ID-ONE COSMO V7.0.1

TERPSICHORE Security Target Lite

For NXP

FQR 110 5145
Issue 4

List of figures

List of tables

1 PREFACE

1.1 OBJECTIVES OF THE DOCUMENT

This Security Target aims to satisfy the requirements of Common Criteria level EAL5 augmented with AVA_VAN.5 and 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.

1.2 SCOPE OF THE DOCUMENT

This document describes the Security Target for the V7.0.1 card, which is running on a Javacard 2.2.2 virtual machine.

This Security Target covers the development of the V7.0.1 platform, which is able to receive and manage different types of applications (IAS, LDS and ID-One Classic,…). This card is consistent with the Java Card 2.2.2 specifications, as well as the GlobalPlatform 2.1.1 specifications.

The objectives of this Security Target are:

- To describe the Target of Evaluation (TOE), its life cycle and to position it in the smart card life cycle.
- To describe the security environment of the TOE including the assets to be protected and the threats to be countered by the TOE and by the operational environment during the development and the platform active phases.
- To describe the security objectives of the TOE and its supporting environment in terms of integrity and confidentiality of sensitive information of the TOE. It includes protection of the TOE and associated documentation during development and active life phases.
- To specify the security requirements which include the TOE functional requirements, the TOE assurance requirements and the security requirements for the environment.
- To describe the summary of the TOE specification including a description of the security functions and assurance measures that meet the TOE security requirements.
- To present evidence that this ST is a complete and cohesive set of requirements that the TOE provides on an effective set of IT security countermeasures within the security environment, and that the TOE summary specification addresses the requirements.

This V7.0.1 platform is able to receive and manage different types of applications (IAS, LDS and ID-One Classic …).

Some of these applications are in ROM (already loaded in the platform), others can be loaded in EEPROM at the Personalisation phase or at the use phase.

1.3 RELATED DOCUMENTS

[1] "Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", September 2006, Version 3.1 revision 1.

[2] "Common Criteria for information Technology Security Evaluation, Part 2: Security Functional requirements", September 2007, Version 3.1 revision 2.

[3] "Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance requirements", September 2007, Version 3.1 revision 2.

[4] “Composite product evaluation for Smart Cards and similar devices”, September 2007, Version 1.0, CCDB-2007-09-001.

[5] PP SUN Java Card™ System Protection Profile Collection *Version 1.0b,* August 2003

[6] "Java Card 2.2.2 - API" Application Programming Interfaces Version 2.2.2 March, 2006, Sun Microsystems [JCAPI]

[7] "Java Card 2.2.2 – JCRE” Runtime Environment Specification Version 2.2.2 March, 2006, Sun Microsystems. [JCRE]

[8] "Java Card 2.2.2 - Virtual Machine Specifications" Version 2.2.2 March, 2006, Sun Microsystems [JCVM]

[9] “GlobalPlatform 2.1.1” Card Specification v2. 1, 01 Mars 2003

[10] “GlobalPlatform 2.1.1” Card Implementation Requirements- Mars 2003

[11] Card Specification – version 2.1.1-February, Mars 2003

[12] Card Specification – Amendment A- February, 2004

[13] "Identification cards - Integrated Circuit(s) Cards with contacts, Part 6: Interindustry data elements for interchange", ISO / IEC 7816-6 (2004)

[14] "Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)", ANSI X9.31-1998, American Bankers Association,

[15] "FIPS PUB 46-3, Data Encryption Standard", October 25, 1999 (ANSI X3.92), National Institute of Standards and Technology

[16] "FIPS PUB 81, DES Modes of Operation", April 17, 1995, National Institute of Standards and Technology

[17] "FIPS PUB 180-2, Secure Hash Standard", August 2002 , National Institute of Standards and Technology

[18] "FIPS PUB 186-2", January 27, 2000, Digital Signature Standard (DSS) Change Notice 1

[19] "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)

[20] “Information technology – Security techniques: Data integrity mechanism using a cryptographic check function employing a block cipher algorithm”, ISO 9797-1 (1999) , International Organization for Standardization

[21] “FIPS PUB 140-2, Security requirements for cryptographic modules”, Mars 2002 , National Institute of Standards and Technology

[22] PKCS#1 The public Key Cryptography standards, RSA Data Security Inc. 1993

[23] Security IC Platform Protection Profile, Version 1.0, reference BSI-PP-0035 (15.06.2007).

[24] Security Target Lite Rev. 1.3 NXP Secure Smart Card Controllers P5CD016/021/041V1A and P5Cx081V1A —BSI-DSZ-CC-0555, 21 September 2009

[25] IEEE Std 1363a-2004
Standard Specification of Public-Key Cryptography

[26] FIPS PUB 197, The Advanced Encryption Standard (AES), U.S. DoC/NIST, November 26, 2001.

The following references are defined in the document SUN PP [5]:

[JAVASPEC], [JCAPI21], [JCVM22], [JCRE22], [JVM], [CC2].

1.4 ABBREVIATIONS AND NOTATIONS

1.4.1 Abbreviations

AES	Advanced Encryption Standard
AID	Applet Identifier
APDU	Application Protocol Data Unit
API	Application Programmer Interface
BIOS	Basic Input/Output System
CC	Common Criteria
CM	Card Manager
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 Memory
ES	Embedded Software
FAMEX	Co-processor for public key crypto calculations
FAT	File Allocation Table
IC	Integrated Circuit
IT	Information Technology
JCP	Java Card Platform
JCRE	Java Card Runtime Environment
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
VM	Virtual Machine
GP	Global Platform

1.4.2 Notations

Active life/phase	period with active security functions and no active code
Applet	Application which can be loaded and executed with the environment of the Java Card platform
Card Issuer	Entity that owns the card and is ultimately responsible for the behavior 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.
Controlling Authority	A Controlling Authority has the privilege to keep the control over the Card Content through the mandating of DAP Verification
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.
Delegated Management	Pre-authorized Card Content Management performed by an approved Application Provider.
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. (Performs the application code loading and related Card Content management and memory management.) The OPEN also manages the installation of applications loaded to the card. The OPEN is responsible for enforcing security principles defined for Card Content management.
Receipt	A cryptographic value provided by the card as proof that a Delegated Management operation has successfully done.
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).
Security Domain Authority	Security Domain with Mandated Dap verification. This privilege allows an Authority Body to control all the application loaded on the card.
Security Domain Delegated	Security Domain with Delegated Management privilege. This privilege allows an Application Providers to perform loading, installing or extraditing application.
Token	A cryptographic value provided by a Card Issuer as proof that a Delegated Management operation has been authorized. This signature is generated by the Card Issuer and used to provide the Card Issuer the control over the Card Content operation. The Token is verified on the card by Card manager.

2 SECURITY TARGET INTRODUCTION

2.1 ST REFERENCE (For ST complete version)

Title: TERPSICHORE Security Target for P5CD041V1A, P5CC081V1A and P5CD081V1A

Name: Cosmo V7.0.1

OCS registration: FQR 110 4933

Version: 5

Authors: Oberthur Technologies

Publication Date: February 2012

2.2 TOE REFERENCE

TOE Name	ID-One Cosmo V7.0.1-n Standard Dual	ID-One Cosmo V7.0.1-n Standard	ID-One Cosmo V7.0.1-n Basic Dual
Mask / Hardware Identification	18 01 1F	18 01 1A	18 01 1B
Label PVCS CODE	ID One Cosmo V701n CC	ID One Cosmo V701n CC	ID One Cosmo V701n CC
Optional code Generic r1.0 on ID-One Cosmo v7.0.1-n	077121	077121	077121
IC reference version	P5CD081 V1A	P5CC081 V1A	P5CD041 V1A
Identication of IC ST	Security Target Lite Rev. 1.3 NXP Secure Smart Card Controllers P5CD016/021/041V1A and P5Cx081V1A - 21 September 2009		
IC Certificate	BSI-DSZ-CC-0555-2009		

Table 1: TOE References

3 TOE OVERVIEW

The Smart Card intended to support the TOE is composed of hardware and software components, as listed below and described in Figure 1.

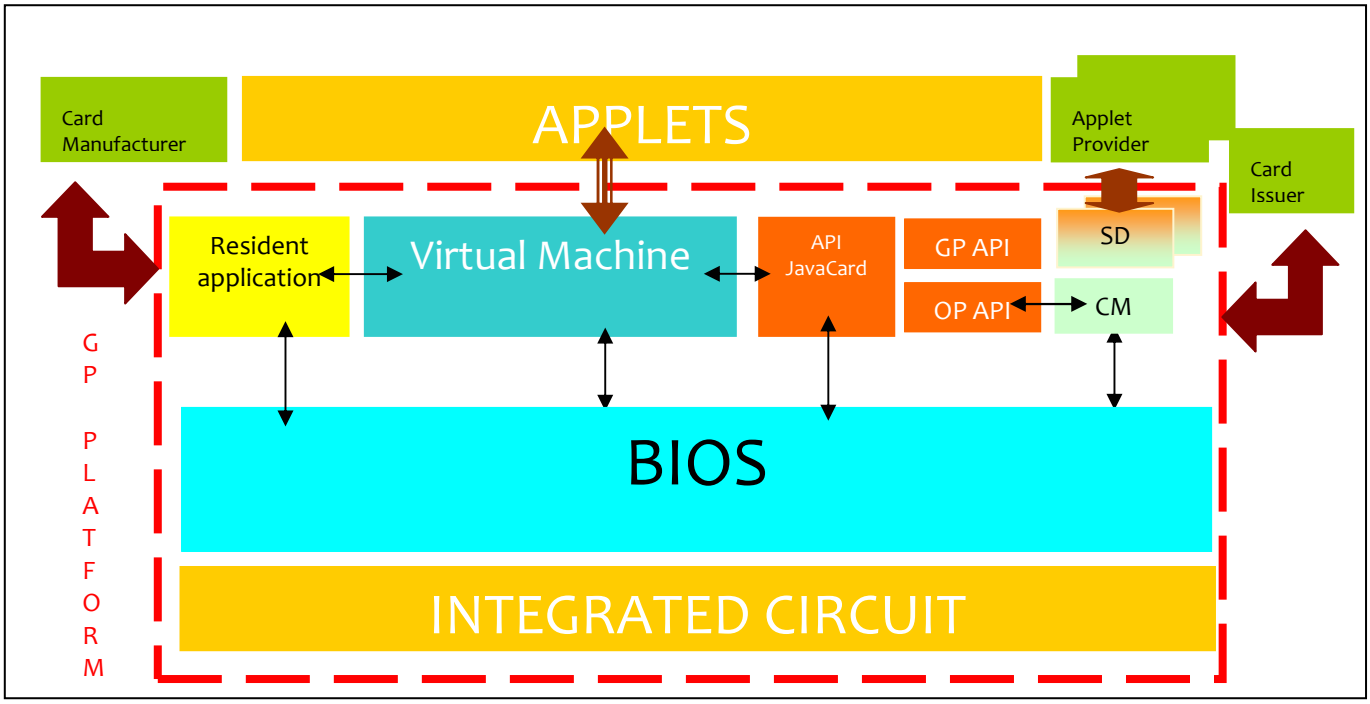

Figure 1: Java Platform Architecture

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

3.1 TOE TYPE

The TOE is **ID-One Cosmo V7.0.1-n** on NXP family is a contact/dual/ javacard platform based on NXP compatible with multi-application ID-One Cosmo product family.

The functional level of the OS will be based on a Java™ based multi-application platform, compliant with **Java Card 2.2.2** and **GlobalPlatform 2.1.1** specifications.

CHIP FAMILY INFORMATION

The operating system works on components specified at chapter TOE REFERENCE 2.2. The differences between the ICs are:

- Chip ID
- Size of EEPROM and address of OTP
- Contactless interface

The IC is the of the microcontroller chip family of NXP Secure Smart Card Controllers P5CD016/021/041V1A and P5Cx081V1A. It also comprises IC Dedicated Test Software for test

purposes and IC Dedicated Support Software, both stored in the Test-ROM of the microcontroller. The Smart Card Controller hardware incorporates an 8-bit processing unit, volatile and non-volatile memories accessible via a Memory Management Unit, cryptographic coprocessors, other security components and two communication interfaces.

Different hardware configurations

Six major configuration options are present in the IC evaluation, which are denoted by IC names:

P5CD016V1A, P5CD021V1A, P5CD041V1A, P5CD081V1A, P5CC081V1A. All of them are equipped with an EEPROM of 80 kBytes and both, the ISO/IEC 7816 contact interface and the ISO/IEC 14443 contactless interface.

