Written by: Boutheina Chetali
Quang-Huy Nguyen

Formal Assurance on the JavaCard Virtual Machine embedded in Usimera Protect

Security Target

PUBLIC VERSION

TABLE OF CONTENTS

Figures

1. ST INTRODUCTION

1.1 ST IDENTIFICATION

Title:	Java Card Virtual Machine embedded on Usimera Protect (SIMEOS JVM)
Version:	1.3 issued September, 2007
Registration:	Ref. D1031392
Origin:	GEMALTO
Commercial name:	USIMERA PROTECT
Product ref.	T1000230 Usimera Protect 128K crypto on Infineon

The **SIMEOSExtended TOE** is composed of a subset of the **Java Card System** of the Usimera Protect software, made of the Java Card Virtual Machine 2.2.1, the Java Card Runtime Environment 2.2.1 and the Java Card API 2.2.1.

The SIMEOS TOE is composed with:

Component	Version number	Supplier
Usimera Protect software	2.1	Gemalto
PSL	0.50.23	Infineon
Chip SLE88CFX4000P	m8830b17	Infineon

1.2 ST OVERVIEW

This Security Target "JAVA CARD VIRTUAL MACHINE ON USIMERA PROTECT: SECURITY TARGET" defines the security requirements for the extension of the evaluation of "USIMERA PROTECT" product already performed according to SIMEOS security target [ST_SIMEOS].

It covers additional evaluation work to give a formal assurance of the correctness of the implementation of the security functions of the **Java Card Virtual Machine (JCVM) embedded in the product "USIMERA PROTECT".**

The product "USIMERA PROTECT" is based on secure Operating System addressing Mobile Communication requirements. It is a SIM/USIM card based on ETSI Release 5 and Release 6 standards both for SCP and 3GPP. The architecture of the product is an UICC with first level applications, such as SIM – addressing 2G network authentication features and USIM – addressing 3G network authentication features.

The product is compliant with Java Card 2.2.1 and Global Platform 2.1.1, enabling value added services to be installed on card.

The **Usimera Protect Security Target** [ST_SIMEOS] focused on the following security functions:

- Hardware Tamper Resistance**:**
  - This is the chip security layer that meets PP SSVG [PP/BSI-0002].
- Secure operation of the Java Card Virtual Machine (meet PP JCS):

- This is the Java Card Virtual Machine and Operating System that meets [PP/JCS].

  The security requirements used from [PP/JCS], Javacard 2.2. standard configuration, are those from the groups: *CoreG* (§5.1.1), *InstG* (§5.1.2), *ADELG* (§5.1.4), *RMIG* (§5.1.5), *LCG* (§5.1.6), *ODELG* (§5.1.7) and *CarG* (§5.1.8).

  The IT environment security requirements are the ones defined in the groups *BCVG* (§5.1.3). Due to the extension of the TOE scope compared to its PP definition, the groups *SCPG* (§5.1.9) and *CMGRG* (§5.1.10) became TOE security requirements.

- **2G/3G network authentication:**
  - The card proposes secure authentication through 2G/3G network so that operators' keys and algorithms shall not be compromised.

The JCVM on Usimera Protect Security Target focuses on the following security functions:

- **Secure operation of the Java Card Virtual Machine that meets the PP JCS:**

  The security requirements used from [PP/JCS], Javacard 2.2. standard configuration, are those from the groups: *CoreG* (§5.1.1), *InstG* (§5.1.2), *LCG* (§5.1.6). However, some security requirements from those groups do not apply for the TOE but for the IT environment. They appear consequently in the SFRs for the IT environment.

  The IT environment security requirements are the ones defined in the groups *BCVG* (§5.1.3), *SCPG* (§5.1.9) and *CMGRG* (§5.1.10).

  This security target adds *ADELG* (§5.1.4), *ODELG* (§5.1.7), *RMIG* (§5.1.5), and *CarG* (§5.1.8) to the IT environment as those security requirements are not in the configuration of the product defined by this security target. Some of them are ensured by the underlying smart card platform and not by the virtual machine.

This Security Target restricts the Usimera Protect Security Target [ST_SIMEOS] to a subset of the Java Card System (JCS).

1.3 CC CONFORMANCE

The compliance is assumed with CC version V2.2 (ISO 15408) ([CC-1], [CC-2], [CC-3]) with Interpretations of January 2004.

This product is a Java Card compliant to the Protection Profile Java Card 2.2 Standard from SUN [PP/JCS] and uses a certified chip conformant to the Protection Profile SSVG (also known as PP/BSI-0002) from Eurosmart, in addition with mobile communication specific functional security requirements, providing the necessary security so that value added applications can be safely loaded and executed on card.

Therefore

- The Simeos ST is conformant to [PP/JCS] Protection Profile and IC is conformant to [PP/BSI-0002] Protection Profile.
- The Simeos Extended ST does not claim any protection profile but it is "based" on the [PP/JCS] in the sense that the TOE is a subset of TOE described in the JC2.2 standard configuration and all the Java Card security features are as described in that Protection Profile.

This ST is CC V2.2 conformant with Part 2.

This ST is CC V2.2 conformant with Part 3 conformant and EAL4 augmented as stated in [PP/JCS], [PP/BSI-0002].

The assurance level for this product is EAL4 augmented by:

- ADV_FSP.4 (Development- Formal functional specification)
- ADV_SPM.3 (Development- Formal TOE security policy model)
- ADV_HLD.5 (Development- Formal high-level design)
- ADV_LLD.2 (Development-Semi-Formal low-level design)
- ADV_INT.3 (Development- Minimisation of complexity)
- ADV_IMP.3 (Development – Structured Implementation of the TSF)
- ADV_RCR.3 (Development- Formal correspondence demonstration)

For a subset of the Java Card System embedded on the product.

Other augmentations are already covered by SIMEOS evaluation:

- ADV_IMP.2 (Development – Implementation of the TSF)
- ALC_DVS.2 (Sufficiency of security measures)
- AVA_VLA.4 (Vulnerability Assessment – Analyse, Highly resistant)
- AVA_MSU.3 (Vulnerability Assessment – Analysis and testing for insecure states).

The augmentations may also be found in section §5.2.

The strength level for the TOE security functional requirements is “SOF high” (Strength Of Functions high).

1.4 REFERENCES

1.4.1 External References [ER]

Reference	Title
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model January 2004, Version 2.2, Revision 256, CCIMB-2004-01-001.
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements January 2004, Version 2.2, CCIMB-2004-01-002.
[CC-3]	Common Criteria for Information Technology security Evaluation Part 3: Security Assurance Requirements January 2004, Version 2.2, Revision 256, CCIMB-2004-01-003.
[CEM]	Common Methodology for Information Technology Security Evaluation. Evaluation Methodology, January 2004, version 2.2, revision 256, CCIMB-2004-01-004.
[PP/JCS]	Java Card System Protection Profile Version 1.0b issued August 2003, standard 2.2 configuration, SUN document
[PP/BSI-0002]	Smart Card IC Platform Protection Profile, version 1.0, registered by BSI in 2001 under PP-BSI-0002, Eurosmart document (SSVG Protection Profile).
[ALGO/CWA]	CEN/ISSS WS/E-Sign Expert Group F: Algorithms and Parameters for Secure Electronic Signatures, V.2.1 Oct 19th 2001
[ST/INFINEON]	Infineon: Security Target of SLE88CFX4000P Integrated Circuit v0.3
[LDR/INFINEON]	SLE88 Flash Loader Security Concept, Infineon Technologies AG, Rev. 0.2, 2004-02-24 (The release version 1.0 is not available at this time).
[ALGO/DES]	FIPS 46-3: DES Encryption Standard (DES and TDES). National Institute of Standards and Technology
[ALGO/AES]	FIPS 197: AES Encryption Standard. National Institute of Standards and Technology.
[ALGO/SHA1]	FIPS PUB 180-1, Secure Hash Standard. National Institute of Standards and Technology 1993 May 11.
[ISO 7816-4]	Identification cards - Integrated circuit(s) cards with contacts, Part 4: Interindustry commands for interchange
[ISO 7816-6]	Identification cards - Integrated circuit(s) cards with contacts, Part 6: Interindustry data elements
[ISO 7816-9]	Identification cards - Integrated circuit(s) cards with contacts, Part 9: Additional Inter industry commands and security attributes.

Reference	Title
[TS 11.11]	Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface, (Release 1999) – v 8.8.0 (2002-09)
[TS 23.040]	Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS); Point to Point (PP) – v 6.0.1 (2002-09)
[TS 23.048]	Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Security Mechanisms for the (U)SIM application toolkit; Stage 2 – v 5.6.0 (2003-03).
[TS 31.102]	3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the USIM Application (Release 5) – v 5.3.0 (2002-12).
[TS 31.111]	3rd Generation Partnership Project; Technical Specification Group Terminals; USIM Application Toolkit (Release 5) – v5.3.0 (2002-12).
[TS 31.115]	3rd Generation Partnership Project; Technical Specification Group Terminals; Secured packet structure for (U)SIM Toolkit applications (Release 6) – v 6.1.0 (2002-09)
[TS 31.116]	3rd Generation Partnership Project; Technical Specification Group Terminals; Remote APDU Structure for (U)SIM Toolkit applications (Release 6) – v 6.1.0 (2002-09)
[TS 31.900]	3rd Generation Partnership Project; Technical Specification Group Terminals; SIM/USIM Internal and External Interworking Aspects – v 5.1.0 (2002-06).
[TS 33.102]	3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (Release 5) – v 5.0.0 (2002-06).
[TS 43.019]	3rd Generation Partnership Project; Technical Specification Group Terminals; Subscriber Identity Module Application Programming Interface. SIM API for Java Card™ stage 2 – v 5.4.0 (2002-09).
[TS 51.011]	3rd Generation Partnership Project; Technical Specification Group Terminals; Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (Release 5) – v 5.0.0 (2001-12).
[TS 51.013]	3rd Generation Partnership Project; Technical Specification Group Terminals; Test specification for SIM API for Java Card™ (Release 4) – v 2.0.0 (2002-09).
[TS 102.221]	Smart cards; UICC-Terminal interface; Physical and logical characteristics (Release 5) – v 5.3.0 (2003-02).
[TS 102.222]	Integrated Circuit Cards (ICC); Administrative commands for telecommunications applications (Release 6) – v 6.0.0 (2003-02).
[TS 102.223]	Smart cards; Card Application Toolkit (CAT) (Release 5) – v5.0.0 (2002-07).
[TS 102.224]	Smart cards; Security mechanisms for UICC based Applications - Functional requirements (Release 6) – v 6.0.0 (2002-04).
[TS 102.225]	Smart cards; Secured packet structure for UICC based applications (Release 6) – v 6.0.0 (2002-04).

Reference	Title
[TS 102.226]	Smart Cards; Remote APDU structure for UICC based applications (Release 6) – v 6.3.0 (2003-02).
[TS 102.240]	Smart Cards; UICC Application Programming Interface and Loader Requirements; Service description; (Release 6) – v 6.0.0 (2002-07).
[JCAPI22]	Java Card™ APIs specification version 2.2, Sun Microsystems, Inc.
[JCRE22]	Java Card™ 2.2 Runtime Environment Specification version 2.2, Sun Microsystems, Inc
[JCVM22]	Java Card™ Virtual Machine Specification version 2.2, Sun Microsystems, Inc
[JCAPI221]	Java Card™ APIs specification version 2.2.1, Sun Microsystems, Inc, June 23, 2003.
[JCAPN221]	Application Programming Notes for the Java Card™ Platform, Sun Microsystems, Inc, version 2.2.1, October 2003.
[JCRE221]	Java Card™ Runtime Environment Specification version 2.2.1, Sun Microsystems, Inc, 2003.
[JCVM221]	Java Card™ Virtual Machine Specification version 2.2.1, Sun Microsystems, Inc, 2003.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[GP]	Global Platform. Card Specification – v2.1.1, March 2003.
[VOP]	Visa Open Platform Card Implementation Requirements Configuration 3 – v2.0

1.4.2 Internal References [IR]

[STF_SIMEOS]	**SIMEOS JVM: Security Target** D1031392
[SPMFTSP_SIMEOS]	**SIMEOS JVM formal Model of the Java card Virtual Machine** (and the informal descriptive document) D1036587 (D1034212_1)
[SPMFi_SIMEOS]	**SIMEOS JVM Formal Model of the Firewall security policy** D1034212_2
[SPMFa_SIMEOS]	**SIMEOS JVM Formal Model of the APIs** (and the informal descriptive document) D1035639 (D1034212_4)
[SPMFb_SIMEOS]	**SIMEOS JVM formal Model of the bytecode verifier** D1034212_3
[SPM(ST_TSP)SIMEOS]	**SIMEOS JVM Correspondence TSS and TSP** D1035636
[RCR(TSS-FSP)l_SIMEOS]	**SIMEOS JVM Representation correspondence between the TSS and the FSP Model for the linker** D1035629
[RCR(TSS-FSP)i_SIMEOS]	**SIMEOS JVM formal correspondence between the TSS and FSP Model for the Interpreter** D1035629
[RCR(TSS-FSP)a_SIMEOS]	**SIMEOS JVM formal correspondence between the TSS and FSP Model for the APIs** D1035629
[RCR(FSP-HLD)Fl_SIMEOS]	**SIMEOS JVM Formal correspondence FSP-HLD for the Linker** (and the informal descriptive document) D1024208 (D1035641)
[RCR(FSP-HLD)Fa_SIMEOS]	**SIMEOS JVM Formal correspondence FSP-HLD for the API** (and the informal descriptive document) D1035639
[LLDF_SIMEOS]	**SIMEOS JVM Formal Low-level Design** D1034209
[LLDFi_SIMEOS]	**SIMEOS JVM Formal Low-level Design of the Interpreter** (and the informal descriptive document) D1034209 (D1034209_2)
[LLDFl_SIMEOS]	**SIMEOS JVM Formal Low-level Design of the Linker** (and the informal descriptive document) D1034209 (D1034209_1)
[LLDFa_SIMEOS]	**SIMEOS JVM Formal Low-level Design of the API** D1035639
[RCR(HLD-LLD)Fi_SIMEOS]	**SIMEOS JVM Formal correspondence HLD-LLD for the Interpreter** (and the informal descriptive document) D1034209 (D1035643)
[RCR(HLD-LLD)Fa_SIMEOS]	**SIMEOS JVM Formal correspondence HLD-LLD for the API** D1035639

