Oberthur Card Systems

LEO

Security Target

Reference:

057991 00 UDD AA

CONTENTS

PREFACE

Objectives of the Document

This document aims to satisfy the requirements of Common Criteria level EAL1 augmented in defining the security enforcing functions of the Target Of Evaluation and describing the environment in which it operates.

Scope of the Document

This document describes the Security Target for the JPH33V2 card, which supports Javacard 2.1.1 features.

Related Document

[1] "Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", August 1999, version 2.1, CCIMB-99-031

[2] "Common Criteria for information Technology Security Evaluation, Part 2: Security Functional requirements", August 1999, version 2.1, CCIMB-99-032

[3] "Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance requirements", August 1999, version 2.1, CCIMB-99-033

[4] "Protection Profile - Smart Card Integrated Circuit with Embedded Software", Version 2.0, Issue June 1999, registered at the French Certification Body under the number PP/9911

[5] “JPH33V2 - Detailed functional specifications”, ref: 0240 55471 00 SRS, revision-issue 1-AA, 31/07/2000, Oberthur Card Systems

[6] "Java Card 2.1.1 - Application Programming Interfaces", May 18 2000, Sun Microsystems

[7] "Java Card 2.1.1 - JCRE", May 18 2000, Sun Microsystems

[8] "Java Card 2.1.1 - Virtual Machine Specifications", May 18 2000, Sun Microsystems

[9] "Open Platform Card Specification", version 2.0.1 December 31, 1999, Visa International

[10] Visa Open Platform Card Implementation Specification", March 8, 1999, Visa International

[11] EMV '96': Integrated Circuit Card Specifications for Payment Systems

[12] "Identification cards - Integrated Circuit(s) Cards with contacts, Part 6: Inter industry data elements", ISO / IEC 7816-6 (1996)

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

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

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

[16] "FIPS PUB 180-1, Secure Hash Standard", April 17, 1995, National Institute of Standards and Technology

[17] "FIPS PUB 184-2", April 17, 1995, National Institute of Standards and Technology

[18] "Information Processing Modes of Operation for a 64-Bit Block Cipher Algorithm", ISO 8372 (1987), International Organization for Standardization

[19] "Banking – Key Management", ISO 8732 (1988), International Organization for Standardization

[20] "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)

[21] "Information technology – Security techniques: Data integrity mechanism using a cryptographic check function employing a block cipher algorithm", ISO 9797 (1994) , International Organization for Standardization

[22] "FIPS PUB 140-1, Security requirements for cryptographic modules", January 11, 1994, National Institute of Standards and Technology

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

Document Structure

This document covers the following topics:

- ST identification
- TOE and environment description
- TOE security environment (assets, assumptions, threats, OSP)
- Security Objectives for the TOE
- Derived security functional requirements
- TOE Summary specification
- PP Claims
- Rationale

Abbreviations and Notations

Abbreviations

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
EEPROM	Electrically Erasable and Programmable Read Only Memory
FAMEX	Co-processor for public key crypto calculations
IC	Integrated Circuit

IT	Information Technology
JCP	Java Card Platform
JCRE	Java Card Runtime Environment
MAC	Message Authentication Code
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
STD	Software Test Description
TOE	Target of Evaluation.
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSP	TOE Security Policy
VM	Virtual Machine
VOP	Visa Open Platform

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 Manager	Main entity which represents the issuer and supervises the whole services available on the card
Security Domain	This entity represents a supplier. It manages the keys and provides cryptographic services for its applets.

CHAPTER 1 – SECURITY TARGET INTRODUCTION

ST Identification

Title:	LEO Security Target
Name:	JPH33V2 card
OCS registration:	FQR 110 963
Version:	2, issue 16/02/01
Component:	P8WE5033 (Philips)
Mask version:	V1

ST Overview

This Security Target covers the development, and the active phases of the JPH33V2 card, which is able to receive and manage different types of applications (debit/credit, wallet, fidelity, Pay-TV). This card is consistent with the Java Card 2.1.1 specifications, as well as the Visa Open Platform 2.0.1 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 an effective set of IT security countermeasures within the security environment, and that the TOE summary specification addresses the requirements.

CC Conformance

This ST is in accordance with the Common Criteria (see related documents [2] and [3]), Part 2 conformant and Part 3 augmented.

The assurance level is EAL1 augmented by AVA_VLA.2.

CHAPTER 2 - TOE DESCRIPTION

This part of the Security Target describes the TOE as an aid to the understanding of its security requirements. It addresses the product type, the intended usage and the main features of the TOE. This part includes :

- TOE overview,
- TOE life-cycle,
- TOE environment,
- Limits of the TOE.

TOE Overview

The TOE to be considered in this ST consists in a VOP Platform called JPH33V2 hosted on smart card IC.

It is a platform based on the Java Card 2.1.1 specifications, on the Open Platform 2.0.1 Card specification, and on the Visa Open Platform Card Implementation Specification.

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

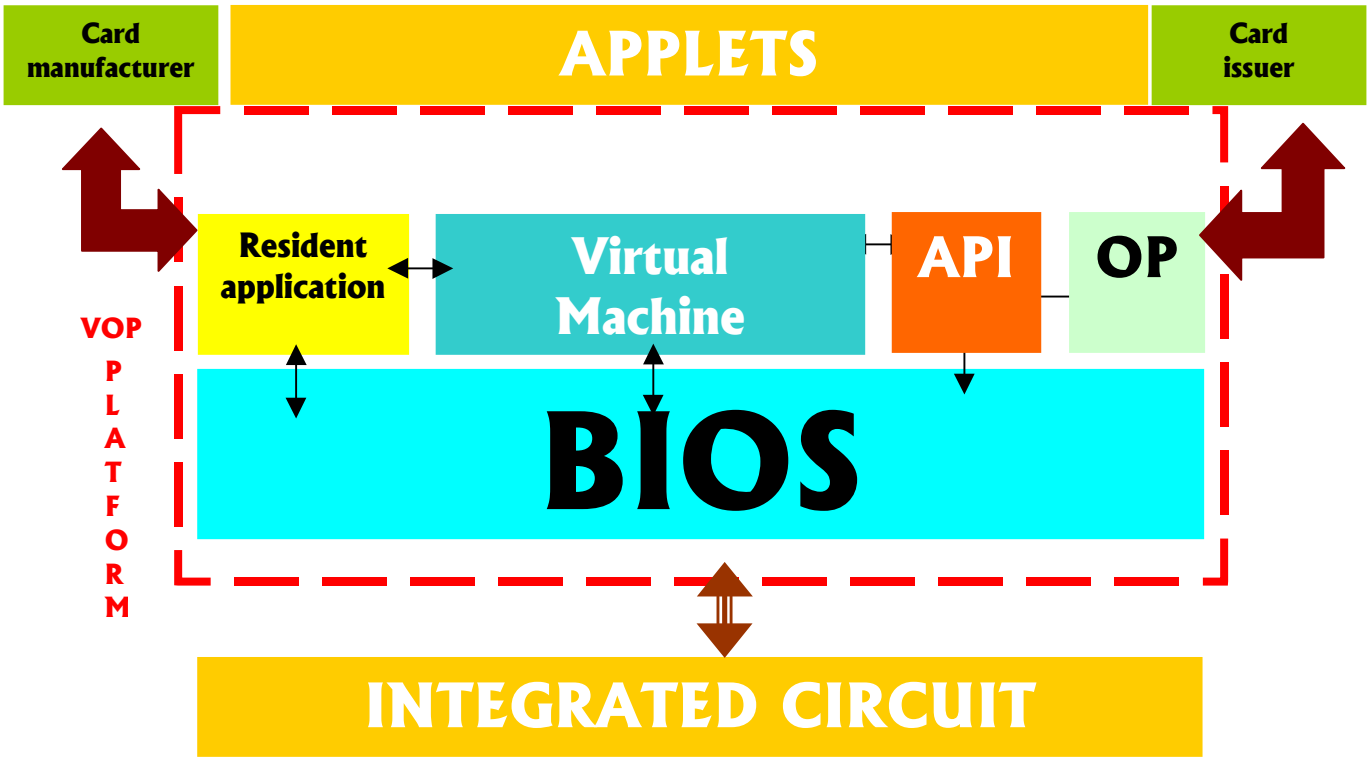

Figure 1 - Java Platform Architecture

The TOE, called “VOP Platform”, includes the BIOS, the Virtual Machine, the APIs, an Open Platform application and the Resident application. Details of components are presented below:

BIOS

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

The BIOS implements the following functionalities:

- APDU management ("T=0", "T=1" protocols),
- Timer management,
- Exceptions management,
- Transaction management,
- EEPROM access,
- Cryptographic modules. The JPH33V2 smart card contains an RSA keys generator 2048 bits; it also implements cryptographic algorithms: DES, RSA and SHA-1 algorithms.

Virtual Machine

The Virtual Machine, which is compliant with the JAVACARD 2.1.1 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 the execution of applets stored in ROM.

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

APIs

The APIs, compliant with the JAVACARD 2.1.1 standard, support key generation, signature and ciphering of messages, and a proprietary API OCSystem.

Open Platform

The "Open Platform OP2.0 configuration1a" application, which consists of the Card Manager, the API OPsystem, is implemented in Java and its byte-code is stored in ROM.

The Open Platform application is activated upon the select of the Card Manager, by the Card Issuer, the API Opsystem can be called at any time by the applets.

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.

TOE Life Cycle

The VOP 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.