3 configurations are used in this project. Their major differences are related to availability of EEPROM space and the contactless interface as detailed below.

Major configuration P5CD041V1A

Its major differences to other configurations are as follows.

The EEPROM space available to the Security IC Embedded Software is reduced to 40 kBytes minus 256 Bytes, which are reserved for Security Rows (128 Bytes) and configuration data (128 Bytes).

The contactless interface is enabled and configured for contactless communication

Major configuration P5CD081V1A

Its major differences to other configurations are as follows.

The physically implemented EEPROM of 80 kBytes is available to the Security IC Embedded Software except for 256 Bytes, which are reserved for Security Rows (128 Bytes) and configuration data of the manufacturer (128 Bytes).

The contactless interface is enabled and configured for contactless communication

Major configuration P5CC081V1A

Its major differences to other configurations are as follows.

The physically implemented EEPROM of 80 kBytes is available to the Security IC Embedded Software except for 256 Bytes, which are reserved for Security Rows (128 Bytes) and configuration data (128 Bytes).

The contactless interface is disabled.

BIOS

The BIOS is an interface between hardware and native components like VM and APIs.

The BIOS implements the following functionalities:

- APDU management ("T=0", "T=1" protocols)
- Timer management,
- Exceptions management,
- Transaction management,
- EEPROM access,

Cryptographic features

- SHA-1, SHA-256, SHA-384, SHA-512
- DES, 3DES (56/112/168 bits)

- AES 128/192/256 bits
- RSA ANSI X9.31 2048 bits + Key Generator (secured)
- RSA SFM (Standard)
- RNG
- ECDSA GF(p)
- ECDH GF(p)
- ECKeyP Generator

Virtual Machine

The Virtual Machine, which is compliant with the JAVACARD 2.2.2 standard, interprets the byte code of JAVACARD 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.

The Java Card Runtime Environment

The Java Card Runtime Environment (JCRE) contains the Java Card Virtual Machine (VM), the Java Card Application Programming Interface (API) classes and industry-specific extensions, and support services. For details please refer to reference [7].

APIs

The APIs, compliant with the JAVACARD 2.2.2 standard, support key generation, Key Agreement, signature, ciphering of messages and proprietary OCS API.

Open Platform

The "GlobalPlatform 2.1.1 configuration full GP", which consists of the Card Manager, the API OPSystem and the API GPSystem is implemented in Java and its byte-code is stored in ROM.

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 and logical channels management.

The dispatcher is always activated. Some card commands (for administration) are only available during prepersonalization phase.

Applets

The applications (Java Card applet) already in the ROM or to be loaded at use phase are out of the TOE scope.

3.2 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 have 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.

The major features are listed in below:

Secure execution and no passability

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.

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.

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.

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 or the Remote Access Control function.

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.

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.

Message Digest Generation

Message digest generation shall be implemented to resist environmental stress and glitches and include measures for preventing information leakage through covert channels.

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.

Random Number Generation

This TOE 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. It shall be compliant to ANSI X9.31 standard.

Data integrity

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

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 securel clear and destroy this information. It concerns Pins, keys, sensitive data, bufferAPDU.

This function is also available to applet.

Unobservability

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

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.

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 Distribution

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

Cardholder Verification

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

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.

Token

The TOE shall ensure the validity of a card content management command when this command is issued to a Security Domain with Delegated Management privilege.

Manufacturer Authentication

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

Resident Application dispatcher

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

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.

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).

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).

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 [IEEE Std 1363a-2004].

Pre-personalization

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.

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.

Secure Boot at power-up

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.

Security Monitor
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 [JIL] 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 conformant to related SUN PP [5], 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).

3.3 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 installation of the file on the card in any case.

3.4 TOE USAGE

Smart cards are mainly used as data carriers that are secure against forgery and tampering. More recent uses also propose them 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, usually 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 is inserted into a card reader. In some configurations of the TOE, the card reader may also be used to enlarge or restrict the set of applications that can be executed on the Java Card platform according to a well defined card management policy.
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, and the balance of an electronic purse is highly sensitive with regard to arbitrary modification (because it represents real money).

This Platform is an open and multi application platform intended to propose security requirements for applications with critical assets, in particular:
- Banking and finance credit / debit smartcards, electronic purse (stored value cards) and electronic commerce, loyalties,
- Network based transaction processing such as mobile phones (GSM SIM cards), pay-TV (subscriber and pay-per-view cards), communication highways (Internet access and transaction processing),
- Transport and ticketing market (access control cards),
- Governmental cards (electronic passports, ID-cards, health-cards, driver license, etc.),
- Multimedia commerce and Intellectual Property Rights protection.

These applications can be loaded at some specific states of the TOE lifecycle defined in the next paragraphs. These applications are not in the scope of the evaluation.

3.5 TOE GUIDANCE

The ID-One Cosmo V7.0.1 is evaluated with its guidance. The guidances of the Platform are listed hereafter:
[GUIDE1] ID-One Cosmo V7.0.1 Security Recommendations FQR 110 4912 Issue: 4
[GUIDE2] ID-One Cosmo V7.0.1 Reference Guide FQR 110 4911 Issue: 4
[GUIDE3] ID-One Cosmo V7.0.1 Pre-Perso Guide FQR 110 4910 Issue: 7

The platform is evaluated without applications. Before loading any application need to pass thru the Verifier.

[GUIDE1]

If the applet needs to have a security certification, the applet must follow recommendations listed in [GUIDE1].
If the applet does not need additional security certification with the platform, the certificate of the Platform is still valid if the applet go thru verifier when this applet is loaded (the security function of the platform are still ok).

[GUIDE2]

This document describes the ID-One Cosmo V7.0.1 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.

[GUIDE3]

This document describes the pre-personalization steps that should be followed to correctly initialize the Cosmo v7.0.1 platforms. The TOE is finalized once it's prepersonalised.

3.6 TOE LIFE CYCLE

The GP platform life-cycle (see Figure 3) is included into the Smart Card Product life-cycle, which is decomposed into 7 phases, according to the Smart Card Integrated Circuit Protection Profile (PP/9806), as described in figure below.

Part of the code named patch code (or codop) shall be added pre-personalisation phase. This code is developed at Oberthur premises (Phase 1, figure 2) delivered and loaded securely at production phase (Phase 5, figure 2).

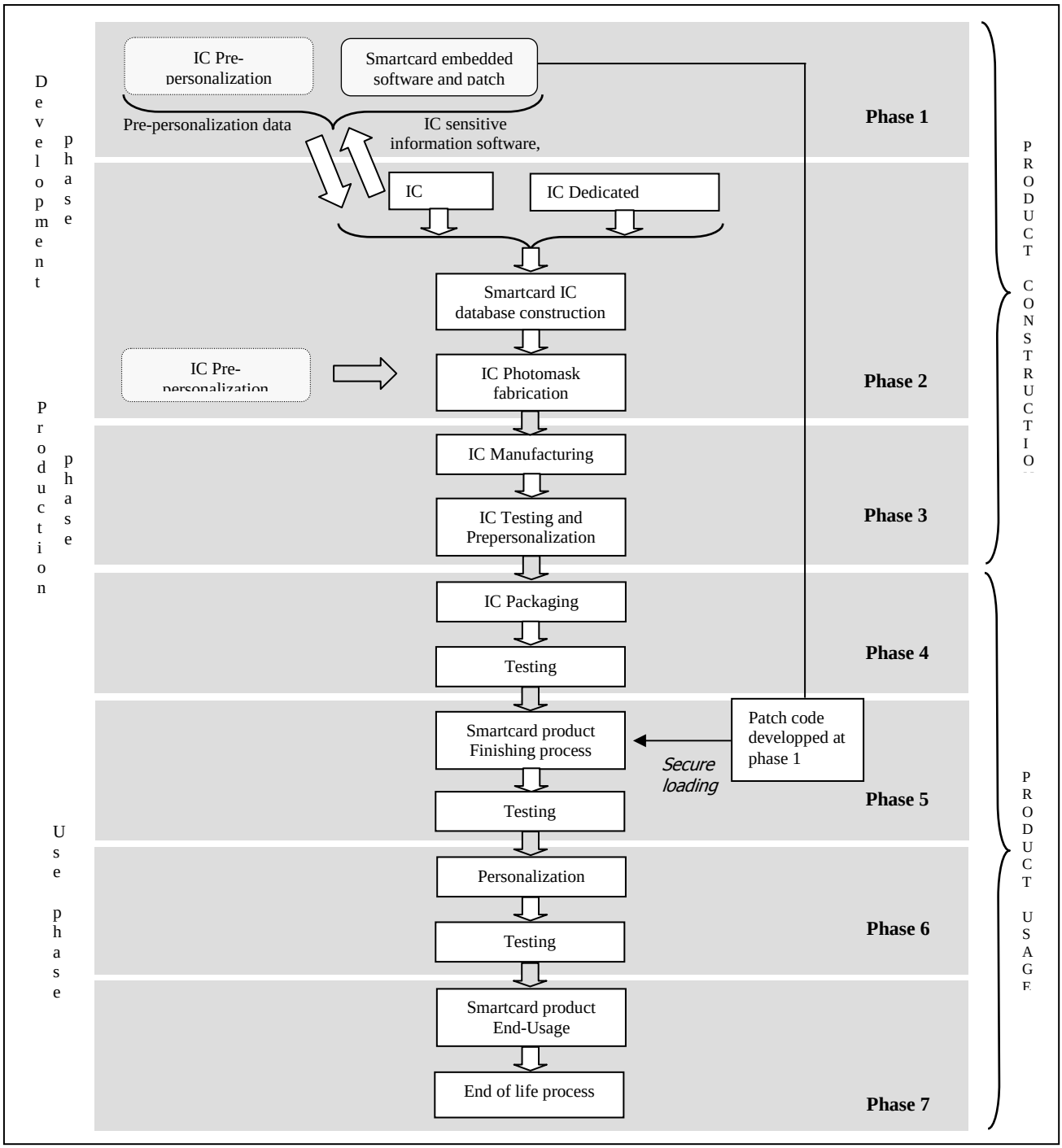

Figure 2: Smart Card Product life-cycle

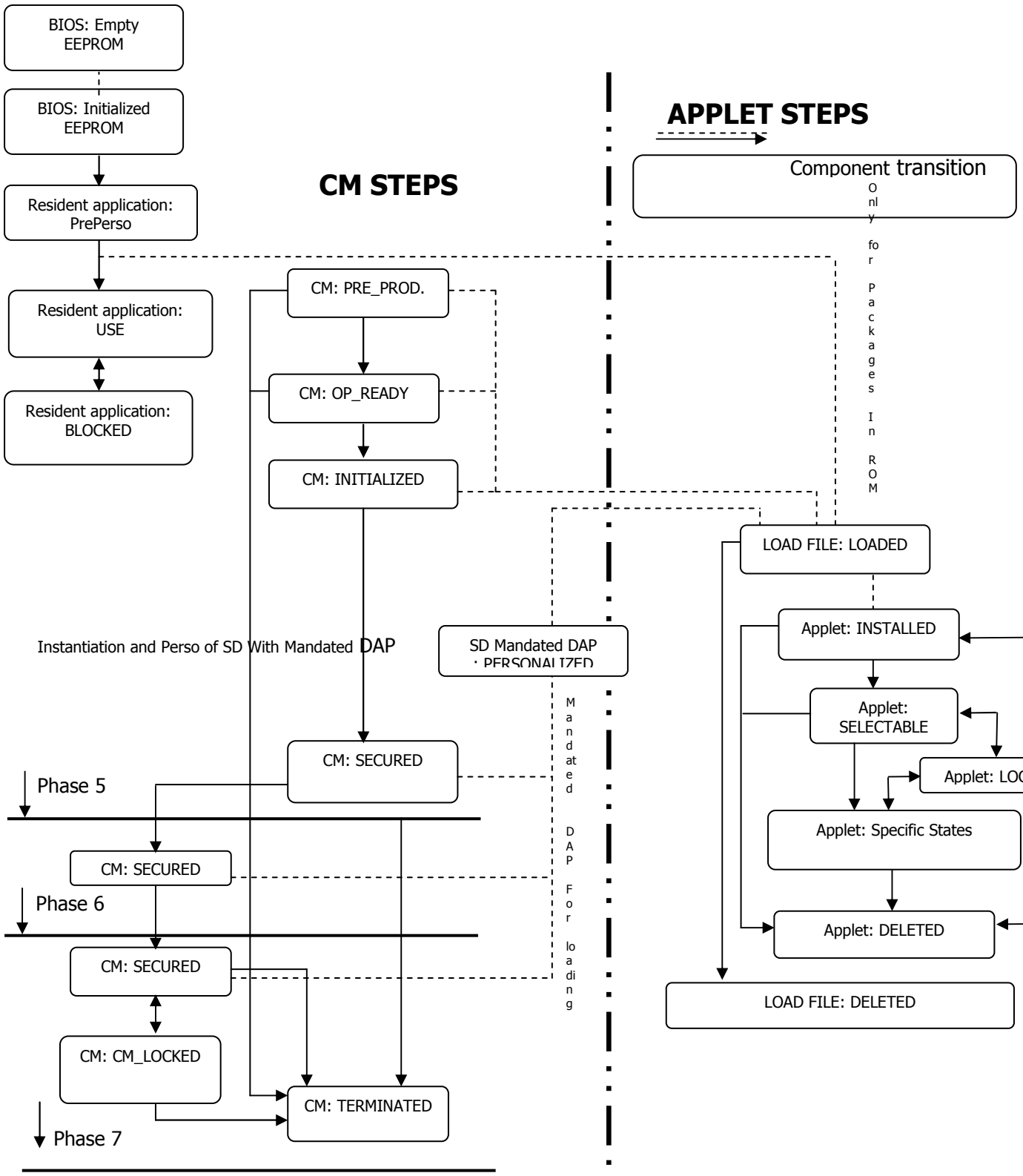

Figure 3: GP platform life-cycles

The point of delivery is fixed to the end of the phase 3 and the loading of the patch is performed in phase 5. The security of this loading is fully enforced by technical measures evaluated by the ITESF. Therefore phases 3 and 5 are completely covered by AGD and not by ALC.

