# Cosmopolic

*2.1 Version 4*

# JavaCard Open Platform

*Security Target*

Manual reference: 057681-03-UDD-AA

# CONTENTS

# ABOUT THIS GUIDE

## Presentation of the Guide

### Purpose

The guide describes the Security Target for the Cosmopolic 2.1 V4 card, running on a Javacard 2.1.1 virtual machine. It defines the:

- Security enforcing functions of the Target Of Evaluation
- Environment in which it operates

### Audience

This guide should be read by all people wishing to understand security implemented in the Cosmopolic Platform.

## Related Documents

The following related documents are available.

| Title | Date, Reference, Version, Issuer |
|---|---|
| Common Criteria for information Technology Security Evaluation, Part 1: Introduction and General Model | August 1999, version 2.1, CCIMB-99-031 |
| Common Criteria for information Technology Security Evaluation, Part 2: Security Functional Requirements | August 1999, version 2.1, CCIMB-99-032 |
| Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance Requirements | August 1999, version 2.1, CCIMB-99-033 |
| Protection Profile – Smart Card Integrated Circuit with Embedded Software | Version 2.0, June 1999 issue, registered at the French Certification Body under the number PP/9911 |
| Java Card 2.1.1 – Application Programming Interfaces | May 18, 2000, Sun Microsystems |
| Java Card 2.1.1 – JCRE | May 18, 2000, Sun Microsystems |
| Java Card 2.1.1 – Virtual Machine Specifications | May 18, 2000, Sun Microsystems |
| Visa Open Platform Card Implementation Specification | March 8, 1999, Visa International (new specifications 04/10/00) |
| Identification cards – Integrated circuit(s) cards with contacts, Part 6: Inter industry data elements | ISO / IEC 7816-6 (1996) |
| Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA) | ANSI X9.31-1998, American Bankers Association |
| FIPS PUB 46-3, Data Encryption Standard | October 25, 1999 (ANSI X3.92), National Institute of Standards and Technology |
| FIPS PUB 81, DES Modes of Operation | April 17, 1995, National Institute of Standards and Technology |
| FIPS PUB 184-2 | April 17, 1995, National Institute of Standards and Technology |
| Information Processing Modes of Operation for a 64-Bit Block Cipher Algorithm | ISO 8372 (1987), International Organisation for Standardisation |
| Banking – Key Management | ISO 8732 (1988), International Organisation for Standardisation |
| 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) |
| Information technology – Security techniques: Data integrity mechanism using a cryptographic check function employing a block cipher algorithm | ISO 9797 (1994) , International Organisation for Standardisation |
| FIPS PUB 140-1, Security requirements for cryptographic modules | January 11, 1994, National Institute of Standards and Technology |
| PKCS#1 The public key cryptography standards | RSA Data Security Inc., 1993 |
| Smart Card Security User Group – Smart Card Protection Profile (SCSUG-SCPP) | Version 3.0, September 9, 2001 |

# Structure of the Guide

## Introduction

This guide contains:

- Six chapters
- A glossary

## Chapters

The chapters in this guide cover the following main topics.

| Chapter | Main Topics |
|---|---|
| Chapter 1 – Security Target Overview | • Identification |
| Chapter 2 – TOE Description | • TOE Overview<br>• TOE Life Cycle<br>• TOE Environment<br>• TOE Limits |
| Chapter 3 – TOE Security Environment | • Roles, Users and Subjects<br>• Assets to be Protected<br>• Assumptions<br>• Threats<br>• Organisational Security Policies |
| Chapter 4 – Security Objectives | • TOE Security Objectives<br>• Environment-Related Security Objectives |
| Chapter 5 – IT Security Requirements | • TOE Security Functional Requirements<br>• TOE Security Assurance Requirements<br>• IT Environment Security Requirements |
| Chapter 6 – TOE Summary Specification | • TOE Security Functions<br>• Assurance Measures |

# CHAPTER 1 – SECURITY TARGET OVERVIEW

## Identification

The security target Lite and Complete Security Target are identified as follows.

| Item | Identification |
|---|---|
| Title | COSMOPOLIC2.1 V4<br>Java Card Open Platform<br>Security Target |
| OCS registration | 057681-03-UDD-AA |
| Name for Complete ST | JPH33V4 ST |
| OCS registration for Complete ST | FQR 110 1254 |
| Version for Complete ST | 1.0, issue 4 |
| Component | P8WE5033 (Philips) |

## Overview

This Security Target covers the development of Cosmopolic 2.1 V4, which receives and manages different types of applications:

- Debit/Credit
- Wallet
- Fidelity
- Pay TV

This card is consistent with the Java Card 2.1.1 and Visa Open Platform 2.0.1 specifications.

The objectives of the Security Target are to describe and specify the:

- Target of Evaluation (TOE), its life cycle, positioning it in the smart card life cycle
- 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 platform active phases
- Security objectives of the TOE and its supporting environment in terms of TOE sensitive information integrity and confidentiality; it includes protection of the TOE and associated documentation during the development and active phases
- Security requirements including TOE functional requirements, TOE assurance requirements and security requirements for the environment
- Summary of the TOE specification, including a description of the security functions and assurance measures that meet the TOE security requirements

## Common Criteria Conformance

The Security Target is in accordance with the Common Criteria, Part 2 conformant and Part 3 augmented.

The assurance level is EAL4 augmented by AVA_VLA.4, ALC_DVS.2 and ADV_IMP.2.

# CHAPTER 2 – TOE DESCRIPTION

## Chapter Overview

This chapter describes the TOE to help understand its security requirements. It addresses the product type, the intended usage and the main TOE features and includes:

- TOE overview
- TOE life cycle
- TOE environment
- TOE limits

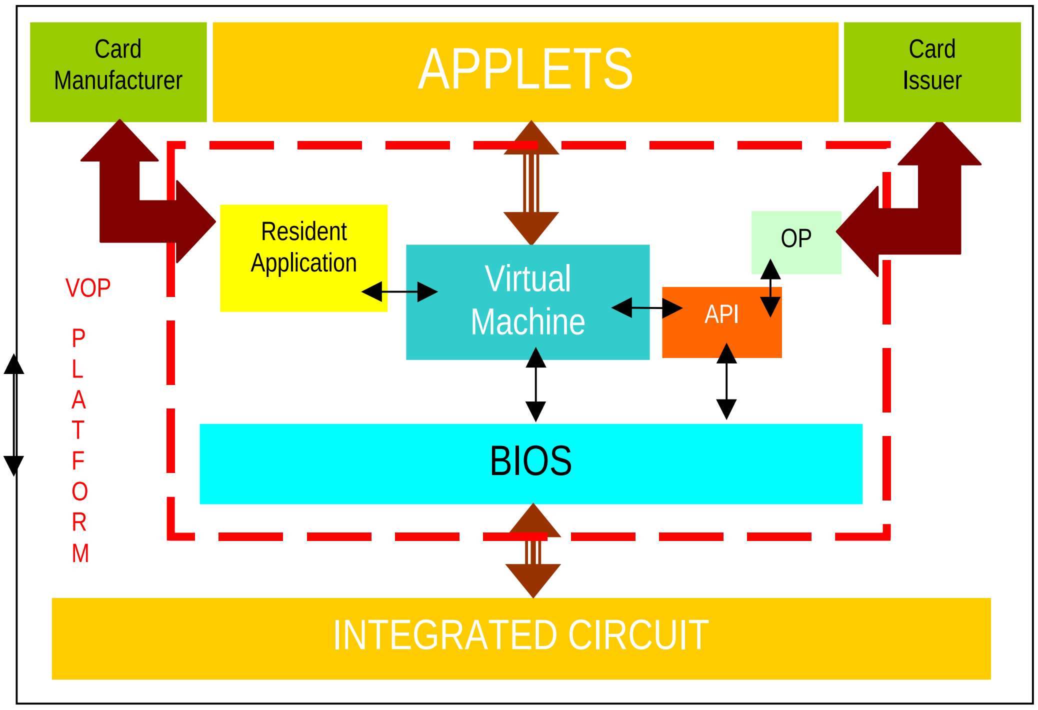
## TOE Overview

The TOE in this ST consists of a VOP Platform called Cosmopolic 2.1 V4 and hosted on the smart card IC.

The platform is based on the:

- Java Card 2.1.1 Specifications
- Open Platform 2.0.1 Card Specifications
- Visa Open Platform Card Implementation Specifications

The smart card supporting the TOE is composed of hardware and software components, as illustrated below.



The TOE, called *VOP Platform*, includes the following components:

- BIOS
- Virtual machine
- APIs
- Open platform application
- Resident application

Each of these components is described in the following paragraphs.

## BIOS

The BIOS is an interface between the hardware and native components, such as the VM, APIs. It implements the following features:

- APDU management (*T=0*, *T=1* protocols)
- Timer management
- Exception management
- Transaction management
- EEPROM access
- Cryptographic modules; the Cosmopolic V4 smart card contains a 2048-bit RSA key generator and also implements the DES, RSA and SHA-1 cryptographic algorithms

## Virtual Machine

The JAVACARD 2.1.1-compliant virtual machine:

- Interprets the JAVACARD applet byte code
- Supports logical channels, allowing one applet to be selected on one channel and another to be selected on another
- Supports the execution of applets loaded in the ROM
- Is activated when an applet is selected

## APIs

The JAVACARD 2.1.1-compliant APIs support:

- Key generation
- Message signature and ciphering
- A proprietary API OCSystem
- A proprietary API FileSystem

## Open Platform Application

The *Open Platform OP2.0 configuration 1b* application:

- Consists of the Card Manager, the API OPsystem and security domains
- Is implemented in Java and its byte-code is stored in ROM
- Is activated when the Card Manager is selected by the Card Issuer

The API Opsystem can be called at any time by the applets.

## Resident Application

The resident application comprises a native code application with a basic main dispatcher to:

- Receive the card commands
- Dispatch them to the application and module functions for execution

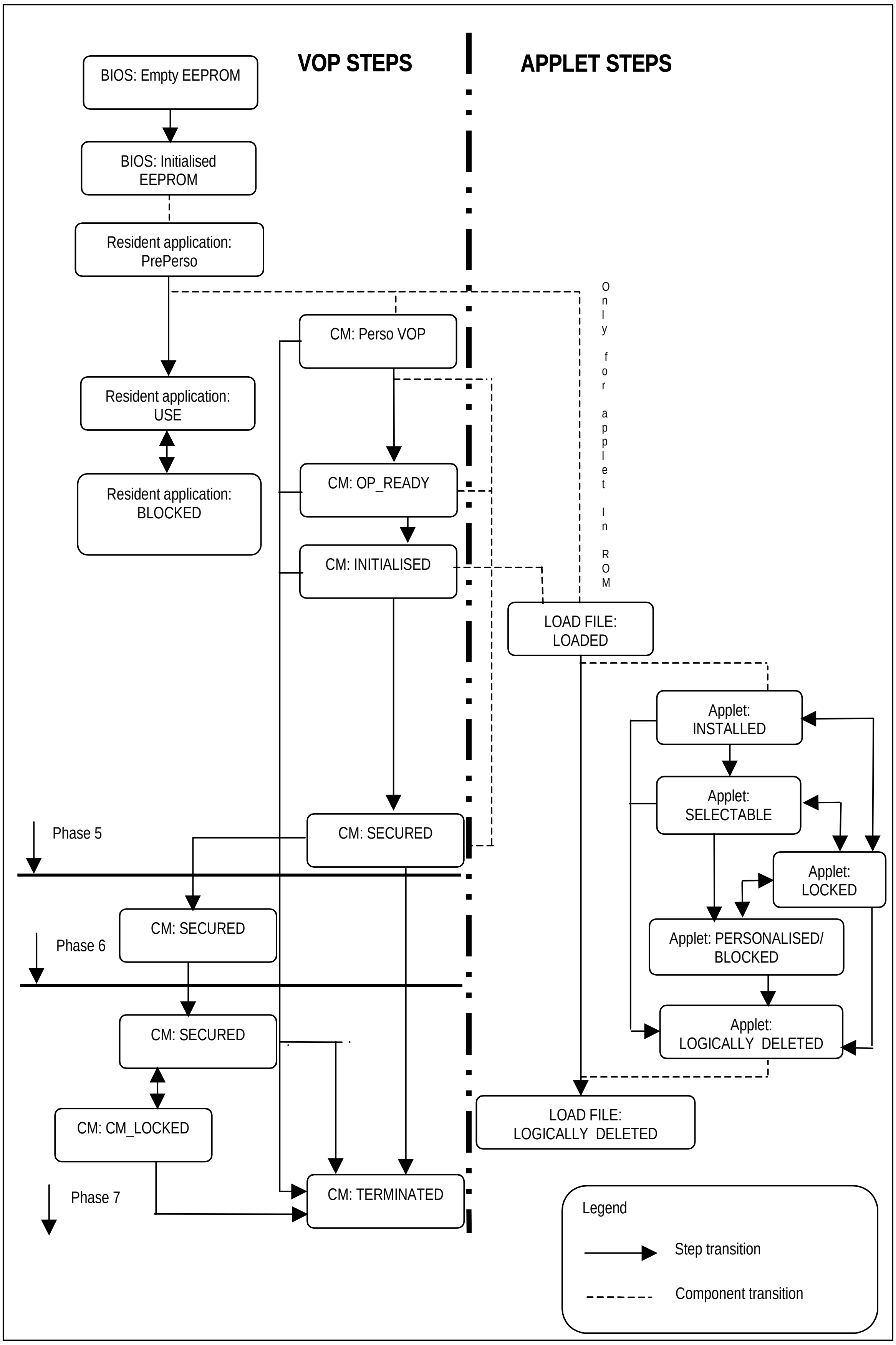
It also handles:

- Card manufacturer authentication
- Logical channel management

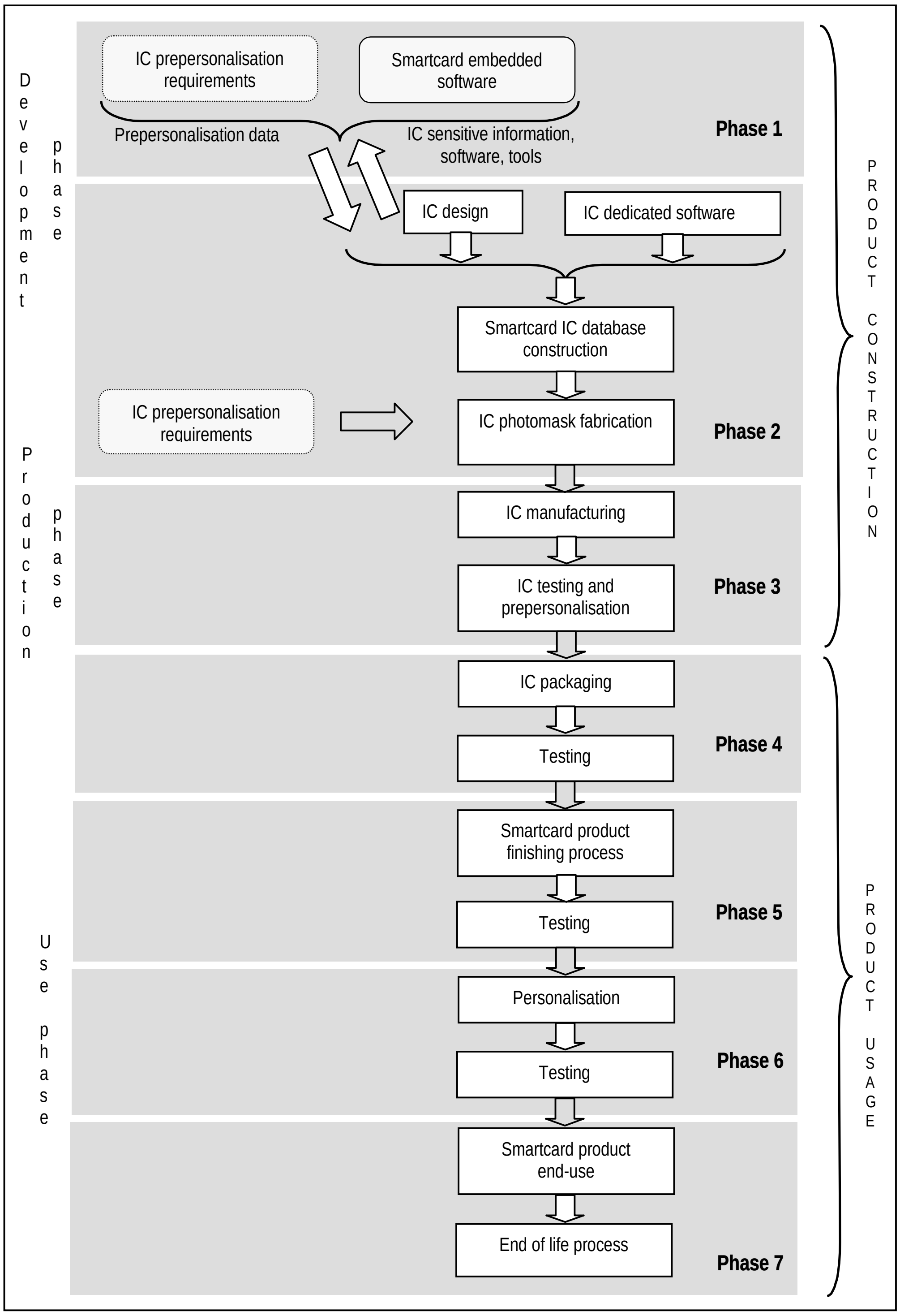
The dispatcher is always activated. Some card administration commands are only available during the prepersonalisation phase.

## TOE Life Cycle

The following illustration presents the VOP life cycle.



The VOP platform life cycle is part of the smart card product life cycle, which is divided into seven phases, in compliance with the Smart Card Integrated Circuit Protection Profile (PP/9806); see the following illustration.



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.

## Card Life Cycle Transitions

The Card Manager maintains the overall security and administration of the card and its contents. Given that the Card Manager plays this supervisory role for the entire card, its life cycle can be considered to be the card's life cycle. The card's life cycle from an Open Platform perspective is only significant at the beginning of the Card Manager life cycle.

The Card Manager:

- Owns and maintains the card life cycle state information
- 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.

| State | Description |
|---|---|
| VOP Personalisation (phase 5) | This life state is the initial state of the Card Manager applet, immediately after it has been installed.<br><br>In this life state, the initialisation key and card and chip CPLC must be loaded before switching to the OP_READY life state. |
| OP_READY (phase 5) | In the OP_READY card life cycle state, all the basic features of the runtime environment are available and the Card Manager is ready to receive, execute and respond to APDU commands.<br><br>The card is assumed to have the following characteristics in the OP_READY state.<br>• The runtime environment is ready for execution.<br>• An initialisation key is available within the Card Manager. |
| INITIALISED (phase 5) | The INITIALISED card life cycle state is an administrative card production state. Most Card Manager personalisation tasks have been carried out when entering this state. |
| SECURED | The SECURED card life cycle state is the normal operating state of the card during issuance. This state is the indicator for the Card Manager to enforce the Card Issuer's security policies related to post-issuance card behaviour, such as applet loading and activation.<br><br>The card is assumed to have the following characteristics in the SECURED state.<br>• The Card Manager contains all necessary key sets and security elements for full functionality.<br>• Card Issuer initiated card content can be changed through the Card Manager.<br>• Post-issuance personalisation of applets belonging to the Card Issuer can be carried out via the Card Manager. |
| CM_LOCKED | The CM_LOCKED state 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 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).<br><br>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 TERMINATED life cycle state to permanently disable all card functionalities, including 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 card expiration.<br><br>The TERMINATED state is irreversible and signals the end of the card's life cycle. |

## BIOS Life Cycle

The BIOS life cycle is divided into two phases.

| Phase | Description |
|---|---|
| Empty EEPROM | When the chip is delivered by the IC manufacturer, the EEPROM is empty, except for the Manufacturer Transport Key (MSK). |
| Initialised EEPROM | On the first power-on, the BIOS initialises its data:<br>• ATR files<br>• Default applet reference<br>• FAT |

## Resident Application Life Cycle

The resident application life cycle is divided into three phases.

| Phase | Description |
|---|---|
| Prepersonalisation state (PP) | The resident application command set (EXTERNAL_AUTHENTICATE, GET_CHALLENGE, GET_DATA, INSTALL, LOAD_APPLET, LOAD_STRUCTURE, MANAGE_CHANNEL) is active. |
| Use | The resident application command set (SELECT, MANAGE_CHANNEL, and GET_DATA only if no applet is selected) is active. |
| Locked/Blocked | All the resident application commands are inactive. |

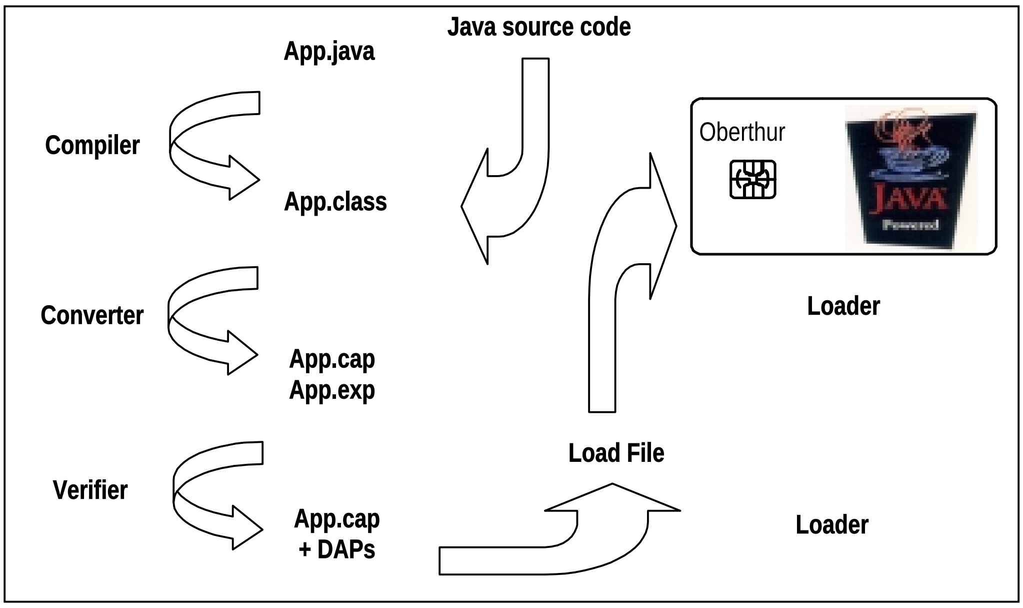
## Load File Life Cycle

The load file life cycle is divided into two phases.

| Phase | Description |
|---|---|
| LOADED | The Card Manager considers that all load files present in the card and available for use either from immutable persistent memory or mutable persistent memory are in the LOADED state. |
| LOGICALLY_ DELETED | If the Card Manager receives a request to delete a load file that cannot be physically deleted (it is stored in the immutable persistent memory for example), the load file is logically deleted by setting its state to LOGICALLY_DELETED.<br><br>Once a load file has been set to the LOGICALLY_DELETED state, it cannot be reversed. The Card Manager considers the LOGICALLY_DELETED state to be equivalent to the physical deletion of the load file. |

## Applet Life Cycle

The delivery of an applet must satisfy a process, using a compiler, converter, verifier and loader. This process is illustrated below.



The applet life cycle begins when an applet is installed in the card. This installation may occur:

- Directly during a loading transaction
- From a load file 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 related to applet management are used to inform the Card Manager of the applet status. This state information must be provided, as the state definitions are:

- Applet-dependent
- Only known to the applet

The Card Manager can then take control of the life cycle again later in the applet life if:

- A security problem is detected by the card or the applet-related Card Issuer
- The applet is to be deleted either physically or logically

The Card Manager:

- Sets the applet life cycle to its initial INSTALLED state during applet installation
- Makes the applet available for selection by setting its life cycle to SELECTABLE

The applet manages life cycle transitions from SELECTABLE to PERSONALISED and optionally to BLOCKED. Applets can be loaded, installed and personalised during phases 5, 6 and 7.

At any point in the applet life cycle, the Card Manager can take control again for security reasons 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 applicable.