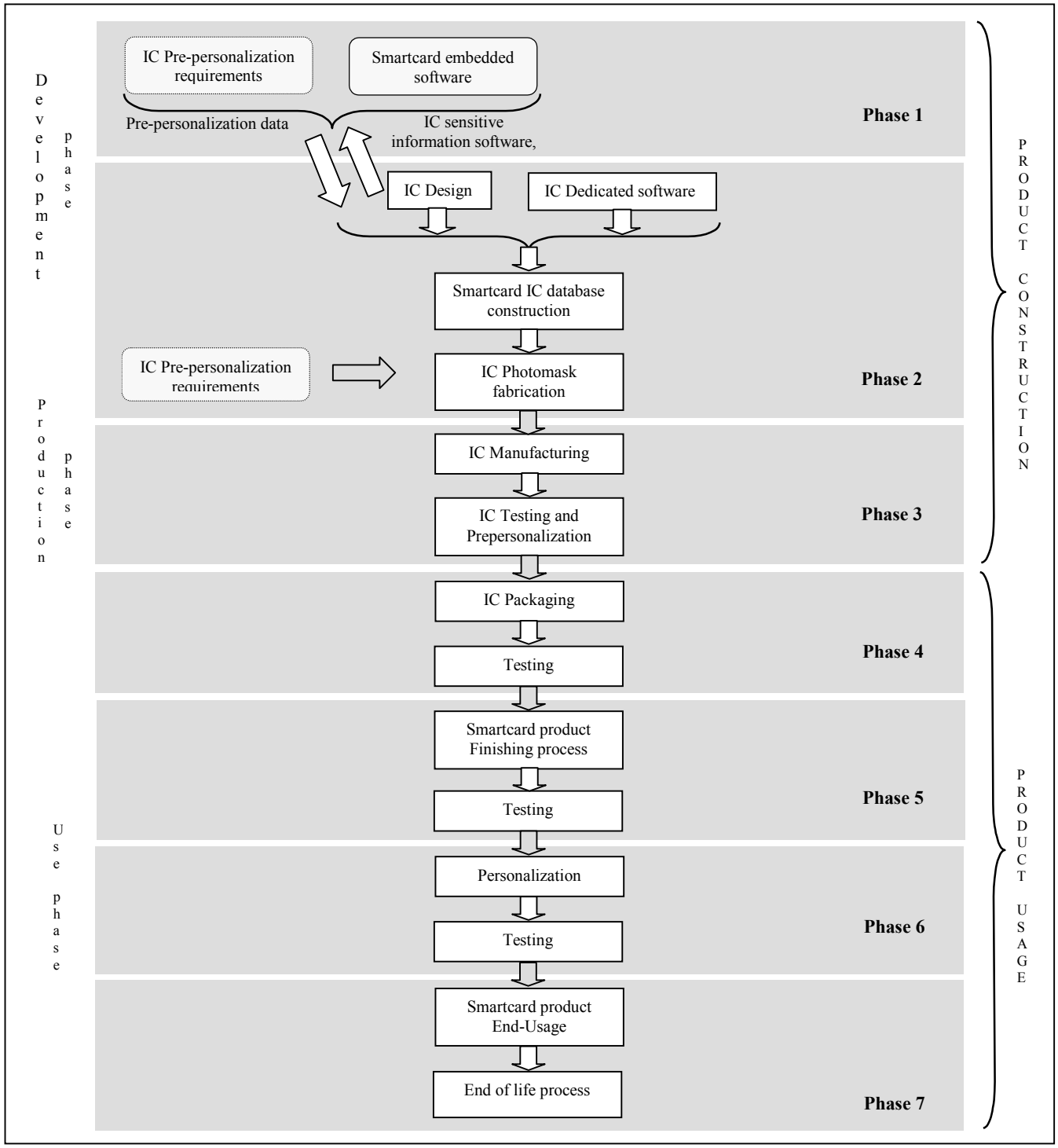

Figure 2 - Smart Card Product life-cycle

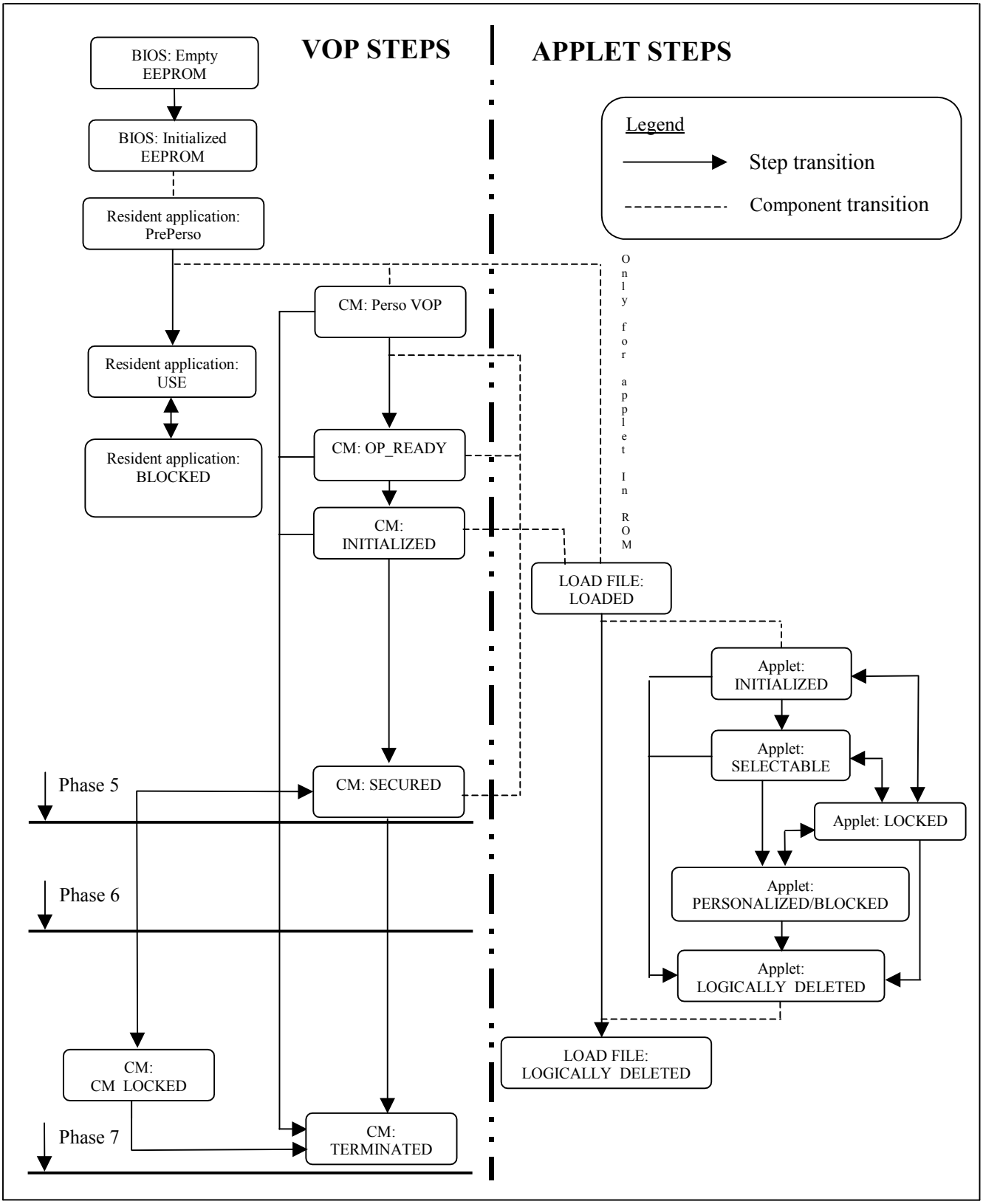

Figure 3 - VOP platform life-cycles

The Open Platform specified by Visa International defines life cycle state models to control the functionality and security of the following components: Card Manager, Global PIN, Card Registry, Key sets, Load Files, and Applets. These life cycle models are presented in this section.

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 an Open Platform 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.

VOP PERSONALIZATION: This life state is the initial life state of the Card Manager applet, just after it has been installed.In this life state, initialization key and card and chip CPLC have to be loaded before switching to OP_READY life state.

OP_READY:In the card life cycle state OP_READY, all the basic functionality of the run-time environment is available and the Card Manager is ready to receive, execute and respond to APDU commands. The card is assumed to have the following functionality in the state OP_READY:

- The run-time environment is ready for execution.
- An Initialization key is available within the Card Manager.

INITIALIZED: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 that state.

SECURED: The Card life cycle state SECURED is the normal operating life cycle state of the card during issuance. This state is the indicator for 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.

CM_LOCKED: The state CM_LOCKED is used to tell the Card Manager to temporarily disable all applets on the card except for the Card Manager. This state is created to provide the Card Issuer with the ability to detect security threats either internal or external to the card and be able to temporarily disable functionality of the card.

Setting the Card Manager to this state means 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 functionality 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 expiration of the card.

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

BIOS Life Cycle

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

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

Resident Application Life Cycle

PP: Prepersonalization state, the set of commands of the resident application (EXTERNAL_AUTHENTICATE, GET_CHALLENGE, GET_DATA, INSTALL, LOAD_APPLET, LOAD_STRUCTURE, 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.

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
- loader

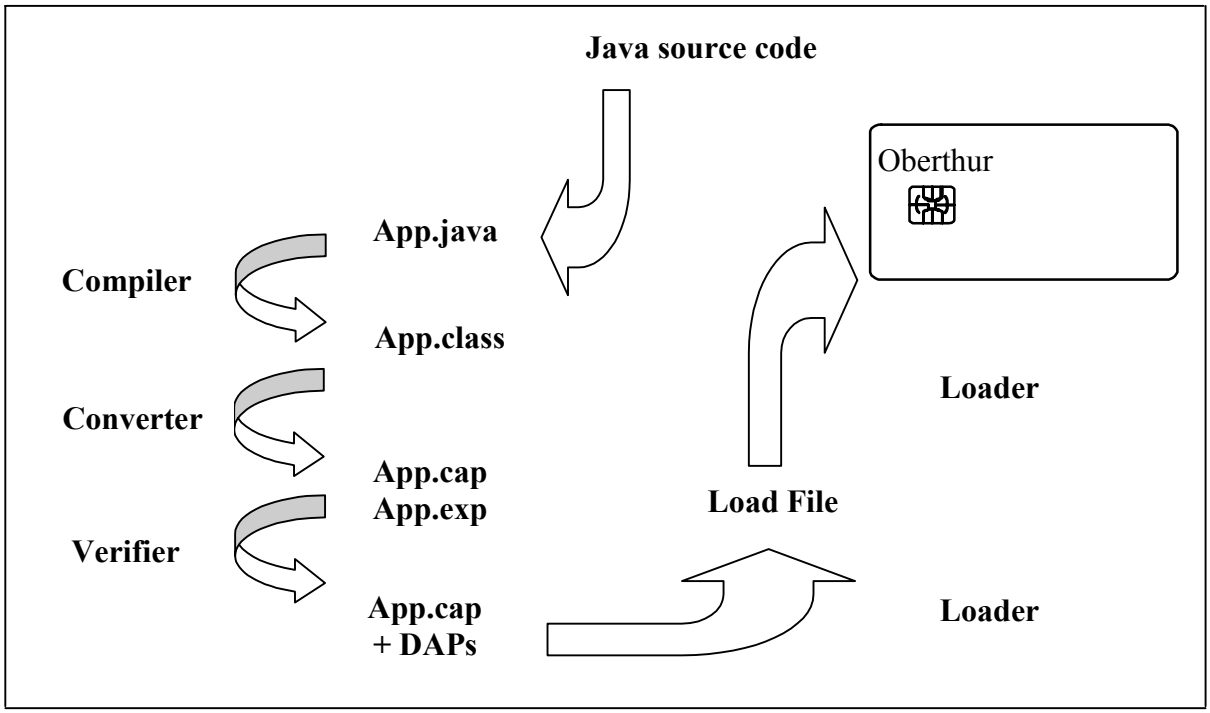

Figure 4 - Applet development process

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.