The GlobalPlatform defines life cycle state models to control the functionality and security of the following components: Card Manager, Security Domains, Global PIN (CVM), Card Registry, Key sets, Load Files, and Applets. These life cycle models are presented in this section.

Card Content Management

Card Content management on a GlobalPlatform card allows Card Content loading, installation, and removal capabilities. The Open as part of the Card manager entity is responsible for the Card Content management even if the initial requests (APDU commands) are issued to a Security Domain with the delegated management privilege.

3.6.1 Card life cycle transitions

The Card Manager is responsible for maintaining the overall security and administration of the card and its contents. Because the Card Manager plays this supervisory role over the entire card, its life cycle can be thought of as the life cycle of the card. The card's life cycle from a GlobalPlatform perspective only has meaning at the beginning of the Card Manager life cycle.

The Card Manager owns and maintains the card life cycle state information and manages the requested state transitions in response to APDU commands. The end of the Card Manager life cycle is considered to be equivalent to the end of the card's life cycle.

Pre_production (phase 5)

This life state is the initial life state of the Card Manager applet, just after it has been installed.

During this life cycle state, initialization key and card and chip CPLC have to be loaded before switching to OP_READY life state.

OP_READY (phase 5)

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.

The card is assumed to have the following functionalites 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.

INITIALIZED (phase 5)

The card life cycle state INITIALIZED is an administrative card production state. Most of the personalization of the Card Manager is performed when entering in this state. Card Content Management is possible in this state.

SECURED (phase 6)

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 behavior 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.

- Post-issuance personalization of applets belonging to the Card Issuer can be carried out via the Card Manager.
- Card Content Management operations are supported

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.

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.

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

Only GET Data (CPLC) command is available.

3.6.2 BIOS life cycle

Empty EEPROM: When the chip is delivered by the IC manufacturer, the EEPROM is empty except for the Manufacturer Transport key (MSK).

Initialized EEPROM: On the first Power-On, the BIOS initializes its data: the ATR files, the default applet reference, the FAT.

3.6.3 Resident Application life cycle

PP: Prepersonalization state, the set of commands of the resident application (EXTERNAL_AUTHENTICATE, INITIALIZE AUTHENTICATION PROCESS, GET_DATA, INSTALL, LOAD_APPLET, LOAD_STRUCTURE, LOAD_SECURE, and MANAGE_CHANNEL) is active.

USE: the set of commands of the resident application (SELECT, MANAGE_CHANNEL, and GET_DATA only if no applet is selected) is active.

LOCKED / BLOCKED: All the commands of the resident application are inactive.

3.6.4 LOAD FILE AND APPLET LIFE CYCLE

The delivery of an applet must satisfy a process (described in Figure 4) using the following tools: compiler, converter, verifier, and loader.

As the verifier is mandatory, a Security Domain with Mandated DAP privilege is instantiated. This privilege enables integrity and authenticity verification during the package loading process.

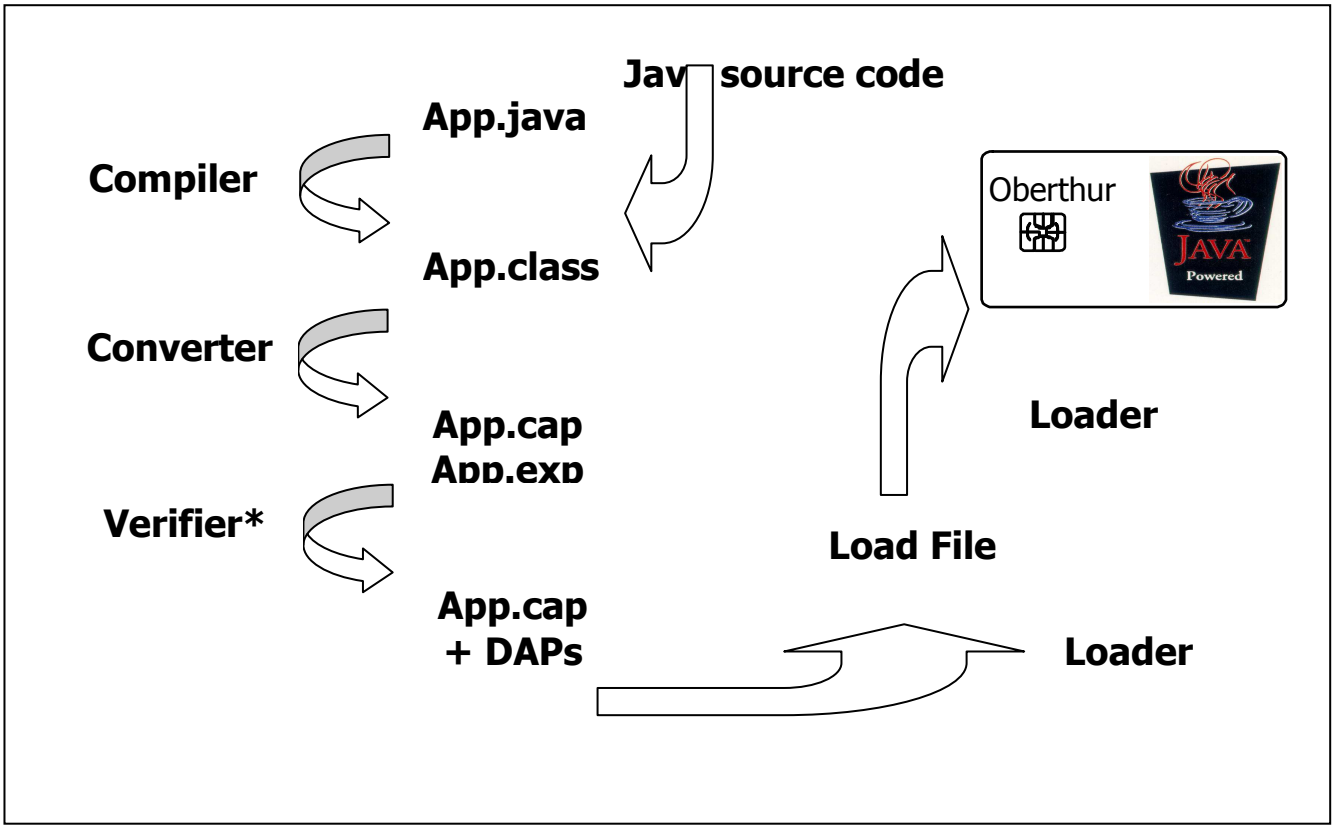

Figure 4: Tools used before delevery applet (compiler, converter, verifier, and loader)

3.6.5 Load File life cycle

LOADED: The Card Manager considers all Load Files which are present in the card and available for use either from immutable persistent memory or mutable persistent memory to be at the state LOADED.

3.6.6 Applet Life cycle

The Applet life cycle begins when an applet is installed in the card. This installation may occur directly during a loading transaction or alternatively from a Load File which is present on the card. Applications can be INSTALLED, SELECTABLE or LOCKED

In addition to these Application Life Cycle 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 GlobalPlatform. 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.

Application Life Cycle State 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 GlobalPlatform 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 personalization of the Application, which may occur as a separate step.

Application Life Cycle State 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 SELECTABLE 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.

Application Life Cycle State 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.

Application Deletion

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 GlobalPlatform 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 Application during these states is determined by the Application itself and is beyond the s e of this Specification. The OPEN does not enforce any control on Application specific Life Cycle State transitions.

3.6.7 Delegated Management

The concept of Delegated Management allows the Card Issuer to empower partnered Application Providers with the ability to initiate approved and pre-authorized Card Content management operations (loading, installing or extraditing). This approval, which is fondamental to the concept of Delegated Management, ensures that only Card Content Management operations that have been previously authorized by the Card Issuer will be accepted and processed by the card. This delegation of control on the Card Content changes gives the Application Provider more flexibility in managing its Applications.

3.6.7.1 Delegated Loading

Delegated loading allows an Application Provider to establish a Secure Channel to transfer Load Files directly to its Security Domain.

The Load Token allows the Card via the Card Manager verification, to ensure that the Card Issuer authorized the load process and the loading of the content of the Load File Data Block.

The Load Token implies that the Card Issuer pre-authorized the data required for the delegated load process. The pre-authorized data includes most of the INSTALL [for load] command and the Load File Data Block Hash. The Load File Data Block Hash links the Token to the actual Load File Data Block.

The response to the last LOAD command identifies the end of the load process. Following the completion of the load process, a Load Receipt is returned to the Security Domain performing the Delegated Management operation and is transmitted by the Security Domain to the off-card entity.

The Application Provider may then forward the Load Receipt to the Card Issuer as a proof that the loading process was successfully performed.

3.6.7.2 Delegated Installation

Delegated Installation allows an Application Provider to establish a Secure Channel to install an Application from an Executable Load File that is associated to the Application Provider's Security Domain. The installation process comprises one INSTALL [for install] command processed by the Security Domain. The Security Domain then forwards the install request to the CardManager for additional verification and processing.

The Install Token allows the card via the CardManager verification, to ensure that the Card Issuer authorized the installation process.

The Install Token implies that the Card Issuer pre-authorized the data required for the delegated installation process. The pre-authorized data includes most of the INSTALL [for install] command. The response to the INSTALL [for install] command identifies the end of the installation process. Following the completion of the installation process, an Install Receipt is returned to the Security Domain performing the Delegated Management operation and is transmitted by the Security Domain to the off-card entity.

3.6.7.3 Delegated Extradition

Delegated Extradition allows an Application Provider to establish a Secure Channel to extradite one of its Applications. The extradition may apply at any time during the Application Life Cycle.

The extradition process comprises one INSTALL [for extradition] command processed by the Security Domain. The Security Domain then forwards the extradition request to the CardManager for additional verification and processing. The Extradition Token allows the Card via the CardManager verification, to ensure that the Card Issuer authorized the extradition process.

The Extradition Token implies that the Card Issuer pre-authorized the data required for the delegated extradition process. The pre-authorized data includes most of the INSTALL [for extradition] command. The response to the INSTALL [for extradition] command identifies the end of the extradition process. Following the completion of the extradition process, an Extradition Receipt is returned to the Security Domain to the off-card entity.

The Application Provider may then forward the Extradition Receipt to the Card Issuer as a proof that the extradition process was successfully performed. The purpose of the optional Extradition Receipt is to assist the Card Issuer in keeping its Card Management System synchronized with its card base.

3.6.8 TOE ENVIRONMENT

The TOE environment is defined as follows:

- TOE development environment corresponding to phase 1
- TOE manufacturing environment corresponding to phase 2 & 3
- End-user environment corresponding to phase 7, where package loading is allowed.

All other phases are out of the scope of the TOE.