The applet life cycle is divided into six phases.

| Phase | Description |
|---|---|
| **INSTALLED** | The INSTALLED state in the open platform context means that:<br>• The applet executable has been properly linked<br>• Any necessary memory allocation has taken place<br>• The applet can be executed<br>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 applet personalisation, which may occur as a separate step. |
| **SELECTABLE** | The SELECTABLE state 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 SELECTABLE state. |
| **PERSONALISED** | The prerequisites for an applet to move to this state are applet-dependent but the state indicates that the applet has been set up with all necessary personalisation data and keys for full runtime functionality.<br>The applet behaviour while in this state is determined by the applet itself. The Card Manager is not involved. |
| **BLOCKED** | The prerequisites for an applet to move to this state are applet-dependent but the state indicates that an applet-specific security problem has been detected either from within the applet or from outside the card.<br>The applet behaviour while in this state is determined by the applet itself. The Card Manager is not involved. |
| **LOCKED** | The LOCKED state is used for security management control by the Card Manager or Issuer to prevent the selection and therefore execution of an applet.<br>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.<br>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.<br>Once an applet is set to LOCKED, it can only be made available for selection once the Card Manager has set it back to the state that it had achieved immediately prior to being set to the LOCKED state. |
| **LOGICALLY_ DELETED** | If the Card Manager receives a request either directly or indirectly from a security domain to delete an applet, which cannot be physically deleted (it is stored in immutable persistent memory for example), the applet is logically deleted by setting its state to LOGICALLY_DELETED.<br>Once an applet has been set to the LOGICALLY_DELETED state, it cannot be reversed. The Card Manager considers the LOGICALLY_DELETED state 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
- Prepersonalisation process environment corresponding to phase 5
- Personalisation environment corresponding to phase 6, personalisation 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

12 The scope of this present security target is:

- TOE development in the OBERTHUR environment during phase 1
- TOE use during phase 7

# CHAPTER 3 – TOE SECURITY ENVIRONMENT

## Chapter Overview

This chapter describes the:

- Security aspects of the environment in which the TOE is to be used
- Assets to be protected
- Secure usage assumptions
- Threats
- Organisational security policies

## Roles, Users and Subjects

### Roles

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

| Role | Description |
|---|---|
| R.Prepersonaliser | Loading of additional data or code<br>Delivery of the card with the Card Manager in the OP_READY state |
| R.Personaliser | 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_AP | Use APIs available on the platform |
| R.Applet_privilege | Modification of the:<br>• CM life cycle<br>• ATR<br>• Global PIN<br>Verification of the mandated DAP |

### Users

The following users are concerned by the TOE.

| User | Description | Roles |
|---|---|---|
| U.Card_manufacturer | Card manufacturer | R.Prepersonaliser<br>R.Personaliser |
| U.Card_Issuer | Administrator of the TOE during its life cycle | R.Personaliser<br>R.Sign_Load_File<br>R.Card_Manager |
| U.Applet | Applet executed on this VOP platform implemented in Java | R.Use_API<br>R.Applet_privilege<br>R.Personaliser |

### Subjects

The following subjects are defined.

| Subject | Description | Users |
|---|---|---|
| S.Resident application | Resident application | U.Card_manufacturer<br>U.Card_Issuer |
| S.Applet | Applet executed on this VOP platform implemented in Java | U.Applet |
| S.CM | Process implementing the VOP specification and activated by the dispatcher | U.Card_manufacturer<br>U.Card_Issuer |

## Assets to be Protected

The TOE sensitive data and code, described in the following paragraphs, must be protected in terms of:

- Confidentiality

  and/or

- Integrity

### User Data

The following user data must be protected.

|  |  |
|---|---|
| D.BYTECOD | 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 Card Manager, used by the DES algorithm |

## TSF Data

The following TSF data must be protected.

| | |
|---|---|
| D.NBAUTHENTIC | Number of authentications |
| D.NB_REMAINTRYOWN | Number of remaining tries for the owner PIN |
| D.NB_REMAINTRYGLB | Number of remaining tries for the global PIN |
| D.CRYPTOGRAM | Indication that cryptograms are used as input for authentication, resulting from key and random computation |
| D.AUDITLOG | Audit log file |
| D.AUDITLOG_SIZE | Audit log size |
| D.FLG_INTEGRITY | Integrity flag |
| | |
| ASG.CARDREG | Card registry: |
| AS.APID | Applet identifier (AID) |
| AS.CMID | Card Manager ID (AID) |
| ASG.APPPRIV | Applet privileges group:<br>Card Manager lock privilege<br>Card terminate privilege<br>Default selected privilege<br>PIN change privilege<br>Security domain privilege<br>Security domain with DAP verification privilege<br>Security domain with mandated DAP verification 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 for a secure channel |
| AS.SECUR_CHANNEL_NUM | Secure channel number |
| AS.MSKEY | Transport key (Manufacturer Secret Key) |
| AS.SECURITY_LEVEL | Secure channel security levels:<br>Confidentiality<br>Integrity<br>Both |
| AS.DAP | Data Authentication Pattern:<br>load file signature |
| AS.SENRST | Sensor reset indicator |
| Check Objects | Check objects (integrity) |
| Check Code | Check code (integrity) |
| Check ROM | Check ROM (integrity) |
| Check FAT | Check FAT (integrity) |
| Quotas EEPROM | Quotas EEPROM |

### Development Data

The following assets must be protected during the development phase:

- VOP Platform specifications
- VOP Platform implementation
- VOP Platform related documentation

## Assumptions

Security systematically concerns the whole system: the weakest element of the chain determines the total system security. Secure usage assumptions must be considered for a secure system using smart card products. These assumptions are made at various levels:

- Tools
- Phases 2, 3 and 4
- Phase 7

| Level | Assumption | Description |
|---|---|---|
| Tools | 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:<br>• Converter<br>• Verifier (all phases)<br>The converter:<br>• Generates verifiable Java Card bytecode, in a well-formed CAP file, which encapsulates the information in Java class files that comprise exactly one Java package<br>• Checks the limits imposed by the JC21 specification on the number of classes, methods and fields<br>• Generates well-formed export files, the conversion process preserving the code semantics of the applet's Java code (all phases)<br>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. |
| Phases 2, 3 and 4 | A.USE_PROD | To maintain confidentiality and integrity of the TOE and prevent any possible copy, modification, retention, theft or unauthorised use, it is assumed that security procedures are used during the:<br>• IC development phase<br>• IC production phase<br>• IC packaging and tests operations through phases 2, 3 and 4 |
| | 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 off-card systems and includes the provision of suitable physical, personnel and procedural measures as well as technical measures. |
| Phase 7 | A.SECURE_LOAD | It is assumed that secure communication protocols and procedures are used between the Manufacturer and the applet developer or 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 also described.

## Attackers

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

For this evaluation (EAL4), these attackers are considered to have a high-level attack potential.

They may also use software (an applet for example), loaded and residing in the smart card memory (also outside the TOE), and attempting to access sensitive data.

These attacks can only be carried out by applet developers with sufficient knowledge to implement an applet passing the Java Card verifier and bypassing the firewall.

## TOE Development Phase

The following threat may be present during the TOE development phase.

| Threat | Description |
|---|---|
| T-1.INFO_DVPT | Unauthorised modification, disclosure during the specification, development and validation phases of the:<br>• TOE specification<br>• TOE design<br>• TOE implementation<br>• Tools used for TOE testing and development<br>• TOE tests results<br>• IC specifications delivered by the manufacturer<br>Unauthorised modification, disclosure during TOE storage and TOE delivery. |

## TOE Active Phases

The following threat may be present during the TOE active phases.

| Threat | Description |
|---|---|
| T0.MOD_MEM | Unauthorised modification of information in ROM and EEPROM. A manipulation or failure of the TOE may modify:<br>• User data<br>• TSF data<br>• OS code |
| T2.MOD_INITKEY | Modification and disclosure of the initialisation key to be loaded. |
| T3.DISCLOS_ MSKEY | Disclosure of the transport key. |

| Threat | Description |
|---|---|
| T4.MOD_PIN_KEYS | Modification or disclosure of the:<br>• Global PIN<br>• Owner PIN<br>• Keysets<br>• Keys |
| T5.LOAD_APP | Unauthorised loading and installation of applets or load files. The load file stored in ROM may be made available in an unauthorised way. |
| T6.DISCLOS_CODE | Applet code disclosure during loading. |
| T7.EXEC_ EXTCODE | Unauthorised execution of the bytecode. |
| T8.MOD_BYTE | Unauthorised modification of the bytecode (applet, load file). |
| T9.MOD_ LIFECYCLE | Unauthorised modification of the life cycle of:<br>• CM<br>• Load file<br>• Resident application<br>• Applet |
| T11.DEL_APP | Unauthorised deletion of the load file or applet. |
| T13.SELECT_APP | Unauthorised selection of an applet: some applet life cycle states forbid the selection of the applet, some Card Manager life cycle states forbid the selection of all applets. |
| T15.MOD_AID | Unauthorised modification of the AID of the CM or an applet. |
| T16.PERSO_APP | Unauthorised applet personalisation using the Card Manager *ProviderSecurityDomain* service (modification of keys, Java objects, PINs, applet life cycle). |
| T17.ASSOC_ SDAPP | Unauthorised association between the security domain and the applet. |
| T18.MOD_PRIV | Modification, disclosure of privileges. |
| T21.USE_IDENT | Identity usurpation by an applet, to access a shareable Java object. |
| T23.ACCES_DATA | Unauthorised accesses when another entity, such as an applet or human user, is reading or writing the data of one applet (PINs, keys, data tables and objects). |
| T24.DISCLOS_KEY | Disclosure of the generated key in the card. |
| T25.PERSO_RESID | Unauthorised Prepersonalisation of the resident application (modification of EEPROM contents). |
| T29.MOD_DATAVM | Unauthorised modification of the VM data and code. |
| T30.MOD_ATR | Unauthorised modification of the ATR files (stored in EEPROM). |
| T32.RESOURCES | Total or partial hoarding by a malicious applet of card resources delivered by the platform. |

## Organisational Security Policies

The TOE must comply with the following organisational policy statements.

| **Policy** | **Description** |
|---|---|
| P.JC_ FRAMEWORK | The TOE must support the core APIs of the Javacard specifications. |
| P.SERVICES | The VOP platform must provide services (cryptographic or others) to allow applets to implement security mechanisms. |

# CHAPTER 4 – SECURITY OBJECTIVES

## Chapter Overview

The security objectives cover the following main aspects:

- Integrity and confidentiality of assets
- Protection of the TOE during its active life via active security functions
- Protection of the TOE development environment and delivery process

## TOE Security Objectives

The following security objectives are defined at TOE level.

| Objective | Description |
|---|---|
| O.DETECT_MEM | The platform must:<br>• Detect loss of integrity in the global EEPROM and user's security information and attributes<br>• Ensure the consistency of all TSF data |
| O.INT_ROM | The platform must ensure the integrity of all code stored in ROM. |

These objectives are divided into four groups:

- Card Manager objectives
- Applet management objectives
- Resident application objectives
- BIOS objectives

### Card Manager Objectives

The following security objectives are defined at Card Manager level.

| Objective | Description |
|---|---|
| O.AUTHE_PERS | The personaliser must be authenticated before executing the commands installing the Card Manager. Loading and installation of applets, including those stored in ROM, require successful authentication during the prepersonalisation phase. |
| O.INITKEY | The TOE must check if the initialisation key is ciphered and signed with the transport key. |
| O.SENS_DATA | The integrity of stored sensitive data must be ensured (audit log file, keyset, global PIN and so on). |
| O.CRYPT_DATA | The TOE must check that the global PIN and keyset are loaded ciphered and signed. The Global PIN or keyset are loaded or deleted after successful authentication in all phases. |
| O.AUTHE_LOAD | An applet or load file is loaded, installed or erased through the Card Manager after successful authentication in all phases. |

| Objective | Description |
|---|---|
| O.SIGN_COD | The TOE must check that all applets loaded have been signed. Otherwise, cryptographic algorithms are not available or restricted. |
| O.PROTEC_COD | During the loading of bytecode on the card provided by an application provider and signed by the card issuer, the TOE guarantees its protection in terms of confidentiality and integrity. |
| O.AUTHE_CMS_OP | The VOP platform can be used by a Card Management System while controlling that the following management operations are only performed by the Card Issuer.<br>• The life cycle state can be changed only after a successful authentication.<br>• The most significant errors are notified and recorded in an audit log file and the Card Issuer can read all recorded errors.<br>• The Card Issuer can request the VOP platform in order to inform it of the applets and load files present on the card and their life cycle states. |
| O.MGT_CYC | Applet and Card Manager life cycle states must be always valid and those states must condition the execution of the applets. Some life cycle evolutions are forbidden. |
| O.AUTHE_AID | An applet or Card Manager AID can only be changed by the Card Manager after successful authentication of the Card Issuer. |

## Applet Management Objectives

The following security objectives are defined at applet management level.

| Objective | Description |
|---|---|
| O.PERSO_APP | An applet can only be personalised by the applet itself:<br>• Using its resources<br>• With delegation to the Card Manager or its security domain |
| O.PROT_PRIV | Applets privileges are protected in terms of integrity. These privileges are:<br>• Applet-security domain association<br>• Default applet<br>• Right to change the CM state to CM_LOCKED |
| O.AUTHE_PRIV | Some applet privileges can be modified by the Card Issuer (via the Card Manager) after successful authentication. |
| O.AUTHE_CMS_APP | An applet can only change states after a successful Card Issuer authentication. |
| O.INTEG_USER | The TOE ensures integrity of users, Java objects and user packages. |
| O.CONF_SENSDATA | The TOE ensures confidentiality of sensitive data: PIN, keys and so on. |
| O.FIREWALL | There is a firewall between applets. Furthermore, the applets cannot modify TOE data and code. |

| Objective | Description |
|---|---|
| O.CRYPT_APP | The TOE provides a set of security features by using the Application Programming Interface for:<br>• Cryptographic operations (DES, RSA); these operations must check the integrity and confidentiality of the keys<br>• Generation of secure RSA key<br>• True random generation<br>• Access control PIN management (creation, update, verification, deletion)<br>• Protection against power loss and tearing through transaction mechanism |
| O.RESOURCES | The TOE must provide the means of controlling the use of resources by its users and subjects so as to prevent permanent unauthorised denial of service.<br>Example: It must prevent a loaded application from taking control of the whole persistent memory (EEPROM), thus prohibiting other loaded applications from using it. |