LOGICALLY_DELETED: If the Card Manager receives a request to delete a Load File which can not be physically deleted (for example because it is stored in immutable persistent memory), the Load File is logically deleted by setting its state to LOGICALLY_DELETED.

Once a Load File has been set to the state LOGICALLY_DELETED, it cannot be reversed. The Card Manager considers the state LOGICALLY_DELETED to be equivalent to the physical deletion of the Load File.

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.

The Card Manager is responsible for managing the initial life cycle state transition of an applet before it is fully functional. Once an applet is available for selection from the outside world, it takes control of managing its own life cycle. The life cycle states under the management of the applet are used as a means to inform the Card Manager as to the status of the applet. This state information must be provided to the Card Manager because the definition of these states are applet dependent and are only known to the applet. The Card Manager can then take control of the applet life cycle again later in the life of an applet if a security problem is detected by the card or the Card Issuer related to the applet, or if the applet is to be deleted either physically or logically.

The Card Manager sets the life cycle of an applet to its initial state of INSTALLED during the applet installation process. The Card Manager is also responsible for making the applet available for selection by setting its life cycle to SELECTABLE. The applet manages its life cycle transitioning from SELECTABLE to PERSONALIZED and optionally to BLOCKED.

At any point in the applet life cycle, the Card Manager can take control again for security protection by setting the applet life cycle state to LOCKED. Also, if the applet is to be removed from the card, the Card Manager manages that process and sets the appropriate resultant life cycle state if any.

INSTALLED: The state INSTALLED in the context of Open Platform means the applet executable has been properly linked and any necessary memory allocation has taken place so that the applet can execute. The Install process specifically does not include establishing the applet as an externally visible applet (SELECTABLE). Moreover, the Install process is not intended to incorporate personalization of the applet which may also occur as a separate step.

SELECTABLE: The state SELECTABLE is used to make an applet available to receive APDUs from outside the card. Applets must be properly installed and functional before they can be set to the life cycle state SELECTABLE.

PERSONALIZED: The definition of what is required for an applet to move to this state is applet dependent but is intended to indicate that the applet has been set up with all necessary personalization data and keys for full runtime functionality.

The behavior of the applet while in this state is determined by the applet itself. The Card Manager is not involved.

BLOCKED: The definition of what is required for an applet to move to this state is applet dependent but is intended to indicate that an applet specific security problem has been detected either from within the applet or from outside the card.

The behavior of the applet while in this state is determined by the applet itself. The Card Manager is not involved.

LOCKED: The life cycle state LOCKED is used as a security management control by the Card Manager or Issuer to prevent the selection, and therefore the execution of an applet.

If the Card Manager detects a threat from within the card and determines that the threat is associated with a particular applet, that applet can be prevented from further selection by setting its state to LOCKED.

Alternatively, the Card Issuer may determine that a particular applet on the card needs to be locked for a business or security reason and can initiate the applet life cycle transition via the Card Manager.

Once an applet is set to LOCKED, it can only be available for selection by the Card Manager setting its life cycle back to the state which it had achieved just prior to being set to the state LOCKED.

LOGICALLY_DELETED: If the Card Manager receives a request either directly or indirectly from a Security Domain to delete an applet which can not be physically deleted (for example because it is stored in immutable persistent memory), the applet is logically deleted by setting its state to LOGICALLY_DELETED.

Once an applet has been set to the state LOGICALLY_DELETED, it cannot be reversed. The Card Manager considers the state LOGICALLY_DELETED to be equivalent to the physical deletion of the applet.

TOE Environment

The TOE environment is defined as follows:

- TOE development environment corresponding to phase 1,
- TOE IT environment, corresponding to the IC,
- Prepersonalization process environment corresponding to phase 5,
- Personalization environment corresponding to phase 6, personalization and testing of the Smart Card with the user data,
- End-user environment corresponding to phase 7.

The loading process is included in phases 5, 6 and 7.

TOE Limits

The scope of this present security target is:

- TOE development phase realized by OBERTHUR in phase 1,
- and TOE active phases included in phases 5, 6 and 7.

As the IC is outside the scope of the TOE, the industrial process, corresponding to phases 2 to 4, is not concerned by the evaluation.

The personalization process is also outside of the TOE.

CHAPTER 3 – TOE SECURITY ENVIRONMENT

This section describes the security aspects of the environment in which the TOE is intended to be used, and addresses the description of the assets to be protected, the secure usage assumptions, the threats and the organizational security policies.

Users and Subjects

Users

Users concerned by the TOE are:

User	Description
U.Card_manufacturer	Card manufacturer. Role: R.Prepersonalizer, R.Personalizer
U.Card_Issuer	Administrator of the TOE during its life cycle. Role: R.Personalizer, R.Sign_Load_File, R.Card_Manager
U.Applet	Applet executed on this VOP platform implemented in Java. Role: R.Use_API, R.Applet_privilege, R.Personalizer

Subjects

Subject	Description
S.Resident application	Resident application. Users: U.Card_manufacturer, U.Card_Issuer
S.Applet	Applet executed on this VOP platform implemented in Java Users: U.Applet
S.CM	This process that implements the VOP specification is activated by the dispatcher. Users: U.Card_manufacturer or U.Card_Issuer

Identified Roles

According to the card life cycle, the users concerned by the TOE may have different roles:

Role	Description
R.Prepersonalizer:	Loading of additional data or code, delivery of the card with Card Manager state in OP_READY
R.Personalizer:	Generation and loading of the Card Manager keys (Card Manufacturer or Card Issuer)
R.Sign_Load_File:	Signature of the Load File
R.Card_Manager:	Manage the secure loading, installing and deleting of applet on-card, the loading of privileges, and manage global card data including Card Manager and applet life cycle state
R.Use_API:	Use APIs available on the platform
R.Applet_privilege:	Modification of CM life cycle, modification of ATR, modification of Global Pin

Assets to be Protected

TOE sensitive data and code have to be protected, some in terms of confidentiality, others in terms of integrity and some others both.

These assets are detailled below.

User Data

Data	Description
D.BYTECOD	Applet Byte code
D.JAVAOBJ	Java objects: • D.ARRAY Array
D.LOADFILE	Load File
D.APPLIFECYC	Applet life cycle state
D.PIN	Pin: • D.GLPIN Global Pin • D.OWNPIN Owner Pin (for applets)
D.KEY	Cryptographic Keys owned by the applets or the Card Manager, used by the DES

TSF Data

Authentication data

Data	Description
D.NBAUTHENTIC	Number of authentications
D.NB_REMAINTRYOWN	Number of remaining tries for owner Pin
D.NB_REMAINTRYGLB	Number of remaining tries for global Pin
D.CRYPTOGRAM	Cryptogram are used as an input for the authentication, resulting of key and random computation

Security attributes (or group of security attributes)

Attribute	Description
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 }
AS.CURCONTEXT	Current context
AS.AUTH_MSK_STATUS	Authentication MSK Status
AS.AUTH_CM_STATUS	Authentication CM Status
AS.CMLIFECYC	Card life cycle state
AS.CMCONTEXT	Card Manager's context
AS.EEPROM_FLAG	EEPROM integrity flag
AS.KEYSET_VERSION	Keyset Version
AS.KEYSET_VALUE	Keyset value
AS.SESSION_KEY	Session key
AS.LOGIC_CHANNEL_NB	Logical channel number (1-4)
AS.MAC	Chained MAC of commands of a secure channel
AS.SECUR_CHANNEL_NUM	Secure channel number
AS.MSKEY	Transport Key (Manufacturer Secret Key)
AS.SECURITY_LEVEL	Security levels of secure channel (Confidentiality, Integrity, or both)

Development Data

Assets to be protected during the development phase are:

- the VOP Platform specifications,
- the VOP Platform implementation,
- the VOP Platform related documentation.

Assumptions

Security is always the matter of the whole system: the weakest element of the chain determines the total system security. Secure usage assumptions described here after have to be considered for a secure system using Smart Card products.

Assumptions on TOE Development

Assumption	Description
A.DEV_TOE	It is assumed that security procedures and secure tools are used during: TOE specification, development, test operations, final validation, masking to maintain confidentiality and integrity of the TOE (to prevent any possible copy, modification, retention, theft or unauthorized use). It is also assumed that people working on/or with the TOE have got the required skill to assure in the working the integrity and the confidentiality of the TOE.

Assumptions on Tools

Assumption	Description
A.APPLET_TOOLS	Applet tools and processes as defined by the Card-Issuer's policy are used to develop applets. More precisely, the development chain of the applets includes a converter and a verifier (All phases) The converter generates verifiable Java Card bytecode, in a well-formed CAP file. The CAP file encapsulates the information contained in Java class files that comprise exactly one Java package. The converter checks the limits imposed by the JC21 specification on the number of classes, methods and fields. It generates well-formed export files. The conversion process preserves the code semantics of the applet's Java code (All phases). The verifier ensures that the CAP file has the correct format. The bytecodes are verified using a simple theorem prover which establishes a set of “structure constraints” on the bytecodes.

