFC.2/CM</a> , <a href="#">FDP_IFF.1/CM</a> , <a href="#">FDP_UIT.1/CM</a> , <a href="#">FTP_ITC.1/CM</a> , <a href="#">FCO_NRO.2/CM_DAP</a>
<a href="#">SF DATA COHERENCY</a>	<a href="#">FPT_RCV.4/SCP</a>
<a href="#">SF DATA INTEGRITY</a>	<a href="#">FDP_SDI.2</a> , , <a href="#">FPT_TST.1</a>
<a href="#">SF ENCRYPTION AND DECRYPTION</a>	<a href="#">FDP_UIT.1/CM</a> , <a href="#">FCS_COP.1</a> , <a href="#">FCS_COP.1/CM</a>
<a href="#">SF ENTITY AUTHENTICATION/SECURE CHANNEL</a>	<a href="#">FIA_AFL.1/CM</a> , <a href="#">FIA_UAU.1/CM</a> , <a href="#">FIA_UAU.4/CardIssuer</a> , <a href="#">FIA_UAU.7/CardIssuer</a> , <a href="#">FMT_SMR.2/CM</a> , <a href="#">FCS_COP.1/CM</a>



TOE Summary Specification	Security Functional Requirements
<a href="#">SF FIREWALL</a>	<a href="#">FMT_MSA.1/REM_REFS</a> , <a href="#">FDP_ACC.2/FIREWALL</a> , <a href="#">FDP_ACF.1/FIREWALL</a> , <a href="#">FDP_IFC.1/JCVM</a> , <a href="#">FDP_IFF.1/JCVM</a> , <a href="#">FMT_MSA.1/JCRE</a> , <a href="#">FMT_MSA.1/JCVM</a> , <a href="#">FMT_MSA.2/FIREWALL_JCVM</a> , <a href="#">FMT_MSA.3/FIREWALL</a> , <a href="#">FMT_MSA.3/JCVM</a> , <a href="#">FMT_SMR.1</a> , <a href="#">FAU_ARP.1</a> , <a href="#">FPT_FLS.1</a> , <a href="#">FIA_UID.2/AID</a> , <a href="#">FIA_USB.1/AID</a>
<a href="#">SF GP DISPATCHER</a>	<a href="#">FIA_UID.1/CM</a> , <a href="#">FTP_ITC.1/CM</a> , <a href="#">FIA_UAU.1/CM</a>
<a href="#">SF HARDWARE OPERATING</a>	<a href="#">FPT_TST.1</a> , <a href="#">FPT_PHP.3/SCP</a>
<a href="#">SF KEY ACCESS</a>	<a href="#">FCS_CKM.3</a>
<a href="#">SF KEY AGREEMENT</a>	<a href="#">FCS_COP.1</a>
<a href="#">SF KEY DESTRUCTION</a>	<a href="#">FCS_CKM.4</a> , <a href="#">FCS_CKM.4/PP</a>
<a href="#">SF KEY DISTRIBUTION</a>	<a href="#">FCS_CKM.2</a>
<a href="#">SF KEY GENERATION</a>	<a href="#">FCS_CKM.1</a> , <a href="#">FCS_CKM.1/PP</a>
<a href="#">SF KEY MANAGEMENT</a>	<a href="#">FMT_MSA.1/CM</a> , <a href="#">FPR_UNO.1/Key_CM</a> , <a href="#">FPT_TDC.1/CM</a>
<a href="#">SF MANUFACTURER AUTHENTICATION</a>	<a href="#">FDP_ACC.2/PP</a> , <a href="#">FDP_ACF.1/PP</a> , <a href="#">FDP_UCT.1/PP</a> , <a href="#">FDP_ITC.1/PP</a> , <a href="#">FIA_AFL.1/PP</a> , <a href="#">FIA_UAU.4/CardManu</a> , <a href="#">FIA_UAU.7/CardManu</a> , <a href="#">FMT_SMR.2/PP</a> , <a href="#">FMT_MSA.3/PP</a> , <a href="#">FCS_COP.1/PP</a> , <a href="#">FMT_MSA.1/PP</a> , <a href="#">FMT_SMF.1/PP</a>
<a href="#">SF MESSAGE DIGEST</a>	<a href="#">FCS_COP.1</a>
<a href="#">SF MEMORY FAILURE</a>	<a href="#">FRU_FLT.1/SCP</a>
<a href="#">SF PREPERSONALISATION</a>	<a href="#">FDP_ACC.2/PP</a> <a href="#">FMT_MOF.1/PP</a> , <a href="#">FMT_MSA.3/PP</a> , <a href="#">FCS_COP.1/PP</a> , <a href="#">FDP_UIT.1/PP</a> , <a href="#">FDP_ITC.1/PP</a> , <a href="#">FCS_CKM.1/PP</a> , <a href="#">FDP_UCT.1/PP</a> , <a href="#">FAU_STG.2</a> , <a href="#">FMT_SMR.2/PP</a>
<a href="#">SF RANDOM NUMBER</a>	<a href="#">FIA_UAU.4/CardIssuer</a> , <a href="#">FIA_UAU.7/CardIssuer</a> , <a href="#">FCS_RNG.1/SCP</a>
<a href="#">SF RESIDENT APPLICATION DISPATCHER</a>	<a href="#">FDP_ACC.2/PP</a> , <a href="#">FIA_UAU.1/PP</a> , <a href="#">FIA_UID.1/PP</a> , <a href="#">FDP_ITC.1/PP</a> , <a href="#">FMT_SMR.2/PP</a>



TOE Summary Specification	Security Functional Requirements
<u>SF_RUNTIME_VERIFIER</u>	<u>FDP_ACC.2/RV Stack,</u> <u>FDP_ACF.1/RV Stack,</u> <u>FMT_MSA.1/RV Stack,</u> <u>FMT_MSA.2/RV Stack,</u> <u>FMT_MSA.3/RV Stack,</u> <u>FMT_SMF.1/RV Stack,</u> <u>FDP_ACC.2/RV Heap,</u> <u>FDP_ACF.1/RV Heap,</u> <u>FMT_MSA.1/RV Heap,</u> <u>FMT_MSA.2/RV Heap,</u> <u>FMT_MSA.3/RV Heap,</u> <u>FMT_SMF.1/RV Heap,</u> <u>FDP_ACC.2/RV Transient,</u> <u>FDP_ACF.1/RV Transient,</u> <u>FMT_MSA.1/RV Transient,</u> <u>FMT_MSA.2/RV Transient,</u> <u>FMT_MSA.3/RV Transient,</u> <u>FMT_SMF.1/RV Transient</u>
<u>SF_SECURITY_FUNCTIONS_OF_THE_IC</u>	<u>FCS_RNG.1/SCP</u>
<u>SF_SIGNATURE</u>	<u>FCS_COP.1</u>
<u>SF_UNOBSERVABILITY</u>	<u>FPR_UNO.1/USE_KEY,</u> <u>FDP_SDI.2/Applet,</u> <u>FPR_UNO.1/Key_CM</u>

Table 17: TSS and SFRs - Coverage