## Resident Application Objectives

The following security objectives are defined at resident application level.

| Objective | Description |
|---|---|
| O.AUTHE_PERS | The personaliser must be authenticated prior to executing the commands installing the Card Manager. The loading and installation of applets, including those stored in ROM, require successful authentication during the prepersonalisation phase. |
| O.AUTHE_CYCAR | The modification of resident application life cycle requires successful authentication. |
| O.EXTEND | The TOE, when properly specified and authorised, must support functionality modification or enhancement. |

## BIOS Objectives

The following security objective is defined at BIOS level.

| Objective | Description |
|---|---|
| O.AUTHE_ATR | The modification of ATR files requires authorisation (authentication of the card manufacturer or applet privileges). |

## Environment-Related Security Objectives

The environment-related security objectives concern the following levels:

- TOE development environment
- TOE environment
- TOE IT environment

### TOE Development Environment

The following security objectives are defined for the TOE development environment.

| Objective | Description |
|---|---|
| O.AUTHO_PEOPLE | Specifications, software, detailed design, schematics/layout or any other design information must be accessible only to authorised personnel (physical, personnel, organisational and technical procedures). |
| O.TOE_DESIGN | The TOE must:<br>• Be designed in a secure manner, that is focusing on program and data integrity<br>• Use all security features and perform security mechanisms as required by the TOE designer (e.g. cryptography)<br>It is assumed that TOE functionalities are suitably tested during phase 1. |
| O.DEV_TOOLS | To guarantee program and data integrity, the TOE must be designed in a secure manner, by exclusively using<br>• Software development tools (compilers, assemblers, linkers, simulators, verifiers)<br>• Software-hardware integration testing tools (emulators) |
| O.SOFT_DLV_IC | The software under development and the IC masked with the software must be delivered through a trusted delivery and verification procedure that must guarantee integrity and confidentiality. |
| O.SOFT_DLV_TR | The software under development and the IC masked with the software must be delivered to the correct party through a trusted delivery and verification procedure that must ensure full traceability. |

### TOE Environment

The following security objectives are defined for the TOE environment.

| Objective | Description |
|---|---|
| O.MSKEY_MGT | The transport key must be:<br>• Stored in a secured area<br>• Exchanged between the platform developer and the IC manufacturer in a secured manner to respect key integrity and confidentiality |
| O.DEV_APPLET | The applets must be designed in a secured manner to respect key integrity and confidentiality. |
| O.VERIF_COD | All applets must be verified by a verifier before signature. All applets loaded must be signed. |
| O.CODE_MGT | The applet code, data and keys must be transmitted in a secured manner to ensure confidentiality and integrity. The actors involved are the applet personaliser, manufacturer, Card Issuer and applet developer. |
| O.TOE_MGT | During phases 2, 3 and 4, the IC manufacturer must guarantee TOE confidentiality and integrity. |

## TOE IT Environment

The following security objectives are defined for the TOE IT environment.

| Objective | Description |
|---|---|
| O.CRYPT_IC | The IC must provide a set of security features:<br>• Cryptographic operations (DES, RSA); these operations must ensure integrity checking and key confidentiality<br>• True random generation<br>• Erasing (deallocation) of cryptographic buffers |
| O.IC_PROT | The IC must protect against manipulation of the:<br>• Hardware<br>• Software and data stored in the chip RAM, ROM and EEPROM |

# CHAPTER 5 – IT SECURITY REQUIREMENTS

## Chapter Overview

This chapter defines the detailed IT security requirements that must 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 Functional Requirements

All functional requirements are drawn from Common Criteria, Version 2.1, Part 2, except for Security Functional Component, FAU_LST.1. The details of this component and rationale for its inclusion are given in this section. This requirement and its corresponding rationale are extracted from the *SCSUG-SCPP Protection Profile* document.

### Explicit Security Requirements

A sequence-related audit list function (FAU_LST.1 – Audit list generation) is defined with the ability to directly specify the audit information to be recorded. It supports TOE security without imposing any unnecessary requirements. This function is defined in its entirety as follows.

| **FAU_LST.1 – Audit list generation** | |
|---|---|
| Management | No management activities foreseen. |
| Audit | No actions identified that should be auditable if FAU_LST Audit list generation is included in the PP/ST. |
| Subfunctions | |
| FAU_LST.1.1 | The TSF must be able to generate an audit list of the following auditable events:<br>• All auditable events for the *minimum*, *basic*, *detailed* or *non-specifie*d audit levels<br>• Assignment: *specifically defined auditable event*s |
| FAU_LST.1.2 | The TSF must record within each audit record at least the following information:<br>• Assignment<br>• Audit relevant information |
| Dependencies | None. |

This definition is required as the TOE is unpowered, except when connected to a CAD device. Any time and date information that may be available is dependent on the CAD, which is not considered to be a trusted source for this information. Audit data cannot, therefore, be linked to time and date but must depend on sequence of operations. This requires the elimination of two of the elements included in FAU_GEN.1 (Audit data generation).

| Element | State |
|---|---|
| FAU_GEN.1.1 | The TSF must be able to generate an audit record of the following auditable events:<br>▪ Startup and shutdown of the audit functions |
| FAU_GEN.1.2 | The TSF must record within each audit record at least the following information:<br>• Date and time of the event<br>• Type of event<br>• Subject identity<br>• Outcome (success or failure) of the event |

The lack of a reliable time and date prevents FAU_GEN.1.2-a element requirement to be met. Likewise, for element FAU_GEN.1.1-a, in the absence of the time and date, the audit list will either exist or the information will not be available. Recording the startup and shutdown of the audit functions makes little sense. Thus, neither of these elements is included in FAU_LST.1. Additionally, the memory capacity of the TOE is extremely limited. It is not practical to impose a requirement that introduces overhead not absolutely essential to the security needs of the product.

Thus, the audit function in its classical sense is not a useful concept for this TOE. At best, the TOE should preserve some information, which could be of use when identifying faults and vulnerabilities. This information can be recorded in the sequence of its occurrence.

FAU_LST.1 (Audit List Generation) is modelled on FAU_GEN.1 (Audit Data Generation), which has a dependency on FPT_STM.1 (Reliable Time Stamps). As discussed in this section and in the previous discussion regarding the justification for unmet dependencies on FPT_STM.1, it is not appropriate to include this dependency on FAU_LST.1. There are therefore no dependencies for FAU_LST.1.

FAU_GEN.1 (Audit Data Generation) is a dependency for a variety of other requirements. The intent of FAU_LST.1 is identical to that of FAU_GEN.1 in that it requires the generation of specific types of audit information, which can then be acted upon by the other requirements. FAU_LST.1 is, therefore, an appropriate substitution for FAU_GEN.1 in meeting these dependencies.

## Class FAU: Security Audit

| FAU_ARP – Security audit automatic response | |
|---|---|
| **FAU_ARP.1 – Security alarms** | |
| Subfunctions | |
| FAU_ARP.1.1 | The TSF must take an action among the following list upon detection of a potential security violation:<br>1. Make the card mute<br>2. Block the action that produced the security violation, throw an exception and lock the responsible applet |
| Dependencies | FAU_SAA.1 Potential violation analysis |

| FAU_LST – Audit list generation | |
|---|---|
| **FAU_LST.1 – Audit list generation** | |
| Subfunctions | |
| FAU_LST.1.1 | The TSF shall be able to generate an audit list of the following auditable events:<br>• All auditable events for the **none** level of audit<br>• **specified in the following list : security exceptions, invalid reference exceptions, object integrity loss** |
| FAU_LST.1.2 | The TSF shall record within each audit record at least the following information:<br>• type of event<br>• subject identity<br>Note : Only failures are recorded |
| Dependencies | No dependencies |

| **FAU_SAA – Security audit analysis** | |
|---|---|
| **FAU_SAA.1 – Potential violation analysis** | |
| Subfunctions | |
| FAU_SAA.1.1/ HARD | The TSF must be able to apply a set of rules when monitoring the audited events and based on these rules must indicate a potential TSP violation. |
| FAU_SAA.1.2/ HARD | The TSF must enforce the following rules when monitoring audited events:<br>• Accumulation or combination of the following auditable events known to indicate a potential security violation; indication by the hardware that the reset was caused by exception sensors<br>• Any other rules: none |
| FAU_SAA.1.1/ SOFT | The TSF must be able to apply a set of rules when monitoring the audited events and based on these rules indicate a potential TSP violation. |
| FAU_SAA.1.2/ SOFT | The TSF must enforce the following rules when monitoring audited events:<br>• Accumulation or combination of the following auditable events known to indicate a potential security violation:<br>1. Card Manager life cycle inconsistency audited through the self-test mechanism and the lifecycle checks in all administration operations (TERMINATED)<br>2. Unauthorised object access outside the active context audited through the firewall mechanism; automatic throw of a security exception<br>3. Invalid access to a reference audited through the object access mechanism; automatic throw of an invalid reference exception<br>4. Inconsistency of the EEPROM integrity flag: make the card mute<br>5. Audit log<br>6. Non-integrity of a key object audited through the key usage mechanism; integrity loss notifications<br>• Any other rules: after the platform reset if $n_{max}$ records are present in the audit log file, the card is mute |
| Dependencies | FAU_GEN.1 Audit data generation |

| **FAU_SAR – Security audit review** | |
|---|---|
| **FAU_SAR.1 – Audit review** | |
| Subfunctions | |
| FAU.SAR.1.1 | The TSF must provide U.Card_Issuer with the capability to read notifications of the occurrence of the following events:<br>• Security exceptions<br>• Invalid reference exceptions<br>• Object integrity loss from the audit records |
| FAU.SAR.1.2 | The TSF must provide the audit records in a manner suitable for the user to interpret the information. |
| Dependencies | FAU_GEN.1 Audit data generation |

| **FAU_STG – Protected audit trail storage** | |
|---|---|
| **FAU_STG.1 – Security audit event storage** | |
| Subfunctions | |
| FAU.STG.1.1 | The TSF must protect the stored audit records from unauthorised deletion. |
| FAU.STG.1.2 | The TSF must be able to prevent and detect modifications to the audit records. |
| Dependencies | FAU_GEN.1 Audit data generation |

## Class FCO: Communication

| FCO_NRO – Non-repudiation of origin | | |
|---|---|---|
| **FCO_NRO.1 – Selective proof of origin** | | |
| Subfunctions | | |
| FCO_NRO.1.1/ SHARINT | The TSF must be able to generate evidence of origin for transmitted method invocation on recipient request. | |
| FCO_NRO.1.2/ SHARINT | The TSF must be able to relate the context of the information originator and the parameters of the information to which the evidence applies. | |
| FCO_NRO.1.3/ SHARINT | The TSF must provide a capability to verify the evidence of origin of the information given to the recipient during invocation. | |
| | Note: The shareable interface must be managed. | |
| FCO_NRO.1.1/ CMOPINI | The TSF must be able to generate evidence of origin for the transmitted D.LOADFILE and ASG.APPPRN on originator request. | |
| FCO_NRO.1.2/ CMOPINI | The TSF must 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 must provide a capability to verify the evidence of origin of the information given to the recipient via the secure channel. | |
| | Note: This function is applicable when the Card Manager life cycle phase is OP_READY or INITIALISED. | |
| Dependencies | FIA_UID.1 | Timing of identification |

| **FCO_NRO.2 – Enforced proof of origin** | | |
|---|---|---|
| Subfunctions | | |
| FCO_NRO.2.1/ DAP | The TSF must enforce the generation of evidence of origin for the transmitted CAP file at all times. | |
| FCO_NRO.2.2/ DAP | The TSF must 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 must provide a capability to verify the evidence of origin of information given to the recipient during CAP file loading. | |
| | Note: DAP verification. | |
| FCO_NRO.2.1/ CMSECURE | The TSF must enforce the generation of evidence of origin for the transmitted D.LOADFILE, AS.KEYSET_VALUE, ASG.APPPRN, D.GLPIN at all times. | |
| FCO_NRO.2.2/ CMSECURE | The TSF must 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 must provide a capability to verify the evidence of origin of the information given to the recipient via the secure channel. | |
| | Note: This function is applicable when the Card Manager life cycle is SECURED (for load files and privileges). | |
| Dependencies | FIA_UID.1 | Timing of identification |