Assumptions on TOE Delivery Process (phase 1)

Assumption	Description
A.DLV_AUDIT	Procedures shall ensure that corrective actions are taken in case of improper operation in the delivery process and storage. Procedures shall also prevent (if applicable) any non-conformance to the confidentiality convention and must have a corrective action system in case any non-conformance or misprocessed procedures are identified.
A.DLV_CONTROL	Procedures shall guarantee the control of the TOE delivery and storage process and conformance to its objectives as described in the following secure usage assumptions. This procedures for material/information under delivery and storage shall be ensured, following these objectives: • Non-disclosure of any security relevant information • Identification of the elements under delivery • Meet confidentiality rules (confidentiality level, transmittal form, and reception acknowledgment) • Integrity of the TOE.
A.DLV_RESP	Procedures shall ensure that people dealing with the procedure for delivery have got the required skill, training and knowledge to meet the procedure requirements and to act to be fully in accordance with the above expectations.

Assumptions on Phases 2, 3, 4

Assumption	Description
A.USE_PROD	It is assumed that security procedures are used during IC development phase and IC production phase, IC packaging and IC tests operations through phases 2, 3, 4 to maintain confidentiality and integrity of the TOE (to prevent any possible copy, modification, retention, theft or unauthorized use).
A.KEY_MGT	The imported cryptographic key (MSK) is assumed to be generated, maintained and used off-card in a secure manner. This is a particular requirement on the off-card systems, and includes the provision of suitable physical, personnel and procedural measures as well as technical measures.

Assumptions on Phase 7

Assumption	Description
A.SECURE_LOAD	It is assumed that secure communication protocols and procedures are used between the Manufacturer and the applet developer or the Card Issuer.

Threats

This section describes all threats to the assets against which specific protection within the TOE or its environment is required. All possible threats that may be encountered are listed. The attackers involved in these threats are described below.

Attackers

Some attackers of the TOE are acting on behalf of a user, using hardware or software methods. These software can be located for example on the terminal using the platform (outside of the TOE).

For this evaluation (EAL1), these attackers are considered to have a low-level potential attack.

The attackers may also use pieces of software (for example an applet), loaded and residing in the Smart Card memory (outside of the TOE too), and trying to access sensitive data.

These attacks can only be realized by applet developers, with a high level of knowledge, to implement an applet that is correct according to the Java Card verifier, and that bypasses the firewall.

TOE Active Phases

Threat	Description
T1.MOD_CODE	Modification of OS code on the card.
T2.MOD_INITKEY	Modification and disclosure of initialization key to be loaded.
T3.DISCLOS_MSKEY	Disclosure of transport key.
T4.MOD_PIN_KEYS	Modification or disclosure of Global Pin code, Owner Pin code, Keysets and keys.
T5.LOAD_APP	Unauthorized loading and installation of applets or Load File. Load File stored in ROM made available in an unauthorized way.
T6.DISCLOS_CODE	Applet code disclosure during the loading.
T7.EXEC_EXTCODE	Unauthorized execution of byte code. For instance Java Standard keywords such as public, protected or package visible gives this authorization.
T8.MOD_BYTE	Unauthorized modification of byte code during the loading (applet, Load File).
T9.MOD_LIFECYCLE	Unauthorized modification of CM, Load file, resident application or applet life cycle.
T11.DEL_APP	Unauthorized deletion of Load File or applet.
T13.SELECT_APP	Unauthorized selection of an applet: some applet life cycle states forbids the selection of the applet, some Card Manager life cycle states forbids the selection of all applets.
T15.MOD_AID	Unauthorized modification of AID of the CM or of an applet.
T16.PERSO_APP	Unauthorized personalization of an applet using the service “ProviderSecurityDomain” of the Card Manager (modification of keys, Java objects, pins, applet life cycle).
T18.MOD_PRIV	Modification of privileges.
T21.USE_IDENT	Identity usurpation by an applet, in order to access a shareable Java object.
T23.ACCES_DATA	Unauthorized accesses in reading, writing to the data of one applet (Pin codes, keys, data tables, and objects) by another entity such as applet or human user.
T24.DISCLOS_KEY	Disclosure of key generated in the card.
T25.PERSO_RESID	Unauthorized Prepersonalization of resident application (modification of EEPROM contents).
T29.MOD_DATAVM	Unauthorized modification of the VM data and code.
T30.MOD_ATR	Unauthorized modification of ATR files (stored in EEPROM).
T31.INT_ROM	Unauthorized modification of EEPROM.

Organizational Security Policies

The TOE must comply with the following organizational policy statements:

Threat	Description
P.JC_FRAMEWORK	The TOE must support the core APIs of the Javacard specifications.
P.SERVICES	The VOP platform shall provide services (cryptographic or others) to allow applets to implement security mechanisms.

CHAPTER 4 –SECURITY OBJECTIVES

The security objectives mainly cover the following aspects:

- integrity and confidentiality of assets,
- protection of the TOE during its active life, that means with active security functions,
- protection of the TOE development environment and delivery process.

TOE Security Objectives

Objective	Description
O.DETECT_MEM	The platform shall detect loss of global EEPROM integrity, and shall ensure the consistency of all TSF data.
O.INT_ROM	The platform shall ensure the integrity of all code stored in ROM memory.

Card Manager

Objective	Description
O.AUTHE_PERS	The personalizer must be authenticated prior to execute the commands for installing the Card Manager. Loading and installation of applets including those stored in ROM need a successful authentication during prepersonalization phase.
O.INITKEY	The initialization Key must be ciphered and signed with the transport key.
O.CRYPT_DATA	Global PIN and KEYSET are loaded ciphered and signed. Loading, deletion of Global PIN or KEYSET are done after successful authentication in all phases
O.AUTHE_LOAD	Loading, installation, erasing of an applet or load file are done through the Card Manager after successful authentication in all phases.
O.SIGN_COD	All applets loaded must have been signed. Otherwise cryptographic algorithms are not available or restricted.
O.PROTEC_COD	During the loading of byte code on the card provided by an application provider and signed by the card issuer, the TOE ensures protection of it in terms of confidentiality and integrity.
O.AUTHE_CMS	The VOP platform can be used by a Card Management System while controlling that following management operations can only be done by the Card Issuer: The evolution of life cycle can be achieved only after a successful authentication. The Card Issuer can request the VOP platform in order to know the applets and the load files present on the card and their life cycle states.
O.MGT_CYC	Applets and Card Manager life cycle states must be always valid, and those states condition the execution of these applets. Some life cycle evolutions are forbidden.
O.AUTHE_AID	The modification of applet AID or Card Manager AID can only be realized by the Card Manager, and after a successful authentication of the Card Issuer.

Applet Management

Objective	Description
O.PERSO_APP	Applet personalization can only be done by the applet itself : • using its resources or • with delegation to the Card Manager.
O.AUTHE_PRIV	Some applets privileges can be modified by the Card Issuer (across the Card Manager) after a successful authentication.
O.AUTHE_CMS	The evolution of life cycle can be achieved only after a successful authentication.
O.CONF_SENSDATA	The TOE ensures confidentiality of sensitive data: PIN, keys.
O.FIREWALL	There is a firewall between applets. Furthermore, the applets can't modify the data and code of the TOE.
O.CRYPT_APP	The TOE provides a set of security features by using Application Programming Interface for: • cryptographic operations (DES, RSA); those operations must ensure integrity checking and confidentiality of keys • generation of secure RSA key • true random generation • access control Pin management (creation, update, verification, deletion) • protection against power loss and tearing through transaction mechanism

Resident Application

Objective	Description
O.AUTHE_PERS	The personalizer must be authenticated prior to execute the commands for installing the Card Manager. Loading and installation of applets including those stored in ROM need a successful authentication during prepersonalization phase.
O.AUTHE_CYCAR	The modification of Resident Application life cycle is submitted to a successful authentication.

BIOS

Objective	Description
O.AUTHE_ATR	The modification of ATR files needs an authorization (authentication of the card manufacturer or Applet privileges).

Security Objectives for the Environment

These objectives are the TOE environment objectives.

Security Objectives for the TOE Development Environment

Objective	Description
O.AUTHO_PEOPLE	Specifications, software, detailed design, schematics/layout or any further design information shall be accessible only by authorized personnel (physical, personnel, organizational and technical procedures).
O.TOE_DESIGN	The TOE shall be designed in a secure manner, that is focusing on integrity of programs and data, shall use all security features and perform security mechanisms as required by the TOE designer (e.g. cryptography); and it is assumed that appropriate functionality testing of the TOE is used in phase1.
O.DEV_TOOLS	The TOE shall be designed in a secure manner, by using exclusively software development tools (compilers assemblers, linkers, simulators, verifier) and software-hardware integration testing tools (emulators) that will result in the integrity of program and data.
O.SOFT_DLV_IC	The software under development must be delivered through a trusted delivery and verification procedure that shall guarantee the integrity and confidentiality.
O.SOFT_DLV_TR	The software under development must be delivered to the correct party through a trusted delivery and verification procedure that shall ensure a full tracability.

Security Objectives on the TOE Environment

Objective	Description
O.MSKEY_MGT	The transport key must be stored in a secured area. It must be exchanged between the developer of the platform and the IC manufacturer in a secured manner to respect key integrity and confidentiality.
O.DEV_APPLET	The applets shall be designed in a secured manner to respect key integrity and confidentiality.
O.VERIF_COD	All applets must have been verified by a verifier before signature.
O.CODE_MGT	The applet code, data and keys must be transmitted in a secured manner to ensure the confidentiality and integrity. The actors involved are the personalizer of applet, the manufacturer, the Card Issuer and the applet developer.
O.TOE_MGT	The manufacturer must guarantee the confidentiality and integrity of the TOE.

Security Objectives on the TOE IT Environment

Objective	Description
O.CRYPT_IC	The IC provides a set of security features: cryptographic operations (DES, RSA); those operations must ensure integrity checking and confidentiality of keys • true random generation • erasing (deallocation) of cryptographic buffers.
O.IC_PROT	The IC protects against manipulation of the hardware as well as software and data stored in the RAM, in the ROM, and in the EEPROM of the chip.