[IMPF_SIMEOS]	**SIMEOS JVM Implementation representation** D1034209
[INTF_SIMEOS]	**SIMEOS JVM Modularity** D1034211
[ALCF_SIMEOS]	**SIMEOS JVM Documentation of development tools** D1034214

The references reused from SIMEOS Security target are:

Reference	Title
[DEF_SIMEOS]	**SIMEOS Definitions and Acronyms** MRD07DEF033000
[FSP_SIMEOS]	**SIMEOS Functional Specification** D1019561 (FSP_D1019561)
[AUT_SIMEOS]	**SIMEOS Partial CM automation** D1021041 (AUT_D1021041)
[CAP_SIMEOS]	**SIMEOS Generation, support and acceptance procedure** D1021043 (AUT_D1021043)
[SCP_SIMEOS]	**SIMEOS Problem tracking CM coverage** D1021044 (SCP_D1021044)
[DEL_SIMEOS]	**SIMEOS Detection of modification** D1021045 (DEL_D1021045)
[IGS_SIMEOS]	**SIMEOS Installation, Generation and Start Up Procedures** D1021046 (IGS_D1021046)
[ADM_SIMEOS]	**SIMEOS Administrator Guidance** D1019545 (ADM_D1019545)
[USR_SIMEOS]	**SIMEOS User Guidance** D1019543 (USR_D1019543)
[HLD_SIMEOS]	**SIMEOS High-level Design (overview document)** D1019563 (HLD_D1019563)
[LLD_SIMEOS]	**SIMEOS Low-level Design (overview document)** D1019566 (LLD_D1019566)
[IMP_SIMEOS]	**SIMEOS Implementation representation** D1019576 (IMP_D1019576)
[SPM_SIMEOS]	**SIMEOS Security Policy Model** D1019578 (SPM_D1019578)
[DVS_SIMEOS]	**SIMEOS Development Security Documentation** D1021035 (DVS_D1021035)
[LCD_SIMEOS]	**SIMEOS Life-cycle definition documentation** D1021039 (LCD_D1021039)
[TAT_SIMEOS]	**SIMEOS Documentation of development tools** D1021040 (TAT_D1021040)
[COV_SIMEOS]	**SIMEOS Analysis of test coverage** D1021030 (COV_D1021030)
[DPT_SIMEOS]	**SIMEOS Analysis of the depth of testing** D1021031 (DPT_D1021031)
[FUN_SIMEOS]	**SIMEOS Test Documentation** D1021032 (FUN_D1021032)
[MSU_SIMEOS]	**SIMEOS Analysis and testing for insecure states** D1021033 (MSU_D1021033)
[ST_SIMEOS]	**SIMEOS Security target** D1019540 (ST_D1019540)
[SOF_SIMEOS]	**SIMEOS Strength of TOE security functions analysis** D1021034 (SOF_D1021034)

Reference	Title
[VLA_SIMEOS]	**SIMEOS Vulnerability Analysis** D1019232 (VLA_D1019232)

1.4.3 Reference of Associated Distribution Sheet

[DS_SIMEOS]	**SIMEOS Security Target Distribution sheet** D1019542

2. TOE DESCRIPTION

This part of the ST describes the TOE as an aid to the understanding of its security requirements.

It addresses the product type, the smart card product life cycle, the TOE environment along the smart card life cycle and the general features of the TOE.

2.1 PRODUCT TYPE

The Target of Evaluation (TOE) is a subset of the Java Card System embedded on the Usimera protect card (evaluated within SIMEOS project). With respect to the Usimera protect product, the scope of the TOE is a subset of the Java Card system made of the JCVM, the JCRE and the API. The integrated circuit, the operating system, the mobile dedicated software and the applications are not part of the TOE. However the Java Card system is used by the applets and interacts with the smart card platform, the card manager and the other components of the Usimera card. All of them are thus part of the TOE IT environment and are included in the scope of evaluation of the Usimera card evaluation [SIMEOS].

The TOE is composed of (in red in the Figure 1):

- Java Card Virtual Machine 2.2.1 [JCVM22],
- Java Card Runtime Environment 2.2.1 [JCRE22] and
- Java Card API 2.2.1 [JCAPI221].

The components static view of the **TOE** is the following: **The linker** (file preparation for execution and interpretation), the **interpreter** (bytecode interpretation), and the native **JC API**.

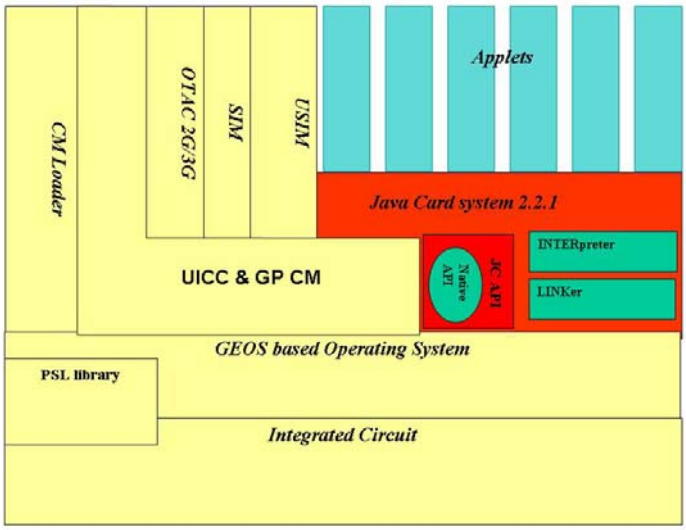

Figure 1 : The TOE and its environment

More precisely, one can view the TOE as the **execution process** of an applet that has been off-card bytecode verified and loaded on the Usimera product. The TSF are those involved in the execution process and correspond to the following three components that compose the Java Card Virtual Machine (and not the Java Card System, that is larger):

- The linker
- The interpreter
- A part of the JC API, in particular the native methods of the package `javacard.framework`

The **TSF** is then made by a subset of **Simeos's Java Card System** (in green on the Figure 1).

2.1.1 The linker

The linker is in charge of the rearrangement of the data structures contained in the Converted APplet file (CAP file) in order to speed up the execution of the applet. The linker first performs a resolution step that is, resolves the external and internal references of a CAP file and replaces them by direct ones. Then it performs the preparation step, allocating the static field image and the static arrays. The later ones are also initialized, thus giving rise to the configuration that will constitute the corresponding initial state of the (JC) interpreter

2.1.2 The interpreter

When an applet is installed, registered and selected, the execution of the applet is carried out by the embedded interpreter. The interpreter mainly consists of a loop that computes the next bytecode to be executed and dispatch the appropriate interpretation functions. Such function modifies the runtimes data areas of the JCVM (the heap, the static field images, the frame stack, etc) according to the semantics of the byte code interpreted.

2.1.3 The Native Java Card APIs

The Java card byte code interpretation done by the interpreter depends in turn on the behaviour of methods of the API. The Java Card APIs consists of a set of customized classes for programming smart card applications according to the ISO 7816 model. The APIs contain four packages:

- The package `java.lang` provides classes that support the design of the Java Card technology subset of the Java programming language.
- The package `javacard.framework` provides framework classes and interfaces for the core functionality of a Java Card applet.
- The packages `javacard.security` and `javacard.crypto` provide classes and interfaces for the Java Card security framework and export-controlled functionality respectively.

Native API methods are usually written in C and are considered as parts of the Java Card platform. The native methods participate in enforcing several essential TOE security functions such as Transaction, Install, Firewall, JCRE, Applet and PIN (Section 6).

Consequently, the native methods of the package `javacard.framework` are then included in the TOE.

The package `javacard.framework.service` that mainly used for RMI functionality is not included because the corresponding security objective (OE.REMOTE) is not considered.

The packages `javacardx.crypto` and `javacard.security` are not included as they are beyond the current state of the art of the formal modelling.

2.1.4 Applet installation

This functionality concerns the Installation of post-issuance applets and thus security aspects of the installation procedure that are related to applet execution. The TOE, made of the virtual machine running on the top of Geos platform collaborates with other card components in order to execute an applet. When executing the code of the applet, the virtual machine communicates with the external world through the Card Manager component. Only **the applet installation process after the loading of the CAP file is in the scope of this ST**. This process belongs to the API (method `Applet.install`) and is invoked by the JCRE to link the CAP file using the existing packages and then, register it using a fresh AID or PID.

The other functionalities, as card life cycle, downloading, communication with the CAD, are out of the scope of this security target.

2.1.5 Assumptions

This security target assumes the following minimum requirements:

- The applets, present on the card, are correctly typed, (all the “must clauses” imposed in chapter 7 of [JCVM221] on the bytecodes and the correctness of the CAP file format are satisfied).
- The applet to be downloaded in post issuance are byte code verified as defined in [PP/JCS].
- The loading of applet pre-issuance is made in a secure environment.

2.2 SMART CARD PRODUCT LIFE-CYCLE

As the lifetime of the JC system is the lifetime of the card, the life phases follow the smart card product life cycle described in the SIMEOS Security Target [ST_SIMEOS].

Usimera Protect product life cycle is described in the following picture:

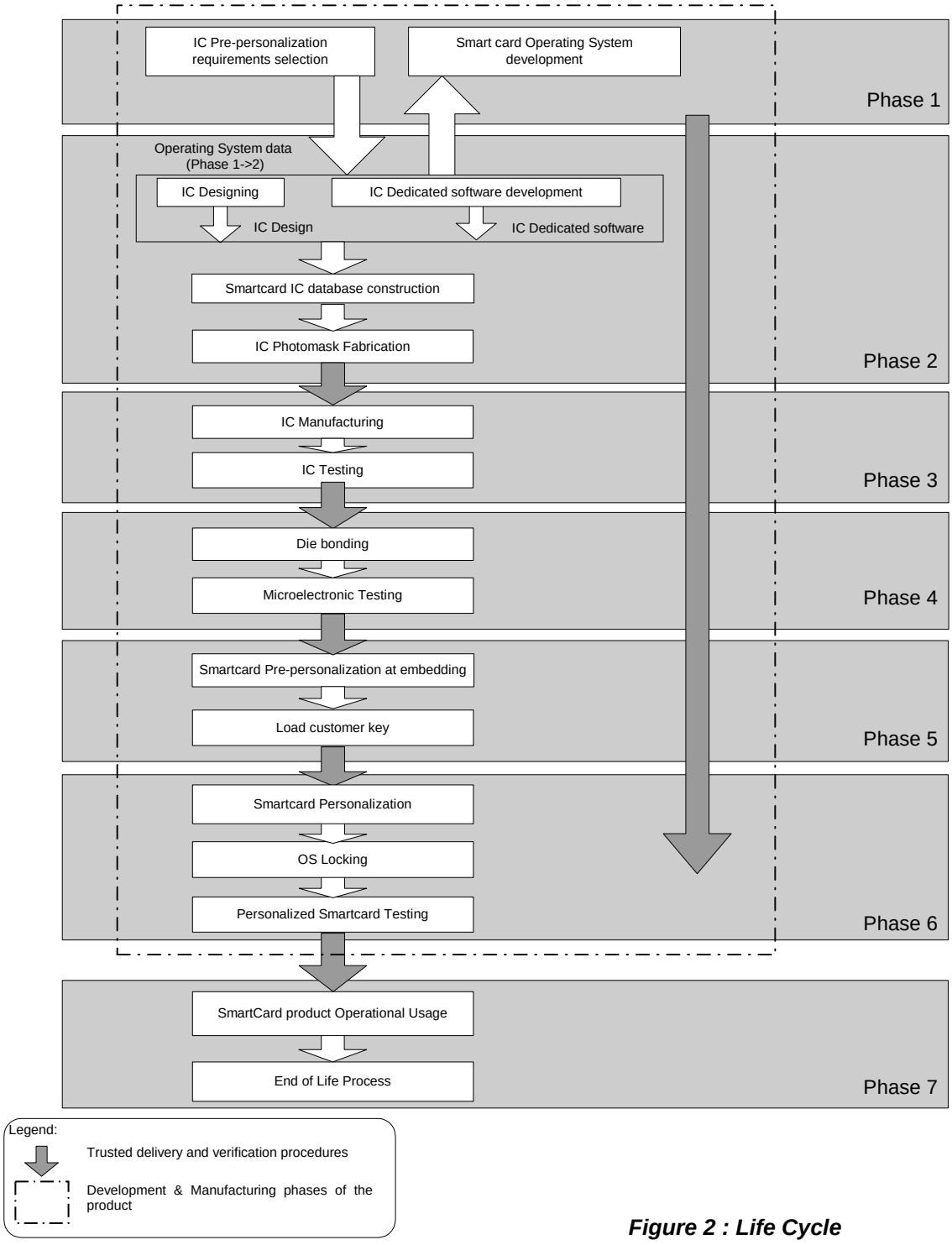

Figure 2 : Life Cycle

The TOE is a subset of the Java Card System embedded on the product at the end of phase 6 as shown in Figure 2. This is the operational Usimera Protect product, as a personalized smart card. As far as the EAL4+ evaluation scope is concerned, phases 1 to 6 are considered as development and manufacturing phases of the product but the TOE is the result of these phases that can consequently be seen as phases of the TOE generation. The scope of this Security Target corresponds to the Phase 7 of the product life cycle.