## Class FCS: Cryptographic support

| **FCS_CKM – Cryptographic key generation** | | |
|---|---|---|
| **FCS_ CKM.1 – Cryptographic key generation** | | |
| Subfunctions | | |
| FCS_CKM.1.1 / RSA | The TSF must generate cryptographic keys in accordance with a specified cryptographic key generation algorithm (RSA) and specified cryptographic key sizes of 512, 768, 1024 or 2048 bits and meeting the ANSI X9.31 standard. | |
| Dependencies | FCS_COP.1 | Cryptographic operation |
| | FCS_CKM.4 | Cryptographic key destruction |
| | FMT_MSA.2 | Secure security attributes |
| **FCS_ CKM.3 – Cryptographic key access** | | |
| Subfunctions | | |
| FCS_CKM.3.1 | The TSF must perform the following types of cryptographic key access in accordance with a specified cryptographic key access method (see the table on the following page) and meeting the following standards: 1. Open Platform Card Specification, Chapters 8 and 9.9 2. Visa Open Platform Card Implementation Specification, Chapter 9.3 3. Java Card 2.1.1 – Application Programming Interfaces, *Javacard.security* and *Javacard.crypto* packages. | |

| FCS_ CKM.3 – Cryptographic key access | | |
|---|---|---|
| | **Cryptographic Key Access Type** | **Cryptographic Key Access Methods/ Commands** |
| | DES | Commands:<br><br>PUT_KEY<br>EXTERNAL AUTHENTICATE<br>INITIALIZE UPDATE<br><br>*ProviderSecurityDomain* key access methods:<br><br>decryptVerifyKey<br>openSecureChannel<br>unwrap<br>verifyExternalAuthenticate<br><br>*APICrypto* key access methods:<br><br>Key.clearKey<br>DES.getKey<br>DES.setKey<br>Signature.init<br>Signature.update<br>Signature.sign<br>Signature.verify<br>Cipher.init<br>Cipher.update<br>Cipher.doFinal |
| | RSA | *ProviderSecurityDomain* key access method:<br><br>DecryptVerifyKey<br><br>*APICrypto* key access methods:<br><br>Key.clearKey<br>RSAPrivateCRTKey.setP<br>RSAPrivateCRTKey.setQ<br>RSAPrivateCRTKey.setPQ<br>RSAPrivateCRTKey.setDP1<br>RSAPrivateCRTKey.setDQ1<br>RSAPrivateCRTKey.getP<br>RSAPrivateCRTKey.getQ<br>RSAPrivateCRTKey.getPQ<br>RSAPrivateCRTKey.getDP1<br>RSAPrivateCRTKey.getDQ1<br>RSAPrivateKey.setModulus<br>RSAPrivateKey.setExponent<br>RSAPrivateKey.getModulus<br>RSAPrivateKey.getExponent<br>RSAPublicKey.setModulus<br>RSAPublicKey.setExponent<br>RSAPublicKey.getModulus<br>RSAPublicKey.getExponent<br>Signature.init<br>Signature.update<br>Signature.sign<br>Signature.verify<br>Cipher.init<br>Cipher.update<br>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** | | |
|---|---|---|
| Subfunctions | | |
| FCS_CKM.4.1 | The TSF must destroy cryptographic keys in accordance with a specified cryptographic key destruction method (that also prevents the destroyed keys from being referenced) and meeting the following standards:<br>• Visa Open Platform Card Implementation Specification, Chapter 6.4.2: Key Renewal and Replacement<br>• ISO 11166 for asymmetric keys (RSA) and ISO 11568 for symmetric keys (DES) | |
| Dependencies | FDP_ITC.1 | Import of user data without security attributes |
| | FMT_MSA.2 | Secure security attributes |

| **FCS_COP – Cryptographic operation** | | |
|---|---|---|
| **FCS_ COP.1 – Cryptographic operation** | | |
| Subfunctions | | |
| FCS_COP.1.1 / DES | The TSF must perform signature, verification of signature, encryption and decryption in accordance with a specified DES cryptographic algorithm and cryptographic key sizes of 56 bits (DES) and 112 bits or 168 bits (triple-DES) and meeting the following standards:<br>• FIPS PUB 46-3, Data Encryption Standard (ANSI X3.92)<br>• FIPS PUB 81, DES Modes of Operation<br>• Information Processing Modes of Operation for a 64-Bit Block Cipher Algorithm, ISO 8372 (1987)<br>• Banking – Key Management, ISO 8732 (1988)<br>• ISO 9797 (1994), Information Technology – Security techniques: Data integrity mechanism using a cryptographic check function employing a block cipher algorithm | |
| FCS_COP.1.1 / RSA | The TSF must perform signature, verification of encryption and decryption in accordance with a specified RSA cryptographic algorithm and cryptographic key sizes of 512, 768, 1024 or 2048 bits and meeting the following standards:<br>• Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), ANSI X9.31<br>• 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<br>• PKCS#1 The Public Key Cryptography Standards, RSA Data Security Inc. 1993 | |
| 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

| **FDP_ACC – Access control policy** | | |
|---|---|---|
| **FDP_ACC.1 – Subset access control** | | |
| Subfunctions | | |
| FDP_ACC.1.1/ JCREPRIV | The TSF must enforce the JCREPRIV access control performed by the API OCSystem on the following list of subjects, objects and operations. | |
| | Subject: | S.CM |
| | Objects: | ATR files<br>Card Manager life cycle<br>Applet life cycle<br>Applet privileges<br>Applet export rights<br>Transport key<br>Applet<br>CAP file<br>Package |
| | Operations: | setATR<br>lockCard<br>useCard<br>SMWithTransportKey<br>delete<br>setDefaultApplet<br>setStatus<br>setAID<br>setAIDRef<br>setRights<br>getRights.getAIDRef<br>getRights<br>Install<br>LoadInit<br>LoadNext<br>LoadEnd |
| FDP_ACC.1.1/ APPPRIV | The TSF must enforce the APPPRIV access control performed by the API OPSystem on the following list of subjects, objects and operations. | |
| | Subject: | S.Applet |
| | Objects: | ATR files<br>Card Manager life cycle<br>Applet life cycle<br>Global PIN |
| | Operations: | setATRHistoricalBytes<br>TerminateCard<br>CMLock<br>lockApplet<br>setCardContentState<br>setPin<br>verifPin |
| Dependencies | FDP_ACF.1 | Security attribute based access control |

| **FDP_ACC.2 – Complete access control** | |
|---|---|
| Subfunctions | |
| FDP_ACC.2.1/ PP | The TSF must enforce the prepersonalisation access control on S.Resident application and for all objects and operations among the subjects and objects covered by the SFP. |
| FDP_ACC.2.2/ PP | The TSF must 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 must enforce the FIREWALL access control on S.Applet and for all D.DATAOBJ objects and the operations among the subjects and objects covered by the SFP. |
| FDP_ACC.2.2/ FIREWALL | The TSF must 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 must enforce the CM access control on S.CM and for the D.LOADFILE, AS.KEYSET_VALUE, D.GLPIN, ASG.APPPRIV, AS.CMLIFECYC, AS.KEYSET_VERSION, D.APPLIFECYC and ASG.CARDREG objects and all operations among the subjects and objects covered by the SFP. |
| FDP_ACC.2.2/ CM | The TSF must 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 |

**FDP_ACF – Access control function**

| **FDP_ACF.1 – Security attribute based access control** | |
|---|---|
| Subfunctions | |
| FDP_ACF.1.1/ JCREPRIV | The TSF must enforce the JCREPRIV access control when writing to objects based on AS.CMCONTEXT. |
| FDP_ACF.1.2/ JCREPRIV | The TSF must enforce the following rules to determine if an operation among controlled subjects and objects is allowed:<br>Current Context = AS.CMCONTEXT |
| FDP_ACF.1.3/ JCREPRIV | The TSF must explicitly authorise access of the subjects to the objects based on the following additional rules:<br>None |
| FDP_ACF.1.4/ JCREPRIV | The TSF must explicitly deny access of the subjects to the objects based on the following additional rules:<br>None |
| FDP_ACF.1.1/ APPPRIV | The TSF must enforce the APPPRIV access control when writing to objects based on ASG.APPPRIV. |
| DP_ACF.1.2/ APPPRIV | The TSF must enforce the following rule to determine if an operation among controlled subjects and controlled objects is allowed:<br>The current applet privileges allow this operation. |
| FDP_ACF.1.3/ APPPRIV | The TSF must explicitly authorise access of the subjects to the objects based on the following additional rules:<br>None |
| FDP_ACF.1.4/ APPPRIV | The TSF must explicitly deny access of the subjects to the objects based on the following additional rules:<br>None |
| FDP_ACF.1.1/ PP | The TSF must enforce the prepersonalisation access control to objects based on AS.AUTH_MSK_STATUS. |
| FDP_ACF.1.2/ PP | The TSF must enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:<br>AS.AUTH_MSK_STATUS = TRUE |

| **FDP_ACF.1 – Security attribute based access control** | |
|---|---|
| Subfunctions *(cont.)* | |
| FDP_ACF.1.3/ PP | The TSF must explicitly authorise access of subjects to objects based on the following additional rules:<br>None |
| FDP_ACF.1.4/ PP | The TSF must explicitly deny access of subjects to objects based on the following additional rules:<br>None |
| FDP_ACF.1.1/ FIREWALL | The TSF must enforce the FIREWALL access control on objects based on AS.CURCONTEXT. |
| FDP_ACF.1.2/ FIREWALL | The TSF must enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed (see Java Card 2.1.1 – JCRE , Section 6)<br>Current Context = Object Context |
| FDP_ACF.1.3/ FIREWALL | The TSF must explicitly authorise access of the subjects to the objects based on the following additional rule:<br>Object context = JCRE context |
| FDP_ACF.1.4/ FIREWALL | The TSF must explicitly deny access of the subjects to the objects based on the following additional rules:<br>None |
| FDP_ACF.1.1/ CM | The TSF must enforce the CM access control to the objects based on AS.AUTH_CM_STATUS and AS.SECURITY_LEVEL. |
| FDP_ACF.1.2/ CM | The TSF must enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:<br>AS.AUTH_CM_STATUS = TRUE<br>If AS.SECURITY_LEVEL and MAC # 0, the integrity of imported objects is ensured<br>If AS.SECURITY_LEVEL and ENC # 0, the confidentiality of imported objects is ensured |
| FDP_ACF.1.3/ CM | The TSF must explicitly authorise access of the subjects to the objects based on the following additional rules:<br>None |
| FDP_ACF.1.4/ CM | The TSF must explicitly deny access of the subjects to the objects based on the following additional rules:<br>None |
| Dependencies | FDP_ACC.1 Subset access control<br>FMT_MSA.3 Static attribute initialisation |

| **FDP_ETC – Export to outside TSF control** | |
|---|---|
| **FDP_ETC.1 – Export of user data without security attributes** | |
| Subfunctions | |
| FDP_ETC.1.1/ AUT | The TSF must enforce the CM access control when exporting user data, controlled under the SFP(s), outside the TSC. |
| FDP_ETC.1.2/ AUT | The TSF must export the user data without the associated security attributes. |
| FDP_ETC.1.1/ FIREWALL | The TSF must enforce the FIREWALL access control when exporting user data, controlled under the SFP(s), outside the TSC. |
| FDP_ETC.1.2/ FIREWALL | The TSF must export the user data without the associated security attributes. |
| Dependencies | FDP_ACC.1 Subset access control |