CHAPTER 5 – IT SECURITY REQUIREMENTS

This section defines the detailed IT security requirements that shall be satisfied by the TOE or its environment. IT security requirements address only the security objectives for the TOE and its IT environment.

TOE Security Requirements

TOE Security Functional Requirements

The TOE IT functional requirements define the functional requirements for the TOE using only functional requirements components drawn from the CC part 2.

Class FAU: Security Audit

Security Audit Automatic Response (FAU_ARP)

FAU_ARP.1 Security Alarms

Functional requirement	Description
FAU_ARP.1.1	The TSF shall take **an action among the following list** upon detection of a potential security violation: **1. Make the card mute** **2. Block the action that produced the security violation, and throw an exception**

Dependencies: FAU_SAA.1 Potential violation analysis

Security Audit Analysis (FAU_SAA)

FAU_SAA.1: Potential Violation Analysis

Functional requirement	Description
FAU_SAA.1.1/SOFT	The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.
FAU_SAA.1.2/SOFT	The TSF shall enforce the following rules for monitoring audited events: A. Accumulation or combination of **the following auditable events** known to indicate a potential security violation: **1. Card Manager lifecycle inconsistency audited through the self-test mechanism and through the lifecycle checks in all administration operations (TERMINATED),** **2. Unauthorized object access outside of the active context audited through the firewall mechanism: automatic throw of a security exception,** **3. Invalid access to a reference audited through the object access mechanism: automatic throw of an invalid reference exception,** **4. Inconsistency of the EEPROM integrity flag: make the card mute;** B. **Any other rules: none**

Dependencies: FAU_GEN.1 Audit data generation

Class FCO: Communication

Non-Repudiation of Origin (FCO_NRO)

FCO_NRO.1. Selective Proof of Origin

Functional requirement	Description
FCO_NRO.1.1/SHARINT **(1)**	The TSF shall be able to generate evidence of origin for transmitted **method invocation** at the request of the **recipient**.
FCO_NRO.1.2/SHARINT	The TSF shall be able to relate the **context** of the originator of the information, and the **parameters** of the information to which the evidence applies.
FCO_NRO.1.3/SHARINT	The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **duration of invocation**.
FCO_NRO.1.1/CMOPINI **(2)**	The TSF shall be able to generate evidence of origin for transmitted **D.LOADFILE**, **ASG.APPPRIV** at the request of the **originator**
FCO_NRO.1.2/CMOPINI	The TSF shall be able to relate the **AS.KEYSET_VALUE** of the originator of the information, and the **APDU command** of the information to which the evidence applies.
FCO_NRO.1.3/CMOPINI	The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **duration of secure channel**.

(1): ***Management of shareable interface.***

(2): ***When the Card Manager life cycle phase is OP_READY or INITIALIZED.***

Dependencies: FIA_UID.1 Timing of identification

FCO_NRO.2. Enforced Proof of Origin

Functional requirement	Description
FCO_NRO.2.1/DAP **(1)**	The TSF shall enforce the generation of evidence of origin for transmitted **CAP file** at all times.
FCO_NRO.2.2/DAP	The TSF shall be able to relate the **AS.KEYSET_VALUE** of the originator of the information, and the **CAP file components** of the information to which the evidence applies.
FCO_NRO.2.3/DAP	The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **indefinite**.
FCO_NRO.2.1/CMSECURE **(2)**	The TSF shall enforce the generation of evidence of origin for transmitted **D.LOADFILE, AS.KEYSET_VALUE, ASG.APPPRIV, D.GLPIN** at all times.
FCO_NRO.2.2/CMSECURE	The TSF shall be able to relate the **AS.KEYSET_VALUE** of the originator of the information, and the **APDU command** of the information to which the evidence applies.
FCO_NRO.2.3/CMSECURE	The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **duration of secure channel**.

(1): DAP verification

(2): When the Card Manager life cycle phase is SECURED (for Load Files and privileges).

Dependencies: FIA_UID.1 Timing of identification

Class FCS: Cryptographic Support

Cryptographic Key Management (FCS_CKM)

FCS_CKM.1: Cryptographic Key Generation

Functional requirement	Description
FCS_CKM.1.1 / RSA	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **(RSA)** and specified cryptographic key sizes **of 512, 768, 1024, 2048 bits** that meet the following standards: **1. Ansi X9.31 (see related document [13]).**

Dependencies: FCS_COP.1 Cryptographic operation

FCS_CKM.4 Cryptographic key destruction

FMT_MSA.2 Secure security attributes

FCS_CKM.3: Cryptographic Key Access

Functional requirement	Description
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 table below)** that meets the following standards: **1. See related document [9], chapter 8 and 9.9;** **2. See related document [10], chapter 9.3;** **3. See related document [6], packages "Javacard.security" and "Javacard.crypto".**

Cryptographic key access type	Cryptographic key access methods/commands
DES	**Commands:** • **PUT_KEY** • **EXTERNAL AUTHENTICATE** • **INITIALIZE UPDATE** **"ProviderSecurityDomain" key acces methods:** • **decryptVerifyKey** • **openSecureChannel** • **unwrap** • **verifyExternalAuthenticate** **"APICrypto" key access methods:** • **Key.clearKey** • **DES.getKey** • **DES.setKey** • **Signature.update** • **Signature.sign** • **Signature.verify** • **Cipher.update** • **Cipher.doFinal**
RSA	**"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.update** • **Signature.sign** • **Signature.verify** • **Cipher.update** • **Cipher.doFinal**

Dependencies: FDP_ITC.1 Import of user data without security attributes

FCS_CKM.4 Cryptographic key destruction

FMT_MSA.2 Secure security attributes

FCS_CKM.4: Cryptographic Key Destruction

Functional requirement	Description
FCS_CKM.4.1	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method (**that also prevent the destroyed keys from being referenced**) that meets the following standards: **1. See related document [10], chapter 6.4.2: Key Renewal and Replacement.**

Dependencies: FDP_ITC.1 Import of user data without security attributes

FMT_MSA.2 Secure security attributes

Cryptographic Operation (FCS_COP)

FCS_COP.1: Cryptographic Operation

Functional requirement	Description
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 **56 bits (DES) and 112 bits or 168 bits (triple-DES)**, that meet the following standards: **1. FIPS PUB 46-3 (ANSI X3.92) (see related document [14])** **2. FIPS PUB 81 (see related document [15])** **3. ISO 8372 (1987), Information Processing Modes of Operation for a 64-Bit Block Cipher Algorithm (see related document [18])** **4. ISO 8732 (1988), Banking – Key Management (see related document [19])** 5. **ISO 9797 (1994), Data integrity mechanism (see related document [21])**
FCS_COP.1.1 / RSA	The TSF shall perform **signature, verification of signature, encryption and decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **of 512, 768, 1024, 2048 bits** that meet the following standards: **1. Ansi X9.31 (see related document [13])** **2. ISO / IEC 9796-1, annex A, section A.4 and A.5, and annex C (see related document [20])** **3. PKCS#1 The public Key Cryptography standards, RSA Data Security Inc. 1993 (see related document [14])**
FCS_COP.1.1/RND	The TSF shall perform **Random Number Generation** in accordance with a specified cryptographic algorithm **(no algorithm)** and cryptographic key sizes **No Key** that meet the following standards: • **FIPS 184.2 (see related document [17]).** ***Note: Entropy of at least 7 bit in each byte.***

Dependencies: FDP_ITC.1 Import of user data without security attributes

FCS_CKM.4 Cryptographic key destruction

FMT_MSA.2 Secure security attributes

Class FDP: User Data Protection

Access Control Policy (FDP_ACC)

FDP_ACC.1 Subset Access Control

Functional requirement	Description
FDP_ACC.1.1/JCREPRIV	The TSF shall enforce the **JCREPRIV access control in writing - done by the API OCSystem -** on **the following list of subjects, objects, and operations**. Subject: **S.CM** Objects: **ATR files, Card Manager life Cycle, Applet life cycle, Applet privileges, Applet export rights, Transport Key** Operations: **setATR, lockCard, useCard, SMWithTransportKey, delete, setDefaultApplet, setStatus, setAID, setAIDRef, getAIDRef, setRights, getRights**
FDP_ACC.1.1/APPPRIV	The TSF shall enforce the **APPPRIV access control in writing - done by the API OPSystem** - on **the following list of subjects, objects, and operations.** Subject: **S.Applet** Objects: **ATR files, Card Manager life Cycle, Global Pin** Operations: **setATRHistoricalBytes, TerminateCard, CMLock, setContentState, setPin.**

Dependencies: FDP_ACF.1 Security attribute based access control

FDP_ACC.2: Complete Access Control

Functional requirement	Description
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 in the TSC and any object within the TSC are covered by an access control SFP.
FDP_ACC.2.1/FIREWALL	The TSF shall enforce the **FireWall access control** on **S.Applet and for all D.JAVAOBJ objects** and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/FIREWALL	The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.
FDP_ACC.2.1/CM	The TSF shall enforce the **CM access control** on **S.CM and for objects D.LOADFILE, AS.KEYSET_VALUE, D.GLPIN, ASG.APPPRIV, AS.CMLIFECYC, AS.KEYSET_VERSION, D.APPLIFECYC, ASG.CARDREG** and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/CM	The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

Dependencies: FDP_ACF.1 Security attribute based access control

Access Control Function (FDP_ACF)

FDP_ACF.1 Security Attribute Based Access Control

Functional requirement	Desription
FDP_ACF.1.1/JCREPRIV	The TSF shall enforce the **JCREPRIV access control in writing** to objects based on **AS.CMCONTEXT**.
FDP_ACF.1.2/JCREPRIV	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **Current Context = AS.CMCONTEXT**.
FDP_ACF.1.3/JCREPRIV	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **None**.
FDP_ACF.1.4/JCREPRIV	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **None**.
FDP_ACF.1.1/APPPRIV	The TSF shall enforce the **APPPRIV access control in writing** to objects based on **ASG.APPPRIV**.
FDP_ACF.1.2/APPPRIV	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **If current applet privileges allows this operation**.
FDP_ACF.1.3/APPPRIV	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **None**.
FDP_ACF.1.4/APPPRIV	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **None**.
FDP_ACF.1.1/PP	The TSF shall enforce the **Prepersonalization access control** to objects based on **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 authorize 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_ACF.1.1/FIREWALL	The TSF shall enforce the **FIREWALL access control** to objects based on **AS.CURCONTEXT**.
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: **see related document [7] (section 6) - ex.: Current Context = Object Context -**.
FDP_ACF.1.3/FIREWALL	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **if object context = JCRE context**.
FDP_ACF.1.4/FIREWALL	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **None.**
FDP_ACF.1.1/CM	The TSF shall enforce the **CM access control** to objects based on **AS.AUTH_CM_STATUS and AS.SECURITY_LEVEL**.
FDP_ACF.1.2/CM	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/CM	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: ***None***.
FDP_ACF.1.4/CM	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: ***None***.

