MultiApp V3
Security Target

UPDATES

Date	Author	Modification
6th May 13	Gemalto	Creating from evaluated ST (V1.2)

CONTENT

FIGURES

TABLES

1 SECURITY TARGET INTRODUCTION

1.1 SECURITYTARGET IDENTIFICATION

Title:	MultiApp V3 JCS Security Target
Version:	1.2p
ST reference:	ST_D1184308
Origin:	Gemalto
Product identification:	MultiApp ID V3
Security Controllers:	M7820 A11 SLE78CLX1600P
TOE identification:	Open Platform on MultiApp V3
TOE documentation:	Operational User Guidance [OPE_JCS] Preparative procedures [PRE_JCS]

The TOE identification is provided by the Card Production Life Cycle Data (CPLCD) of the TOE, located in OTP and in EEPROM. These data are available by executing a dedicated command.

The TOE and the product differ, as further explained in §2

- The TOE is the JCS open platform of the MultiApp V3 product.
- The MultiApp V3 product also includes 2 applets and 1 native application in ROM.

CPLC field	Length	Value
IC Fabricator	2	IFX
IC Type	2	M7820 A11 SLE78CLX1600P
Operating System Identifier	2	n.a.
Operating System release date	2	n.a.
Operating System release level	2	n.a.
IC Fabrication Date	2	n.a.
IC Serial Number	4	Unique identification of the chip written by the ICC Manufacturer
IC Batch Identifier	2	n.a.
IC Module Fabricator	2	n.a.
IC Module Packaging Date	2	n.a.
ICC Manufacturer	2	‘Gemalto’
IC Embedding Date	2	n.a.
IC Pre-personalizer	2	‘Gemalto’

CPLC field	Length	Value
IC Pre-personalization Date	2	n.a.
IC Pre-personalization Equipment Identifier	4	n.a.
IC Personalizer	2	n.a.
IC Personalization Date	2	n.a.
IC Personalization Equipment Identifier	4	n.a.

Table 1: Card Production Life Cycle Data

IT Security Evaluation scheme: Serma Technologies
IT Security Certification scheme: Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)

1.2 SECURITY TARGET OVERVIEW

This TOE provides the security of an EAL5+ evaluated card with the flexibility of an open platform.
It allows for the loading of applets before or after the issuance of the card. These applets MAY or MAY NOT be evaluated on this platform.
The applications using only certified applets will BE certified even if NOT-certified applets are loaded on the platform.
The applications using a NOT-certified applet will NOT BE certified.

The Issuer can forbid the loading of applets before or after the issuance of the card.
The Java Card technology combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices [JCVM222]. The Java Card platform is a smart card platform enabled with Java Card technology (also called a "Java card"). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications for the Java Card platform ("Java Card applications") are called applets.

The main objectives of this ST are:

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

1.3 REFERENCES

1.3.1 External References

[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, CCMB-2009-07-001, version 3.1 rev 4, September 2012

[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, CCMB-2009-07-002, version 3.1 rev 4, September 2012
[CC-3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, CCMB-2009-07-003, version 3.1 rev 4 September 2012
[CEM]	Common Methodology for Information Technology Security Evaluation Methodology CCMB-2009-07-004, version 3.1 rev 4 September 2012
[ST-IC]	[ST-IC-M7820]
[ST-IC-M7820]	ST of M7820 A11 & M11 SLE78CLX1600P -*Version 1.6* – 2012-08-28
[CR-IC]	[CR-IC-M7820]
[CR-IC-M7820]	*Certification Report, SLE78CLX1600P* / M7820 A11 & M11 *BSI-DSZ-CC-0829-2012*
[FIPS180-2]	*Federal Information Processing Standards Publication 180-2 SECURE HASH STANDARD (+Change Notice to include SHA-224),* U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 2002 August 1
[FIPS197]	*Federal Information Processing Standards Publication 197 ADVANCED ENCRYPTION STANDARD (AES),* 2001 November 26
[SP800-67]	NIST Special Publication 800-67 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher – version 1 – May 2004
[TR-03111]	Technical Guideline TR-03111, Elliptic Curve Cryptography based on ISO 15946, v1.00 Bundesamt für Sicherheit in der Informationstechnik
[ISO15946-1]	*ISO/IEC 15946: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General,* 2002
[ISO15946-2]	*ISO/IEC 15946: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 2: Digital Signatures,* 2002
[ISO15946-3]	*ISO/IEC 15946: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 3: Key establishment,* 2002
[ISO7816]	*ISO 7816, Identification cards – Integrated circuit(s) cards with contacts, Part 4: Organization, security and commands for interchange*, FDIS2004
[ISO9796-2]	*ISO/IEC 9796: Information technology – Security techniques – Digital Signature Schemes giving message recovery – Part 2: Integer factorisation based mechanisms,* 2002
[ISO9797-1]	*ISO/IEC 9797: Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher,* 1999
[IEEE-P1363]	Standard Specifications for Public Key Cryptography, Institute of Electrical and Electronic Engineers, 2000 : IEEE 1363
[PKCS#3]	*PKCS #3: Diffie-Hellman Key-Agreement Standard,* An RSA Laboratories Technical Note, Version 1.4, Revised November 1, 1993
[GP211]	Global Platform Card Specification v 2.1.1 - March 2003
[GP221]	GlobalPlatform Card Technology Secure Channel Protocol 03 Card Specification v 2.2 – Amendment D Version 1.0 Public Release April 2009