The following table gives a description of the product life cycle and explains where the authorities are involved:

Phase 1	Smart Card OS development	The Embedded Software developer is responsible for the development of the Operating System and the specification of initialisation requirements (OS options).
Phase 2	IC design, IC database construction and IC photomask fabrication (IC & DS development)	The IC manufacturer is responsible for these operations, taking as an input the embedded software data given by Embedded Software developer.
Phase 3	IC manufacturing and testing	The IC manufacturer is responsible for producing and testing the IC.
Phase 4	Die bonding. Microelectronic testing.	The IC packaging manufacturer is responsible for die bonding and microelectronic testing. The transport key is changed to GEMALTO key.
Phase 5	Pre-personalization (Embedding)	The Pre-personaliser is responsible to store customer loader key encrypted with GEMALTO loader key.
Phase 6	Smart card Personalization	The Personaliser is responsible for the Smart Card personalization (change customer key with OS key, loading the OS code and data using the Infineon Loader, locking the capability of code downloading) and for final tests.
Phase 7	Smart card operational-usage	This is the operational usage of the product in the Mobile Communication environment, including the over the air customisation of the product by the end-usage administrator.

2.3 TOE ENVIRONMENT

Considering the product, the environment is defined as follows:

- Development environment corresponding to phases 1 and 2;
- Production and Personalization environments corresponding to phases 3 to 6:
    - Manufacturing environment including the IC test operations, IC packaging, testing and pre-personalization (phases 3 to 5),
    - Personalization environment corresponding to the loading by the Infineon Loader of the OS in the flash memory, personalization and testing of the Smart Card with the user data (phase 6).
- User environment corresponding to the card use by a subscriber on a 2G or 3G network (phase 7).

2.3.1 TOE Development Environment

The TOE is a subset of the software embedded on Usimera protect. Therefore, the TOE development environment is the one of Usimera Protect card described in the SIMEOS Security Target [ST_SIMEOS]

2.3.2 TOE Production and Personalization Environment

The TOE is a subset of the software embedded on Usimera protect. Therefore, the TOE Production and Personalization environment is the one of Usimera Protect card described in the SIMEOS Security Target [ST_SIMEOS]

2.3.3 TOE User Environment

Smart Cards are used in a wide range of applications to assure authorized conditional access. The Usimera card is to be used on terminals such as GSM and UMTS handsets or smart card readers.

The end-user environment therefore covers an unprotected environment, thus making it difficult to avoid any abuse of the TOE. The product is prepared accordingly to mitigate such attacks in this environment.

Considering the TOE as the Java Card System, it is fully operational in the user environment. The only abuse that we concentrate on in this Security Target is the **applet isolation property**: the main requirement that has to be full filled is that each applet is protected from illegal modification from the neighbour applet and that the underlying system is protected from fraudulent and corrupted applet.

2.3.4 The actors and roles

The actors are split in:

Product Developers

The IC designer and DS developer designs the chip and its Dedicated Software (DS). Here it is INFINEON.

The Embedded Software Developer designs the Operating System (OS) according to IC and DS specifications. Here it is GEMALTO.

Product Manufacturer

The IC manufacturer -or founder- designs the photomask, manufactures the IC with the Dedicated Software(PSL). Here the founder and IC manufacturer are INFINEON.

The IC die bounding manufacturer is responsible for the die bounding from the ICs provided by the founder. For this product, the die bounding manufacturer is GEMALTO.

The Smart Card product manufacturer (or Card manufacturer) is responsible to obtain a pre-personalized card from a packaged IC.

For this product, the Smart Card product manufacturer is GEMALTO.

Personalizer

The Smart Card Personalizer personalizes the card by loading two categories of data:

1. the code and data (OS) belonging to the Developer and Manufacturer of the Card ;
2. the Cardholder data as well as cryptographic keys and PINs.

The Personalizer may also load card issuer applets during this phase. Here it is GEMALTO.

Phase	Administrator	Environment
6	Smart Card Personalizer	Production Environment

The Mobile Communications applications are activated during this phase. At the end of this phase, only applets may be loaded. The card is issued in OP_SECURED state.

Card Issuer

The Card Issuer -short named “issuer”- is the Mobile Communication cellular network operator (GSM or UMTS network). The operator issues cards to its customers who are the “Cardholders”. The card belongs to the Card Issuer and its role is to administrate the card in the End Usage Phase. Therefore, the Card Issuer is responsible for selecting and managing the personalization, for managing applets (load, install and delete) as well as for distribution and invalidation of the card.

End User

The Cardholder is a customer of the Card Issuer. The card is personalized with the Cardholder identification and secrets. It is usually the final user.

The roles (administration and usage) are defined in the following tables. During the delivery between phases the responsibility is transferred from the current phase administrator to the next phase administrator.

Phase	Administrator	Environment
7	Card Issuer	Usage Environment

Phase	User	Environment
7	End User	Usage Environment

2.4 TOE INTENDED USAGE

Usimera Protect is a 2G and 3G product, which is compliant with the Release 5 and Release 6 of ETSI Mobile Communication standards. Depending on the handset and network capabilities, it can be used as a USIM card, a SIM card, or both. This product can also support Java Card applets that need security like DES/TDES authentication applets to enable secure access to banking services.

M-Commerce, trusted projects are typical use cases.

The Java card System, the TOE in this ST, embedded in Usimera protect, is intended to provide a framework in which the Java card applets, present or to be downloaded, coexist and interact safely. Those applications can be selected for execution when the card is inserted into a card reader.

While the Java Card virtual machine (JCVM) is responsible for ensuring language-level security, the JCRE provides additional security features for the Usimera protect. The basic runtime security feature imposed by the JCRE enforces isolation of applets using the Java Card **firewall**. It prevents objects created by one applet from being used by another applet without explicit sharing. This prevents unauthorized access to the fields and methods of class instances, as well as the length and contents of arrays.

The firewall is considered as the most important security feature of the JCS. It enables complete isolation between applets or controlled communication through additional mechanisms that allow them to share objects when needed. The JCRE allows such sharing using the concept of “shareable interface objects” (SIO) and static public variables. The JCVM should ensure that the only way for applets to access any resources are either through the JCRE or through the Java Card API.

3. TOE SECURITY ENVIRONMENT

3.1 ASSETS

3.1.1 User data

D.APP_CODE

The code of the *applet*s and libraries loaded on the card.

To be protected from unauthorized modification.

D.APP_C_DATA

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

To be protected from unauthorized disclosure.

D.APP_I_DATA

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

To be protected from unauthorized modification.

D.PIN

Any end-user's PIN.

To be protected from unauthorized disclosure and modification.

D.APP_KEYs

Cryptographic keys owned by the *applets*.

To be protected from unauthorized disclosure and modification.

3.1.2 TSF Data

D.JCS_CODE

The code of the *Java Card System*.

To be protected from unauthorized disclosure and modification.

D.JCS_DATA

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

In particular for the

Linker: The complete linked format of the loaded package, including all the auxiliary structures used to link the package and then thrown away (linked constant pool, descriptor component and reference location component).

Interpreter: Security sensitive data used by the interpreter includes:

- the control information of the frame stack, like the current pc and the active firewall context of each execution frame;
- the header of each Java object in the heap, containing the owner of the object and its type;

To be protected from monopolization and unauthorized disclosure or modification.

D.SEC_DATA

The runtime security data of the *JCRE*, like, for instance, the *AID*s used to identify the installed *applets*, the *Currently selected applet*, the *current context* of execution, the owner of each object, java objects owned by the JCRE and private fields of the API, The content of the operand stack and the local variables of those frames where the active context is the JCRE and The table of registered applets.

To be protected from unauthorized disclosure and modification.

D.API_DATA

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

To be protected from unauthorized disclosure and modification.

D.JCS_KEYs

Cryptographic keys used when loading a file into the card.

To be protected from unauthorized disclosure and modification.

D.CRYPTO

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

To be protected from unauthorized disclosure and modification.

3.2 SUBJECTS

The main *subjects* of the Java Card System are the following ones:

S.PACKAGE

Packages used on the Java Card platform that act on behalf of the applet Developer. These subjects are involved in the ***FIREWALL security policy*** and ***static access control security policy*** defined in [ISP_SIMEOS] and they should be understood as instances of the subject *S.PACKAGE*. An application is identified by the name and version of the package that contains it. The package is the basic security unit of the TOE, which makes no difference between two applets contained in the same package. For the ***static access control security policy***, *the subject is called "S.applet".*

S.JCRE

The *JCRE*, which acts on behalf of the card issuer. This subject is involved in several security policies defined in [PP/JCS], and in [ISP_SIMEOS] and is always represented by the subject *S.JCRE*.

It is represented in the card by the native code implementing the card manager, the on card part of the loading protocol, the linker, the API and the interpreter, plus those packages of the API that are written in Java Card.

S.BCV

The bytecode verifier (BCV), which acts on behalf of the verification authority. This subject is involved in the **PACKAGE LOADING security policy** defined in *CarG* Security Functional Requirements (§ 5.3.9) and is represented by the subject S.BCV.

Application note:

The bytecode verifier is off-card and out of the scope of this ST.

S.CRD

The installer, which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets. It could play the role of the on-card entity in charge of package loading, which is involved in the **PACKAGE LOADING security policy** defined in *CarG* Security Functional Requirements (§ 5.3.9) and is represented by the subject S.CRD.

Application note:

The installer is on-card but into the IT security environment of the TOE.

S.ADEL

The applet deletion manager, which also acts on behalf of the card issuer. This subject is involved in the **ADEL Manager security policy** defined in *ADELG* Security Functional Requirements (§ 5.3.7).

Application note:

The ADEL Manager is in the IT security environment of the TOE.

S.CAD

The *CAD* is involved in the ***JCRMI security policy***.

Application note:

The JCRMI is in the IT security environment of the TOE.

S.SPY

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

Application note:

The package loading is in the IT security environment of the TOE.

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

3.3 ASSUMPTIONS

These Assumptions are those described in [PP/JCS] except *A.NATIVE*. *A.NATIVE* is just removed because native API belongs to the TOE and the assumptions described in *A.NATIVE* have been transferred to *OSP.NATIVE*.

A.VERIFICATION

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

A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM22], §3.3) outside the API.

3.4 THREATS

A threat agent wishes to abuse the assets either by functional attacks or by environmental manipulation, by specific hardware manipulation, by a combination of hardware and software manipulations or by any other type of attacks.

The threats of the Usimera card on which the TOE is embedded are defined and evaluated in [ST_SIMEOS].

The threats described hereafter are those specific to the TOE as defined in chapter 2 and defined in [PP/JCS].

T.PHYSICAL

The attacker **discloses** or **modifies** the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means.

This threat includes IC failure analysis, electrical probing, unexpected tearing, [Note: extension of the original text from the PP "and DP analysis"] fault injection, side channel attacks using Power and Electromagnetic analysis.

That also includes the modification of the runtime execution of *Java Card System* or *SCP* software through alteration of the intended execution order of (set of) instructions through physical tampering techniques.

As the TOE is subset of the software embedded on the SIMEOS product, this threat is covered by SIMEOS evaluation and evaluated in [ST_SIMEOS].

3.4.1 Confidentiality

T.CONFID-JCS-CODE

The attacker executes an application without authorisation to disclose the TOE embedded code (*Java Card System* code).

T.CONFID-APPLI-DATA

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

T.CONFID-JCS-DATA

The attacker executes an application without authorisation to disclose data belonging to the TOE *(Java Card System).*

3.4.2 Integrity

T.INTEG-APPLI-CODE

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

T.INTEG-JCS-CODE

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

T.INTEG-APPLI-DATA

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

T.INTEG-JCS-DATA

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

T.INTEG-APPLI-CODE.2

The attacker modifies (part of) its own or another application code when an application *package* is transmitted to the card for installation.

T.INTEG-APPLI-DATA.2

The attacker modifies (part of) the initialisation data contained in an application *package* when the package is transmitted to the card for installation.

3.4.3 Identity Usurpation

T.SID.1

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

T.SID.2

The attacker modifies the identity of the privileged roles.

3.4.4 Unauthorized Executions

T.EXE-CODE.1

An *applet* performs an unauthorized execution of a method.

T.EXE-CODE.2

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

T.NATIVE

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

T.EXE-CODE-REMOTE

The attacker performs an unauthorized **remote execution** of a **method** from the *CAD*.

3.4.5 Denial of Service

T.RESOURCES

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

3.4.6 Modifications of the Set of Applications

T.INSTALL

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

3.4.7 Card Management

T.DELETION

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

3.4.8 Services

T.OBJ-DELETION

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

3.5 ORGANISATIONAL SECURITY POLICIES

The Organisational security policies used for SIMEOS evaluation are studied in [ST_SIMEOS].

OSP.VERIFICATION

This policy shall ensure the adequacy between the export files used in the verification and those used for installing the verified file. The policy must also ensure that no modification of the file is performed in between its verification and the signing by the verification authority.

OSP.NATIVE

Any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

Application note:

Note 1. *#.NATIVE* from [PP/JCS] is reduced as follows:

- No un-trusted native code may reside on the card.
- Loading of native code is forbidden.

OSP.LOAD

This policy shall ensure that the package loading is performed safely without loss, substitution, addition, modification, repetition of data or any other integrity failure on the loaded data such as a wrong order in the delivery of data by incoming APDUs.

OSP.DEV

Procedures shall ensure that relevant security data are used in a secure manner during TOE construction.

4. SECURITY OBJECTIVES

4.1 SECURITY OBJECTIVES FOR THE TOE

4.1.1 Identification

OT.SID

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

4.1.2 Execution

OT.FIREWALL

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

OT.SHRD_VAR_INTEG

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

4.1.3 Services

OT.ALARM

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

OT.TRANSACTION

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

Application note:

1. Transactions are provided to *applets* as Java Card class libraries.
2. See also the *OE.KEY-MNGT* Application note.

4.1.4 Applet Management

OT.INSTALL

The TOE shall ensure that the installation of an *applet* is safe. More precisely, each newly loaded package references only packages that have been already loaded on the card.

Application note:

This objective concerns the linking in the installation process.