Dependencies: FDP_ACC.1 Subset access control.

FMT_MSA.3 Static attribute initialization.

Export to Outside TSF Control (FDP_ETC)

FDP_ETC.1: Export of User Data Without Security Attributes

Functional requirement	Description
FDP_ETC.1.1/AUT	The TSF shall enforce the **CM access control** when exporting user data, controlled under the SFP(s), outside of the TSC.
FDP_ETC.1.2/AUT	The TSF shall export the user data without the user data's associated security attributes.
FDP_ETC.1.1/FIREWALL	The TSF shall enforce the **Firewall access control** when exporting user data, controlled under the SFP(s), outside of the TSC.
FDP_ETC.1.2/FIREWALL	The TSF shall export the user data without the user data's associated security attributes.

Dependencies: FDP_ACC.1 Subset access control.

Import From Outside TSF Control (FDP_ITC)

FDP_ITC.1: Import of User Data Without Security Attributes

Functional requirement	Description
FDP_ITC.1.1/FIRE_1	The TSF shall enforce the **FireWall access control on DES KEY, RSA KEY, D.PIN Value import, Applet Data, and D.BYTECOD** when importing user data, controlled under the SFP, from outside of the TSC. **Note: for Byte Code: putfield, <t>astore.**
FDP_ITC.1.2/FIRE_1	The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.
FDP_ITC.1.3/FIRE_1	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: **None**.
FDP_ITC.1.1/CM	The TSF shall enforce the **CM access control** when importing user data, controlled under the SFP, from outside of the TSC.
FDP_ITC.1.2/CM	The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.
FDP_ITC.1.3/CM	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: **None**.
FDP_ITC.1.1/APPPRIV	The TSF shall enforce the **APPPRIV access control** when importing user data, controlled under the SFP, from outside of the TSC.
FDP_ITC.1.2/APPPRIV	The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.
FDP_ITC.1.3/APPPRIV	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: **None**.

Dependencies: FDP_ACC.1 Subset access control.

FMT_MSA.3 Static attribute initialization.

FDP_ITC.2: Import of User Data with Security Attributes

Functional requirement	Description
FDP_ITC.2.1/CM_CAPFILE	The TSF shall enforce the **CM access control** when importing user data, controlled under the SFP, from outside of the TSC.
FDP_ITC.2.2/CM_CAPFILE	The TSF shall use the security attributes associated with the imported user data.
FDP_ITC.2.3/CM_CAPFILE	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/CM_CAPFILE	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/CM_CAPFILE	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: **None**.

Note:	***For Import of CAP File.***
Dependencies:	FDP_ACC.1 Subset access control, FTP_TRP.1 Trusted path, FPT_TDC.1 Inter-TSF basic TSF data consistency.

Residual Information Protection (FDP_RIP)

FDP_RIP.1: Subset Residual Information Protection

Functional requirement	Description
FDP_RIP.1.1/DEALL_JAVAOBJ	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: • **All Java objects, and no access path to the transients objects.**
FDP_RIP.1.1/ALL_OBJTRANS	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: • **Transients objects.**
FDP_RIP.1.1/DEALL_GARB	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: • **Garbage collector.**
FDP_RIP.1.1/ALL_APDU	The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource to** the following objects: • **The resource concerned is the APDU buffer.**

Dependencies:	No dependencies.

Stored Data Integrity (FDP_SDI)

FDP_SDI.2: Stored Data Integrity Monitoring and Action

Functional requirement	Description
FDP_SDI.2.1/EEPROM	The TSF shall monitor user data stored within the TSC for **EEPROM integrity error** on all objects, based on the following attributes: **AS.EEPROM_FLAG**.
FDP_SDI.2.2/EEPROM	Upon detection of a data integrity error, the TSF shall **make the card mute**.

Dependencies: No dependencies.

Inter-TSF User Data Confidentiality Transfer Protection (FDP_UCT)

FDP_UCT.1 Basic Data Exchange Confidentiality

Functional requirement	Description
FDP_UCT.1.1/PP	The TSF shall enforce the **prepersonalization access control** to be able to **receive** objects in a manner protected from unauthorized disclosure.
FDP_UCT.1.1/CM	The TSF shall enforce the **CM access control** to be able to **receive** objects in a manner protected from unauthorized disclosure.

Dependencies: FTP_TRP.1 Trusted path

FDP_ACC.1 Subset access control

Inter-TSF User Data Integrity Transfer Protection (FDP_UIT)

FDP_UIT.1 Data Exchange Integrity

Functional requirement	Description
FDP_UIT.1.1/SECURE	The TSF shall enforce the **CM access control** to be able to **receive** user data in a manner protected from **modification** errors.
FDP_UIT.1.2/ SECURE **(1)**	The TSF shall be able to determine on receipt of user data, whether **modification** has occurred.
FDP_UIT.1.1/KEYCHECK	The TSF shall enforce the **CM access control** to be able to **receive** user data in a manner protected from **modification** errors.
FDP_UIT.1.2/KEYCHECK **(2)**	The TSF shall be able to determine on receipt of user data, whether **modification** has occurred.
FDP_UIT.1.1/DAP	The TSF shall enforce the **CM access control** to be able to **receive** user data in a manner protected from **modification** errors.
FDP_UIT.1.2/DAP **(3)**	The TSF shall be able to determine on receipt of user data, whether **modification** has occurred.

(1): ***If modification has occurred, it has been during secure channel transmission.***

(2): ***If modification has occurred, it has occurred between generation of the key and its reception.***

(3): ***If modification has occurred, it has occurred between verification of the CAP File and its reception***

Dependencies: FDP_ACC.1 Subset access control

FTP_TRP.1 Trusted path

Class FIA: Identification and Authentication

Authentication Failures (FIA_AFL)

FIA_AFL.1: Authentication Failure Handling

Functional requirement	Description
FIA_AFL.1.1/CM	The TSF shall detect when **1** unsuccessful authentication attempts occur related to **U.Card_Issuer authentication.**
FIA_AFL.1.2/CM	When the defined number of unsuccessful authentication attempts has been met or surpassed, 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_AFL.1.1/APP	The TSF shall detect when **user defined maximum (1 to 127)** unsuccessful authentication attempts occur related **to any user authentication using a PIN.**
FIA_AFL.1.2/APP	When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall **block the PIN**.

Dependencies: FIA_UAU.1 Timing of authentication.

User Attribute Definition (FIA_ATD)

FIA_ATD.1: User Attribute Definition

Functional requirement	Description
FIA_ATD.1.1/CARD_MANUF	The TSF shall maintain the following list of security attributes belonging to individual users:**AS.AUTH_MSK_STATUS**.
FIA_ATD.1.1/CARD_ISSUER	The TSF shall maintain the following list of security attributes belonging to individual users: **AS.CMLIFECYC, AS.CMCONTEXT, AS.KEYSET_VERSION, AS.KEYSET_VALUE**.
FIA_ATD.1.1/APPLET	The TSF shall maintain the following list of security attributes belonging to individual users: **ASG.APPPRIV, AS.CURCONTEXT**.

Dependencies: No dependencies.

Specification of Secrets (FIA_SOS)

FIA_SOS.1: Verification of Secrets

Functional requirement	Description
FIA_SOS.1.1/RSA	The TSF shall provide a mechanism to verify that secrets meet **RSA generation key metric (Miller-Rabin method**).

Dependencies: No dependencies.

FIA_SOS.2: TSF Generation of Secrets

Functional requirement	Description
FIA_SOS.2.1/RANDOM	The TSF shall provide a mechanism to generate secrets that meet **random metric (see related document [22])**.
FIA_SOS.2.2/RANDOM	The TSF shall be able to enforce the use of TSF generated secrets for: **Card Manager and Card Issuer authentication, Secure channel management.**

Dependencies: No dependencies.

User Authentication (FIA_UAU)

FIA_UAU.1: Timing of Authentication

Functional requirement	Description
FIA_UAU.1.1	The TSF shall allow **the TSF-mediated actions of the following list** on behalf of the user to be performed before the user is authenticated. **Resident application: Get Challenge, Get Data, Manage Channel, Select Applet.** **Card Manager: Get Data, Initialize Update.**
FIA_UAU.1.2	The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Dependencies: FIA_UID.1 Timing of identification

FIA_UAU.3: Unforgeable Authentication

	Description
FIA_UAU.3.1	The TSF shall **prevent** use of authentication data that has been forged by any user of the TSF.
FIA_UAU.3.2	The TSF shall **prevent** use of authentication data that has been copied from any other user of the TSF.

Dependencies: No dependencies.

FIA_UAU.4: Single-Use Authentication Mechanisms

Functional requirement	Description
FIA_UAU.4.1/CARD_MANUF	The TSF shall prevent reuse of authentication data related to the **Card Manufacturer authentication mechanism**.
FIA_UAU.4.1/CARD_ISSUER	The TSF shall prevent reuse of authentication data related to the **Card Issuer authentication mechanism**.

Dependencies: No dependencies.

FIA_UAU.7: Protected Authentication Feedback

Functional requirement	Description
FIA_UAU.7.1/CARD_MANUF	The TSF shall provide only **the result of the authentication (NOK) and the random** to the user while the authentication is in progress.
FIA_UAU.7.1/CARD_ISSUER	The TSF shall provide only **the result of the authentication (NOK), the keyset version, the starting key index, the card random and the card cryptogram** to the user while the authentication is in progress.