[JAVASPEC]	The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[JCBV]	Java Card Platform, version 2.2 Off-Card Verifier. June 2002. White paper. Published by Sun Microsystems, Inc.
[JCRE222]	Java Card 2.2.2 Runtime Environment (JCRE) Specification – 15 March 2006 - Published by Sun Microsystems, Inc.
[JCVM222]	Java Card 2.2.2 Virtual Machine (JCVM) Specification – 15 March 2006 - Published by Sun Microsystems, Inc.
[JCAPI222]	Java Card 2.2.2 Application Programming Interface - March 2006 - Published by Sun Microsystems, Inc.
[PP-IC-0035]	Security IC Platform Protection Profile – BSI-PP-0035
[PP-JCS-Open]	Java Card System Protection Profile – Open Configuration ANSSI-PP-2010-03M01, Version 3.0, May 2012
[PP-MRTD-EAC]	Machine Readable Travel Document with „ICAO Application", Extended Access Control with PACE (EAC PP) PP-BSI-0056-V2-2012-v131 Version 1.3.1, 22th March 2012
[PP-MRTD-BAC]	Machine Readable Travel Document with „ICAO Application", Basic Access Control BSI-CC-PP-0055 Version 1.10, 25th March 2009
[PP-MRTD-PACE]	Machine Readable Travel Document using Standard Inspection Procedure with PACE(PACE PP) PP-BSI-0068-V2-2011 Version 1.0, 2nd November 2011
[RGS-B1]	Référentiel Général de sécurité version 1.0: Annexe B1 Mécanismes cryptographiques, règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques; version 1.20 du 26 janvier 2010
[PP-SSCD-T2]	*Protection Profile – Secure Signature-Creation Device Type2* BSI-PP-0005, Version 1.04, 25th July 2001
[PP-SSCD-T3]	*Protection Profile – Secure Signature-Creation Device Type3* BSI-PP-0006, Version 1.05, 25th July 2001
[PP-SSCD-KG]	Protection profiles for secure signature creation device - Part 2: Device with key generation BSI-CC-PP-0059-2009-MA-01 version 2.0.1 21 February 2012
[PP-SSCD-KI]	Protection profiles for secure signature creation device – Part3: Device with key import BSI-CC-PP-0075-2012, Version 1.02, July 2012

1.3.2 Internal References

[PRE_JCS]	Preparation Guidance
[OPE_JCS]	Operational Guidance

1.4 ACRONYMS AND GLOSSARY

AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAD	Card Acceptance Device
CC	Common Criteria
CPU	Central Processing Unit
DES	Data Encryption Standard

EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
EEPROM	Electrically-Erasable Programmable Read-Only Memory
ES	Embedded Software
GP	Global Platform
IC	Integrated Circuit
IT	Information Technology
JCRE	JavaCard Runtime Environment
JCS	JavaCard System
JCVM	JavaCard Virtual Machine
NVM	Non-Volatile Memory
OP	Open Platform
PIN	Personal Identification Number
PP	Protection Profile
RMI	Remote Method Invocation
RNG	Random Number Generator
ROM	Read-Only Memory
RSA	Rivest Shamir Adleman
SAR	Security Assurance Requirement
SC	Smart Card
SCP	Secure Channel Protocol
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
TOE	Target Of Evaluation
TSF	TOE Security Functionality

2 TOE DESCRIPTION

2.1 ARCHITECTURE OF THE SMARTCARD CONTAINING THE TOE

The TOE is part of the MultiApp ID V3.0 smartcard. This smartcard contains the software dedicated to the operation of:

- The MultiApp ID V3.0 javacard platform, which supports the execution of the personalized applets and provides the smartcard administration services.
- The identity applets: IAS Classic V4, MOCA Server 1.0, and eTravel 2.0 native application..
- Additionally, other applets – not determined at the moment of the present evaluation – may be loaded on the smartcard before or after issuance.