3.6.8.1 Software development (phase 1)

This environment is limited to OBERTHUR sites at Nanterre, Levallois and Bordeaux.
To ensure security, access to development tools and products elements (PC, emulator, card reader, documentation, source code including patches, etc...) is protected. The protection is based on measures for prevention and detection of unauthorized access. Two levels of protection are applied:

- Access control to OBERTHUR offices and sensitive areas.
- Access to developpement data through the use of a secure computer system to design, implement and test software

3.6.8.2 Hardware development & IC manufacturing (phase 2 & 3)

The environment of hardware development and IC manufacturing is limited to NXP sites
The IC development environment is described in [24].
The IC is certified EAL5+ and the IC certificate references are listed in the TOE identification.

4 CONFORMANCE CLAIM

4.1 CONFORMANCE CLAIM TO CC

This Security Target claims conformance to **CC version 3.1** with the following documents:

[1] "Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", July 2009, Version 3.1 revision 3 Final

[2] "Common Criteria for information Technology Security Evaluation, Part 2: Security Functional requirements", July 2009, Version 3.1 revision 3 Final

[3] "Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance requirements", July 2009, Version 3.1 revision 3 Final

Conformance is claimed as follows:

- Part 1: conformant
- Part 2: extended with the FCS_RNG.1. All the other Security requirements have been drawn from the catalogue of requirements in Part 2
- Part 3: conformant EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

4.2 CONFORMANCE CLAIM TO A PP

This security target is conformant to [5] configuration. The configuration chosen is ***Java Card System Standard 2.1.1*** ++.
It contains: COREG, the Smart card platform, the Installer, the Logical channels Object deletion, the Applet deletion, the secure carrier and the Card Manager.

4.3 CONFORMANCE CLAIM RATIONALE

The TOE in this ST is conformant to TOE type in the PP. It's a java card platform.
The security problem definition of this ST consistent with Security problem definition of the PP as it's taken from the [5].

All threats defined in the [5] are present in the ST. Two elements are added but they are not in contradiction with [5] SPD:

1- T.CONFIGURATION The attacker tries to observe or modify configuration information exchanged between the TOE and its environnment.

The TOE in this phase must protect it self from modification or theft. Even the field is protected by assurance measures, each operations realised in this phase has to be protected.

This threat is extended to include pre-personalisation and personalization phases of the TOE.

2- T.CONF_DATA_APPLET The attacker tries to observe the operation of comparaison between two byte array in order to catch confidential information manipulated.

All the objectives of the PP are present in this ST so the statement of security objectives is consistent with the statement of security objectives in the PP for which conformance is being claimed.

Two objectives are added. They are listed below:

1- O.Resident Application: This objective concerns the 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 that implement the application commands.

2-O.Secure_Compare: The TOE shall provide the applets a mean to securely compare two byte arrays.

These objectives aren't in contradiction with existing ones.

All security requirement of the PP are definied in this ST, but as the PP is defined with previous Common Criteria version V2.1, some requirements are adapted to CC version 3.1.

The ST contains also additional SFR. It is an additional refinement of existing SFR.

The additional refinement concerns RSA key size,

In the PP [5] RSA key size are restricted to 1024;1280;1536;1984 and 2048, in this ST, we extended to other values: from 1024 to 2048 with a step of 32 bits.

We consider that these values are also secure and are not in contraction of the SFR on this ST.

5 Security problem definition

5.1 Assets

Assets are security-relevant elements to be directly protected by the TOE. Confidentiality of assets is always intended with respect to untrusted people or software, as various parties are involved during the first stages; 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 source code (one asset) or compiled code (another asset), and may exist in various formats (digital supports, printed paper) at different stages of its development. 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). The kind of dangers that weigh on each asset is specified.

5.1.1 User Data

D.APP_CODE

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

To be protected from unauthorized modification.

D.APP_C_DATA

Confidential sensitive data of the applications, like the data contained in an object, the 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, the 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.PIN

Any end-user's PIN and GlobalPIN.

To be protected from unauthorized disclosure and modification.

D.APP_KEYs

Cryptographic keys owned by the applets.

To be protected from unauthorized disclosure and modification.

5.1.2 TSF Data

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 JCVM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated to an array or any pointer used to chain data structures.

To be protected from monopolization and unauthorized disclosure or modification.

D.SEC_DATA

The runtime security data of the JCRE, 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.

D.API_DATA

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

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.

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.CONFIG

The configuration DATA are put at pre-personalization Phase. These elements of configuration have to be loaded securely.

D.SENSITIVE_DATA

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

- D.NBAUTHENTIC: Number of authentications. This number is specified in the SFR.
- D.NB_REMAINTRYOWN: Number of remaining tries for owner Pin. This number is specified in the SFR.
- D.NB_REMAINTRYGLB: Number of remaining tries for global Pin. This number is specified in the SFR.
- D.AUDITLOG: Audit log file
- ASG.CARDREG: Card Registry {AS.APID: Applet Identifier (AID), AS.CMID: Card Manager ID (AID)}
- ASG.APPPRIV: Applet privileges group {Card Manager lock privilege, Card terminate privilege, Default selected privilege, PIN Change privilege, Security Domain privilege, Security Domain with DAP verification privilege, Security Domain with Mandated DAP verification privilege},
- AS.AUTH_MSK_STATUS: Authentication MSK Status,
- AS.AUTH_CM_STATUS: Authentication CM Status,
- AS.CMLIFECYC: Card life cycle state,
- AS.MSKEY: Transport Key (Manufacturer Secret Key),

- AS.SECURITY_LEVEL: Security levels of a Secure Channel (Confidentiality, Integrity, or both).

D.ARRAY

Applets are enabled to store confidential data.

5.2 Users

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 or the verification authority), hardware (like the CAD where the card is inserted) 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.

S.PACKAGE

Packages used on the Java Card platform that act on behalf of the applet developer. These subjects are involved in the FIREWALL security policy and they should be understood as instances of the subject S.PACKAGE.

S.JCRE

The JCRE, which acts on behalf of the Card Issuer. This subject is involved in several security policies defined in this document and is always represented by the subject S.JCRE.

S.BCV

The bytecode verifier (BCV), which acts on behalf of the verification authority. This subject is involved in the PACKAGE LOADING security policy and is represented by the subject S.BCV.

S.CRD

The installer, which acts on behalf of the Card Issuer. This subject is involved in the loading of packages and installation of applets. It could play the role of the on-card entity in charge of package loading, which is involved in the PACKAGE LOADING security policy and is represented by the subject S.CRD.

S.ADEL

The applet deletion manager, if the configuration contains such components, which also acts on behalf of the Card Issuer. This subject is involved in the ADEL security policy and is represented by the subject S.ADEL.

S.CAD

The CAD is involved in the JCRMI security policy and is represented by the subject S.CAD.

With the exception of packages, the other subjects have special privileges and play key roles in the security policies of the TOE.

A special subject is involved in the PACKAGE LOADING security policy, which acts as the entity that may potentially intercept, modify, or permute the messages exchanged between the verification authority and the on-card entity in charge of package loading.

5.3 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 means used in the attack. The classification is also inspired by the components of the TOE that are supposed to counter each threat.

5.3.1 PHYSICAL THREAT

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 DP analysis. 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 threat refers to the security aspect #.SCP.7: the IC must be 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. It also refers to all the security aspects related to confidentiality and integrity of code and data.

. Directly threatened asset(s): **all the identified assets**.

5.3.2 CONFIDENTIALITY

T.CONFID-JCS-CODE

The attacker executes an application without authorization to disclose the Java Card System code.

. This threat refers to the security aspect #.CONFID-JCS-CODE: Java Card System code must be protected against unauthorized disclosure. 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.

. Directly threatened asset(s): **D.JCS_CODE.**

T.CONFID-APPLI-DATA

The attacker executes an application without authorization to disclose data belonging to another application.

. This threat refers to the security aspect #.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.

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

T.CONFID-JCS-DATA

The attacker executes an application without authorization to disclose data belonging to the Java Card System.

. This threat refers to the security aspect #.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 runtime environment, the virtual machine and the internal data of Java Card API classes as well.

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

5.3.3 INTEGRITY

T.INTEG-APPLI-CODE

The attacker executes an application to alter (part of) its own or another application's code.

. This threat refers to the security aspect #.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.

. Directly threatened asset(s): **D.APP_CODE**

T.INTEG-JCS-CODE

The attacker executes an application to alter (part of) the Java Card System code.

.This threat refers to the security aspect #.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.

.Directly threatened asset(s): **D.JCS_CODE**.

T.INTEG-APPLI-DATA

The attacker executes an application to alter (part of) another application's data.

. This threat refers to the security aspect #.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. If the configuration allows 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.

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

T.INTEG-JCS-DATA

The attacker executes an application to alter (part of) Java Card System or API data.

. This threat refers to the security aspect #.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 runtime environment, the virtual machine and the internal data of Java Card API classes as well.

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

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. In this configuration the integrity of a package's code is covered by the objective O.LOAD. The objective O.CARD-MANAGEMENT contributes to cover the T.INTEG-APPLICODE.LOAD threat.

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. In this configuration the integrity of a package's

code is covered by the objective O.LOAD. The objective O.CARD-MANAGEMENT contributes to cover the T.INTEG-APPLIDATA.LOAD threat.

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.

Application note:

For conformity with [5], we used T.INTEG-APPLI-DATA.**LOAD** instead of T.INTEG-APPLI-DATA.**2** (in the PP) And T.INTEG-APPLI-CODE.**LOAD** instead of T.INTEG-APPLI-CODE.**2** (in the PP) In the 2 cases, we replaced '2' by 'load', as it concerns integrity of aplication code and data when the package is transmitted [**LOADED**] to the card for installation.

5.3.4 IDENTITY USURPATION

T.SID.1

An applet impersonates another application, or even the JCRE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal.

. This threat refers to the security aspect #.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 JCRE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2). A change of identity, especially standing for an administrative role (like an applet impersonating the JCRE), is a severe violation of the TOE Security Policy (TSP). 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.

. 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.APP_KEYs** and **D.JCS_KEYs**.

T.SID.2

The attacker modifies the identity of the privileged roles.

. This threat refers to the security aspect #.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 JCRE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2). A change of identity, especially standing for an administrative role (like an applet impersonating the JCRE), is a severe violation of the TOE Security Policy (TSP). 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.

. Directly threatened asset(s): **D.SEC_DATA** (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

5.3.5 UNAUTHORIZED EXECUTION

T.EXE-CODE.1

An applet performs an unauthorized execution of a method.

. This threat refers to the security aspect #.EXE-JCS-CODE: Java Card System (byte)code must be protected against unauthorized execution. Java Card System (byte)code includes any code of the JCRE or API. This concerns:

- (1) invoking a method outside the scope of the visibility rules provided by the public/private 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.

It also refers to the security aspect #.EXE-APPLI-CODE: Application (byte) code must be protected against unauthorized execution. This concerns:

- (1) invoking a method outside the scope of the visibility rules provided by the public/private 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;
- (3) unauthorized execution of a remote method from the CAD.

. Directly threatened asset(s): **D.APP_CODE**.

T.EXE-CODE.2

An applet performs an unauthorized execution of a method fragment or arbitrary data.

. This threat refers to the security aspect #.EXE-JCS-CODE: Java Card System (byte) code must be protected against unauthorized execution. Java Card System (byte)code includes any code of the JCRE or API. This concerns:

- (1) invoking a method outside the scope of the visibility rules provided by the public/private 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.

It also refers to the security aspect #.EXE-APPLI-CODE: Application (byte)code must be protected against unauthorized execution. This concerns:

- (1) invoking a method outside the scope of the visibility rules provided by the public/private 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;
- (3) unauthorized execution of a remote method from the CAD.

. Directly threatened asset(s): **D.APP_CODE**.

T.NATIVE

An applet executes a native method to bypass a security function such as the firewall.

. This threat refers to the security aspect #.NATIVE: Because the execution of native code is outside of the TOE Scope Control (TSC), it must be secured so as to not provide ways to bypass the TSFs. No untrusted native code may reside on the card. Loading of native code, which is as well outside the TSC, 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.

. Directly threatened asset(s): **D.JCS_DATA**.

5.3.6 DENIAL OF SERVICE

T.RESSOURCES

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

. Directly threatened asset(s): **D.JCS_DATA**.

5.3.7 MODIFICATIONS OF THE SET OF APPLICATIONS

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.

. This threat refers to the security aspect #.INSTALL: Installation of a package or an applet is secure.

- (1) The TOE must be able to return to a safe and consistent state should the installation fail or be cancelled (whatever the reasons).
- (2) Installing an application 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 a secure atomic operation, and 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.

. Directly threatened asset(s): **D.SEC_DATA** (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

5.3.8 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).

. This threat refers to the security aspect #.DELETION: Deletion of applets must be secure.

- (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 TSPs. 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 TSPs.