| **FDP_ITC – Import from outside TSF control** | | |
|---|---|---|
| **FDP_ITC.1 – Import of user data without security attributes** | | |
| Subfunctions | | |
| FDP_ITC.1.1/ FIRE_1 | The TSF must enforce the FIREWALL access control on the DES key, RSA key, D.PIN value import, applet data and D.BYTECOD when importing user data, controlled under the SFP, from outside the TSC.<br>Note: For bytecode: putfield, <t>astore. | |
| FDP_ITC.1.2/ FIRE_1 | The TSF must ignore any security attributes associated with the user data when imported from outside the TSC. | |
| FDP_ITC.1.3/ FIRE_1 | The TSF must enforce the following rules when importing user data controlled under the SFP from outside the TSC:<br>None | |
| FDP_ITC.1.1/ CM | The TSF must enforce the CM access control when importing user data, controlled under the SFP, from outside the TSC. | |
| FDP_ITC.1.2/ CM | The TSF must ignore any security attributes associated with the user data when imported from outside the TSC. | |
| FDP_ITC.1.3/ CM | The TSF must enforce the following rules when importing user data controlled under the SFP from outside the TSC:<br>None | |
| FDP_ITC.1.1/ APPPRIV | The TSF must enforce the APPPRIV access control when importing user data, controlled under the SFP, from outside the TSC. | |
| FDP_ITC.1.2/ APPPRIV | The TSF must ignore any security attributes associated with the user data when imported from outside the TSC. | |
| FDP_ITC.1.3/ APPPRIV | The TSF must enforce the following rules when importing user data controlled under the SFP from outside the TSC:<br>None | |
| Dependencies | FDP_ACC.1 | Subset access control |
| | FMT_MSA.3 | Static attribute initialisation |
| **FDP_ITC.2 – Import of user data with security attributes** | | |
| Subfunctions | | |
| FDP_ITC.2.1/ CM_CAPFILE | The TSF must enforce the CM access control when importing user data, controlled under the SFP, from outside the TSC. | |
| FDP_ITC.2.2/ CM_CAPFILE | The TSF must use the security attributes associated with the imported user data. | |
| FDP_ITC.2.3/ CM_CAPFILE | The TSF must 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 must 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 must enforce the following rules when importing user data controlled under the SFP from outside the TSC:<br>None<br>Note: This function is applicable when importing the CAP file. | |
| Dependencies | FDP_ACC.1 | Subset access control |
| | FTP_TRP.1 | Trusted path |
| | FPT_TDC.1 | Inter-TSF basic TSF data consistency |

| FDP_RIP – Residual information protection | |
|---|---|
| **FDP_RIP.1 – Subset residual information protection** | |
| Subfunctions | |
| FDP_RIP.1.1/ DEALL_JAVAOBJ | The TSF must ensure that any previous information content of a resource is no longer available to the following objects when the resource is deallocated:<br>• All Java objects<br>The TSF must ensure that there is no access path to the transient objects. |
| FDP_RIP.1.1/ ALL_OBJTRANS | The TSF must ensure that any previous information content of a resource is no longer available to the following objects when the resource is allocated:<br>• Transient objects |
| FDP_RIP.1.1/ DEALL_GARB | The TSF must ensure that any previous information content of a resource s no longer available to the following objects when the resource is deallocated:<br>• Garbage collector |
| FDP_RIP.1.1/ DEALL_CRYPTO | The TSF must ensure that any previous information content of a resource s no longer available to the following objects when the resource is deallocated:<br>• Cryptographic buffers |
| FDP_RIP.1.1/ DEALL_TRANS | The TSF must ensure that any previous information content of a resource s no longer available to the following objects when the resource is deallocated.<br>• Transaction buffer |
| FDP_RIP.1.1/ ALL_APDU | The TSF must ensure that any previous information content of a resource s no longer available to the following objects when the resource is deallocated:<br>• APDU buffer |
| Dependencies | None. |

| FDP_SDI – Stored data integrity | |
|---|---|
| **FDP_SDI.2 – Stored data integrity monitoring and action** | |
| Subfunctions | |
| FDP_SDI.2.1/ EEPROM | The TSF must monitor user data stored within the TSC for EEPROM integrity errors on all objects, based on the following attributes:<br>AS.EEPROM_FLAG |
| FDP_SDI.2.2/ EEPROM | On detection of a data integrity error, the TSF must make the card mute. |
| FDP_SDI.2.1/ AUDITLOG | The TSF must monitor user data stored within the TSC for audit log integrity errors on all objects, based on the following attributes:<br>Audit log checksum |
| FDP_SDI.2.2/ AUDITLOG | On detection of a data integrity error, the TSF must make the card mute. |
| FDP_SDI.2.1/ ROM | The TSF must monitor user data stored within the TSC for ROM integrity errors on all objects, based on the following attributes:<br>ROM code checksum |
| FDP_SDI.2.2/ ROM | On detection of a data integrity error, the TSF must make the card mute. |
| FDP_SDI.2.1/ JAVAOBJ | The TSF must monitor user data stored within the TSC for D.JAVAOBJ integrity errors on all objects, based on the following attributes:<br>D.JAVAOBJ checksum |

| **FDP_SDI.2 – Stored data integrity monitoring and action** | |
|---|---|
| Subfunctions *(cont.)* | |
| FDP_SDI.2.2/ JAVAOBJ | On detection of a data integrity error, the TSF must record the error in the audit log file and notify the error by throwing an exception. |
| FDP_SDI.2.1/ BYTECOD | The TSF must monitor user data stored within the TSC for D.BYTECOD integrity errors on all objects, based on the following attributes: D.BYTECOD checksum |
| FDP_SDI.2.2/ BYTECOD | On detection of a data integrity error, the TSF must record the error in the audit log file, lock the selected applet and notify the error by throwing an exception. |
| Dependencies | None. |

**FDP_TCT – Inter-TSF user data confidentiality transfer protection**

| **FDP_UCT.1 – Basic data exchange confidentiality** | | |
|---|---|---|
| Subfunctions | | |
| FDP_UCT.1.1/ PP | The TSF must enforce the prepersonalisation access control to be able to receive objects in a manner protected from unauthorised disclosure. | |
| FDP_UCT.1.1/ CM | The TSF must enforce the CM access control to be able to receive objects in a manner protected from unauthorised disclosure. | |
| Dependencies | FTP_TRP.1 | Trusted path |
| | FDP_ACC.1 | Subset access control |

**FDP_UIT – Inter-TSF user data integrity transfer protection**

| **FDP_UIT.1 – Data exchange integrity** | | |
|---|---|---|
| Subfunctions | | |
| FDP_UIT.1.1/ SECURE | The TSF must enforce the CM access control to be able to receive user data in a manner protected from modification errors. | |
| FDP_UIT.1.2/ SECURE | On receipt of user data, the TSF must be able to determine whether a modification has occurred. <u>Note</u>: If a modification has been made, it took place during secure channel transmission. | |
| FDP_UIT.1.1/ KEYCHECK | The TSF must enforce the CM access control to be able to receive user data in a manner protected from modification errors. | |
| FDP_UIT.1.2/ KEYCHECK | On receipt of user data, the TSF must be able to determine whether a modification has occurred. <u>Note</u>: If a modification has been made, it took place between generation of the key and its reception. | |
| FDP_UIT.1.1/ DAP | The TSF must enforce the CM access control to be able to receive user data in a manner protected from modification errors. | |
| FDP_UIT.1.2/ DAP | On receipt of user data, the TSF must be able to determine whether a modification has occurred. <u>Note</u>: If a modification has been made, it took place 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

| **FIA_AFL – Authentication failures** | |
|---|---|
| **FIA_AFL.1 – Authentication failure handling** | |
| Subfunctions | |
| FIA_AFL.1.1/CM | The TSF must detect when one unsuccessful authentication attempt occurs related to U.Card_Issuer authentication. |
| FIA_AFL.1.2/CM | When the defined number of unsuccessful authentication attempts has been reached or exceeded, the TSF must slow down the next authentication in accordance with the following function.<br>The waiting time is exponential with a maximum number of unsuccessful authentications of 15. |
| FIA_AFL.1.1/PP | The TSF must detect when three unsuccessful authentication attempts occur related to U.Card_manufacturer authentication. |
| FIA_AFL.1.2/PP | When the defined number of unsuccessful authentication attempts has been reached or exceeded, the TSF must make the card mute. |
| FIA_AFL.1.1/ APP | The TSF must detect when the user-defined maximum number of unsuccessful authentication attempts is reached related to any user authentication using a PIN (1 to 127 for OwnerPIN and 3 to 15 for GlobalPIN). |
| FIA_AFL.1.2/ APP | When the defined number of unsuccessful authentication attempts has been reached or exceeded, the TSF must block the PIN. |
| Dependencies | FIA_UAU.1 Timing of authentication |

| **FIA_ATD – User attribute definition** | |
|---|---|
| **FIA_ATD.1 – User attribute definition** | |
| Subfunctions | |
| FIA_ATD.1.1/ CARD_MANUF | The TSF must maintain the following list of security attributes belonging to individual users:<br>AS.AUTH_MSK_STATUS |
| FIA_ATD.1.1/ CARD_ISSUER | The TSF must maintain the following list of security attributes belonging to individual users:<br>AS.CMLIFECYC<br>AS.CMCONTEXT<br>AS.KEYSET_VERSION<br>AS.KEYSET_VALUE |
| FIA_ATD.1.1/ APPLET | The TSF must maintain the following list of security attributes belonging to individual users:<br>ASG.APPPRIV<br>AS.CURCONTEXT |
| Dependencies | None. |

| **FIA_SOS – Specification of secrets** | |
|---|---|
| **FIA_SOS.1 – Verification of secrets** | |
| Subfunctions | |
| FIA_SOS.1.1/ RSA | The TSF must provide a mechanism to verify that secrets meet the RSA generation key metric (Miller-Rabin method). |
| Dependencies | None. |
| **FIA_SOS.2 – TSF generation of secrets** | |
| Subfunctions | |
| FIA_SOS.2.1/ RANDOM | The TSF must provide a mechanism to generate secrets that meet the random metric (see FIPS PUB 140-1, Security requirements for cryptographic modules). |
| FIA_SOS.2.2/ RANDOM | The TSF must be able to enforce the use of TSF generated secrets for: • Card Manager and Card Issuer authentication • Secure channel management |
| Dependencies | None. |

| **FIA_UAU – User authentication** | | |
|---|---|---|
| **FIA_UAU.1 – Timing of authentication** | | |
| Subfunctions | | |
| FIA_UAU.1.1 | The TSF must 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<br>Get Data<br>Manage Channel<br>Select Applet |
| | Card Manager: | Get Data<br>Initialise Update |
| FIA_UAU.1.2 | The TSF must 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** | | |
| Subfunctions | | |
| FIA_UAU.3.1 | The TSF must prevent use of authentication data that has been forged by any user of the TSF. | |
| FIA_UAU.3.2 | The TSF must prevent use of authentication data that has been copied from any other user of the TSF. | |
| Dependencies | None. | |
| **FIA_UAU.4 – Single-use authentication mechanisms** | | |
| Subfunctions | | |
| FIA_UAU.4.1/ CARD_MANUF | The TSF must prevent reuse of authentication data related to the Card Manufacturer authentication mechanism. | |
| FIA_UAU.4.1/ CARD_ISSUER | The TSF must prevent reuse of authentication data related to the Card Issuer authentication mechanism. | |
| Dependencies | None. | |

| **FIA_UAU.7 – Protected authentication feedback** | |
|---|---|
| Subfunctions | |
| FIA_UAU.7.1/ CARD_MANUF | The TSF must 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 must 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 |

| **FIA_UID – User identification** | |
|---|---|
| **FIA_UID.1 – Timing of identification** | |
| Subfunctions | |
| FIA_UID.1.1 | The TSF must allow the Card Manager to be executed on behalf of the user before the user is identified. |
| FIA_UID.1.2 | The TSF must require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user. <br> Note: This execution is possible only if the CM is the default applet. |
| Dependencies | None. |

| **FIA_USB – User-subject binding** | |
|---|---|
| **FIA_USD.1 – User-subject binding** | |
| Subfunctions | |
| FIA_USB.1.1 | The TSF must associate the appropriate user security attributes with the subjects acting on behalf of that user. |
| Dependencies | FIA_ATD.1 User attribute definition |