Therefore, the architecture of the smartcard software and application data can be represented as follows:

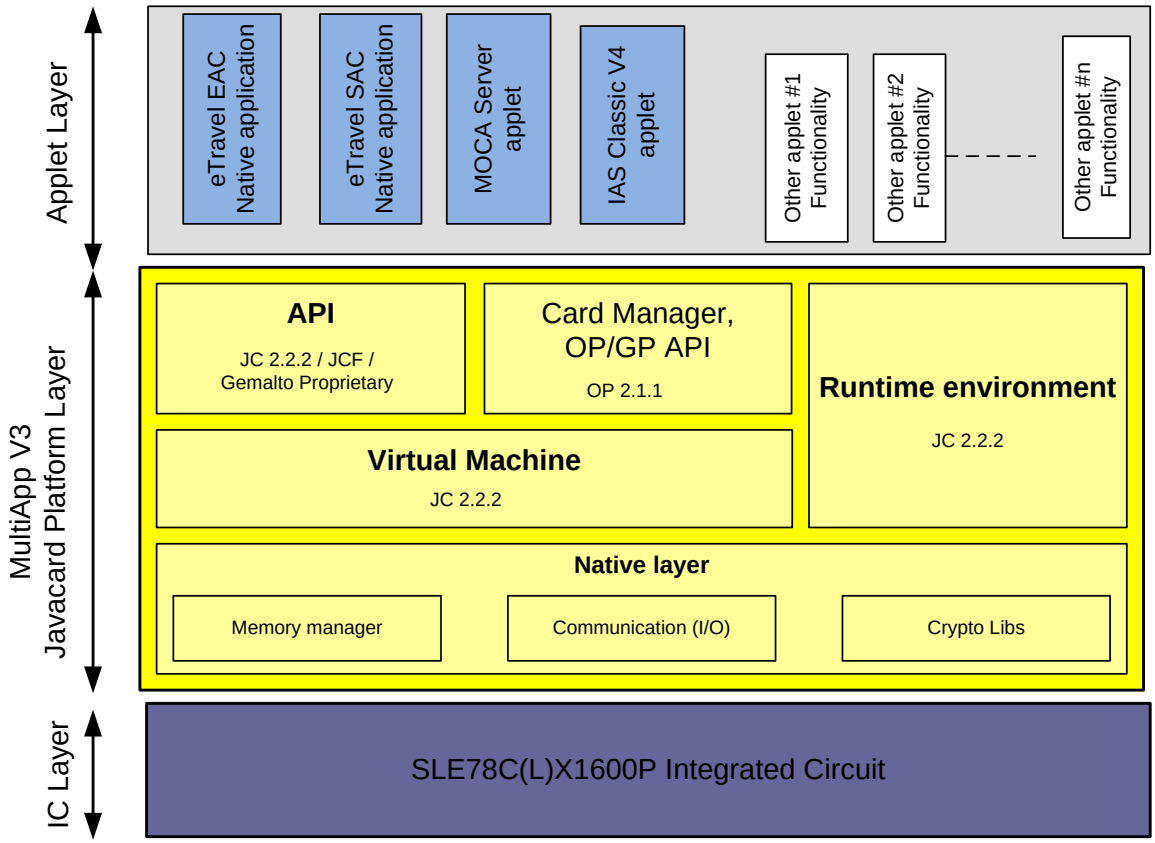

Figure 1: MultiApp ID V3.0 smartcard architecture

The IAS Classic V4 applet, eTravel 2.0 native application, MOCA server1.0 applet and the MultiApp V3.0 javacard platform, are located in ROM.
Any additional applet's executable code (loaded before or after issuance) would also be located in EEPROM, which might also contain any software patch that would be needed in the future.
All the data (related to the applets or to the javacard platform) are located in EEPROM. The separation between these data is ensured by the javacard firewall as specified in [JCRE222].