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).

. Directly threatened asset(s): **D.SEC_DATA** and **D.APP_CODE**.

5.3.9 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.

. This threat refers to the security aspect #.OBJ-DELETION: Deallocation of objects must be secure.

- (1) It 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.

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

T.CONF_DATA_APPLET

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

Sensitive data: D.ARRAY.

5.3.10 CONFIGURATION OF THE TOE

T.CONFIGURATION

T.CONFIGURATION The attacker tries to observe or modify configuration information exchanged between the TOE and its environnment.

The TOE in this phase must protect it self 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**

5.4 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 adequacy 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.

. This organisational security policy refers to the security aspect #.VERIFICATION: All bytecode must be verified prior to being executed. Bytecode verification includes:

- (1) how well-formed CAP file is and the verification of the typing constraints on the bytecode,

- (2) binary compatibility with installed CAP files and the assurance that the export files used to check the CAP file correspond to those that will be present on the card when loading occurs.

5.5 Assumptions

This section introduces the assumptions made on the environment of the TOE for each of the configurations considered in this document.

A.NATIVE

Those parts of the APIs written in native code as well as any pre-issuance native application on the card are assumed to be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

. This assumption refers to the security aspect #.NATIVE: Because the execution of native code is outside of the TOE Scope Control (TSC), it must be secured so as to not provide ways to bypass the TSFs. No untrusted native code may reside on the card. Loading of native code, which is as well outside the TSC, 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.

A.VERIFICATION

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

A.APPLET

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

6 Security objectives

6.1 Security objectives for the TOE

The security obectives presented below are taken from SUN PP [5]. We added O.Resident Application wich concerns configuration phase of the TOE: prepersonalisation and personalisation.

6.1.1 IDENTIFICATION

O.SID

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

6.1.2 EXECUTION

O.OPERATE

The TOE must ensure continued correct operation of its security functions.

. This security objective refers to the security 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.

O.RESOURCES

The TOE shall control the availability of resources for the applications.

. This security objective refers to the security aspect #.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.

O.FIREWALL

The TOE shall ensure controlled sharing of data containers owned by applets of different packages, and between applets and the TSFs.

This security objective refers to the security aspect #.FIREWALL: The Java Card System shall ensure controlled sharing of class instances, and isolation of their data and code between packages (that is, controlled execution contexts).

- (1) An applet shall neither read, write nor compare a piece of data belonging to an applet that is not in the same context, nor execute one of the methods of an applet in another context without its authorization.

O.NATIVE

The **only** means that the JCVM shall provide for an application to execute native code is the invocation of a method of the Java Card API, or any additional API.

This security objective refers to the security aspect #.NATIVE: Because the execution of native code is outside of the TOE Scope Control (TSC), it must be secured so as to not provide ways to bypass the TSFs. No untrusted native code may reside on the card. Loading of native code, which is as well outside the TSC, 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.

O.REALLOCATION

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

O.SHRD_VAR_CONFID

The TOE shall ensure that any data container that is shared by all applications is always cleaned after the execution of an application. Examples of such shared containers are the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API.

O.SHRD_VAR_INTEG

The TOE shall ensure that only the currently selected application may grant write access to a data memory area that is shared by all applications, like the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API. Even though the memory area is shared by all applications, the TOE shall restrict the possibility of getting a reference to such memory area to the application that has been selected for execution. The selected application may decide to temporarily hand over the reference to other applications at its own risk, but the TOE shall prevent those applications from storing the reference as part of their persistent states.

6.1.3 APPLET MANAGEMENT

O.INSTALL

The TOE shall ensure that the installation of an applet is safe.

This security objective refers to the security aspect #.INSTALL: Installation of a package or an applet is secure.

- (1) The TOE must be able to return to a safe and consistent state should the installation fail or be cancelled (whatever the reasons).
- (2) Installing an application 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 a secure atomic operation, and 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.

O.LOAD

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

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.DELETION

The TOE shall ensure that both applet and package deletion are safe.

This security objective refers to the security aspect #.DELETION: Deletion of applets must be secure.

- (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 TSPs. 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 TSPs.

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).

6.1.4 OBJECT DELETION

O.OBJ-DELETION

The TOE shall ensure the object deletion shall not break references to objects.

This security objective refers to the security aspect #.OBJ-DELETION: Deallocation of objects must be secure.

- (1) It 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.

6.1.5 SERVICES

O.ALARM

The TOE shall provide appropriate feedback information upon detection of a potential security violation.

This security objective refers to the security aspect #.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 JCVM, or any other security-related event occurring during the execution of a TSF.

O.TRANSACTION

The TOE must provide a means to execute a set of operations atomically.

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

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.

This security objective refers to the security aspect #.CIPHER: The TOE shall provide a means to the upper layers 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.

O.PIN-MNGT

The TOE shall provide a means to securely manage PIN objects.

This security objective refers to the security aspect #.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.

O.KEY-MNGT

The TOE shall provide a means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys.

This security objective refers to the security aspect #.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.

O.SECURE_COMPARE

The TOE shall provide to applet a means to securely compare two byte arrays

6.1.6 Miscellaneous

O.SCP.RECOVERY

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

This security objective of the TOE refers to the security aspect #.SCP.1: The smart card platform must be secure with respect to the TSP. 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.CARD-MANAGEMENT

The card manager is in the scope of the TOE. IT 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. 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.SCP.SUPPORT

The SCP is a part of the TOE, it supports TSFs of the upper layer of the TOE.

This security objective for the TOE refers to the security aspect #.SCP.2-5:

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

O.SCP.IC

The SCP shall possess IC security features. It is included in the scope of this evaluation by composition with existing certificate.

This security objective refers to the security arspect #.SCP.7:

- (7) We finally require 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.Resident Application

This objective concerns resident application. This resident application is in the scope of the TOE.

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 and logical channels management. The dispatcher is always activated. Some card commands (for administration) are only available during prepersonalization phase. It ensures the Personalizer authentication before allowing operations in writing of the resident application.

6.1.7 Conclusion

Security objectives of PP claimed in this evaluation are not in contradiction with IC PP [5] and SUN PP [5].

6.2 Security objectives for the operational environment

This section introduces the security objectives to be achieved by the environment associated to each TOE configuration.

OE.NATIVE

Those parts of the APIs written in native code as well as any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

This security objective for the environment refers to the security aspect #.NATIVE: Because the execution of native code is outside of the TOE Scope Control (TSC), it must be secured so as to not provide ways to bypass the TSFs. No untrusted native code may reside on the card. Loading of native code, which is as well outside the TSC, 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.

OE.VERIFICATION

The TOE shall ensure that any bytecode is verified prior to being executed.

This security objective for the TOE refers to the security aspect #.VERIFICATION: All bytecode must be verified prior to being executed. Bytecode verification includes:

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

OE.APPLET

No applet loaded post-issuance shall contain native methods. As already mentioned in §2.1.3 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 card content management (loading, installation, deletion) at invalid states of the card or carried out by non-authorized actors. It shall also enforce security policies established by the card issuer.

This section introduces the security objectives to be achieved by the environment associated to each TOE configuration.

7 Extended requirements

7.1 Extended families

7.1.1 Extended family FCS_RNG - Generation of random numbers

7.1.1.1 Description

See [23] section 5.1.

7.1.1.2 Extended components

Extended component FCS_RNG.1

FCS_RNG.1 Random number generation

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

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

Dependencies: No dependencies.

Application Note: 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.

8 Security requirements

8.1 Security functional requirements

8.1.1 CoreG Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall. This policy essentially concerns the security of installed applets, along with a small part that relates to the installation procedure. The policy focuses on the execution of bytecodes.

8.1.1.1 Firewall Policy

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.LOCAL, S.MEMBER, I.DATA and OP.PUT(S1,S2,I)**.

Refinement:

Subjects(*) (prefixed with an "S") and information (prefixed with an "I") covered by this policy are:

Subject/Information	Description
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
I.DATA	JCVM Reference Data: objectref addresses of temporary JCRE Entry Point objects and global arrays.

There is a unique operation in this policy:

Operation	Description
OP.PUT(S1, S2, I)	Transfer a piece of information I from S1 to S2.

(*) Information flow policies control the flow of information between "subjects". This is a purely terminological choice; those "subjects" can merely be passive containers. They are not to be confused with the "active entities" of access control policies.

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: **(1) the currently active context**.

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

- **An operation OP.PUT(S1, S.MEMBER, I) is allowed if and only if the active context is "JCRE"; other OP.PUT operations are allowed regardless of the active context's value**.

FDP_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**.

FDP_RIP.1/OBJECTS Subset residual information protection

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

FMT_MSA.2/JCRE Secure security attributes

FMT_MSA.2.1/JCRE The TSF shall ensure that only secure values are accepted for **Active context**.

FMT_MSA.3/FIREWALL Static attribute initialisation

FMT_MSA.3.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP and 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/FIREWALL 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_SMR.1/JCRE Security roles

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

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

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes**:**

- **the active context,**
- **the SELECTed applet Context**
- **and the ActiveApplets** to **the JCRE (S.JCRE)**.

8.1.1.2 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 requirement should only apply to the implemented subset.

FCS_CKM.1/ECKeyP Cryptographic key generation

FCS_CKM.1.1/ECKeyP The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **ECKeyP** and specified cryptographic key sizes **from 160 to 521 bits** that meet the following: IEEE Std 1363a-2004.

Application note:

FCS_CKM.1/ JCAPI: the TSF generates 4 types of cryptographic keys. All these keys are specified in [JCAPI21]: ECKeyP, DES, AES and RSA and specified cryptographic key sizes (defined in the 4 components: /ECKEyP, /RSA, /AES and DES).

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 refinements below-** that meets the following:

1. See related document [7], chapter 9.5.
2. See related document [9], chapter 4.3.
3. See related document [6], packages "Javacard.security" and "Javacard.crypto".

Refinement:

Type of cryptographic key access Cryptographic key access methods (or commands) DES The following commands: PUT_KEY, EXTERNAL AUTHENTICATE, INITIALIZE UPDATE, The following "ProviderSecurityDomain" key access methods: decryptVerifyKey, openSecureChannel, unwrap, verifyExternalAuthenticate The following SecureChannel key access methods Unwrap, decryptData, encryptData, resetSecurity The following "APICrypto" key access methods:

Key.clearKey, DES.getKey, DES.setKey, Signature.init, Signature.update, Signature.sign, Signature.verify, Cipher.init, Cipher.update, Cipher.doFinal AES The following commands: PUT_KEY, EXTERNAL AUTHENTICATE, INITIALIZE UPDATE, The following "ProviderSecurityDomain" key access methods: decryptVerifyKey, openSecureChannel, unwrap, verifyExternalAuthenticate The following?SecureChannel? key access methods Unwrap, decryptData, encryptData, resetSecurity The following "APICrypto" key access methods: Key.clearKey, DES.getKey, DES.setKey, Signature.init, Signature.update, Signature.sign, Signature.verify, Cipher.init, Cipher.update, Cipher.doFinal RSA The following "ProviderSecurityDomain" key access methods: DecryptVerifyKey, The following "APICrypto" key access methods: Key.clearKey, RSAPrivateCRTKey.setP, RSAPrivateCRTKey.setQ, RSAPrivateCRTKey.setPQ, RSAPrivateCRTKey.setDP1, RSAPrivateCRTKey.setDQ1, RSAPrivateCRTKey.getP, RSAPrivateCRTKey.getQ, RSAPrivateCRTKey.getPQ, RSAPrivateCRTKey.getDP1, RSAPrivateCRTKey.getDQ1, RSAPrivateKey.setModulus, RSAPrivateKey.setExponent, RSAPrivateKey.getModulus, RSAPrivateKey.getExponent, RSAPublicKey.setModulus, RSAPublicKey.setExponent, RSAPublicKey.getModulus, RSAPublicKey.getExponent, Signature.init, Signature.update, Signature.sign, Signature.verify, Cipher.init, Cipher.update, Cipher.doFinal ECkeyP The following "APICrypto" key access methods: Key.clearKey, ECPrivateKey.setFieldFP,ECPrivateKey.setA,ECPrivateKey.setB,ECPrivateKey.setG,ECPrivateKey.setR, ECPrivateKey.setK,ECPrivateKey.getField, ECPrivateKey.getA, ECPrivateKey.getB, ECPrivateKey.getG, ECPrivateKey.getR, ECPrivateKey.getK,ECPrivateKey.setS,ECPrivateKey.getS, ECPublicKey.setFieldFP, ECPublicKey.setA, ECPublicKey.setB, ECPublicKey.setG, ECPublicKey.setR,ECPublicKey.setK, ECPublicKey.getField,ECPublicKey.getA,ECPublicKey.getB,ECPublicKey.getG,ECPublicKey.getR, ECPublicKey.getK,ECPublicKey.setW,ECPublicKey.getW, Signature.init,Signature.update, Signature.sign, Signature.verify KeyAgreement.init, KeyAgreement.generateSecret

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 in accordance with [JCAPI] in class Key with the method clearKey(). Any access to a cleared key attempting to use it for ciphering or signing shall throw an exception** that meets the following: **[JCAPI]**.