4.2 SECURITY OBJECTIVES FOR THE ENVIRONMENT

4.2.1 Common

OE.DEVELOPMENT

During phases 1 to 6, procedures shall be used suitably to maintain the integrity and confidentiality of the assets of the TOE.

OE.APPLET

No *applet* loaded post-issuance contains native methods.

OE.VERIFICATION

Any byte-code must be verified prior to being loaded, and installed before the execution in order to ensure that each bytecode is valid at execution time.

4.2.2 IC and Native

OE.NATIVE.2

Any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

OE.SCP.RECOVERY

If there is a loss of power, or if the smart card is withdrawn from the *CAD* while an operation is in progress, the *SCP* must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state (#.SCP.1 from [PP/JCS]).

Application note: The smart card Platform is composed of a micro-controller and an operating system. It provides memory management functions, I/O functions that are compliant with ISO standards, transaction facilities and secure implementation of cryptographic functions. It also contains dedicated software, which provides an interface with the integrated software.

OE.SCP.SUPPORT

The SCP shall provide functionalities that support the well-functioning of the TSFs of the TOE (avoiding they are bypassed or altered) and by controlling the access to information proper of the TSFs. In addition, the smart card platform should also provide basic services, which are required by the runtime environment to implement security mechanisms such as atomic transactions, management of persistent and transient objects and cryptographic functions. These mechanisms are likely to be used by security functions implementing the security requirements defined for the TOE.

OE.SCP.IC

The *SCP* shall possess IC security features.

Application note:

The TOE must be resistant to physical attack and prevent use of security relevant assets derived from these attacks.

OE.CARD-MANAGEMENT

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

The card manager is an application with specific rights, which is responsible for the administration of the smart card. This component is part of the TOE environment, and enforces some of its security functions. Typically the card manager is in charge of the life cycle of the whole card, as well as that of the installed applications (*applet*s). The card manager should prevent that card content management (loading, installation, deletion) is carried out, for instance, at invalid states of the card or by non-authorized actors. It shall also enforce security policies established by the card issuer.

The following objectives were objectives for the TOE into [simeos]. They become Objectives for the environment because the TOE does not include the corresponding functionalities and groups in this ST.

4.2.3 Execution

OE.OPERATE

The IT environment must ensure continued correct operation of its security functions.

OE.RESOURCES

The IT environment (Card manager) controls the availability of resources for the applications.

Application note:

Here the resources include those of the *S.JCRE*.

OE.REALLOCATION

The IT environment (Card manager) shall ensure that the re-allocation of a memory block for the runtime areas of the *JCVM* does not disclose any information that was previously stored in that block.

Application note:

To be made unavailable means to be physically erased with a default value. Except for local variables that do not correspond to method parameters, the default values to be used are specified in [JCVM22].

OE.NATIVE

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

OE.SHRD_VAR_CONFID

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

4.2.4 Services

OE.CIPHER

The SCP (the Crypto APIs) shall provide means to cipher sensitive data for applications in a secure way. In particular, the SCP must support cryptographic algorithms consistent with cryptographic usage policies and standards.

Application note:

See also the *OE.KEY-MNGT* Application note.

OE.KEY-MNGT

The SCP (the Crypto APIs) shall provide means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys.

Application note:

OE.KEY-MNGT, OE.PIN-MNGT, OT.TRANSACTION and *OE.CIPHER* are actually provided to *applets* in the form of Java Card API's.

OE.PIN-MNGT

The IT environment shall provide means to securely manage PIN objects.

Application note:

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

2. See also the *OE.KEY-MNGT* Application note.

OE.REMOTE

The IT environment shall provide means to restrict remote access from the *CAD* to the services implemented by the applets on the card. This particularly concerns the *RMI* services introduced in version 2.2 of the Java Card platform.

4.2.5 Applet management

OE.LOAD

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

Application note:

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

OE.DELETION

The Deletion Manager shall ensure that both *applet* and *package* deletion are safe.

OE.OBJ-DELETION

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

5. IT SECURITY REQUIREMENTS

5.1 TOE SECURITY FUNCTIONAL REQUIREMENTS

5.1.1 Core (CoreG)

5.1.1.1 Firewall Policy

The following requirements are modified by and therefore included in *LCG*:

- *FDP_ACC.2/FIREWALL*;
- *FDP_ACF.1/FIREWALL*;
- *FMT_MSA.1/JCRE*.

FDP_IFC.1/JCVM Subset information flow control

FDP_IFC.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** on **the following subjects, information and operations**.

Non editorial refinement:

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

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

There is a unique operation in this policy:

Operation	Description
OP.PUT(S_1, S_2, I)	Transfer a piece of information *I* from S_1 to S_2.

FDP_IFF.1/JCVM Simple security attributes

FDP_IFF.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** based on the following types of subject and information security attributes: **S.LOCAL, S.MEMBER, I.DATA and the currently active Context**.

FDP_IFF.1.2/JCVM The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **An operation *OP.PUT(S_1, S.MEMBER, I)* is allowed if and only if the active context is "JCRE"; other *OP.PUT* operations are allowed regardless of the active context's value**.

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

FDP_IFF.1.4/JCVM The TSF shall provide the following **none**.

FDP_IFF.1.5/JCVM The TSF shall explicitly authorise an information flow based on the following rules: **all JCRE Permanent Entry Point Object may be stored in a S.MEMBER**.

FDP_IFF.1.6/JCVM The TSF shall explicitly deny an information flow based on the following rules: **the storage of the reference of an object with attribute JCRE Temporary Entry Point Object or Global Array in a static field, instance field or array element is forbidden**.

FMT_MSA.2/JCRE Secure security attributes

FMT_MSA.2.1/JCRE The TSF shall ensure that only secure values are accepted for security attributes.

Application note:

Secure values conform to the following rules:

- The Context attribute of a **.JAVAOBJECT* must correspond to that of an installed applet or be "JCRE".
- An *O.JAVAOBJECT* whose Sharing attribute is *JCRE Entry Point* or Global Array necessarily has "JCRE" value for its Context security attribute.
- An *O.JAVAOBJECT* whose Sharing attribute value is Global Array necessarily have "array of primitive Java Card type" as JavaCardClass security attribute's value.
- Any *O.JAVAOBJECT* whose Sharing attribute value is not "Standard" has a PERSISTENT-LifeTime attribute's value.
- Any *O.JAVAOBJECT* whose LifeTime attribute value is not PERSISTENT has an array type as JavaCardClass attribute's value.

FMT_MSA.3/FIREWALL Static attribute initialisation

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

FMT_MSA.3.2/FIREWALL The TSF shall allow the **authorized identified roles: None** to specify alternative initial values to override the default values when an object or information is created.

5.1.1.2 Application Programming Interface

FDP_RIP.1/ABORT Subset residual information protection

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

*Global refinement:*The resource is *any reference to an object instance created during an aborted transaction*.

FDP_ROL.1/FIREWALL Basic rollback

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

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process() or install() call, notwithstanding the restrictions given in [JCRE22], §7.7, within the bounds of the Commit Capacity ([JCRE22], §7.8), and those described in [JCAPI22]**.

5.1.1.3 Card Security Management

FAU_ARP.1/JCS Security alarms

FAU_ARP.1.1/JCS The TSF shall take **the following actions: throw an exception, lock the card session or reinitialise the JCS and its data** upon detection of a potential security violation.

Global refinement:

Potential security violation is refined to one of the following events:

- Applet life cycle inconsistency
- Card tearing (unexpected removal of the Card out of the *CAD*) and power failure
- Abortion of a transaction in an unexpected context (see (abortTransaction(), [JCAPI22] and ([JCRE22], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Other runtime errors related to applet's failure (like uncaught exceptions)

Application note:

The thrown exceptions and their related events are described in [JCRE22], [JCAPI22], and [JCVM22].

5.1.1.4 AID Management

FMT_MTD.1/JCRE Management of TSF data

FMT_MTD.1.1/JCRE The TSF shall restrict the ability to **modify** the **list of registered applets' *AID*** to **the JCRE**.

FMT_MTD.3 Secure TSF data

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

FIA_ATD.1/AID User attribute definition

FIA_ATD.1.1/AID The TSF shall maintain the following list of security attributes belonging to individual users: **the *AID* and version number of each *package*, the *AID* of each registered *applet*, and whether a registered *applet* is currently selected for execution ([JCVM22], §6.5)**.

FIA_UID.2/AID User identification before any action

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

Application note:

The User here mentioned are those associated to Packages or Applets.

FIA_USB.1 User-subject binding

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

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **each *S.Package* has a different context independent of the one of the JCRE**.

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

Application note:

For *S.PACKAGE*s, the Context security attribute plays the role of the appropriate user security attribute; see *FMT_MSA.1.1/JCRE*.

5.1.2 Installer (InstG)

This group includes the Security Functional Requirements as described by the [PP/JCS], except:

- *FMT_SMR.1/Installer*, merged into *FMT_SMR.1/JCS*;

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

FDP_ITC.2.1/Installer The TSF shall enforce the **FIREWALL access control SFP** when importing user data, controlled under the SFP, from outside of the TSC.

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

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

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

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

- A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs. The loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM22], §4.5.2). The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM22], §4.4).
- The References export rules

5.1.3 Logical channels (LCG)

5.1.3.1 Firewall Policy

Except for *FDP_ACC.2/FIREWALL*, *FDP_ACF.1/FIREWALL* and *FMT_MSA.1/JCRE* included hereafter, this policy includes unchanged the functional requirements specified in the ***FIREWALL*** *access control* ***SFP*** of the group *CoreG*.

FDP_ACC.2/FIREWALL Complete access control

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

Non editorial refinement:

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

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

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

Operation	Description
OP.ARRAY_ACCESS (O.JAVAOBJECT, field)	Read/Write an array component.
OP.INSTANCE_FIELD (O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language
OP.INVK_VIRTUAL (O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object)
OP.INVK_INTERFACE (O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.THROW (O.JAVAOBJECT)	Throwing of an object (**athrow**).
OP.TYPE_ACCESS (O.JAVAOBJECT, class)	Invoke **checkcast** or **instanceof** on an object.
OP.JAVA (...)	Any access in the sense of [JCRE22], §6.2.8. In our formalization, this is one of the preceding operations.
OP.CREATE (Sharing, LifeTime)	Creation of an object (**new** or **makeTransient call**).

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

FDP_ACC.2.2/FIREWALL The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1/FIREWALL Security attribute based access control

FDP_ACF.1.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** to objects based on the following: **(1) the security attributes of the covered subjects and objects, (2) the currently active context, (3) the SELECTed applet Context, and (4) the attribute ActiveApplets, which is a list of the active applets' AIDs**.

Non editorial refinement:

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

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

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

Name	**Description**
Context	*Package AID* or "JCRE"
Selection Status	Multiselectable, Non-multiselectable or "None"
Sharing	Standard, SIO, JCRE Entry Point, or Global Array
LifeTime	CLEAR_ON_DESELECT or PERSISTENT.
SELECTed applet Context	*Package AID* or "None"
ActiveApplets	List of package's *AID*s

The Java Card platform, version 2.2, introduces the possibility for an *applet* instance to be selected on multiple *logical channels* at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as *multiselectable applet*s. *Applet*s that

belong to a same *package* are either all multiselectable or not ([JCVM22], §2.2.5). Therefore, the selection mode can be regarded as an attribute of *package*s. No selection mode is defined for a library *package*.

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

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

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

R.JAVA.1 ([JCRE22] §6.2.8) An *S.PACKAGE* may freely perform any of *OP.ARRAY_ACCESS*, *OP.INSTANCE_FIELD*, *OP.INVK_VIRTUAL*, *OP.INVK_INTERFACE*, *OP.THROW* or *OP.TYPE_ACCESS* upon any *O.JAVAOBJECT* whose Sharing attribute has value "JCRE Entry Point" or "Global Array".

R.JAVA.2 ([JCRE22] §6.2.8) An *S.PACKAGE* may freely perform any of *OP.ARRAY_ACCESS*, *OP.INSTANCE_FIELD*, *OP.INVK_VIRTUAL*, *OP.INVK_INTERFACE* or *OP.THROW* upon any *O.JAVAOBJECT* whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if *O.JAVAOBJECT* 's Context attribute has the same value as the active context.

R.JAVA.3 ([JCRE22] §6.2.8.10) An *S.PACKAGE* may perform *OP.TYPE_ACCESS* upon an *O.JAVAOBJECT* whose sharing attribute has value "SIO" only if *O.JAVAOBJECT* is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.

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

R.JAVA.20 ([JCRE22] §6.2.8.6) An *S.PACKAGE* may perform *OP.INVK_INTERFACE* upon an *O.JAVAOBJECT* whose Sharing attribute has the value "SIO", and whose Context attribute has the value " *Package AID* ", only if one of the following applies:

a) The value of the attribute Selection Status of the package whose AID is " *Package AID* " is "Multiselectable",

b) The value of the attribute Selection Status of the package whose AID is " *Package AID* " is "Non-multiselectable", and either " *Package AID* " is the value of the currently selected applet or otherwise " *Package AID* " does not occur in the attribute Active applets,

and in either of the cases above the invoked *interface* method extends the Shareable *interface*.

FDP_ACF.1.3/FIREWALL The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **The subject *S.JCRE* can freely perform *OP.JAVA(...)* and *OP.CREATE*, with the exception given in FDP_ACF.1.4/FIREWALL**.

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

1) Any subject with *OP.JAVA* upon an *O.JAVAOBJECT* whose LifeTime attribute has value "CLEAR_ON_DESELECT" if *O.JAVAOBJECT* 's Context attribute is not the same as the SELECTed applet Context.

2) Any subject with *OP.CREATE* and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the SELECTed applet Context.

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes **the active context, the SELECTed applet Context and the ActiveApplets security attributes** to **the JCRE (*S.JCRE*)**.

Application note: The modification of the active context, SELECTed applet Context and ActiveApplets security attributes should be performed in accordance with the rules given in [JCRE22], §4 and ([JCVM22], §3.4.