2.2 TOE BOUNDARIES

The Target of Evaluation (TOE) is the JCS open platform of the MultiApp V3 product. It is defined by:

- The Java Platform 2.2.2 based on JLEP Operating System;
- The underlying Integrated Circuit;

The applications eTravel EAC, eTravel SAC, IAS classic V4 and MOCA, stored in ROM in MultiApp V3 are outside the TOE.
The Applets loaded pre issuance or post issuance are outside the TOE, Other smart card product elements, (such as holograms, magnetic stripes, security printing,…) are outside the scope of this Security Target.

Java Card RMI is not implemented in the TOE.

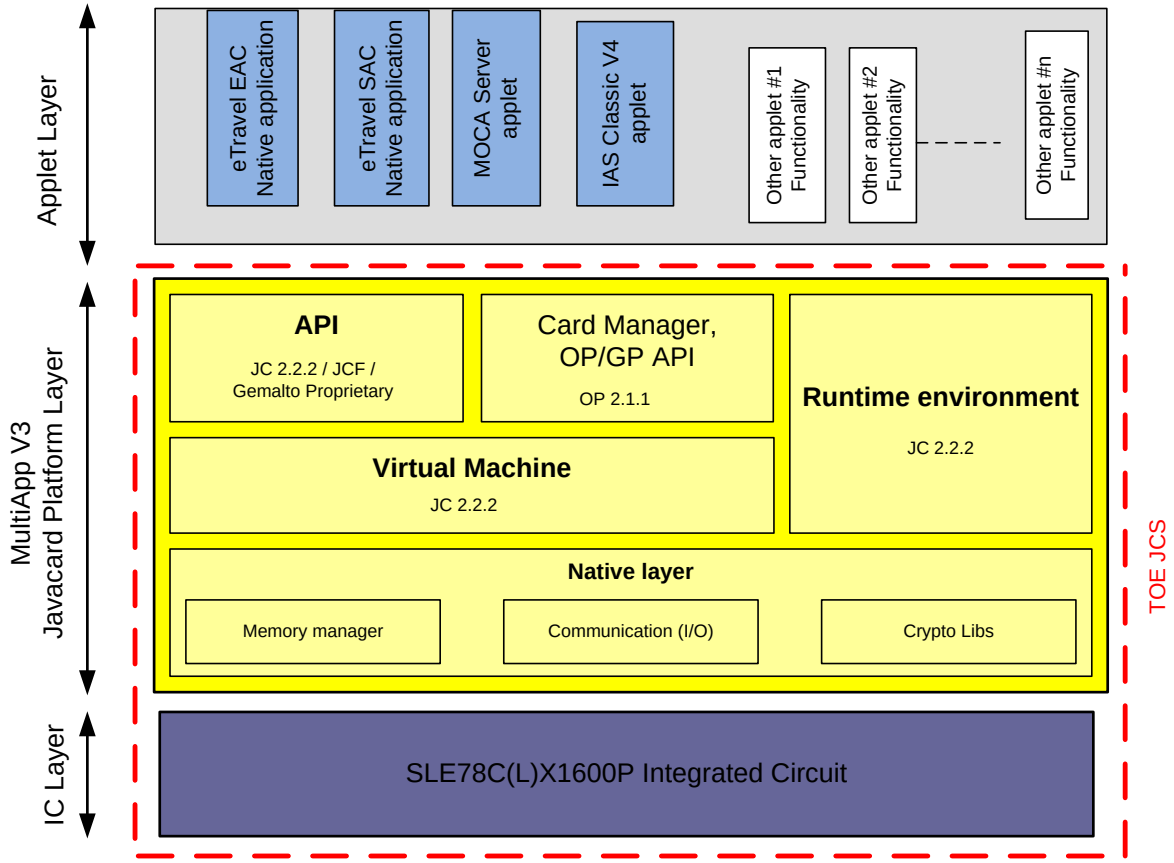

Figure 2: JCS TOE boundaries

2.3 MULTIAPP ID V3.0 JAVACARD PLATFORM DESCRIPTION

The MultiApp ID V3.0 platform is a smart card operating system that complies with two major industry standards:

- Sun's Java Card 2.2.2, which consists of the Java Card 2.2.2 Virtual Machine [JCVM222], the Java Card 2.2.2 Runtime Environment [JCRE222] and the Java Card 2.2.2 Application Programming Interface [JCAPI222].
- The Global Platform Card Specification version 2.1.1 [GP211].