Refinement:

That also prevent the destroyed keys from being referenced.

FCS_COP.1/DES Cryptographic operation

FCS_COP.1.1/DES The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **DES** and cryptographic key sizes **of 112 bits or 168 bits (triple-DES),** that meet the following: **1. FIPS PUB 46-3 (ANSI X3.92) (see related document** [15]**), 2. FIPS PUB 81 (see related document** [16]**), 3. ISO 9797 (1999), Data integrity mechanism (see related document** [20]**)**.

FDP_RIP.1/APDU Subset residual information protection

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

FDP_RIP.1/bArray Subset residual information protection

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

FDP_RIP.1/ABORT Subset residual information protection

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

FDP_RIP.1/KEYS Subset residual information protection

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

FDP_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.JAVAOBJECTs**.

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process() or install() call, not with standing the restrictions given in [JCRE], §7.7, within the bounds of the Commit Capacity ([JCRE], §7.8), and those described in [JCAPI]**.

FCS_CKM.1/RSA Cryptographic key generation

FCS_CKM.1.1/RSA_PP The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **RSA** and specified cryptographic key sizes **1024;1280;1536;1984 and 2048-bit** that meet the following: **ANSI X9.31**.

FCS_CKM.1.1/RSA_ST The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **RSA** and specified cryptographic key sizes **from 1024 to 2048 bits with a step of 32-bit** that meet the following: **ANSI X9.31**.

In the rest of the document, there is no distinction between **FCS_CKM.1/RSA_PP** and **FCS_CKM.1/RSA_ST**).

FCS_COP.1/RSA Cryptographic operation

FCS_COP.1.1/RSA_PP The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes: **1024;1280;1536;1984 and 2048-bit** that meet the following: **1. Ansi X9.31 (see related document** [14]**), 2. ISO / IEC 9796-1, annex A, section A.4 and A.5, and annex C (see related document** [19]**) 3. PKCS#1 The public Key Cryptography standards, RSA Data Security Inc. 1993 (see related document** [22]**)**.

FCS_COP.1.1/RSA_ST The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **from 1024 to 2048 bits with a step of 32-bit** that meet the following: **1. Ansi X9.31 (see related document** [14]**), 2. ISO / IEC 9796-1, annex A, section A.4 and A.5, and annex C (see related document** [19]**) 3. PKCS#1 The public Key Cryptography standards, RSA Data Security Inc. 1993 (see related document** [22]**)**.

In the rest of the document, there is no distinction between **FCS_COP.1/RSA_PP** and **FCS_COP.1/RSA_ST**).

FCS_COP.1/AES Cryptographic operation

FCS_COP.1.1/AES The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **AES** and cryptographic key sizes **from 128 to 256 bits with a step of 64 bits** that meet the following: **FIPS PUB 197 (see related document [26])**.

FCS_COP.1/EC Cryptographic operation

FCS_COP.1.1/EC The TSF shall perform **signature, verification of signature** in accordance with a specified cryptographic algorithm **EC** and cryptographic key **from 160 to 521** bits that meet the following standards: **IEEE Std 1363a-2004*, ANSI X9.62, ANSI X9.63***

FCS_COP.1/SHA Cryptographic operation

FCS_COP.1.1/SHA The TSF shall perform **Hash functions** in accordance with a specified cryptographic algorithm **SHA-1, SHA-256, SHA-384 and SHA-512** and cryptographic key sizes **no keys** that meet the following: **FIPS PUB 180-2**. A proprietary cryptographic algorithm **SHA-224** is available (based on the **SHA-256** algorithm).

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 **command setKey** that meets the following: **[JCAPI] specification**.

FCS_CKM.1/DES Cryptographic key generation

FCS_CKM.1.1/DES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **TDES** and specified cryptographic key sizes **112 bits or 168 bits** that meet the following: **1. FIPS PUB 46-3 (ANSI X3.92) (see related document** [15]**), 2. FIPS PUB 81 (see related document** [16]**), 3. ISO 9797 (1999), Data integrity mechanism (see related document** [20]**)**.

FCS_CKM.1/AES Cryptographic key generation

FCS_CKM.1.1/AES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **AES** and specified cryptographic key sizes **from 128 to 256 bits with a step of 64 bits** that meet the following: **FIPS PUB 197 (see related document [26]).**

8.1.1.3 Card Security Management

The following SFRs are related to the security requirements at the level of the whole card, in contrast to the previous ones, that are somewhat restricted to the TOE alone. For instance, a potential security violation detected by the virtual machine may require a reaction that does not only concern the virtual machine, such as blocking the card (or request the appropriate security module with the power to block the card to perform the operation).

FAU_ARP.1/JCS Security alarms

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

- **throw an exception,**
- **lock the card session**
- **or reinitialize the Java Card System and its data** upon detection of a potential security violation.

Refinement:

Potential security violation is refined to 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
- Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI] and ([JCRE], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Other runtime errors related to applet's failure (like uncaught exceptions)

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 Kill card file**.

Application note:

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

1. PINs (i.e. objects instance of class OwnerPin or subclass of interface PIN)

2. keys (i.e. objects instance of classes implemented the interface Key)

3. SecureStores (i.e. objects instance of class SecureStore)

4. packages Java Card

5. patches

If the maximum is reached (15) the the Kill card is launched.

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 (shared between the card manager and the TOE), the bytecode and its data arguments (shared with applets and API packages),** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use **the following rules:**

- **The [JCVM] specification;**
- **Reference export files;**
- **The ISO 7816-6 rules;**
- **The EMV specification** when interpreting the TSF data from another trusted IT product.

FPT_FLS.1/JCS Failure with preservation of secure state

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

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **any user** are unable to observe the operation **Cardholder authentication** on **D.PIN** by **No user and no subject**.

FPT_TST.1/RESET TSF testing

FPT_TST.1.1/RESET [Editorially Refined] The TSF shall run a suite of self tests **during initial start-up (at each power on)** to demonstrate the correct operation of the TSF.

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

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

Application note:

At each start up, a self test of the hardware is done. Random, DES, and CRC functional modules are tested always at each start up: implements a know calculus and checks if the result is correct. SHA, RSA, AES and ECC functional modules are tested at each start up or at first used: implements a know calculus and checks if the result is correct.

FDP_SDI.1 Stored data integrity monitoring

FDP_SDI.1.1 The TSF shall monitor user data stored in containers controlled by the TSF for **Integrity errors** on all objects, based on the following attributes: **integrityData**.

Application note:

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

SecureStores (i.e. objects instance of class SecureStore)

FPT_TST.1FIRST_USED TSF testing

FPT_TST.1.1FIRST_USED [Editorially Refined] The TSF shall run a suite of self tests **at first use** to demonstrate the correct operation of the TSF.

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

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

Application note:

Each crypto functional module used SHA, RSA, AES and ECC is tested as first used if not at start up.

8.1.1.4 AID Management

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' AID** to **the JCRE**.

FMT_MTD.3 Secure TSF data

FMT_MTD.3.1 The TSF shall ensure that only secure values are accepted for **TSF data**.

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: **the AID and version number of each package, the AID of each registered applet, and whether a registered applet is currently selected for execution ([JCVM], §6.5) ASG.APPPRIV and CONTEXT**.

FIA_UID.2/AID User identification before any action

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

FIA_USB.1 User-subject binding

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

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **rules are defined in FDP_ACC.2/Firewall and ADP_ACF/Firewall**.

FIA_USB.1.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **none**.

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_SMF.1/JCRE Specification of Management Functions

FMT_SMF.1.1/JCRE The TSF shall be capable of performing the following management functions: **Modify the active context and the SELECTed applet Context. o Modify the list of registered applets' AID**.

8.1.2 ADELG Security Functional Requirements

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

8.1.2.1 Applet Deletion Manager Policy

FDP_ACC.2/ADEL Complete access control

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

Refinement:

Subjects (prefixed with an "S") and objects (prefixed with an "O") covered by this policy are:

Subject/Object	Description
S.ADEL	The applet deletion manager. It may be an applet ([JCRE22], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique.

Subject/Object	Description
O.CODE_PKG	The code of a package, including all linking information. On the Java Card platform, a package is the installation unit.
O.APPLET	Any installed applet, its code and data.
O.JAVAOBJECT	Java class instance or array.

Operations (prefixed with "OP") of this policy are described in the following table.

Operation	Description
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.

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:

- **(1) the security attributes of the covered subjects and objects,**
- **(2) the list of AIDs of the applet instances registered on the card,**
- **(3) the attribute ResidentPackages, which journals the list of AIDs of the packages already loaded on the card, and**
- **(4) the attribute ActiveApplets, which is a list of the active applets' AIDs**.

Refinement:

The following table presents some of the security attributes associated to the subjects/objects under control of the policy. However, they are mostly implementation independent.

Subject/Object	Attributes
O.CODE_PKG	package's AID, dependent packages' AIDs, Static References
O.APPLET	Selection state
O.JAVAOBJECT	Owner, Remote

The package's AID identifies the package defined in the CAP file.

When an export file is used during preparation of a CAP file, the version numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM], §4.5.2): the dependent packages AIDs attribute allows the retrieval of those identifications.

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

An applet instance can be in two different selection states: selected or deselected. If the applet is selected (in some logical channel), then in turn it could either be currently selected or just active. At any time there could be up to four active applet instances, but only one currently selected. This latter is the one that is processing the current command ([JCRE22], §4).

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).

An object is said to be a Remote if it is an instance of a class that directly or indirectly implements the interface java.rmi.Remote.

Finally, there are needed security attributes that are not attached to any object or subject of the TSP: (1) the ResidentPackages Versions (or Resident Image,[JCVM],§4.5) and AIDs. They describe the packages that are already on the card, (2) the list of registered applet instances and (3) the ActiveApplets security attribute. They are all attributes internal to the VM, that is, not attached to any specific object or subject of the SPM. These attributes are TSF data that play a role in the SPM.

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: **by the ADEL SFP**

- **R.JAVA.14 ([JCRE22], §11.3.4.1, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) O.APPLET is deselected and**
  - **(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.15 ([JCRE22],§11.3.4.1, Multiple Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) every O.APPLET being deleted is deselected and**
  - **(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.16 ([JCRE22], §11.3.4.2, Applet/Library Package Deletion). The S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PCKG only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PCKG that is an instance of a class that belongs to O.CODE_PCKG exists on the card and**
  - **(3) there is no package loaded on the card that depends on O.CODE_PCKG.**

- **R.JAVA.17 ([JCRE22], §11.3.4.3, Applet Package and Contained Instances Deletion). The S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PCKG only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PCKG, which is an instance of a class that belongs to O.CODE_PCKG exists on the card,**
  - **(3) there is no package loaded on the card that depends on O.CODE_PCKG and**
  - **(4) for every O.APPLET of those being deleted it holds that:**

- **(i) O.APPLET is deselected and**

- **(ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE22],§8.5) O.JAVAOBJECT is remote reachable**.

Refinement:

The subjects of this policy is S.ADEL.

Some basic common specifications are required in order to allow Java Card applets and packages to be deleted without knowing the implementation details of a particular deletion manager. In particular, this policy introduces a notion of reachability, which provides a general means to describe objects that are referenced from a certain applet instance or package.

In the context of this policy, an object O is reachable if and only if either:

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

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 the S.ADEL to O.CODE_PKG or O.APPLET for the purpose of deleting it from the card**.

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 **ActiveApplets** to **the JCRE (S.JCRE)**.

FMT_SMR.1/ADEL Security roles

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

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

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 ActiveApplets security attribute**.

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.

8.1.2.2 Additional Deletion Requirements

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 **allocation of the resource to** the following objects: **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**.

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 [JCRE22], §11.3.4**.

8.1.3 LCG Security Functional Requirements

The security issues introduced by logical channels are mainly related to the access to SIO objects owned by legacy applets as well as to the clearing of transient data which is shared by applet instances which are concurrently active in different logical channels. Accordingly, this group introduces a reformulation of the FIREWALL SFP specified in the group CoreG and a modification to a component of the security requirement for residual information protection (FDP_RIP.1.1/TRANSIENT).