5.2 TOE SECURITY ASSURANCE REQUIREMENTS

The [PP/BSI-0002] requirements for the IC evaluation are consistent with these requirements.

The SIMEOS security assurance requirement level is EAL4 augmented with AVA_MSU.3, AVA_VLA.4, ADV_IMP.2 and ALC_DVS.2 for the TOE and are already evaluated in SIMEOS.

The "FORMAL ASSURANCE on SIMEOS JVM" augments only the assurance requirements on the Javacard System software with:

- ADV_FSP.4 (Development- Formal functional specification)
- ADV_SPM.3 (Development- Formal TOE security policy model)
- ADV_HLD.5 (Development- Formal high-level design)
- ADV_LLD.2 (Development-Semi-formal low-level design)
- ADV_INT.3 (Development- Minimisation of complexity)
- ADV_IMP.3 (Development – Structured Implementation of the TSF)
- ADV_RCR.3 (Development- formal correspondence demonstration)

5.2.1 ACM Configuration management

5.2.1.1 ACM_AUT CM automation

ACM_AUT.1 Partial CM automation

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are extended here to deliverables required for the FORMAL evidences and associated process.

5.2.1.2 ACM_CAP CM capabilities

ACM_CAP.4 Generation support and acceptance procedures

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are extended here to deliverables supplied for the FORMAL evidences.

5.2.1.3 ACM_SCP CM scope

ACM_SCP.2 Problem tracking CM coverage

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.2 ADO Delivery and operation

5.2.2.1 ADO_DEL Delivery

ADO_DEL.2 Detection of modification

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.2.2 ADO_IGS Installation, generation and start-up

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

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.3 ADV Development

5.2.3.1 ADV_FSP Functional specification

ADV_FSP.4 Formal Functional Specification

ADV_FSP.4.1D The developer shall provide a functional specification.

ADV_FSP.4.1C The functional specification shall describe the TSF and its external interfaces using a **formal** style, supported by informal, explanatory text where appropriate.

ADV_FSP.4.2C The functional specification shall be internally consistent.

ADV_FSP.4.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.4.4C The functional specification shall completely represent the TSF.

ADV_FSP.4.5C The functional specification shall include rationale that the TSF is completely represented.

ADV_FSP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.4.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.2 ADV_HLD High-level design

ADV_HLD.5 Formal high-level design

ADV_HLD.5.1D The developer shall provide the high-level design of the TSF.

ADV_HLD.5.1C The presentation of the high-level design shall be formal.

ADV_HLD.5.2C The high-level design shall be internally consistent.

ADV_HLD.5.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.5.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.5.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.5.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.5.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.5.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.5.9C The high-level design shall describe the separation of the TOE into TSPenforcing and other subsystems.

ADV_HLD.5.10C The high-level design shall justify that the identified means of achieving separation, including any protection mechanisms, are sufficient to ensure a clear and effective separation of TSP-enforcing from non-TSP-enforcing functions.

ADV_HLD.5.11C The high-level design shall justify that the TSF mechanisms are sufficient to implement the security functions identified in the high-level design.

ADV_HLD.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.5.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.3 ADV_IMP Implementation representation

ADV_IMP.3 Structured Implementation of the TSF

ADV_IMP.3.1D The developer shall provide the implementation representation for the entire TSF.

ADV_IMP.3.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.3.2C The implementation representation shall be internally consistent.

ADV_IMP.3.3C The implementation representation shall describe the relationships between all portions of the implementation.

ADV_IMP.3.4C The implementation representation shall be structured into small and comprehensible sections.

ADV_IMP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.3.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.4 ADV_INT TSF Internal

ADV_INT.3 Minimisation of complexity

ADV_INT.3.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.3.2D The developer shall provide an architectural description.

ADV_INT.3.3D The developer shall design and structure the TSF in a layered fashion that minimises mutual interactions between the layers of the design.

ADV_INT.3.4D The developer shall design and structure the TSF in such a way that minimises the complexity of the entire TSF.

ADV_INT.3.5D The developer shall design and structure the portions of the TSF that enforce any access control and/or information flow control policies such that they are simple enough to be analysed.

ADV_INT.3.6D The developer shall ensure that functions whose objectives are not relevant for the TSF are excluded from the TSF modules.

ADV_INT.3.1C The architectural description shall identify the modules of the TSF and shall specify which portions of the TSF enforce the access control and/or information flow control policies.

ADV_INT.3.2C The architectural description shall describe the purpose, interface, parameters, and side-effects of each module of the TSF.

ADV_INT.3.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.3.4C The architectural description shall describe the layering architecture.

ADV_INT.3.5C The architectural description shall show that mutual interactions have been minimised, and justify those that remain.

ADV_INT.3.6C The architectural description shall describe how the entire TSF has been structured to minimise complexity.

ADV_INT.3.7C The architectural description shall justify the inclusion of any non-TSPenforcing modules in the TSF.

ADV_INT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.3.2E The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.

ADV_INT.3.3E The evaluator shall confirm that the portions of the TSF that enforce any access control and/or information flow control policies are simple enough to be analysed.

5.2.3.5 ADV_LLD Low-level design

ADV_LLD.2 Semi-Formal low-level design

ADV_LLD.2.1D The developer shall provide the low-level design of the TSF.

ADV_LLD.2.1C The presentation of the low-level design shall be semiformal.

ADV_LLD.2.2C The low-level design shall be internally consistent.

ADV_LLD.2.3C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.2.4C The low-level design shall describe the purpose of each module.

ADV_LLD.2.5C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.2.6C The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.2.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.2.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.2.9C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_LLD.2.10C The low-level design shall describe the separation of the TOE into TSPenforcing and other modules.

ADV_LLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.2.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.6 ADV_RCR Representation correspondence

ADV_RCR.3 formal correspondence demonstration

ADV_RCR.3.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.3.1C For those corresponding portions of representations that are formally specified, the developer shall prove that correspondence.

ADV_RCR.3.2C For each adjacent pair of provided TSF representations, the analysis shall prove or demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.3.3C For each adjacent pair of provided TSF representations, where portions of one representation are semi-formally specified and the other at least semi-formally specified, the demonstration of correspondence between those portions of the representations shall be semi-formal.

ADV_RCR.3.4C For each adjacent pair of provided TSF representations, where portions of both representations are formally specified, the proof of correspondence between those portions of the representations shall be formal.

ADV_RCR.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence. ADV_RCR.3.2E The evaluator shall determine the accuracy of the proofs of correspondence by selectively verifying the formal analysis.

5.2.3.7 ADV_SPM Security policy modeling

ADV_SPM.3 formal TOE security policy model

ADV_SPM.3.1D The developer shall provide a TSP model.

ADV_SPM.3.2D The developer shall demonstrate or prove, as appropriate, correspondence between the functional specification and the TSP model.

ADV_SPM.3.1C The TSP model shall be formal.

ADV_SPM.3.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.3.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.3.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.3.6C If the functional specification is formal, the proof of correspondence between the TSP model and the functional specification shall be formal.

ADV_SPM.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 AGD Guidance documents

5.2.4.1 AGD_ADM Administrator guidance

AGD_ADM.1 Administrator guidance

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.4.2 AGD_USR User guidance

AGD_USR.1 User guidance

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.5 ALC Life cycle support

5.2.5.1 ALC_DVS Development security

ALC_DVS.2 Sufficiency of security measures

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.5.2 ALC_LCD Life cycle definition

ALC_LCD.1 Developer defined life-cycle model

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.5.3 ALC_TAT Tools and techniques

ALC_TAT.1 Well-defined development tools

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.6 ATE Tests

The present TOE is a subset of SIMEOS TOE. A tracability document is provided to describe the link between both TOEs.

5.2.6.1 ATE_COV Coverage

ATE_COV.2 Analysis of coverage

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.6.2 ATE_DPT Depth

ATE_DPT.1 Testing: high-level design

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.6.3 ATE_FUN Functional tests

ATE_FUN.1 Functional testing

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.6.4 ATE_IND Independent testing

ATE_IND.2 Independent testing - sample

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.7 AVA Vulnerability assessment

5.2.7.1 AVA_MSU Misuse

AVA_MSU.3 Analysis and testing for insecure states

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.7.2 AVA_SOF Strength of TOE security functions

AVA_SOF.1 Strength of TOE security function evaluation

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.2.7.3 AVA_VLA Vulnerability analysis

AVA_VLA.4 Highly resistant

The details of requirements may be found in [CC-3].

The requirements already evaluated in SIMEOS are not extended.

5.3 SECURITY REQUIREMENTS FOR THE IT ENVIRONMENT

5.3.1 Core (CoreG)

This group includes some Security Functional Requirements as described by the [PP/JCS] in coreG but related to the IT environment of this TOE as described in §1.2.

The *FDP_RIP.1/TRANSIENT* requirement was substituted by that one included in *LCG*.

5.3.1.1 Firewall Policy

FDP_RIP.1/OBJECTS Subset residual information protection

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

Application note:

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

5.3.1.2 Card Security Management

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored within the TSC for **integrity errors** on all objects, based on the following attributes: **JCS integrity checked stored data**.

Non editorial refinement:

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

1. PINs
2. keys
3. application sensitive data

4. applet code.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **warn the connected entity**.

FPT_FLS.1/JCS Failure with preservation of secure state

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

FPR_UNO.1 Unobservability

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

FPT_TDC.1 Inter-TSF basic TSF data consistency

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **the CAP files (shared between the Byte Code Verifier and the TOE), the bytecode and its data arguments (shared with applets and API packages),** when shared between the TSF and another trusted IT product.

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

- **The [JCVM22] specification;**
- **Reference export files;**
- **The ISO 7816-6 rules;**
- **The [GP] specification**

when interpreting the TSF data from another trusted IT product.

FPT_TST.1 TSF testing

FPT_TST.1.1 The TSF shall run a suite of self tests **during initial start-up** to demonstrate the correct operation of **the TSF**.

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

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

Application note: "Initial start-up" means at each power on.

5.3.1.3 AID Management

FIA_UID.1/JCS Timing of identification

FIA_UID.1.1/JCS The TSF shall allow **JCAPI with already installed applets** on behalf of the user to be performed before the user is identified.

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

By users here it must be understood the ones associated to the *packages* (or *applets*), which act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected *applet* or the *package* that is the subject's owner. Means of identification are provided during the loading procedure of the *package* and the registration of *applet* instances.

FMT_SMR.1/JCS Security roles

FMT_SMR.1.1/JCS The TSF shall maintain the roles **JCRE, Operator, Card Manager, applet (RMIG)**.

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

Application note:

1. This SFR combines all *FMT_SMR.1* from [PP/JCS] except the BCV requirement:
   - *FMT_SMR.1/JCRE*;
   - *FMT_SMR.1/Installer*;
   - *FMT_SMR.1/ADEL*;
   - *FMT_SMR.1/JCRMI*
   - *FMT_SMR.1/CM*,
   - *FMT_SMR.1/CMGR*.

2. The Operator includes the *CAD* and the Bytecode Verifier (*FMT_SMR.1/BCV*).

The Card Manager (*FMT_SMR.1/CMGR*) includes the Installer (*FMT_SMR.1/Installer*) and the Applet Deletion Manager (*FMT_SMR.1/ADEL*).

FPT_RCV.3/JCS Automated recovery without undue loss

FPT_RCV.3.1/JCS When automated recovery from **a failure** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/JCS For **Applet loading, installation and deletion failure; Sensitive data loading failure**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/JCS The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **any security attribute** for loss of TSF data or objects within the TSC.

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

Application note:

1. This SFR combines *FPT_RCV.3/SCP* and *FPT_RCV.3/Installer* from [PP/JCS].

2. The TOE has no maintenance mode. Therefore all operations must be interrupted.

2. The secure state is the mute state.

3. The actual recovery services are provided by the Operating System.

5.3.1.4 *Application Programming Interface*

FCS_CKM.1/AES Cryptographic key generation

FCS_CKM.1.1/AES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **AES Key generation** and specified cryptographic key sizes **128 bits** that meet the following: **Random Generation**.

FCS_CKM.1/TDES Cryptographic key generation

FCS_CKM.1.1/TDES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **TDES Key generation** and specified cryptographic key sizes **112 bits for TDES 2 keys, 168 bits for TDES 3 keys** that meet the following: **Random Generation**.

FCS_CKM.2/AES Cryptographic key distribution

FCS_CKM.2.1/AES The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **JC API getkey()** that meets the following: **none**.

FCS_CKM.2/TDES Cryptographic key distribution

FCS_CKM.2.1/TDES The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **JC API getkey()** that meets the following: **none**.

FCS_CKM.3/AES Cryptographic key access

FCS_CKM.3.1/AES The TSF shall perform **access to AES keys** in accordance with a specified cryptographic key access method **secure reading in Memory** that meets the following: **none**.

FCS_CKM.3/TDES Cryptographic key access

FCS_CKM.3.1/TDES The TSF shall perform **access to TDES keys** in accordance with a specified cryptographic key access method **Secure reading in Memory** that meets the following: **none**.

FCS_CKM.4/AES Cryptographic key destruction

FCS_CKM.4.1/AES The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method**: physical irreversible destruction of the stored key value** that meets the following: **no standard**.

FCS_CKM.4/TDES Cryptographic key destruction

FCS_CKM.4.1/TDES The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method**: physical irreversible destruction of the stored key value** that meets the following: **no standard**.

FCS_COP.1/AES Cryptographic operation

FCS_COP.1.1/AES The TSF shall perform **AES encryption and decryption** in accordance with a specified cryptographic algorithm **AES** and cryptographic key sizes **128 bits** that meet the following: [**ALGO/AES]**.

FCS_COP.1/TDES Cryptographic operation

FCS_COP.1.1/TDES The TSF shall perform **TDES encryption and decryption** in accordance with a specified cryptographic algorithm **TDES-CBC, TDES-EBC** and cryptographic key sizes **112 bits for TDES 2 keys, 168 bits for TDES 3 keys** that meet the following: [**ALGO/DES]**.