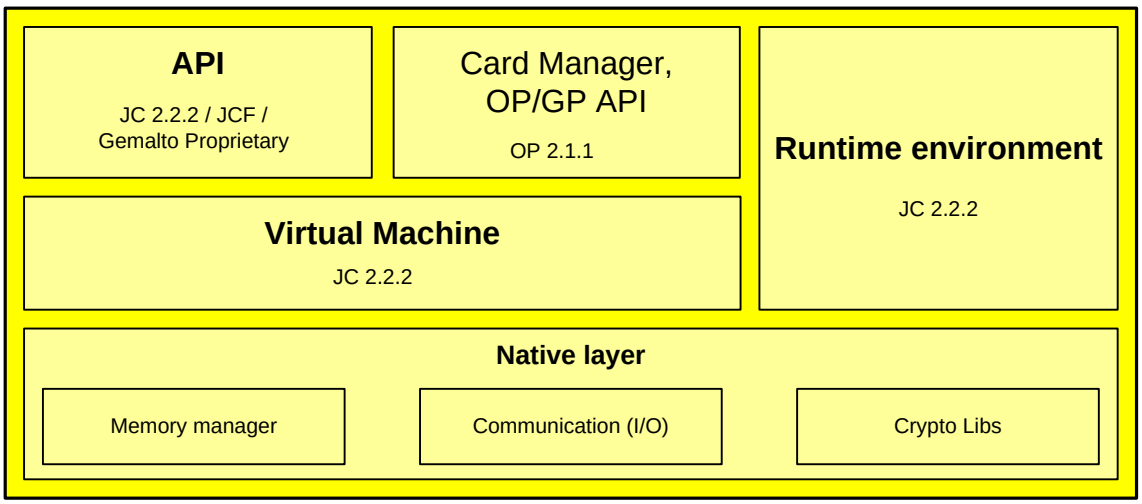

Figure 3: MultiApp ID V3.0 javacard platform architecture

As described in figure 3, the MultiApp ID V3.0 platform contains the following components:

- The **Native Layer**

  It provides the basic card functionalities (memory management, I/O management and cryptographic primitives) with native interface with the underlying IC. The cryptographic features implemented in the native layer encompass the following algorithms:

  - DES, Triple-DES
  - RSA up to 4096 (CRT method), 2048 (Std method)
  - DH up to 2048
  - AES 128, 192, 256
  - SHA1, SHA224, SHA256, SHA384, SHA512
  - ECC up to P521 (Standard JC2.2.2 method)
  - Pseudo-Random Number Generation (PRNG) & Software random.

- The **Javacard Runtime Environment**

  It conforms to [JCRE222] and provides a secure framework for the execution of the Java Card programs and data access management (firewall).
  Among other features, multiple logical channels are supported, as well as extradition, DAP, Delegated management, SCP01, SCP02 and SCOP3.

- The **Javacard Virtual Machine**

It conforms to [JCVM222] and provides the secure interpretation of bytecodes.

- The **API**

It includes the standard javacard API [JCAPI222] and the Gemalto proprietary API.

- The **Open Platform Card Manager**

It conforms to [GP211] and provides card, key and applet management functions (contents and life-cycle) and security control.

The MultiApp ID V3.0 platform provides the following services:

- Initialization of the Card Manager and management of the card life cycle
- Secure loading and installation of the applets under Card Manager control
- Deletion of applications under Card Manager control
- Extradition services to allow several applications to share a dedicated security domain
- Secure operation of the applications through the API
- Management and control of the communication between the card and the CAD
- Card basic security services as follows:
    - Checking environmental operating conditions using information provided by the IC
    - Checking life cycle consistency
    - Ensuring the security of the PIN and cryptographic key objects
    - Generating random numbers
    - Handling secure data object and backup mechanisms
    - Managing memory content
    - Ensuring Java Card firewall mechanism

2.4 LIFE-CYCLE

2.4.1 Product Life-cycle

2.4.1.1 Actors

Actors	Identification
Integrated Circuit (IC) Developer	IFX
Embedded Software Developer	Gemalto
Integrated Circuit (IC) Manufacturer	IFX
Initializer	Gemalto or IFX

Actors	Identification
Pre-personalizer	Gemalto or IFX
Inlay manufacturer	Gemalto or another Inlay manufacturer
Personalization Agent	The agent who is acting on the behalf of the issuing State or Organization and personalize the MRTD for the holder by activities establishing the identity of the holder with biographic data.
Card Holder	The rightful holder of the card for whom the issuer personalizes it.