8.1.3.1 Firewall Policy

Except for the requirements explicitly introduced in what follows, this policy includes unchanged the functional requirements specified in the FIREWALL access control SFP of the group CoreG.

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, O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

Refinement:

Subjects (prefixed with an "S") and objects (prefixed with an "O") covered by this policy are:

Subject	Description
S.PACKAGE	Any package, which is the security unit of the firewall policy.
S.JCRE	The JCRE. This is the process that manages applet selection and de-selection, along with the delivery of APDUs from and to the smart card device. This subject is unique.
O.JAVAOBJECT	Any object. Note that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.

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

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object)
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow).
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object.
OP.JAVA(...)	Any access in the sense of [JCRE], §6.2.8. In our formalization, this is one of the preceding operations.

Operation	Description
OP.CREATE(Sharing, LifeTime)	Creation of an object (new or makeTransient call).

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

FDP_ACF.1/FIREWALL Security attribute based access control

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

- **(1) the security attributes of the covered subjects and objects,**
- **(2) the currently active context,**
- **(3) the SELECTed applet Context, and**
- **(4) the attribute ActiveApplets, which is a list of the active applets' AIDs**.

Refinement:

The following table describes which security attributes are attached to which subject/object of our policy.

Subject/Object	Attributes
S.PACKAGE	Context, Selection Status
S.JCRE	None
O.JAVAOBJECT	Sharing, Context, LifeTime

The following table describes the possible values for each security attribute.

Name	Description
Context	Package AID, or "JCRE"
Sharing	Standard, SIO, JCRE entry point, or global array
LifeTime	CLEAR_ON_DESELECT or PERSISTENT(*).
SELECTed applet Context	Package AID, or "None"
Selection Status	Multiselectable, Non-multiselectable or "None"
ActiveApplets	List of package's AIDs

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

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

The Sharing attribute defines four categories of objects:

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

When a new object is created, it is associated with the currently active context. But the object is owned by the applet instance within the currently active context when the object is instantiated ([JCRE], §6.1.2). 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).

Finally both "the currently active context" and "the SELECTed applet context" are security attributes internal to the VM, that is, not attached to any specific object or subject of the Security Policy Model ("SPM"). They are TSF data that play a role in the SPM.

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

While the expression "selected applet" refers to a specific installed applet, the relevant aspect to the policy is the context of the selected applet; that is why the associated security attribute is a package AID.

([JCRE] §6.1.1) At any point in time, there is only one active context within the VM (this is called the currently active context).

This should be identified in our model with the acting S.PACKAGE's context. This value is in one-to-one correspondence with AIDs of packages (except for the JCRE context, of course), which appears in the model in the "Context" attribute of both subjects and objects of the policy. The reader should note 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 in this case.

The Java Card platform, version 2.2, 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 ([JCVM22], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

Support for multiple logical channels (with multiple selected applet instances) requires a change to the Java Card System, version 2.1.1, concept of selected applet. Since more than one applet instance can be selected at the same time, and one applet instance can be selected on different logical channels simultaneously, it is necessary to differentiate the state of the applet instances in more detail. An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel, up to a maximum of four. 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. ([JCRE22],§4).

The ActiveApplets security attribute is internal to the VM that is, not attached to any specific object or subject of the SPM. The attribute is TSF data that plays a role in the SPM.

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: **by the FIREWALL SFP**

- **R.JAVA.1 ([JCRE]§6.2.8) An S.PACKAGE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**
- **R.JAVA.2 ([JCRE]§6.2.8) An S.PACKAGE may freely perform any of OP.ARRAY_ACCESS,OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**
- **R.JAVA.3 ([JCRE]§6.2.8.10) An S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**
- **R.JAVA.20 ([JCRE22], §6.2.8.6,) An S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if one of the following applies:**
  - **a) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",**
  - **b) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets,**

    and in either of the cases above the invoked interface method extends the Shareable interface.
- **R.JAVA.5 An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".**
- **At last, rules governing access to and creation of O.JAVAOBJECTs by S.JCRE are essentially implementation-dependent (however, see FDP_ACF.1.3/FIREWALL.)**.

Refinement:

(*) For this operation, there is no accessed object; the "Sharing value" thus refers to the parameter of the operation. This rule simply enforces that shareable transient objects are not allowed. Note: parameters can be seen as security attributes whose value is under the control of the subject. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

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

- **The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL**.

FDP_ACF.1.4/FIREWALL The TSF shall explicitly deny access of subjects to objects based on the **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 with OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the SELECTed applet Context**.

8.1.3.2 Additional Requirements on Logical Channels

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**.

8.1.4 ODELG Security Functional Requirements

The following requirements are concerned with the secure deletion of information provoked by the object deletion mechanism. This mechanism is triggered by the applet who owns the deleted objects by invoking a specific API method.

FDP_RIP.1/ODEL Subset residual information protection

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

FPT_FLS.1/ODEL Failure with preservation of secure state

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

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

8.1.5 SCPG Security Functional Requirements

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

The TSF and TSC stated in the following components refer to that of the SCP.

FPT_FLS.1/SCP Failure with preservation of secure state

FPT_FLS.1.1/SCP The TSF shall preserve a secure state when the following types of failures occur: **1. Invalid reference exception; 2. Code or data integrity failure; 3. Power loss while processing. 4. worm on or dead EEPROM, full security area, false CRC For each problem the TOE sends a specific exception status or doesn't start**.

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: **Lack of EEPROM**.

FPT_PHP.3/SCP Resistance to physical attack

FPT_PHP.3.1/SCP The TSF shall resist **changing operational conditions every times: the frequency of the external clock, power supply, and temperature** to the **the chip elements** by responding automatically such that the SFRs are always enforced.

FPT_RCV.3/SCP Automated recovery without undue loss

FPT_RCV.3.1/SCP 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.

FPT_RCV.3.2/SCP For **all cases**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/SCP The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **100%** for loss of TSF data or objects under the control of the TSF.

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

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.

FCS_RNG.1/IC_SOFT Random number generation

FCS_RNG.1.1/IC_SOFT The TSF shall provide a **hybrid** random number generator that implements: **None**.

FCS_RNG.1.2/IC_SOFT The TSF shall provide random numbers that meet **khi2 test**.

FPR_UNO.1/USE_KEY Unobservability

FPR_UNO.1.1/USE_KEY The TSF shall ensure that **all subjets** are unable to observe the operation **Use** on **Key** by **D.KEY**.

8.1.6 Additional requirements for CM

FCO_NRO.2/CM_TOKEN Enforced proof of origin

FCO_NRO.2.1/CM_TOKEN [Editorially Refined] The TSF shall enforce the generation of evidence of origin for transmitted **Pre-Authorisation for the delegated operation** at all times.

FCO_NRO.2.2/CM_TOKEN [Editorially Refined] The TSF shall be able to relate the **Card Issuer** of the originator of the information, and the **AS.KEYSET_Value** the information to which the evidence applies.

FCO_NRO.2.3/CM_TOKEN The TSF shall provide a capability to verify the evidence of origin of information to **originator** given **all time**.

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 [Editorially Refined] 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 [Editorially Refined] The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **during Cap File loading**.

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 **surpassed and met**, the TSF shall **slow down the next authentication in accordance with the following function: The waiting time is exponential with a maximum number of unsuccessful authentications of 15**.

FIA_UAU.1/CM Timing of authentication

FIA_UAU.1.1/CM The TSF shall allow **Get Data, Initialize Update** 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 **the Card Issuer authentication mechanism**.

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 subjets** are unable to observe the operation **Import** on **Key** by **D.KEY**.

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, Packages** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2/CM The TSF shall use **the PUT KEY data format** when interpreting the TSF data from another trusted IT product.

Application note:

when importing a key from another Key generator

FAU_SAR.1/CM Audit review

FAU_SAR.1.1/CM The TSF shall provide **all user (getData)** with the capability to read **D.AuditLog file (Authentification failure)** from the audit records.

FAU_SAR.1.2/CM The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

FAU_GEN.1/CM Audit data generation

FAU_GEN.1.1/CM The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the **not specified** level of audit; and

c) **authentication failure**.

FAU_GEN.1.2/CM The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, **AID of user; number of authentication failure present in the D.AUDITLOG file**.

8.1.7 Resident application

FDP_ACC.2/PP Complete access control

FDP_ACC.2.1/PP The TSF shall enforce the **Prepersonalization access control** on **S.Resident application and for all objects** and all operations among subjects and objects covered by the SFP.

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.

FDP_ACF.1/PP Security attribute based access control

FDP_ACF.1.1/PP The TSF shall enforce the **Prepersonalization access control** to objects based on the following: **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: **AS.AUTH_MSK_STATUS = TRUE**.

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 **none**.

FDP_UCT.1/PP Basic data exchange confidentiality

FDP_UCT.1.1/PP The TSF shall enforce the **Prepersonalization access control** to be able to **receive** user data in a manner protected from unauthorised disclosure.

Application note:

The TOE is able to receive crypted INIT KEY

FDP_ITC.1/PP Import of user data without security attributes

FDP_ITC.1.1/PP The TSF shall enforce the **Prepersonalization 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 **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, 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 **Prepersonalization access control** to restrict the ability to **modify** the security attributes **AS.MSKEY** to **R.Prepersonalizer**.

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 security attributes**.

FIA_ATD.1/CardManu User attribute definition

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

FIA_UAU.4/CardManu Single-use authentication mechanisms

FIA_UAU.4.1/CardManu The TSF shall prevent reuse of authentication data related to **the Card Manufacturer authentication mechanism**.

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 **disable** the functions **of resident application: INITIALIZE AUTHENTICATION PROCESS, EXTERNAL AUTHENTICATE, LOAD STRUCTURE, INSTALL, LOAD APPLET, GET DATA** to **R.Prepersonalizer**.

FMT_MOF.1/PP_TOE Management of security functions behaviour

FMT_MOF.1.1/PP_TOE The TSF shall restrict the ability to **modify the behaviour of** the functions **All functions** to **R.Prepersonalizer**.

Application note:

This function permits on PP phase to load patch. This patch is in the scope of the evaluation.

FMT_SMR.2/PP Restrictions on security roles

FMT_SMR.2.1/PP The TSF shall maintain the roles: **R.Prepersonaliser, R.Personaliser**.

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 rafinement below** are satisfied.

Refinement:

Roles //Condition for this role// :

-R.Prepersonalizer //Successful authentication (of Card Manufacturer) using Transport key and card still in prepersonalization state)//

- R.Personalizer //Successful authentication (of Card Manufacturer or Card Issuer) using a key set of the Card Manager, with CM life cycle phase from OP_READY to SECURED //

FMT_MSA.3/PP Static attribute initialisation

FMT_MSA.3.1/PP The TSF shall enforce the **Prepersonalization access control** to provide **same rights by** 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/DES_PP Cryptographic operation

FCS_COP.1.1/DES_PP The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **DES** and cryptographic key sizes **of 112 bits or 168 bits (triple-DES),** that meet the following: **1. FIPS PUB 46-3 (ANSI X3.92) (see related document** [15]**), 2. FIPS PUB 81 (see related document** [16]**), 3. ISO 9797 (1999), Data integrity mechanism (see related document** [20]**)**.

FCS_COP.1/AES_PP Cryptographic operation

FCS_COP.1.1/AES_PP The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **AES** and cryptographic key sizes **from 128 to 256 bits with a step of 64 bits** that meet the following: **FIPS PUB 197 (see related document [26])**.

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:

MSKKey and its associated ckecksum are set to null. It permits to never reuse it.

8.1.8 CarG and CMGR Security Functional Requirements

The 2 groups concerning the Cad Manager are merged:

As the bytecode verifier is not embedded on the card, the groupe CMGR is included. It includes requirements for preventing the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.

The second group CARG helps define a policy for controlling access to card content management operations and for expressing card issuer security policies.

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:

While entity (Card Manager or Security Domain) is selected, this function tests for every command if the secure channel is open.

If a secure channel is opened, this function tests according to Card Manager life cycle for every command if secure messaging is required.

If the secure channel is not open then only the command establishing a secure channel channel and the Get data command are available.

FDP_ACF.1/CMGR_SD Security attribute based access control

FDP_ACF.1.1/CMGR_SD The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to objects based on the following: **AS.AUTH_CM_STATUS and AS.SECURITY_LEVEL**.

FDP_ACF.1.2/CMGR_SD The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **1/ AS.AUTH_CM_STATUS = TRUE, 2/ if AS.SECURITY_LEVEL and MAC # 0, integrity of imported objects is ensured, 3/ if AS.SECURITY_LEVEL and ENC # 0, confidentiality of imported objects is ensured**.