Dependencies: FIA_UAU.1 Timing of authentication.

User Identification (FIA_UID)

FIA_UID.1: Timing of Identification

Functional requirement	Description
FIA_UID.1.1	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	The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Note: **This execution is possible only if the CM is the default applet.**

Dependencies: No dependencies.

User-Subject Binding (FIA_USB)

FIA_USB.1: User-Subject Binding

Functional requirement	Description
FIA_USB.1.1	The TSF shall associate the appropriate user security attributes with subjects acting on behalf of that user.

Dependencies: FIA_ATD.1 User attribute definition.

Class FMT: Security Management

Management of Functions in TSF (FMT_MOF)

FMT_MOF.1 Management of Security Functions Behaviour

Functional requirement	Description
FMT_MOF.1.1/RES_APP	The TSF shall restrict the ability to **disable** the functions **of resident application: GET CHALLENGE, EXTERNAL AUTHENTICATE, LOAD STRUCTURE, INSTALL, LOAD APPLET, GET DATA** to **R.Prepersonalizer.**

Dependencies: FMT_SMR.1 Security roles.

Management of Security Attributes (FMT_MSA)

FMT_MSA.1: Management of Security Attributes

Functional requirement	Description
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 **the R.Prepersonalizer**.
FMT_MSA.1.1/CM_MOD **(1)**	The TSF shall enforce the **CM access control to** restrict the ability to **modify** the security attributes **AS.KEYSET_VERSION, AS.KEYSET_VALUE, Default selected privilege, AS.CMLIFECYC** to **the R.Card_Manager**.
FMT_MSA.1.1/CM_DEL	The TSF shall enforce the **CM access control** to restrict the ability to **delete** the security attributes **AS.KEYSET_VERSION**, **AS.KEYSET_VALUE** to **the R.Card_Manager**.
FMT_MSA.1.1/PRIV_MOD	The TSF shall enforce the **APPPRIV access control** to restrict the ability to **modify** the security attributes **AS.CMLIFECYC** to **the R.Applet_privilege**.

(1) ***Other privileges cannot be modified.***

Dependencies: FDP_ACC.1 Subset access control.
FMT_SMR.1 Security roles.

FMT_MSA.2: Secure Security Attributes

Functional requirement	Description
FMT_MSA.2.1	The TSF shall ensure that only secure values are accepted for security attributes.

Dependencies: ADV_SPM.1 Informal TOE security policy model

FDP_ACC.1 Subset access control

FMT_MSA.1 Management of security attributes

FMT_SMR.1 Security roles

FMT_MSA.3 Static Attribute Initialisation

Functional requirement	Description
FMT_MSA.3.1	The TSF shall enforce the **CM access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2	The TSF shall allow the **R.Personalizer** to specify alternative initial values to override the default values when an object or information is created.

Note: ***The personalizer can only specify initial values for keyset.***

Dependencies: FMT_MSA.1 Management of security attributes.

FMT_SMR.1 Security roles.

Management of TSF Data (FMT_MTD)

FMT_MTD.1: Management of TSF Data

Functional requirement	Description
FMT_MTD.1.1/CI	The TSF shall restrict the ability to **modify** the **AS.CMID**, **AS.APID** to **the R.Personalizer**.
FMT_MTD.1.1/ CARDREG	The TSF shall restrict the ability to **query** the **AS.APID** to **the R.Card_Manager and the R.Use_API.**

Dependencies: FMT_SMR.1 Security roles.

FMT_MTD.2: Management of Limits on TSF Data

Functional requirement	Description
FMT_MTD.2.1/ GLBPIN	The TSF shall restrict the specification of the limits for **D.NB_REMAINTRYGLB** to **the R.Card_Manager**.
FMT_MTD.2.2/ GLBPIN	The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: **block the D.GLPIN**.
FMT_MTD.2.1/ OWNPIN	The TSF shall restrict the specification of the limits for **D.NB_REMAINTRYOWN** to **R.Use_API**.
FMT_MTD.2.2/ OWNPIN	The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: **block the D.OWNPIN**.

Dependencies: FMT_MTD.1 Management of TSF data.

FMT_SMR.1 Security roles.

Security Management Roles (FMT_SMR)

FMT_SMR.1: Security Roles

Functional requirement	Description
FMT_SMR.1.1	The TSF shall maintain the roles **R.Sign_Load_File**.
FMT_SMR.1.2	The TSF shall be able to associate users with roles.

Dependencies: FIA_UID.1 Timing of identification

FMT_SMR.2: Restrictions on Security Roles

Hierarchical to: FMT_SMR.1

Functional requirement	Description	
FMT_SMR.2.1	The TSF shall maintain the roles **defined below**.	
FMT_SMR.2.2	The TSF shall be able to associate users with roles.	
FMT_SMR.2.3	The TSF shall ensure that the conditions **defined below** are satisfied.	
	Roles	Conditions
	R.Prepersonalizer	**Successful authentication (of Card Manufacturer) using Transport key and card still in prepersonalization state, except for loading of Initialization Key**
	R.Personalizer	**Successful authentication (of Card Manufacturer or Card Issuer) using a keyset of the Card Manager, with CM life cycle phase from OP_READY to SECURED**
	R.Card_Manager	**Successful authentication (of Card Issuer) using a keyset, with CM life cycle phase from OP_READY to SECURED**
	R.Use_API	**Successful identification (of Applet), with Applet life cycle phase after SELECTABLE**
	R.Applet_privilege	**Modification of CM life cycle, ATR, and also Global Pin, is only possible for privileged applet**

Dependencies: FIA_UID.1 Timing of identification.

Class FPR: Privacy

Unobservability (FPR_UNO)

FPR_UNO.1: Unobservability

Functional requirement	Description
FPR_UNO.1.1	The TSF shall ensure that **any users** are unable to observe the **following operations** on **the following objects** by **the following subjects:**

Subject	Operation	Object
S.Applet	**Comparison of PIN value**	**D.GLPIN, D.OWNPIN**
S.Applet, S.CM	**Import and use**	**D.KEY**
S.Applet	**Comparison of two byte arrays**	**D.ARRAY**

Dependencies: No dependencies.

Class FPT: Protection of the TOE Security Functions

Fail secure (FPT_FLS)

FPT_FLS.1: Failure With Preservation of Secure State

Functional requirement	Description
FPT_FLS.1.1	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**

Dependencies: ADV_SPM.1 Informal TOE security policy model

Trusted recovery (FPT_RCV)

FPT_RCV.4 Function Recovery

Functional requirement	Description
FPT_RCV.4.1	The TSF shall ensure that **anti-tearing failure scenarios** have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

Note: ***See related documentation [7] for details about the secure state of the environment of an applet.***

Dependencies: ADV_SPM.1 Informal TOE security policy model.

Reference Mediation (FPT_RVM)

FPT_RVM.1 Non-bypassability of the TSP

Functional requirement	Description
FPT_RVM.1.1	The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Dependencies: No dependencies.

Domain Separation (FPT_SEP)

FPT_SEP.1: TSF Domain Separation

Functional requirement	Description
FPT_SEP.1.1	The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.
FPT_SEP.1.2	The TSF shall enforce separation between the security domains of subjects in the TSC.

Note: ***There is a separation of Security Domain between Card Manager and applets (the CM is written with Java Language), and between transients of different logical channels.***

Dependencies: No dependencies.

Inter-TSF TSF Data Consistency (FPT_TDC)

FPT_TDC.1: Inter-TSF Basic TSF Data Consistency

Functional requirement	Description
FPT_TDC.1.1	The TSF shall provide the capability to consistently interpret **AS.KEYSET_VALUE**, when shared between the TSF and another trusted IT product.
FPT_TDC.1.2	The TSF shall use **the PUT KEY data format** when interpreting the TSF data from another trusted IT product: Key generator.

Dependencies: No dependencies.

TSF self test (FPT_TST)

FPT_TST.1: TSF Testing

Functional requirement	Description
FPT_TST.1.1/RESET	The TSF shall run a suite of self tests **at each card reset** to demonstrate the correct operation of the TSF.
FPT_TST.1.2/RESET	The TSF shall provide authorized users with the capability to verify the integrity of the TSF data.
FPT_TST.1.3/RESET	The TSF shall provide authorized users with the capability to verify the integrity of stored TSF executable code.

Dependencies: FPT_AMT.1: Abstract machine testing.

Class FTA: TOE Access

Limitation on Scope of Selectable Attributes (FTA_LSA)

FTA_LSA.1 Limitation on Scope of Selectable Attributes

Functional requirement	Description
FTA_LSA.1.1/SECURE	The TSF shall restrict the scope of the session security attributes **AS.SESSION_KEY** based on **AS.KEYSET_VERSION, AS.KEYSET_VALUE, AS.CURCONTEXT, AS.LOGIC_CHANNEL_NB**.

Dependencies: No dependencies.

Class FTP: Trusted Path/Channels

Trusted Path (FTP_TRP)

FTP_TRP.1 Trusted Path

Functional requirement	Description
FTP_TRP.1.1	The TSF shall provide a communication path between itself and **local** users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure.
FTP_TRP.1.2	The TSF shall permit **local users** to initiate communication via the trusted path.
FTP_TRP.1.3	The TSF shall require the use of the trusted path for **loading of keysets and D.GLPIN**.

Note: ***Applet may use trusted path for loading of data and byte code, or for checking integrity of applet commands, data, keys and privileges.***

Dependencies: No dependencies.

TOE Security Assurance Requirements

For this evaluation, TOE security assurance requirements are low, and the assurance level is EAL 1, augmented with an additional assurance component: AVA_VLA.2.

AVA_VLA.2: Independent Vulnerability Analysis

Developer Action Elements

Assurance requirement	Description
AVA_VLA.2.1D	The developer shall perform and document an analysis of the TOE deliverables searching for **ways** in which a user can violate the TSP.
AVA_VLA.2.2D	The developer shall document the disposition of **identified** vulnerabilities.