Application note:

The TOE can also encrypt and decrypt using DES algorithm with 56 bits key, but this is to be considered as a service. The DES algorithm is no longer considered as resistant to high level attacks. The applet developer must take into account this consideration.

FCS_COP.1/SHA-1 Cryptographic operation

FCS_COP.1.1/SHA-1 The TSF shall perform **secure hashing** in accordance with a specified cryptographic algorithm **SHA-1** and cryptographic key sizes **none** that meet the following: [**ALGO/SHA1]**.

Global refinement:

This cryptographic algorithm does not use a key.

FDP_RIP.1/APDU Subset residual information protection

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

FDP_RIP.1/bArray Subset residual information protection

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

FDP_RIP.1/KEYS Subset residual information protection

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

5.3.2 Installer (InstG)

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a *package* / *applet* as described in [JCRE22] §11.1.5**.

FRU_RSA.1/Installer Maximum quotas

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

Global refinement:

Subjects here are the packages.

Application note:

1. A *package* may import at most 128 packages.

2. A *package* may declare at most 255 classes and interfaces.

3. A *class* can implement a maximum of 128 public or protected instance methods, and a maximum of 128 instance methods with *package* visibility. These limits include inherited methods.

4. A *class* instance can contain a maximum of 255 fields, where an int data type is counted as occupying two fields ([JCVM21], §2.2.4.2).

5.3.3 Additional Requirements on Logical Channels

The following element *FDP_RIP.1/TRANSIENT* substitutes the requirement from *CoreG*:

FDP_RIP.1/TRANSIENT Subset residual information protection

FDP_RIP.1.1/TRANSIENT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any transient object**.

Application note:

The events that provoke the de-allocation of any transient object are described in [JCRE22], §5.1.

The clearing of CLEAR_ON_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable *applet* instances, CLEAR_ON_DESELECT memory segments may be attached to *applet*s that are active in different logical channels. Multiselectable applet instances within a same *package* must share the transient memory segment if they are concurrently active ([JCRE22], §4.2.

5.3.4 Smart card platform (SCPG)

FPT_AMT.1/SCP Abstract machine testing

FPT_AMT.1.1/SCP The TSF shall run a suite of tests **at each Power-on** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

FPT_FLS.1/SCP Failure with preservation of secure state

FPT_FLS.1.1/SCP The TSF shall preserve a secure state when the following types of failures occur: **same as in FPT_FLS.1/JCS**.

FRU_FLT.1/SCP Degraded fault tolerance

FRU_FLT.1.1/SCP [Editorially Refined] The TSF shall ensure **all operations but package loading, applet installation and extension (need the creation of a new chained container)** when the following failures occur: **not enough memory left**.

FPT_PHP.3/SCP Resistance to physical attack

FPT_PHP.3.1/SCP [Editorially Refined] The TSF shall resist to the **physical attacks** on the **TOE** by responding automatically such that the TSP is not violated.

FPT_SEP.1/SCP TSF domain separation

FPT_SEP.1.1/SCP The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.1.2/SCP The TSF shall enforce separation between the security domains of subjects in the TSC.

Application note:

The use of "security domain" here refers to execution space, and should not be confused with other meanings of security domains.

FPT_RVM.1/SCP Non-bypassability of the TSP

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

Application note:

This component supports *OE.SCP.SUPPORT*, which in turn contributes to the secure operation of the TOE, by ensuring that these latter as well as the supporting platform security mechanisms cannot be bypassed.

FPT_RCV.4/SCP Function recovery

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

5.3.5 Card manager (CMGRG)

FDP_ACC.1/CMGR Subset access control

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

FDP_ACF.1/CMGR Security attribute based access control

FDP_ACF.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to objects based on the following:

- **Subjects: Byte Code Verifier, Operator**
- **Objects: applets, keys and PINs.**
- **Security Attributes: DAP for applets; type and KEK for keys**.

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

- **the Byte Code Verifier loads applets into the card**
- **the Operator deletes applets in the card**.

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

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

FMT_MSA.1/CMGR Management of security attributes

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

FMT_MSA.3/CMGR Static attribute initialisation

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

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

5.3.6 RMI (RMIG)

This group includes the Security Functional Requirements as described by the PP, except *FMT_SMR.1/JCRMI*. merged into *FMT_SMR.1/JCS*.

5.3.6.1 JCRMI Policy

FDP_ACC.2/JCRMI Complete access control

FDP_ACC.2.1/JCRMI The TSF shall enforce the **JCRMI access control SFP** on **S.CAD, S.JCRE, O.APPLET, O.REMOTE_OBJ, O.REMOTE_MTHD, O.ROR, O.RMI_SERVICE** and all operations among subjects and objects covered by the SFP.

Non editorial refinement:

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

S.CAD The *CAD*. In the scope of this policy it represents the actor that requests, by issuing commands to the card, for RMI services.

S.JCRE The *JCRE* is responsible on behalf of the card issuer of the bytecode execution and runtime environment functionalities. In the context of this security policy, the *JCRE* is in charge of the execution of the commands provided to (1) obtain the initial remote reference of an applet instance and (2) perform Remote Method Invocation.

O.APPLET Any installed *applet*, its code and data.

O.REMOTE_OBJ A remote object is an instance of a class that implements one (or more) remote interfaces. A remote interface is one that extends, directly or indirectly, the interface java.rmi.Remote ([JCAPI22]).

O.ROR A remote object reference. It provides information concerning: (i) the identification of a remote object and (ii) the Implementation class of the object or the interfaces implemented by the class of the object. This is the object's information to which the *CAD* can access.

O.REMOTE_MTHD A method of a remote interface.

O.RMI_SERVICE These are instances of the class javacardx.rmi.RMIService. They are the objects that actually process the RMI services.

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

Operation	**Description**
OP.GET_ROR(O.APPLET,...)	Retrieves the initial remote object reference of a RMI based *applet*. This reference is the seed which the CAD client application needs to begin remote method invocations
OP.INVOKE(O.RMI_SERVICE,...)	Request a remote method invocation on the remote object.

FDP_ACC.2.2/JCRMI The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1/JCRMI Security attribute based access control

FDP_ACF.1.1/JCRMI The TSF shall enforce the **JCRMI access control SFP** to objects based on the following: **(1) the security attributes of the covered subjects and objects, (2) the list of AIDs of the applet instances registered on the card and (3) the attribute ActiveApplets, which is a list of the active applets'AIDs**.

Non editorial refinement:

The following table presents the security attributes associated to the objects under control of the policy:

Object	Attributes
O.APPLET	Package's AID or none
O.REMOTE_OBJ	Owner, class, Identifier, Exported
O.REMOTE_MTHD	Identifier
O.RMI_SERVICE	Owner, Returned References

For *FDP_ACF.1.2:* **R.JAVA.18** The *S.CAD* may perform *OP.GET_ROR* upon an *O.APPLET* only if *O.APPLET* is the currently selected applet, and there exists an *O.RMI_SERVICE* with a registered initial reference to an *O.REMOTE_OBJ* that is owned by *O.APPLET*.

R.JAVA.19 The S.JCRE may perform OP.INVOKE upon O.RMI_SERVICE, O.ROR and O.REMOTE_MTHD, only if, O.ROR is valid (as defined in [JCRE22], §8.5) and belongs to the value of the attribute Returned References of O.RMI_SERVICE, and the attribute Identifier of O.REMOTE_MTHD matches one of the remote methods in the class, indicated by the security attribute class, of the O.REMOTE_OBJECT to which O.ROR makes reference.

FDP_ACF.1.2/JCRMI The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **by the JCRMI SFP: R.JAVA.18, R.JAVA.19**.

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

FDP_ACF.1.4/JCRMI The TSF shall explicitly deny access of subjects to objects based on the **any subject but S.JCRE to O.REMOTE_OBJ and O.REMOTE_MTHD for the purpose of performing a remote method invocation**.

FDP_IFC.1/JCRMI Subset information flow control

FDP_IFC.1.1/JCRMI The TSF shall enforce the **JCRMI information flow control SFP** on **the following subjects, information and operations (see refinement)**.

Non editorial refinement:

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

Subject/Information	Description
S.JCRE	As in the Access control policy
S.CAD	As in the Access control policy
I.RORD	Remote object reference descriptors

A remote object reference descriptor provides information concerning: (i) the identification of the remote object and (ii) the implementation class of the object or the interfaces implemented by the class of the object. The descriptor is the only object's information to which the *CAD* can access.

There is a unique operation in this policy:

Operation	Description
OP.RET_RORD(S.JCRE,S.CAD,I.RORD)	Send a remote object reference descriptor to the *CAD*.

A remote object reference descriptor is sent from the card to the *CAD* either as the result of a successful SELECT FILE command ([JCRE22], §8.4.1), and in this case it describes, if any, the initial remote object reference of the selected applet; or as the result of a remote method invocation ([JCRE22], §8.3.5.1).

FDP_IFF.1/JCRMI Simple security attributes

FDP_IFF.1.1/JCRMI The TSF shall enforce the **JCRMI information flow control SFP** based on the following types of subject and information security attributes: **S.JCRE, S.CAD, ExportedInfo**.

Non editorial refinement:

The following table summarizes which security attribute is attributed to which subject/information:

Subject/Information	Attributes
S.JCRE	None
S.CAD	None
I.RORD	ExportedInfo (Boolean value)

The ExportedInfo attribute of an *I.RORD* indicates whether the *O.REMOTE_OBJ* which *I.RORD* identifies is exported or not (as indicated by the security attribute Export of the *O.REMOTE_OBJ*).

FDP_IFF.1.2/JCRMI The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **An operation *OP.RET_RORD(S.JCRE, S.CAD, I.RORD)* is permitted only if the attribute ExportedInfo *I.RORD* has the value "true" ([JCRE22], §8.5)**.

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

FDP_IFF.1.4/JCRMI The TSF shall provide the following **none**.

FDP_IFF.1.5/JCRMI The TSF shall explicitly authorise an information flow based on the following rules: **OP.INVOKE is allowed if a successful OP.GET_ROR operation was previously successfully executed on the O.ROR supplied in OP.INVOKE and if O.ROR has not been revoked**.

FDP_IFF.1.6/JCRMI The TSF shall explicitly deny an information flow based on the following rules: **OP.INVOKE is denied if O.ROR supplied is not valid. OP.INVOKE is denied if the remote method identifier supplied with O.ROR is not the one of a method belonging to the remote object referenced by O.ROR**.

FMT_MSA.1/JCRMI Management of security attributes

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

Non editorial refinement:

The intent is to have none of the identified roles to have privileges with regards to the modification of the security attributes.

FMT_MSA.1/EXPORT Management of security attributes

FMT_MSA.1.1/EXPORT The TSF shall enforce the **JCRMI access control SFP and the JCRMI information flow control SFP** to restrict the ability to **modify** the security attributes **export of an O.REMOTE_OBJ** to **its owner**.

Application note:

The Exported status of a remote object can be modified by invoking its methods export() and unexport(), and only the owner of the object may perform the invocation without raising a SecurityException

(javacard.framework.service.CardRemoteObject). However, even if the owner of the object may provoke the change of the security attribute value, the modification itself could be performed by the JCRE.

FMT_MSA.1/REM_REFS Management of security attributes

FMT_MSA.1.1/REM_REFS The TSF shall enforce the **JCRMI access control SFP and the JCRMI information flow control SFP** to restrict the ability to **modify** the security attributes **Returned References of an O.RMI_SERVICE** to **its owner**.

FMT_MSA.3/JCRMI Static attribute initialisation

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

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

FMT_REV.1/JCRMI Revocation

FMT_REV.1.1/JCRMI [Editorially Refined] The TSF shall restrict the ability to revoke the **Returned References security attribute of an O.RMI_SERVICE** to the **JCRE**.

FMT_REV.1.2/JCRMI The TSF shall enforce the rules **that determine the lifetime of remote object references**.

Application note:

The rules previously mentioned are described in [JCRE22], §8.5.

5.3.7 Applet deletion (ADELG)

This group includes the Security Functional Requirements as described by the PP, except *FMT_SMR.1/ADEL*. merged into *FMT_SMR.1/JCS*.

5.3.7.1 Applet Deletion Manager Policy

FDP_ACC.2/ADEL Complete access control

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

Non editorial refinement:

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

S.ADEL The *applet deletion manager*. *This subject is unique.*

O.CODE_PKG The code of a *package*, including all linking information. On the Java Card platform, a package is the installation unit.

O.APPLET Any installed *applet*, its code and data.

O.JAVAOBJECT Java class instance or array.

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

Operation	Description
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed *applet* and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a *package*, either logically or physically
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a *package* and its installed *applets*, either logically or physically.

FDP_ACC.2.2/ADEL The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1/ADEL Security attribute based access control

FDP_ACF.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to objects based on the following: **(1) the security attributes of the covered subjects and objects, (2) the list of AIDs of the applet instances registered on the card, (3) the attribute ResidentPackages, which journals the list of AIDs of the packages already loaded on the card and (4) the attribute ActiveApplets, which is a list of the active applets' AIDs**.

Non editorial refinement:

The following table presents the security attributes associated to the subjects/objects under control of the policy:

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

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

When an export file is used during preparation of a CAP file, the version numbers and *AIDs* indicated in the export file are recorded in the CAP files ([JCVM21], §4.5.2): the dependent packages AIDs attribute allows the retrieval of those identifications.

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

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

The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package).

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

Finally, there are needed security attributes that are not attached to any object or subject of the TSP: (1) the ResidentPackages Versions (or Resident Image, [JCVM21],§4.5) and *AID*s. They describe the packages that are already on the card, (2) the list of registered applet instances and (3) the ActiveApplets security attribute. They are all attributes internal to the VM, that is, not attached to any specific object or subject of the SPM. These attributes are TSF data that play a role in the SPM.

FDP_ACF.1.2/ADEL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **by the ADEL SFP: see corresponding refinement**.