## Class FMT: Security Management

| **FMT_MOF – Management of functions in TSF** | |
|---|---|
| **FMT_MOF.1 – Management of security function behaviour** | |
| Subfunctions | |
| FMT_MOF.1.1/ RES_APP | The TSF must restrict the ability to disable the functions of the resident application to R.Prepersonaliser: <br> • GET CHALLENGE <br> • EXTERNAL AUTHENTICATE <br> • LOAD STRUCTURE <br> • INSTALL <br> • LOAD APPLET <br> • GET DATA |
| FMT_MOF.1.1/ TOE | The TSF must restrict the ability to modify the behaviour of the TOE functions: <br> • All functions to R.Prepersonaliser. <br> Note: The evaluated application (TOE) does not contain any additional code. |
| Dependencies | FMT_SMR.1 Security roles |

| **FMT_MSA – Management of security attributes** | | |
|---|---|---|
| **FMT_MSA.1 – Management of security attributes** | | |
| Subfunctions | | |
| FMT_MSA.1.1/ PP | The TSF must enforce the prepersonalisation access control to restrict the ability to modify the AS.MSKEY security attributes to R.Prepersonaliser. | |
| FMT_MSA.1.1/ CM_MOD | The TSF must enforce the CM access control to restrict the ability to modify the AS.KEYSET_VERSION, AS.KEYSET_VALUE, Default selected privilege and AS.CMLIFECYC security attributes to R.Card_Manager. | |
| | Note: Other privileges cannot be modified. | |
| FMT_MSA.1.1/ CM_DEL | The TSF must enforce the CM access control to restrict the ability to delete the AS.KEYSET_VERSION and AS.KEYSET_VALUE security attributes to R.Card_Manager. | |
| FMT_MSA.1.1/ PRIV_MOD | The TSF must enforce the APPPRIV access control to restrict the ability to modify the AS.CMLIFECYC security attributes to R.Applet_privilege. | |
| Dependencies | FDP_ACC.1 | Subset access control |
| | FMT_SMR.1 | Security roles |
| **FMT_MSA.2 – Secure security attributes** | | |
| Subfunctions | | |
| FMT_MSA.2.1 | The TSF must 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** | | |
| Subfunctions | | |
| FMT_MSA.3.1 | The TSF must 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 must allow the R.Personaliser to specify alternative initial values to override the default values when an object or information is created. | |
| | Note: The personaliser can only specify initial values for the keyset. | |
| Dependencies | FMT_MSA.1 | Management of security attributes |
| | FMT_SMR.1 | Security roles |

| **FMT_MTD – Management of TSF data** | | |
|---|---|---|
| **FMT_MTD.1 – Management of TSF data** | | |
| Subfunctions | | |
| FMT_MTD.1.1/CI | The TSF must restrict the ability to modify the AS.CMID, AS.APID, AS.KEYSET_VALUE attributes for DAP verification to R.Personaliser. | |
| FMT_MTD.1.1/ CARDREG | The TSF must restrict the ability to query the AS.APID attribute for R.Card_Manager and R.Use_API. | |
| FMT_MTD.1.1/ AUDIT | The TSF must restrict the ability to delete all information included in the audit log, except integrity errors, for R.Card_Manager. | |
| | Note: This function is available if the state of the Card Manager is not SECURED or LOCKED. | |
| Dependencies | FMT_SMR.1 | Security roles |

| **FMT_MTD.2 – Management of limits on TSF data** | | |
|---|---|---|
| Subfunctions | | |
| FMT_MTD.2.1/ GLBPIN | The TSF must restrict the specification of the limits for D.NB_REMAINTRYGLB to R.Card_Manager. | |
| FMT_MTD.2.2/ GLBPIN | The TSF must take the following actions, if the TSF data has reached or exceeded the indicated limits: | |
| | Block D.GLPIN | |
| FMT_MTD.2.1/ OWNPIN | The TSF must restrict the specification of the limits for D.NB_REMAINTRYOWN to R.Use_API. | |
| FMT_MTD.2.2/ OWNPIN | The TSF must take the following actions, if the TSF data has reached or exceeded the indicated limits: | |
| | Block D.OWNPIN | |
| Dependencies | FMT_MTD.1 | Management of TSF data |
| | FMT_SMR.1 | Security roles |

| **FMT_SMR – Security management roles** | | |
|---|---|---|
| **FMT_SMR.1 – Security roles** | | |
| Subfunctions | | |
| FMT_SMR.1.1 | The TSF must maintain the R.Sign_Load_File roles. | |
| FMT_SMR.1.2 | The TSF must be able to associate users with roles. | |
| Dependencies | FIA_UID.1 | Timing of identification |
| **FMT_SMR.2 – Restrictions on security roles** | | |
| Subfunctions | | |
| FMT_SMR.2.1 | The TSF must maintain the roles defined in the table below. | |
| FMT_SMR.2.2 | The TSF must be able to associate users with roles. | |
| FMT_SMR.2.3 | The TSF must ensure that the conditions defined in the table below are satisfied. | |
| | **Roles** | **Conditions** |
| | R.Prepersonaliser | Successful authentication (of the Card Manufacturer) using the transport key and card still in the prepersonalisation state |
| | R.Personaliser | Successful authentication (of the Card Manufacturer or Card Issuer) using a keyset of the Card Manager, with the CM life cycle phase changing from OP_READY to SECURED |
| | R.Card_Manager | Successful authentication (of the Card Issuer) using a keyset, with the CM life cycle phase changing from OP_READY to SECURED |
| | R.Use_API | Successful identification (of the applet), with the applet life cycle phase after SELECTABLE |
| | R.Applet_privilege | Modification of the CM life cycle, ATR and Global PIN only possible for a privileged applet |
| Dependencies | FIA_UID.1 | Timing of identification |
| Hierarchy | Hierarchical to FMT_SMR.1 | |

## Class FPR: Privacy

| FPR_UNO – Unobservability | | | |
|---|---|---|---|
| **FPR_UNO.1 – Unobservability** | | | |
| Subfunctions | | | |
| FPR_UNO.1.1 | The TSF must 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<br>D.OWNPIN |
| | S.Applet<br>S.CM | Import and use | D.KEY |
| | S.Applet | Comparison of two byte arrays | D.ARRAY |
| Dependencies | None. | | |

## Class FPT: Protection of the TOE Security Functions

| FPT_FLS – Fail secure | |
|---|---|
| **FPT_FLS.1 – Failure with preservation of secure state** | |
| Subfunctions | |
| FPT_FLS.1.1 | The TSF must preserve a secure state when the following types of failures occur:<br>• Invalid reference exception<br>• Code or data integrity failure<br>• Power loss while processing |
| Dependencies | ADV_SPM.1 Informal TOE security policy model |

| FPT_RCV – Trusted recovery | |
|---|---|
| **FPT_RCV.4 – Function recovery** | |
| Subfunctions | |
| FPT_RCV.4.1 | The TSF must ensure that an anti-tearing failure scenario has the property that the SF either completes successfully, or for the indicated failure scenario, recovers to a consistent and secure state. |
| Dependencies | ADV_SPM.1 Informal TOE security policy model |

| FPT_RVM – Reference mediation | |
|---|---|
| **FPT_RVM.1 – Non-bypassability of the TSP** | |
| Subfunctions | |
| FPT_RVM.1.1 | The TSF must ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed. |
| Dependencies | None |

| **FPT_SEP – Domain separation** | |
|---|---|
| **FPT_SEP.1 – TSF domain separation** | |
| Subfunctions | |
| FPT_SEP.1.1 | The TSF must maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects. |
| FPT_SEP.1.2 | The TSF must enforce separation between the security domains of subjects in the TSC. <br> Note: There is a separation of the security domain between the Card Manager and the applets (the CM is written in Java) and between transients of different logical channels. |
| Dependencies | None |

| **FPT_TDC – Inter-TSF TSF data consistency** | |
|---|---|
| **FPT_TDC.1 – Inter-TSF basic TSF data consistency** | |
| Subfunctions | |
| FPT_TDC.1.1 | The TSF must 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 must use the PUT KEY data format when interpreting the TSF data from another trusted IT product: key generator. |
| Dependencies | None |

| **FPT_TST – TSF self test** | |
|---|---|
| **FPT_TST.1 – TSF testing** | |
| Subfunctions | |
| FPT_TST.1.1/ RESET | The TSF must 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 must provide authorised users with the capability to verify the integrity of the TSF data. |
| FPT_TST.1.3/ RESET | The TSF must provide authorised users with the capability to verify the integrity of stored TSF executable code. |
| FPT_TST.1.1/ TOE | The TSF must run a suite of self tests while executing the TOE to demonstrate the correct operation of the TSF. |
| FPT_TST.1.2/ TOE | The TSF must provide authorised users with the capability to verify the integrity of the TSF data. |
| FPT_TST.1.3/ TOE | The TSF must provide authorised users with the capability to verify the integrity of stored TSF executable code. |
| Dependencies | FPT_AMT.1 Abstract machine testing |

## Class FRU: Resource Utilisation

| **FRU_RSA – Resource allocation** | |
|---|---|
| **FRU_RSA.1 – Maximum quotas** | |
| Subfunctions | |
| FRU_RSA.1.1 | The TSF must enforce maximum quotas of the following resources: EEPROM that S.Applet can use throughout the applet life time. |
| Dependencies | None. |

## Class FTA: TOE Access

| FTA_LSA – Limitation on scope of selectable attributes | |
|---|---|
| **FTA_LSA.1 – Limitation on scope of selectable attributes** | |
| Subfunctions | |
| FTA_LSA.1.1/ SECURE | The TSF must restrict the scope of the AS.SESSION_KEY session security attributes based on:<br>AS.KEYSET_VERSION<br>AS.KEYSET_VALUE<br>AS.CURCONTEXT<br>AS.LOGIC_CHANNEL_NB |
| Dependencies | None. |

## Class FTP: Trusted Path/Channels

| FTP_TRP – Trusted path | |
|---|---|
| **FTP_TRP.1 – Trusted path** | |
| Subfunctions | |
| FTP_TRP.1.1 | The TSF must provide a communication path between itself and local users that:<br>• Is logically distinct from other communication paths<br>• Provides assured identification of its end points and protection of the communicated data from modification or disclosure |
| FTP_TRP.1.2 | The TSF must permit local users to initiate communication via the trusted path. |
| FTP_TRP.1.3 | The TSF must require the use of the trusted path for loading of keysets and D.GLPIN.<br>Note: The applet may use the trusted path to load data and the bytecode, or to check the integrity of applet data, keys and privileges. |
| Dependencies | None. |

## TOE Security Assurance Requirements

For this evaluation, TOE security assurance requirements are high and the assurance level is EAL 4, augmented with additional assurance components:

- ADV_IMP.2
- ALC_DVS.2
- AVA_VLA.4

### ADV_IMP.2 – Implementation of the TSF

The TSF is implemented as follows.

| Element | Implementation |
|---|---|
| Developer action elements | The developer must provide the implementation representation for the entire TOE security functions. |
| Contents and presentation of evidence elements | The implementation representation must:<br>• Unambiguously define the TOE security functions with such a level of details that the TOE security functions can be generated without further design decisions<br>• Be internally consistent<br>• Describe the relationships between all portions of the implementation |
| Evaluator action elements | The evaluator must:<br>• Confirm that the information provided meets all requirements for content and presentation of evidence<br>• Determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements |

### ALC_DVS.2 – Sufficiency of Security Measures

The security measures are implemented as follows.

| Element | Implementation |
|---|---|
| Developer action elements | The developer must produce development security documentation. |
| Contents and presentation of evidence elements | The development security documentation:<br>• Describe all the physical, procedural, personnel and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment<br>• Provide evidence that these security measures are followed during the development and maintenance of the TOE<br>The evidence must justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE. |
| Evaluator action elements | The evaluator must confirm that the:<br>• Information provided meets all requirements for content and presentation of evidence<br>• Security measures are being applied |

## AVA_VLA.4 – Highly resistant

High resistance is implemented as follows.

| Element | Implementation |
|---|---|
| Developer action elements | The developer must:<br>• Perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP<br>• Document the disposition of identified vulnerabilities |
| Contents and presentation of evidence elements | The document must show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.<br>The documentation must justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.<br>The evidence must show that the search for vulnerabilities is systematic.<br>The analysis documentation must provide a justification that the analysis completely addresses the TOE deliverables. |
| Evaluator action elements | The evaluator must:<br>• Confirm that the information provided meets all requirements for content and presentation of evidence<br>• Conduct penetration testing, building on the developer vulnerability analysis, to ensure that identified vulnerabilities have been addressed<br>• Perform an independent vulnerability analysis<br>• Perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment<br>• Determine that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential |

# IT Environment Security Requirements

| **FAU_ARP.1 – Security alarms** | | |
|---|---|---|
| Subfunctions | | |
| FAU_ARP.1.1/ PHIL | The TSF must take an action among the following list on detection of a potential security violation:<br>• Reset the card | |
| Dependencies | FAU_SAA.1 | Potential violation analysis |
| **FAU_SAA.1 – Potential violation analysis** | | |
| Subfunctions | | |
| FAU_SAA.1.1/ PHIL | The TSF must be able to apply a set of rules when monitoring the audited events and based on these rules indicate a potential violation of the TSP. | |
| FAU_SAA.1.2/ PHIL | The TSF must enforce the following rules when monitoring audited events:<br>• Low frequency of clock input<br>• High frequency of clock input<br>• Low voltage power supply<br>• High voltage power supply<br>• Low temperature<br>• High temperature<br>• High voltage for the write process to the EEPROM<br>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** | | |
| Subfunctions | | |
| FCS_COP.1.1/ DES_PHIL | The TSF must perform encryption and decryption in accordance with a specified Triple DES cryptographic algorithm and cryptographic key sizes of 112 bits that meet the following standard:<br>• FIPS PUB 46-3 (ANSI X3.92), KEYING option 2 | |
| FCS_COP.1.1/ RSA_PHIL | The TSF must raise an integer to a power modulo in accordance with a specified RSA cryptographic algorithm and cryptographic key sizes of 1024 bits that meet the following standard:<br>• ISO / IEC 9796-1, Annex A, Sections A.4 and A.5 and Annex C | |
| FCS_COP.1.1/ RND | The TSF must perform random number generation in accordance with a specified cryptographic algorithm (no algorithm) and cryptographic key sizes (no key) that meet the following standard:<br>• FIPS 184.2<br>Note: An entropy of at least 7 bits is required 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** | |
|---|---|
| Subfunctions | |
| FDP_RIP.1.1/ PHIL | The TSF must ensure that any previous information content of a resource is no longer available to the following objects when the resource is deallocated: <br>• DES cryptoprocessor registers <br>Note: The DES cryptoprocessor registers are deallocated by the IC. |
| Dependencies | None |
| **FPR_UNO.1 – Unobservability** | |
| Subfunctions | |
| FPR_UNO.1.1/ PHIL | The TSF must ensure that any users are unable to observe the operations using the CPU, DES coprocessor or FAMEX coprocessor on the data stored in EEPROM or RAM or generated by the random number generator by the TOE. |
| Dependencies | None. |
| **FPT_PHP.2 – Notification of physical attack** | |
| Subfunctions | |
| FPT_PHP.2.1/ PHIL | The TSF must provide unambiguous detection of physical tampering that might compromise the TSF. |
| FPT_PHP.2.2/ PHIL | The TSF must provide the capability to determine whether physical tampering with the TSF devices or elements has occurred. |
| FPT_PHP.2.3/ PHIL | For the power supply block, internal frequency generation and chip temperature, the TSF must monitor the devices and elements and notify the platform by setting the sensor reset bit when physical tampering with the TSF devices or elements has occurred. |
| Hierarchy | Hierarchical to FPT_PHP.1. |
| Dependencies | FMT_MOF.1 Management of security functions behaviour |
| **FPT_PHP.3 – Resistance to physical attack** | |
| Subfunctions | |
| FPT_PHP.3.1/ PHIL | The TSF must resist changing operational conditions at all times: <br>• Frequency of the external clock <br>• Power supply <br>• Temperature <br>It must respond automatically so that that the TSP is not violated. |
| Dependencies | None. |

# Chapter 6 – TOE Summary Specification

## Chapter Overview

This chapter reviews the:

- TOE security functions and their strength levels
- Assurance measures

## TOE Security Functions

### F1 – Exceptions Management

A potential attack analysis automatically throws an exception. This stops the current process.

It notifies the error by the following actions.

- It writes it in the audit log if its type can be analysed as a security violation.
- It locks the applet that caused the security exception.
- It executes a procedure to process exceptions written by the applet developer (see *Exception Handling* in the Java Card 2.1.1 – Virtual Machine Specifications document).
- Otherwise, it outputs an error status.

### F2 – Integrity of the CAP File

The CAP file must be signed. This signature is checked by the TOE when loading the CAP file. Loading is denied if the CAP file integrity check fails.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

### 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.

The following services are also available for applets.

| Service | Description |
|---|---|
| F3 – Integrity of 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.<br><br>This function uses a probabilistic mechanism and is consequently SOF –HIGH. |
| F4 – Confidentiality of code and data during loading (secure channel) | The confidential data is encrypted using a DES algorithm.<br><br>This function uses a probabilistic mechanism and is consequently SOF –HIGH. |

## 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.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## 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 and garbage collector
- By erasing the EEPROM while deallocating
- By erasing the 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 number of remaining tries for the PIN is decremented before the comparison to avoid attack by tearing.

## F8 – Ratification

This TSF:

- Manages the number of remaining tries and the PIN validation flag
- Slows down the Card Issuer authentication timing
- Records unsuccessful authentication of the Card Manufacturer

## F11 – EEPROM Quota

The card issuer can determine a limit for non-volatile data space per applet for its entire life.

## F12 – Sensitive Data Integrity

The integrity of the keys, PIN and sensitive applet data is checked. This operation is protected against disclosure of manipulated data.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F13 – Objects Integrity

Before use, the integrity of the Java objects, Card Registry objects (AID privileges), keyset versions and audit log files is checked.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F14 – Package Integrity

Before executing an applet, its package integrity is checked.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F15 – ROM Code Integrity

The ROM code integrity is checked:

- At each reset (partial check)
- During manufacturer authentication via an EXTERNAL AUTHENTICATE command (full check)

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F17 – Internal Role Management: Card Registry

The internal roles for applets is managed using privileges stored in the card registry.

## F18 – Startup Coherence

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

- Inconsistency of Card Manager life cycle
- Bad result for test of integrity of EEPROM
- Loss of integrity of audit log file (F13)
- Loss of integrity of ROM code (F15)
- Number of records in audit log file equals or exceeds the limits (F1, F12, and F13)
- Loss of integrity of optional code area
- Blocked random generator
- Incorrect operation of the cryptographic module
- Loss of integrity of FAT (check FAT)
- Throw of an exception

## F19 – Audit Log File Assessments

This function tests whether the number of records in the audit log file equals or exceeds the limits (F1, F12, and F13), and mutes the card if the test fails.

## F20 – Record of Security Information in Audit Log

If an exception, the type of which can be analysed as a security violation, occurs, its type and the reference of the current applet are recorded in the audit log file.

## F24 – Card Manufacturer Authentication

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

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F26 – Resident Application Dispatcher

During the prepersonalisation phase, this function determines whether manufacturer authentication is required for each command.

## F28 – Key Integrity from its Generation: KeyCheck Value

This function verifies the key integrity using a key check value algorithm as defined in the Visa Open Platform Card Specification, Chapter 9.3.4.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F29 – Card Manager Dispatcher

While the Card Manager is selected, this function determines whether card issuer authentication is required for each command.

If a secure channel is opened, this function determines whether secure messaging is required for each command, depending on the Card Manager life cycle.

## F30 – Read the Audit Log File

This function reads the audit log file and exports it in a comprehensive form. It requires successful authentication of the card issuer.

## F31 – Secret Generation

| Function | Description |
|---|---|
| Random generation | This function based on the IC random number generator generates a random number. |
| Session key generation | To ensure a high level of secure communication for each session involving the Card Manager, this function generates a session key. DES session keys are used in support of secure channel operations. |

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F32 – RSA Key Generation

The TOE provides applets with a service for RSA key generation. This service uses the IC RSA coprocessor.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F33 – DES Algorithm

The TOE implements this function based on DES hardware.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## F34 – RSA Algorithm

The TOE implements this function accelerated by the FAMEX coprocessor.

This function uses a probabilistic mechanism and is consequently SOF –HIGH.

## Firewall

| Function | Description |
|---|---|
| F36 – Applet isolation | The TOE supports isolation of contexts and applets.<br>Isolation means that one applet cannot access the fields or objects of an applet in another context, unless the other applet explicitly provides an interface for access.<br>It implements applet isolation as defined in the Java Card 2.1.1 – Virtual Machine Specifications, Section 7, and Java Card 2.1.1 – JCRE, Section 6. |
| F37 – JCRE privileges | Given that the JCRE context is the *system* context, it 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. |
| F38 – JCRE entry point | The JCRE entry points are objects owned by the JCRE context, but they have been flagged as containing entry point methods.<br>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.<br>In the TOE the JCRE entry points are the APDU object and card runtime exceptions.<br>If the object is a JCRE entry point, the usual rules for applet isolation (F36) are changed to permit general access under the control of the current context. |
| F39 – Global arrays | Global arrays are owned by the JCRE context, but can be accessed from any context.<br>In the TOE, the only global array is the APDU buffer.<br>If the object is a global array, the usual rules for applet isolation (F36) are changed to permit general access under the control of the current context. |
| F40 – Shareable interface | The shareable interface is used to identify all shared objects. Any object that needs to be shared through the applet firewall must directly or indirectly implement this interface. Only those methods specified in a shareable interface are available through the firewall.<br>If the applet calls *getPreviousContextAID* from a method that may be accessed either from within the applet itself or via a shareable interface from an external applet, it identifies the caller identity. |

## F41 – Keyset Version Management

The loading of a keyset can update, delete or add a former keyset.

### F43 – DES Key Access

Access to the DES key is in accordance with the standards defined in the Java Card 2.1.1 – Application Programming Interfaces, Open Platform Card Specification and Visa Open Platform Card Implementation Specification documents. This access is protected against key disclosure.

### F44 – RSA Key Access

Access to the RSA key is in accordance with the standards defined in the Java Card 2.1.1 – Application Programming Interfaces document. This access is protected against key disclosure.

### F45 –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 high and the scale of evaluation levels constructed using these components are EAL 4 augmented by the following additional assurance components:

- ADV_IMP.2
- ALC_DVS.2
- AVA_VLA.4

These components give an augmented confidence in security function efficiency.

### Configuration Management

The configuration management tool (PVCS and its procedures) used by the developers meets the following requirements:

ACM_AUT.1 Partial CM automation

ACM_CAP.4 Generation support and acceptance procedures

ACM_SCP.2 Problem tracking CM coverage

### Delivery and Operation

TOE and its associated documentation are given to users in compliance with the following procedures:

ADO_DEL.2 Detection of modification

ADO_IGS.1 Installation, generation and startup procedures

## Development

TOE development documentation be drawn up to include:

- Functional specifications
- High and Low level design
- Implementation of the entire TSF
- TOE security policy model
- At the end of each document listed, the correspondence between all adjacent pairs of TSF representation

This documentation is sufficient to meet Assurance Class ADV:

ADV_FSP.2 Fully defined external interfaces

ADV_HLD.2 Security enforcing high-level design

ADV_IMP.2 Implementation of the TSF

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ADV_SPM.1 Informal TOE security policy model

## Guidance Documents

The expected information required to meet this requirement are present in the following documentation:

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

## Life Cycle Support

OCS procedures specify the method enabling the integrity and confidentiality of the TOE and its documentation to be guaranteed during the development phase.

The life cycle model used for TOE development is the *V cycle*. This cycle is clearly defined in a specific procedure. Maintenance is not applicable to this TOE.

The following languages are used to develop the TOE:

- Java
- C
- Assembly 8051

The compiler is KEIL C51. Documentation is available to the evaluator. This documentation is sufficient to meet the following requirements:

ALC_DVS.2 Sufficiency of security measures

ALC_LCD.1 Developer defined life-cycle model

ALC_TAT.1 Well-defined development tools

## Tests

The *Doc 170* will contain all test specifications and associated results (expected and obtained). Tests concern coverage and its analysis (test of high level design and all functional tests).

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

Doc 170 and the TOE are sufficient to meet the following requirements:

ATE_COV.2 Analysis of coverage

ATE_DPT.1 Testing: high-level design

ATE_FUN.1 Functional testing

ATE_IND.2 Independent testing – sample

## Vulnerability Assessment

The following requirements are met by the documentation presenting an analysis of the guidance documentation, the strength of the TOE security functions and the analysis of the TOE identified vulnerabilities:

AVA_MSU.2 Validation of Analysis

AVA_SOF.1 Strength of TOE security function evaluation

AVA_VLA.4 Highly resistant

# GLOSSARY

### Active life/phase

Period with active security functions and no active code.

### AID

Applet Identifier.

### APDU

Application Protocol Data Unit.

### API

Application Programmer Interface.

### Applet

Application that can be loaded and executed with the environment of the Java Card platform.

### BIOS

Basic Input/Output System.

### Card Manager

Main entity representing the issuer and supervising all services available on the card.

### CC

Common Criteria.

### CM

Card Manager.

### CPLC

Card Production Life Cycle.

### DAP

Data Authentication Pattern.

### DES

*Data Encryption Standard* cryptographic module.

### EAL

Evaluation Assurance Level.

### EEPROM

Electrically Erasable and Programmable Read Only Memory.

### ES

Embedded Software.

### FAMEX

Coprocessor for public key cryptographic calculations.

### FAT

File Allocation Table.

### IC

Integrated Circuit.

### IT

Information Technology.

### JCP

Java Card Platform.

### JCRE

Java Card Runtime Environment.

### OSP

Organisational Security Policy.

### PP

Protection Profile.

## RNG

Random Number Generation.

## ROM

Read Only Memory.

## RSA

*Rivest, Shamir, Adleman* cryptographic module.

## Security Domain

Entity representing a supplier, managing the keys and providing cryptographic services for its applets. A Security Domain is the on-card representative of an application provider. It is a special key management application that may provide cryptographic services for all the applications owned by a particular application provider.

## SF

Security Function.

## SFP

Security Function Policy.

## SHA-1

*Secure hash standard* cryptographic module.

## ST

Security Target.

## TOE

Target of Evaluation.

## TSC

TSF Scope of Control.

## TSF

TOE Security Functions.

### TSP

TOE Security Policy.

### VM

Virtual Machine.

### VOP

Visa Open Platform.