Content and Presentation of Evidence Elements

Assurance requirement	Description
AVA_VLA.2.1C	The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.
AVA_VLA.2.2C	The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

Evaluator Action Elements

Assurance requirement	Description
AVA_VLA.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VLA.2.2E	The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure **the identified** vulnerabilities have been addressed.
AVA_VLA.2.3E	The evaluator shall perform an independent vulnerability analysis.
AVA_VLA.2.4E	The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.
AVA_VLA.2.5E	The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a low attack potential.

Security Requirements For The IT Environment

FAU_ARP.1 Security Alarms

Assurance requirement	Description
FAU_ARP.1.1/PHIL	The TSF shall take **an action among the following list** upon detection of a potential security violation: • **Reset the card.**

Dependencies: FAU_SAA.1 Potential violation analysis.

FAU_SAA.1: Potential Violation Analysis

Assurance requirement	Description
FAU_SAA.1.1/PHIL	The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.
FAU_SAA.1.2/PHIL	The TSF shall enforce the following rules for monitoring audited events: **1. Low frequency of clock input** **2. High frequency of clock input** **3. Low voltage power supply** **4. High voltage power supply** **5. Low temperature** **6. High temperature** **7. High voltage for the write process to the EEPROM**

Note: ***Limits are chosen such that proper and secure function within these limits is guaranteed.***

Dependencies: FAU_GEN.1 Audit data generation.

FCS_COP.1: Cryptographic Operation

Assurance requirement	Description
FCS_COP.1.1/ DES_PHIL	The TSF shall perform **encryption and decryption** in accordance with a specified cryptographic algorithm **Triple DES** and cryptographic key sizes **of 112 bits** that meet the following standards: • **FIPS PUB 46-3 (ANSI X3.92), KEYING option 2 (see related document [14]).**
FCS_COP.1.1/ RSA_PHIL	The TSF shall perform **raising to a power modulo an integer** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **of 1024 bits** that meet the following standards: • **ISO / IEC 9796-1, annex A, section A.4 and A.5, and annex C (see related document [20]).**
FCS_COP.1.1/RND	The TSF shall perform **Random Number Generation** in accordance with a specified cryptographic algorithm **(no algorithm)** and cryptographic key sizes **No Key** that meet the following standards: • **FIPS 184.2 (see related document [17]).** ***Note: Entropy of at least 7 bits in each byte.***

Dependencies: FDP_ITC.1 Import of user data without security attributes

FCS_CKM.4 Cryptographic key destruction

FMT_MSA.2 Secure security attributes

FDP_RIP.1: Subset Residual Information Protection

Assurance requirement	Description
FDP_RIP.1.1/PHIL	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: • **DES cryptoprocessor registers.**

Note: ***The deallocation of DES cryptoprocessor registers is done by the IC.***

Dependencies: No dependencies.

FPR_UNO.1: Unobservability

Assurance requirement	Description
FPR_UNO.1.1/PHIL	The TSF shall ensure that **any users** are unable to observe the operations: **all operations CPU, DES coprocessor, FAMEX coprocessor** on **the data stored in EEPROM, RAM, or generated by the random number generator** by **the TOE**.

Dependencies: No dependencies.

FPT_PHP.3: Resistance to Physical Attack

Assurance requirement	Description
FPT_PHP.3.1/PHIL	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 TSP is not violated.

Dependencies: No dependencies.

CHAPTER 6 - TOE SUMMARY SPECIFICATION

TOE Security Functions

Security Functions

F1 - Exceptions Management

A potential attack analysis throws automatically an exception.

This stops the current process.

It notifies the error by the following actions:

- Execute a procedure to process exceptions written by the applet developer (see "Exception Handling" in related document [8]),
- Otherwise output an error status.

F2 - Integrity of CAP File

The CAP file must be signed. This signature is checked by the TOE during the loading of the CAP file. The cryptographic operations (RSA and triple DES) are denied for applets instantiated from this CAP file if check of the CAP file integrity fails. The CM has to be installed with restricted cryptographic rights.

Secure Channel

The TOE provides security services related to information exchanged between the TOE and external users. The life cycle of the Card Manager determines the level of security requirements for exchanges with the Card Issuer. Those services are also available for applets.

F3 - Integrity of Command, Data, Keys and Privileges (Secure Channel)

A MAC of the data transmitted along with the data insures that the data transmitted by the Card Issuer is received unaltered by the TOE.

F4 - Confidentiality of Code and Data During Loading (Secure Channel)

The confidential data is encrypted using a DES algorithm.

F5 - Card Issuer Authentication (Administrator Authentication)

Mutual authentication at the beginning of a communication session, establishing a secure channel is mandatory prior to any relevant data being transferred to the TOE.

F6 - Sensitive Data Confidentiality

Confidentiality is ensured during comparison of two memory blocks in RAM or in EEPROM: PIN values, bytes arrays.

The TOE ensures the confidentiality of residual data:

- With FAT management + Garbage collector,
- By Erasing of EEPROM while deallocating,
- By Erasing of transient arrays while allocating.

F7 - Anti Tearing and Transactions

A transaction is a logical set of updates of persistent data. The TOE provides robust support for atomic transactions, so that data is automatically restored to its original pre-transaction state if the transaction does not complete normally. This mechanism protects against events such as power loss in the middle of a transaction.

The decrementation of remaining tries for PIN is done before the comparison to avoid attack by tearing.

F8 - Ratification

This TSF ensures:

- The management of remaining tries and of validation flag of PIN
- Slow down of Card Issuer authentication timing.

F9 - Internal Roles Management: Card Registry:

The management of internal roles for applets is done using privileges stored in Card Registry.

F10 - Startup Coherence

During startup sequence, if any of the following events occurs, the card mutes itself:

- Inconsistency of Card Manager life cycle
- Bad result for partial test of integrity of EEPROM
- Throw of an exception.

F11 - Card 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.

F12 - Resident Application Dispatcher

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

F13 - Key Integrity From Its Generation: KeyCheck Value

This function verifies the key integrity using key check value algorithm as defined in related document [10] at chapter 9.3.4.

F14 - Card Manager Dispatcher

While Card Manager is selected, this function tests for every command if card issuer authentication is required.

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

F15 - Secret Generation

Random Generation

This function based on the IC random number generator generates random number.

Session Keys Generation

In order to ensure a high level of secure communication for each session involving the Card Manager, this function generates session key. Session DES Keys are used in support of Secure Channel operations.

F16 - RSA Keys Generation

The TOE provides to applets a service for RSA keys generation. This service uses the FameX co-processor.

F17 - DES Algorithm

The TOE provides to applets a DES algorithm based on DES hardware.

F18 - RSA Algorithm

The TOE provides to applets a RSA algorithm based on FameX Co-processor.

Firewall

F19 - Applet Isolation

The TOE supports isolation of contexts and applets.

Isolation means that one applet can not access the fields or objects of an applet in another context unless the other applet explicitly provides an interface for access.

It implements the Applet isolation as defined in the related documents [8], section 7, and [7], section 6.

F20 - JCRE Privileges

Because it is the “system” context, the JCRE context has a special privilege. It can invoke a method of any object on the card. In the TOE, the Card Manager context is the JCRE context.

F21 - JCRE Entry Point

These are objects owned by the JCRE context, but they have been flagged as containing entry point methods. The firewall protects these objects from access by applets. The entry point designation allows the methods of these objects to be invoked from any context.

In the TOE the JCRE Entry point are the APDU object and Card Runtime Exceptions. If the object is a JCRE Entry point the usual rules for Applet Isolation (F19) are changed in order to permit general access under the control of the current context.

F22 - Global Arrays

These objects are owned by the JCRE context, but can be accessed from any context. In the TOE the only global array is the APDU buffer. If the object is a global array the usual rules for Applet Isolation (F19) are changed in order to permit general access under the control of the current context.

F23 - Shareable Interface

The Shareable interface serves to identify all shared objects. Any object that needs to be shared through the applet firewall shall directly or indirectly implement this interface. Only those methods specified in a shareable interface are available through the firewall.

If the applet calls “getPreviousContextAID” from a method that may be accessed either from within the applet itself or when accessed via a shareable interface from an external applet, it identifies the caller identity.

F24 - Keyset Version Management

Loading of a Keyset can update, delete, or add a former Keyset.

F25 - DES Key Access

Access to DES Key is in accordance with the standards defined in related documents [6], [9], and [10]. This access is protected against disclosure of key.

F26 - RSA Key Access

Access to RSA Key is in accordance with the standards defined in related document [6]. This access is protected against disclosure of key.

F27 - Transient Arrays Management in Logical Channel

This function ensures isolation of CLEAR_ON_DESELECT transient arrays belonging to applet(s) executed on different logical channels.

Assurance Measures

TOE security assurance requirements must be low and the scale of evaluation levels constructed using these components is EAL 1 augmented of additional assurance component AVA_VLA.2.

This component gives an augmented confidence on the security functions efficiency.

Configuration Management

This requirement is satisfied by the configuration management tool used: PVCS and its procedures.

ACM_CAP.1 Version numbers

Generation support and acceptance procedures

Delivery and Operation

TOE and its associated documentation are given to users with respect towards procedures.

ADO_IGS.1 Installation, generation, and start-up procedures

Development

TOE development documentation including requirements listed below will be realized:

functional specification.

This documentation is sufficient to meet Assurance Class: ADV.

ADV_FSP.1 Informal functional specification

ADV_RCR.1 Informal correspondence demonstration

Guidance Documents

The expected information needed to satisfy this requirement are present in documentations listed below:

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Tests

The "STD" (OCS documentation) will contain all tests specification and its associated results (expected and obtained).

The TOE will be given to an evaluator for independent testing.

ATE_IND.1 Independent testing – conformance

Vulnerability Assessment

This requirement will be satisfied by a documentation presenting the identified vulnerabilities and an analysis of different penetration attacks performed by an attacker possessing a low attack potential:

AVA_VLA.2 Independent vulnerability analysis.

CHAPTER 7 - PP CLAIMS

No claim of PP compliance is made. No PP claims are included.

PP Reference

Not applicable.

PP Refinements

Not applicable.

PP Additions

Not applicable.

CHAPTER 8 – RATIONALE

This chapter is the OBERTHUR CARD SYSTEMS property.