Non editorial refinement:

The subject of this policy is *S.ADEL*.

Some basic common specifications are required in order to allow Java Card *applet*s and *package*s to be deleted without knowing the implementation details of a particular deletion manager. In particular, this

policy introduces a notion of **reachability**, which provides a general means to describe objects that are referenced from a certain *applet* instance or *package*.

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

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

R.JAVA.14 ([JCRE22], §11.3.4.1, **Applet Instance Deletion**).Deletion). The *S.ADEL* may perform *OP.DELETE_APPLET* upon an *O.APPLET* only if, (1) *S.ADEL* is currently selected, (2) *O.APPLET* is deselected and (3) there is no *O.JAVAOBJECT* owned by *O.APPLET* such that either *O.JAVAOBJECT* is reachable from an applet instance distinct from O.APPLET, or *O.JAVAOBJECT* is reachable from a package P, or ([JCRE22], §8.5) *O.JAVAOBJECT* is remote reachable.

R.JAVA.15 ([JCRE22],§11.3.4.1, **Multiple Applet Instance Deletion**). The *S.ADEL* may perform *OP.DELETE_APPLET* upon several *O.APPLET* only if, (1) *S.ADEL* is currently selected, (2) every *O.APPLET* being deleted is deselected and (3) there is no *O.JAVAOBJECT* owned by any of the *O.APPLET* being deleted such that either *O.JAVAOBJECT* is reachable from an applet instance distinct from any of those *O.APPLET*, or *O.JAVAOBJECT* is reachable from a package P, or ([JCRE22], §8.5) *O.JAVAOBJECT* is remote reachable.

R.JAVA.16 ([JCRE22], §11.3.4.2, **Applet/Library Package Deletion**). The *S.ADEL* may perform *OP.DELETE_PCKG* upon an *O.CODE_PCKG* only if, (1) *S.ADEL* is currently selected, (2) no reachable *O.JAVAOBJECT*, from a package distinct from *O.CODE_PCKG* that is an instance of a class that belongs to *O.CODE_PCKG* exists on the card and (3) there is no package loaded on the card that depends on *O.CODE_PCKG*.

R.JAVA.17 ([JCRE22], §11.3.4.3, **Applet Package and Contained Instances Deletion**). The *S.ADEL* may perform *OP.DELETE_PCKG_APPLET* upon an *O.CODE_PCKG* only if, (1) *S.ADEL* is currently selected, (2) no reachable *O.JAVAOBJECT*, from a package distinct from *O.CODE_PCKG*, which is an instance of a class that belongs to *O.CODE_PCKG* exists on the card, (3) there is no package loaded on the card that depends on *O.CODE_PCKG* and (4) for every *O.APPLET* of those being deleted it holds that: (i) *O.APPLET* is deselected and (ii) there is no *O.JAVAOBJECT* owned by *O.APPLET* such that either *O.JAVAOBJECT* is reachable from an applet instance not being deleted, or *O.JAVAOBJECT* is reachable from a package not being deleted, or ([JCRE22],§8.5) *O.JAVAOBJECT* is remote reachable.

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

FDP_ACF.1.4/ADEL The TSF shall explicitly deny access of subjects to objects based on the**: any subject but the *S.ADEL* to *O.CODE_PKG* or *O.APPLET* must not have access for the purpose of deleting it from the card**.

FMT_MSA.1/ADEL Management of security attributes

FMT_MSA.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to restrict the ability to **modify** the security attributes **ActiveApplets** to **the JCRE**.

FMT_MSA.3/ADEL Static attribute initialisation

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

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

5.3.7.2 Additional Deletion Requirements

FDP_RIP.1/ADEL Subset residual information protection

FDP_RIP.1.1/ADEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**.

Application note:

Requirements on de-allocation during *applet* / *package* deletion are described in [JCRE22] §11.3.4.1, §11.3.4.2 and §11.3.4.3.

FPT_FLS.1/ADEL Failure with preservation of secure state

FPT_FLS.1.1/ADEL The TSF shall preserve a secure state when the following types of failures occur: **the *applet deletion manager* fails to delete a *package / applet* as described in [JCRE22], §11.3.4**.

5.3.8 Object deletion (ODELG)

FDP_RIP.1/ODEL Subset residual information protection

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

Application note:

Freed data resources resulting from the invocation of the method javacard.framework.JCSystem.requestObjectDeletion() may be reused. Requirements on *de-allocation* after the invocation of the method are described in [JCAPI22].

FPT_FLS.1/ODEL Failure with preservation of secure state

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

5.3.9 Secure carrier (CarG)

This group includes the Security Functional Requirements as described by the PP, except:

3. *FMT_SMR.1/CM* merged into *FMT_SMR.1/JCS*.

This requirement is not listed by [PP/SUN], but that is needed as a dependency for already included SFRs. [PP/SUN] seems not to be complete.

FMT_SMF.1/CM Specification of management functions

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

- **the selection of applets and opening of logical channels;**

- **the loading and the installing of the applets, with their DAP and AID by the Operator**.

Application note:

This requirement satisfies the dependencies with the *FMT_MSA.1* requirements.

FCO_NRO.2/CM Enforced proof of origin

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

FCO_NRO.2.2/CM The TSF shall be able to relate the **identity** of the originator of the information, and the **application package** of the information to which the evidence applies.

FCO_NRO.2.3/CM [Editorially Refined] The TSF shall provide a capability to verify the evidence of origin of information to the **recipient** given **that the related cryptographic keys have been loaded using a secure process**.

Non editorial refinement:

The related cryptographic keys are usually TDES keys that serve to load *applets* / *packages* through a secure channel.

Application note:

If a new application package is received by the card for installation, the card manager shall first check that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification (*S.BCV*).

If there are library packages, they are considered to be included into application packages.

FDP_IFC.2/CM Complete information flow control

FDP_IFC.2.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** on **S.CRD, S.BCV, S.SPY** and all operations that cause that information to flow to and from subjects covered by the SFP.

Non editorial refinement:

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

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

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

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

The information (prefixed with an *"I"*) controlled by the typing policy is the APDUs exchanged by the subjects through the communication channel linking the card and the *CAD*. Each of those messages contain part of an application package that is required to be loaded on the card (either S.BCV or S.SPY), as well as any control information used by the subjects in the communication protocol.

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

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

FDP_IFF.1/CM Simple security attributes

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

Subject / Information	Attribute	Value
S.BCV	DAPKey, OPKeys	Valid
S.CRD	DAPKey, OPKeys	Valid
S.SPY	none	none
I.APDU	SecureLevel	None, MAC, ENC

,

Subject / Object	Attribute	Value
user	role	Operator, Signatory, None
applet	Transfer	OTA, Local
applet	checked	Boolean
DAP Key	OK	Boolean

.

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

- **The user with the security attribute role set to Operator can load an applet with the security attribute Transfer set to OTA.**
- **The user with the security attribute role set to Service Provider can load an applet with the security attribute Transfer set to Local.**
- **Only applets with the security attribute Checked set to YES can be transferred.**
- **The DAP key OK security attribute must be set to TRUE to check the integrity and the origin of the applet**.

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

FDP_IFF.1.4/CM The TSF shall provide the following **None**.

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

- **The user with the security attribute role set to Operator can load an applet with the security attribute Transfer set to OTA.**
- **The user with the security attribute role set to Service Provider can load an applet with the security attribute Transfer set to Local.**
- **The DAP key OK security attribute must be set to TRUE to check the integrity and the origin of the applet**.

FDP_IFF.1.6/CM The TSF shall explicitly deny an information flow based on the following rules: **No user can load an applet with the security attribute Checked set to NO**.

Global refinement:

S.SPY A subject without the correct DAP attributes cannot modify the transmitted information.

FDP_UIT.1/CM Data exchange integrity

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

FDP_UIT.1.2/CM The TSF shall be able to determine on receipt of user data, whether **replay, insertion, deletion and modification** have occurred.

Non editorial refinement:

The modification, deletion, insertion, replay apply for some of the pieces of the application sent by the CAD.

FMT_MSA.1/CM Management of security attributes

FMT_MSA.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to restrict the ability to **modify** the security attributes **AIDs** to **none**.

FMT_MSA.3/CM Static attribute initialisation

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

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

FTP_ITC.1.2/CM The TSF shall permit **the remote trusted IT product** to initiate communication via the trusted channel.

Non editorial refinement:

The **remote trusted IT product** extends the *CAD* placed in the card issuer secured environment notion from the [PP/JCS] with the OTA server, belonging to the Network Operator.

FTP_ITC.1.3/CM The TSF shall initiate communication via the trusted channel for **installing a new application package on the card**.

Application note:

There is no dynamic package loading on the Java Card platform. New packages can be installed on the card only on demand of the Card issuer or Network operator.

5.3.10 Bytecode verification (BCVG)

FDP_IFC.2/BCV Complete information flow control

FDP_IFC.2.1/BCV The TSF shall enforce the **TYPING information flow control SFP** on **S.LOCVAR, S.STCKPOS, S.FLD, S.MTHD** and all operations that cause that information to flow to and from subjects covered by the SFP.

Non editorial refinement:

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

Subject	Description
S.LOCVAR	Any local variable of the currently executed method.
S.STCKPOS	Any operand stack position of the currently executed method.
S.FLD	Any field declared in a package loaded on the card.
S.MTHD	Any method declared in a package loaded on the card.

The operations (prefixed with " *OP* ") that make information flow between the subjects are all bytecodes. For instance, the *aload_0* bytecode causes information to flow from the local variable 0 to the top of the operand stack; the bytecode *putfield(x)* makes information flow from the top of the operand stack to the field x; and the *return_a* bytecode makes information flow out of the currently executed method.

Operation	Description
OP.BYTECODE(BYTCD)	Any bytecode for the Java Card platform ("Java Card bytecode").

The information (prefixed with an " *I* ") controlled by the typing policy are the bytes, shorts, integers, references and return addresses contained in the different storage units of the JCVM (local variables, operand stack, static fields, instance fields and array positions).

Information	Description
I.BYTE(BY)	Any piece of information that can be encoded in a byte.
I.SHORT(SH)	Any piece of information that can be encoded in a short value.
I.INT(W1,W2)	Any piece of information that can be encoded in an integer value, which in turn is encoded in two words w1 and w2.
I.REFERENCE(RF)	Any reference to a class instance or an array.
I.ADDRESS(ADRS)	Any return address of a subroutine.

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

FDP_IFF.2/BCV Hierarchical security attributes

FDP_IFF.2.1/BCV The TSF shall enforce the **TYPING information flow control SFP** based on the following types of subject and information security attributes: **(1) type attribute of the information, (2) type attribute of the storage units of the JCVM, (3) class attribute of the fields and methods, (4) bounds attribute of the methods**.

Non editorial refinement:

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

Subject / Information	Attributes
S.LOCVAR	*TYPE*
S.STCKPOS	*TYPE*
S.FLD	*TYPE, CLASS*
S.MTHD	*TYPE, CLASS, BOUNDS*
I.BYTE(BY)	*TYPE*
I.SHORT(SH)	*TYPE*
I.INT(W1,W2)	*TYPE*
I.REFERENCE(RF)	*TYPE*
I.ADDRESS(ADRS)	*TYPE*

The following table describes the security attributes.

Attribute Name	Description
TYPE	Either the type attached to the information, or the type held or declared by the subject.
CLASS	The class where a field or method is declared.
BOUNDS	The start and end of the method code inside the method component of the CAP file where it is declared.

The *TYPE* security attribute attached to local variables and operand stack positions is the type of information they currently hold. The *TYPE* attribute of the fields and the methods is the type declared for them by the programmer.

The *BOUNDS* attribute of a method is used to prevent control flow to jump outside the currently executed method.

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

Name	Description
TYPE	byte, short, int_1, int_2, any class name *C*, *T*[] with *T* any type in the Java Card platform ("Java Card type"), T_0 (T_1 x_1,... T_n x_n) with T_0,... T_n any Java Card type, RetAddrs(adrs), Top, Null, ⊥.
CLASS	The name of a class, represented as a reference into the class Component of one of the packages loaded on the card.
BOUNDS	Two integers marking a rank into the method component of a package loaded on the card.

Byte values have type ***byte*** and short values have type ***short***. The first and second halves of an integer value has respectively type ***int_1***, and ***int_2***. The type of a reference to an instance of the class *C* is ***C*** itself. A reference to an array of elements of type *T* has type ***T[]***. From the previous basic types it is possible to build the type T_0 (T_1 x_1,... T_n x_n) of a method. A return address *adrs* of a subroutine has type ***RetAddrss (adrs)***. Finally, the former Java Card types are extended with three extra types **Top**, **Null** and ⊥, so that the domain of types forms a complete lattice. **Top** is the type of any piece of data, that is, the maximum of the lattice. **Null** is the type of the default value null of all the reference types (classes and arrays). ⊥ is the type of an element that belongs to all types (for instance the value 0, provided that null is represented as zero).

FDP_IFF.2.2/BCV The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules, based on the ordering relationships between security attributes hold:

The following rules constitute a synthetic formulation of the information flow control:

R.JAVA.6 If the bytecode pushes values from the operand stack, then there are a sufficient number of values on the stack and the values of the attribute *TYPE* of the top positions of the stack is appropriate with respect to the ones expected by the bytecode.

R.JAVA.7 If the bytecode pushes values onto the operand stack, then there is sufficient room on the operand stack for the new values. The values, with the appropriate attribute *TYPE* value are added to the top of the operand stack.

R.JAVA.8 If the bytecode modifies a local variable with a value with attribute *TYPE* T, it must be recorded that the local variable now contains a value of that type. In addition, the variable shall be among the local variables of the method.

R.JAVA.9 If the bytecode reads a local variable, it must be ensured that the specified local variable contains a value with the attribute *TYPE* specified by the bytecode.

R.JAVA.10 If the bytecode uses a field, it must be ensured that its value is of an appropriate type. This type is indicated by the *CLASS* attribute of the field.