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

FDP_ACF.1.4/CMGR_SD The TSF shall explicitly deny access of subjects to objects based on the **none**.

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.CRD, S.BCV, S.SPY** and all operations that cause that information to flow to and from subjects covered by the SFP.

Refinement:

Subjects (prefixed with an "S") covered by this policy are those involved in the reception of an application package by the card through a potentially unsafe communication channel:

Subject	Description
S.BCV	The subject representing who is in charge of the bytecode verification of the packages (also known as the verification authority).
S.CRD	The on-card entity in charge of package downloading.
S.SPY	Any other subject that may potentially intercept, modify, or permute the messages exchanged between the former two subjects.

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

Operation	Description

Operation	Description
OP.SEND(M)	A subject sends a message M through the communication channel.
OP.RECEIVE(M)	A subject receives a message M from the communication channel.

The information (prefixed with an "I") controlled by the typing 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 (either S.BCV or S.SPY) in the communication protocol.

Information	Description
I.APDU	Any APDU sent to or from the card through the communication channel.

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

FDP_IFF.1/CM Simple security attributes

FDP_IFF.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** based on the following types of subject and information security attributes: **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: **[assignment: the rules describing the communication protocol used by the CAD and the card for transmitting a new package]**.

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

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

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

FDP_UIT.1/CM Data exchange integrity

FDP_UIT.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to be able to **receive** user data in a manner protected from **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.

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**.

Application note:

The Other privileges cannot be modified.

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 **Security Domain Status = Locked** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/CM The TSF shall allow the **R.CardManager** to specify alternative initial values to override the default values when an object or information is created.

Application note:

when the Security Domain Status = Locked, it means package loading is forbidden.

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles **R.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 **installing a new application package on the card**.

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 **Card Issuer** given **none**.

FMT_MSA.1/CM_DEL Management of security attributes

FMT_MSA.1.1/CM_DEL The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to restrict the ability to **delete** the security attributes **S.KEYSET_VERSION, AS.KEYSET_VALUE** to **R.CARD_Manager**.

Application note:

All other privileges cannot be modified.

FMT_SMR.2/CM Restrictions on security roles

FMT_SMR.2.1/CM The TSF shall maintain the roles: **see elements in 2.3**.

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 delails below:**

Roles //Condition for this role//

-R.Personalizer //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.

FDP_ACC.2/CMGR Complete access control

FDP_ACC.2.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** on **D.LOADFILE, AS.KEYSET_VALUE, D.GLPIN, ASG.APPPRIV, AS.CMLIFECYC, AS.KEYSET_VERSION, D.APPLILIFECYC, ASG.CARDREG** and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/CMGR 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.

FMT_MSA.1/CMGR-PRIVMOD Management of security attributes

FMT_MSA.1.1/CMGR-PRIVMOD The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to restrict the ability to **modify** the security attributes **AS.CMLIFECYC** to **R.Card_Manager**.

FIA_UID.1/CM_SD Timing of identification

FIA_UID.1.1/CM_SD The TSF shall allow **execution of 'issuer Security Domain, execution of supplementary Security Domain** on behalf of the user to be performed before the user is identified.

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

Application note:

While entity is selected (Card Manager or Security Domain) is selected, these function tests for every command if the secure channel is open.

If a secure channel is opened, these functions test according to Card Manager life cycle for every command if secure messaging is required.

If the secure channel is not open then only the command establishing a secure channel and the get data command are available.

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 security attributes**.

8.1.9 InstG Security Functional Requirements

This group bulks the SFRs related to the installation of the applets, which addresses security aspects outside the runtime. The idea here is that installation of applets is a critical phase, which lies partially out of the boundaries of the firewall, and therefore has to be deserved specific treatment. In the Common Criteria model, loading a package or installing an applet was considered as being an importation of user data (that is, user application's data) with its security attributes (such as the parameters of the applet used in the firewall rules).

See also FIA_ATD.1, FIA_USB.1, FMT_MTD.1, FMT_SMR.1 for various information about applet installation.

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

FDP_ITC.2.1/Installer The TSF shall enforce the **FIREWALL access 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:

A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs. The loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM],§4.5.2). The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM], §4.4).

FMT_SMR.1/Installer Security roles

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

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

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [JCRE] §10.1.4**.

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **An applet (i.e. a package) is considered as loaded, once its reference is written is the list of the loaded packages (i.e. instantiated applets) This is the ultimate stage of the applet/package installation, done when everything has succeeded before (verification, initialisation, object instantiation) If an error occur before registration, everything must be rolled**

back. For package installation, the garbage collector will automatically remove the package code since we stopped installation before the package recording. For applet installation, we mainly relies on garbage collector, as it is done for package, to remove the applet instance and AID objects (since the applet is not on the root of persistence, these objects are unreachable). On applet installation, its install method is called which can lead to change the states of the VM objects. To rollback the modifications eventually made in field of other persistent objects, the installation is surrounded by a transaction (that is aborted). Finally, we have additional mechanisms to rollback modifications eventually done is the field of transient arrays since they are not covered but the transaction (volatile data is not in the scope of Java Card transaction) is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/Installer For **installation of the applet**, 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 **100%** 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.

FRU_RSA.1/Installer Maximum quotas

FRU_RSA.1.1/Installer [Editorially Refined] The TSF shall enforce maximum quotas of the following resources: **imported packages and declared classes, methods and fields** that **packages** can use **simultaneously**.

8.1.10 Additional Fonctional Requirements for the applets

Its recommended to applet to use this fonction to ensure security

FIA_AFL.1/Pin Authentication failure handling

FIA_AFL.1.1/Pin The TSF shall detect when **an administrator configurable positive integer within user defined maximum 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**.

Application note:

This requirement implies that the TOE must be able to track unsuccessful authentication and to 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**.

Application note:

Global Pin = CVM

FPR_UNO.1/applet Unobservability

FPR_UNO.1.1/applet The TSF shall ensure that **S.Applet** are unable to observe the operation **Comparison** on **of two byte arrays** by **D.ARRAY**.

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 **Owner PIN** 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 PIN**.

Application note:

This requirement implies that the TOE must be able to track unsuccessful authentication and to block the PIN.

8.1.11 Requirement for the IC

FPT_PHP.3/IC Resistance to physical attack

FPT_PHP.3.1/IC The TSF shall resist **changing operational conditions every times: the frequency of the external clock, power supply, and temperature** to the **chip elements** by responding automatically such that the SFRs are always enforced.

FCS_COP.1/IC Cryptographic operation

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

Refinement of FCS_COP.1.1	List of operations	list of algorithms	key sizes	list of standards
TDES (IC)	TDES encryption and decryption	Triple Data Encryption (TDES)	112-bits	E-D-E two-key triple-encryption implementation of the Data Encryption Standard, FIPS PUB 46-3, 25 Oct. 1999
AES (IC)	AES encryption and decryption	Advanced Encryption Standard (AES)	128-192-256 bits	FIPS PUB 197, Nov 2001

FCS_RNG.1/IC Random number generation

FCS_RNG.1.1/IC The TSF shall provide a **physical** random number generator that implements: **none**.

FCS_RNG.1.2/IC The TSF shall provide random numbers that meet **Fips 140-2**.

Application note: The FCS_RNG.1.1/IC is based on FCS_RND.1_IC from [24].

8.2 Security assurance requirements

This composite ST has a set of assurance requirements at EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

This level allows a developer to attain a reasonably high assurance level without the need for highly specialized processes and practices. It corresponds to a white box analysis and it can be considered as a reasonable level that can be applied to an existing product line without undue expense and complexity. The TOE is intended to operate in open environments, where attackers can easily exploit vulnerabilities. According to the claimed intended usage of the TOE, it is very likely that it may represent a significant value and then constitute an attractive target for attacks. In some malicious usages of the TOE the statistical or probabilistic mechanisms in the TOE, for instance, may be subjected to analysis and attack in the normal course of operation. This level seems to be the reasonable minimum level for cards hosting sensitive applications.

9 TOE Summary Specification

9.1 TOE Summary Specification

The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. The assurance component ADV_IMP.2 has been chosen for the platform implementation (rom code) because the evaluation of the TOE must ensure that its security functional requirements are completely and accurately addressed by the implementation representation of the TSF.The evaluation of the TOE may be performed, for instance, because the product hosts one or several sensitive applications, such as financial and health recording ones, which contain, represent, or provide access to valuable assets. In addition to that the TOE may not be directly under the control of trained and dedicated administrators.

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 [JCRE222]. 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_Card_Content_Management

This TSF controls the loading, installation, and deletion of packages and applet instances: o CARD CONTENT LOADING. 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 CARD CONTENT INSTALLATION. This function allows the Installer to make actually selectable a previously loaded Applet subclass. In order to do this, an instance of such class, called an applet instance, has to be created: 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 CARD CONTENT DELETION. 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. If this is the 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.

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 include 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 trace of which are the currently active ones. It therefore handles sensitive TSF data of other security functions, like the Firewall or the Remote Access Control function.

SF_Signature

This TSF provides 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. The ciphering algorithms are available to the applets through the javacard.Signature class of the Java Card API. 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.

SF_Ciphering

This TSF provides the applet instances with a mechanism 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. 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.

SF_Message_Digest

This TSF provides the applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. 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. Message digest generation is implemented to resist to environmental stress and glitches and include measures for preventing information leakage through covert channels.

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

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_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 are, the integrity value is checked and maintained valid. In case of incoherency, an exception is raise to prevent the bad use of the data.

SF_Clearing_of_Sensitive_Information

This TSF clears all the data containers that hold sensitive information when that information is no longer used. 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 All the information contained in the packages that is not necessary for executing the code of the applets, like the Descriptor Component, the Reference Location Component and the Constant Pool of the CAP files. o The contents of the APDU buffer after processing an APDU command. o The contents of the bArray argument of the Applet.install method after a new applet instance is installed. o The contents 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 contents of all transient objects after a card reset. o The reason code contained in the instances of a CardException or CardRuntimeException classes after a card reset. o The validated flag of the PINs after a card reset. o The contents of the cryptographic buffer after performing cryptographic operations. o The contents of the input parameters of a remote method invocation after returning the response to the terminal.

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_Atomic_Transactions

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 Java Card API. This TSF also performs the actions necessary to roll back to a safe state upon interruption of the following procedures, for example because of a card withdrawal or an unexpected fatal error: 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. 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 [JCRE222] and in the documentation associated to the JCSystem class in the [JCAPI222].

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_Distribution

This TSF enforces the distribution of all the cryptographic keys of the card using the method specified in that SFR.

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.

SF_Hardware_Operating

When needed, at each start up or before first use, a self test of each hardware functional module is done, i.e.: DES, RSA, RNG implements a know calculus and checks if the result is correct.

When executing, external hardware event can be trigged to prevent attacks or bad use. Temperature, frequency, voltage, light, glitch are considered as abnormal environmental conditions and put the card in frozen state

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_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_Exception

In case of abnormal event: data unavailable on an allocation, illegal access to a data, the system owns an internal mechanism that's allows to stop the code execution and raise an exception.

SF_DAP_Verification

An Application Provider may require that their 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 detailed in this Specification 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 this Specification provides this service on behalf of the Controlling Authority. The key and algorithm to be used for DAP Verification or Mandated DAP Verification are implicitly known by the corresponding Security Domain.

SF_TOKEN

This function ensures the validity of the card content management command when this command is issued from an Security Domain with Delegated Management.

SF_Manufacturer_Authentication

At prepersonalization phase, manufacturer authentication at the beginning of a communication session is mandatory prior to any relevant data being transferred to the TOE.

SF_Resident_Application_Dispatcher

During prepersonalization phase, this function tests for every command if manufacturer authentication is required.

SF_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).

SF_GP_Dispatcher

While the card manager or a security domain is selected, this function tests for every command if security requirements are needed: If secure channel is required. If its required, the function tests

according to the Security Domain state and the Card state for every command if secure messaging is required.

SF_KEY_MANAGEMENT

this function gives le management of key set: Loading keys, add a new key set (version and value of the key) or update a key set (update key value).

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 [24].

SF_Unobservability

This function assures that processing based on secure elements of the TOE does not reveal any information on those elements. For example, observation of a PIN verification cannot reveal the PIN value, observation a cryptographic computation cannot give information on the key.

SF_Prepersonalization

This function is in charge of pre-initializing the internal data structures, loading the configuration of the card and loading patch code if need.

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 Std 1363a-2004].