R.JAVA.11 If the bytecode modifies a field, then it must be ensured that the value to be assigned is of an appropriate type. This type is indicated by the *CLASS* attribute of the field

R.JAVA.12 If the bytecode is a method invocation, it must be ensured that it is invoked with arguments of the appropriate type. These types are indicated by the *TYPE* and *CLASS* attributes of the method.

R.JAVA.13 If the bytecode is a branching instruction, then the bytecode target must be defined within the *BOUNDS* of the method in which the branching instruction is defined.

FDP_IFF.2.3/BCV The TSF shall enforce the **(following additional information flow control SFP rules): none**.

FDP_IFF.2.4/BCV The TSF shall provide the following **(list of additional SFP capabilities): none.**

FDP_IFF.2.5/BCV The TSF shall explicitly authorise an information flow based on the following rules: **none**.

FDP_IFF.2.6/BCV The TSF shall explicitly deny an information flow based on the following rules: **none**.

FDP_IFF.2.7/BCV The TSF shall enforce the following relationships for any two valid information flow control security attributes:

a) There exists an ordering function that, given two valid security attributes, determines if the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable; and

b) There exists a "least upper bound" in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is greater than or equal to the two valid security attributes; and

c) There exists a "greatest lower bound" in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is not greater than the two valid security attributes.

Application note:

FDP_IFF.2.2

The rules described above are strongly inspired in the rules described in section 4.9 of [JVM], Second Edition. The complete set of typing rules can be derived from the "Must" clauses from Chapter 7 of [JCVM21] as instances of the rules defined above.

FDP_IFF.2.7

The order relationship between Java Card types is described, for instance, in the description of the **checkcast** bytecode of [JCVM22]. That relation is with the following rules:

4. **Top** is the maximum of all types;
5. **Null** is the minimum of all classes and array types;
6. ⊥ is the minimum of all types.

These three extra types are introduced in order to satisfy the two last items in requirement FDP_IFF.2.7.

FMT_MSA.1/BCV.1 Management of security attributes

FMT_MSA.1.1/BCV.1 The TSF shall enforce the **TYPING information flow control SFP** to restrict the ability to **modify** the security attributes **TYPE security attribute of the fields and methods** to **none**.

FMT_MSA.1/BCV.2 Management of security attributes

FMT_MSA.1.1/BCV.2 The TSF shall enforce the **TYPING information flow control SFP** to restrict the ability to **modify** the security attributes **TYPE security attribute of local variables and operand stack position** to **the role Bytecode Verifier**.

Application note:

1. See *FMT_SMR.1.1* for the roles.

2. *FMT_MSA.1.1/BCV.2*. The TYPE attribute of the local variables and the operand stack positions is identified to the attribute of the information they hold. Therefore, this security attribute is possibly modified as information flows. For instance, the rules of the typing function enable information to flow from a local variable *lv* to the operand stack by the operation *sload*, provided that the value of the type attribute of *lv* is ***short***. This operation hence modifies the type attribute of the top of the stack. The modification of the security attributes should be done according to the typing rules derived from Chapter 7 of [JCVM21].

FMT_MSA.2/BCV Secure security attributes

FMT_MSA.2.1/BCV The TSF shall ensure that only secure values are accepted for security attributes.

Application note:

During the type verification of a method, the bytecode verifier makes intensive use of the information provided in the CAP format like the sub-class relationship between the classes declared in the package, the type and class declared for each method and field, the rank of exceptions associated to each method, and so on. All that information can be thought of as security attributes used by the bytecode verifier, or as information relating security attributes. Moreover, the bytecode verifier relies on several properties about the CAP format. All the properties on the CAP format required by the bytecode verifier could, for instance, be completely described before starting type verifications. Examples of such properties are:

7. Correspondences between the different components of the CAP file (for instance, each class in the class component has an entry in the descriptor component).
8. Pointer soundness (example: the index argument in a static method invocation always has an entry in the constant pool);
9. Absence of hanged pointers (example: each exception handler points to the beginning of some bytecode);
10. Redundant information (enabling different ways of searching for it);
11. Conformance to the Java Language Specification respecting the access control features mentioned in §2.2 of [JCVM22].
12. Packages that are loaded post-issuance can not contain native code.

FMT_MSA.3/BCV Static attribute initialisation

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

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

Application note:

FMT_MSA.3.1 The TYPE attribute of the fields and methods is fixed by the application provider and never modified. When a method is invoked, the operand (type) stack is empty. The initial type assigned to those local variables that correspond to the method parameters is the type the application provider declared for those parameters. Any other local variable used in the method is set to the default value Top.

FMT_MSA.3.2 The intent is to have none of the identified roles to have privileges with regards to the default values of the TYPE attributes.

FMT_SMR.1/BCV Security roles

FMT_SMR.1.1/BCV The TSF shall maintain the roles **Bytecode Verifier**.

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

FRU_RSA.1/BCV Maximum quotas

FRU_RSA.1.1/BCV The TSF shall enforce maximum quotas of the following resources: **the operand stack and the local variables** that **individual user** can use **simultaneously**.

Non editorial refinement:

Individual user is **a method** here.

6. TOE SUMMARY SPECIFICATION

6.1 TOE SECURITY FUNCTIONS

The security functions of the product are those of both the hardware and the software platforms. The Security Functions that reply to the SFR of the IC are described in [ST/Infineon]. The Security Functions that reply to the SFR of the smart card platform are described in [ST/SIMEOS].

We consider in the following the security functions implemented by the subset of the JavaCard System defining the TOE of this security Target (see §1.2). The security functions that reply to the SFRs of the JCS parts not included in this TOE are described in [ST/SIMEOS].

The minimum strength for the security functions is SOF-high.

SF.Transaction

This SF manages transactions in the:

- JCVM. It enables to create Java Objects within a transaction. Transaction management includes the rollback of operations according to [JCRE221].

The TSF preserves a secure state when failures occur.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Attributes

This SF manages the values of the following security attributes: resident applets, active applets and currently selected applet. It also checks the consistency of applets' life cycle.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Firewall

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

This SF participates to information confidentiality.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Install

It modifies the CAP files in a safe way and performs coherency checks on the CAP files. It verifies the export references of the packages upon linking them[1].

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.JCRE

The *JCRE* owns JCRE entry points objects of which methods may be accessed by any context. Their fields may only be accessed by the JCRE context.

The reference of temporary JCRE entry point objects and global arrays cannot be stored in class variables, instance fields or array components.

[1] This security function takes care of the linking part of the installing process described in [PP/JCS]..

The JCRE is the only one allowed to create JCRE entry point objects and global arrays and to define them as temporary or permanent.

The *JCRE* supplies transient memory management through JC System services.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Applet

The SF defines the behaviour of each Java Card application through a strictly defined interface named Applet class and to the management of the *AID*s. Each new application inherits its behaviour and the associated constraints from the Applet class model.

This SF realises applet code interpretation. From the interpretation point of view, the *applet* 's code is considered as data to read.

This SF participates to information confidentiality, information integrity and availability.

This SF does not use probabilistic or permutational effects.

This function has no strength.

6.2 ASSURANCE MEASURES

6.2.1 ASE

[STF_SIMEOS] Formal Assurance on SIMEOS JVM: Security Target

We reused the results of SIMEOS evaluation as described in [ST_SIMEOS]

6.2.2 ACM

[ACMF_SIMEOS]

SIMEOS Configuration management (Extension)

We reused the results of SIMEOS evaluation as using the deliverables [AUT_SIMEOS], [CAP_SIMEOS], [SCP_SIMEOS].

6.2.3 ADO

We reused the results of SIMEOS evaluation using the deliverables [DEL_SIMEOS] and [IGS_SIMEOS].

6.2.4 ADV

[FSPFi_SIMEOS] SIMEOS JVM Formal Functional Specification of the Interpreter
[FSPFl_SIMEOS] SIMEOS JVM Formal Functional Specification of the Linker
[FSPFa_SIMEOS] SIMEOS JVM Formal Functional Specification of the API

[HLDFi_SIMEOS] SIMEOS JVM Formal High level design of the Interpreter
[HLDFl_SIMEOS] SIMEOS JVM Formal High level design of the Linker
[HLDFa_SIMEOS] SIMEOS JVM Formal High level design of the API

[RCR(FSP-HLD)Fi_SIMEOS] SIMEOS JVM Formal correspondence FSP-HLD for the Interpreter
[RCR(FSP-HLD)Fl_SIMEOS] SIMEOS JVM Formal correspondence FSP-HLD for the Linker
[RCR(FSP-HLD)Fa_SIMEOS] SIMEOS JVM Formal correspondence FSP-HLD for the API

[LLDFi_SIMEOS] SIMEOS JVM Formal Low-level Design of the Interpreter
[LLDFl_SIMEOS] SIMEOS JVM Formal Low-level Design of the Linker
[LLDFa_SIMEOS] SIMEOS JVM Formal Low-level Design of the API

[RCR(HLD-LLD)Fi_SIMEOS] SIMEOS JVM Formal correspondence HLD-LLD for the Interpreter
[RCR(HLD-LLD)Fl_SIMEOS] SIMEOS JVM Formal correspondence HLD-LLD for the Linker
[RCR(HLD-LLD)Fa_SIMEOS] SIMEOS JVM Formal correspondence HLD-LLD for the API

[INTF_SIMEOS] SIMEOS JVM Modularity (INT.3)

[SPMFi_SIMEOS] SIMEOS JVM formal Model of the Firewall security policy
[SPMFTSP_SIMEOS] SIMEOS JVM formal Model of the Java card Virtual Machine
[SPMFa_SIMEOS]SIMEOS JVM Formal Model of the APIs

[SPMFCl_SIMEOS] SIMEOS JVM formal correspondence between the FSP and TSP Model for the linker
[SPMFCi_SIMEOS] SIMEOS JVM formal correspondence between the FSP and TSP Model for the Interpreter
[SPMFCa_SIMEOS] SIMEOS JVM formal correspondence between the FSP and TSP Model for the APIs

[RCR(TSS-FSP)l_SIMEOS] SIMEOS JVM Representation correspondence between the TSS and the FSP Model for the linker
[RCR(TSS-FSP)i_SIMEOS] SIMEOS JVM formal correspondence between the TSS and FSP Model for the Interpreter
[RCR(TSS-FSP)a_SIMEOS] SIMEOS JVM formal correspondence between the TSS and FSP Model for the APIs

6.2.5 AGD

We reused the results of SIMEOS evaluation using the deliverables [USR_SIMEOS] and [AGD_SIMEOS].

6.2.6 ALC

We reused the results of SIMEOS evaluation using the deliverables [DVS_SIMEOS] and [TAT_SIMEOS].

We extend the results of SIMEOS evaluation using the deliverables [LCD_SIMEOS] by [LCDF_SIMEOS].

6.2.7 ATE

We reused the results of SIMEOS evaluation using the deliverables [COV_SIMEOS], [FUN_SIMEOS] and [DPT_SIMEOS] and [IND_SIMEOS] as Independent testing – sample.

[ATE_LINK_SIMEOS] SIMEOS JVM tracability between Tests and Formal Models

6.2.8 AVA

We reused the results of SIMEOS evaluation using the deliverables [MSU_SIMEOS], [SOF_SIMEOS] and [VLA_SIMEOS].

7. PP CLAIMS

[PP/JCS] Java Card™ System, Protection Profile Collection, Version 1.0b

7.1 PP REFERENCE

No PP is claimed here. However, the present TOE is a subset of the Java Card TOE defined in [PP/JCS] for the standard 2.2 configuration. The groups defined in the PP are sets of identified security requirements.

The groups of security requirements of the present TOE are (a part of) the Core (CoreG), Installer (InstG) and Logical channels (LCG).

The IT environment of the TOE is made of (the complementary part of) the Core (CoreG), the smart card platform (SCPG) upon which the TOE is implemented, the Byte-code verification (BCVG) for applet verification before the download and the Card Manager (CMGR) for card content management, Object deletion (ODELG) for garbage collection and applet deletion (ADELG) for erasing applet from the card.

7.2 PP ADDITIONS

Additional Security Functional Requirements

The following SFR have been added to the ST main groups of SFRs compared to [PP/JCS]:

FMT_SMF.1/CM. It is required as dependency by *FMT_MSA.1*.

Operations that have been completed on SFRs from [PP/JCS]:

CoreG:

- Firewall Policy: *FDP_IFF.1/JCVM*;
- Application Programming Interface: *FCS_CKM.1*, *FCS_CKM.2*, *FCS_CKM.3*, *FCS_CKM.4* and *FCS_COP.1* have been instantiated with the algorithms available on the card.
- Card Security Management: *FDP_SDI.2*, *FPR_UNO.1*.

RMI (RMIG):

- JCRMI Policy: *FDP_IFF.1/JCRMI*

Secure carrier (CarG):

- FCO_NRO.2/CM, FDP_IFF.1/CM, FMT_MSA.1/CM.

Smart card platform (SCPG):

- FPT_FLS.1/SCP, FRU_FLT.1/SCP, FPT_PHP.3/SCP.

Card manager (CMGRG):

- FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR.

The following SFRs have been merged with respect to [PP/JCS] : FIA_UID.1/CM and FIA_UID.1/CMGR merged into FIA_UID.1/JCS

Additional Assurance requirements

[PP/JCS] requires EAL4 augmented with:

- ADV_IMP.2
- AVA_VLA.3.

Additionally, this ST requires:

- ADV_IMP.3
- AVA_VLA.4
- AVA_MSU.3
- ALC_DVS.2
- ADV_FSP.4
- ADV_HLD.5
- ADV_LLD.2
- ADV_INT.3
- ADV_RCR.3
- ADV_SPM.3

8. RATIONALE

8.1 SECURITY OBJECTIVES RATIONALE

Not delivered in public version.

8.2 SECURITY REQUIREMENTS RATIONALE

Not delivered in public version.

8.3 TOE SUMMARY SPECIFICATION RATIONALE

Not delivered in public version.