Table 2: Identification of the actors

2.4.1.2 *Initialisation on module at Gemalto site*

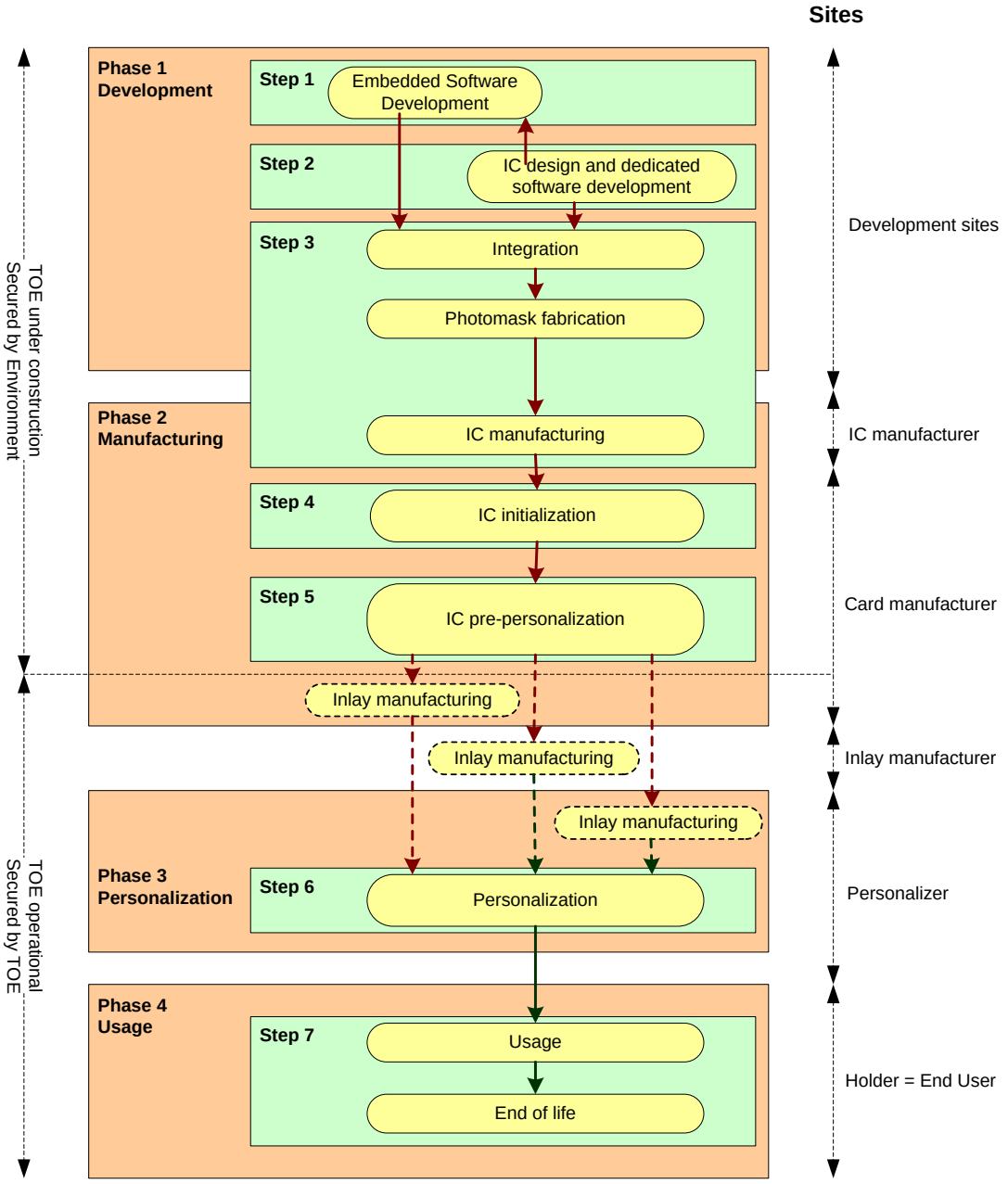

Figure 4: LC1: Init on module at Gemalto site

Figure 4: LC1: Init on module at Gemalto site~~Figure 4: LC1: Init on module at Gemalto site~~ describes the standard Life Cycle. The module is manufactured at the founder site. It is then shipped to Gemalto site where it is initialized and pre-personalized and then shipped to the Personalizer directly or after through the Inlay manufacturer (optional step).
During the shipment from Gemalto to the Personalizer, the module is protected by a diversified key.

2.4.1.3 *Initialization at Founder site*

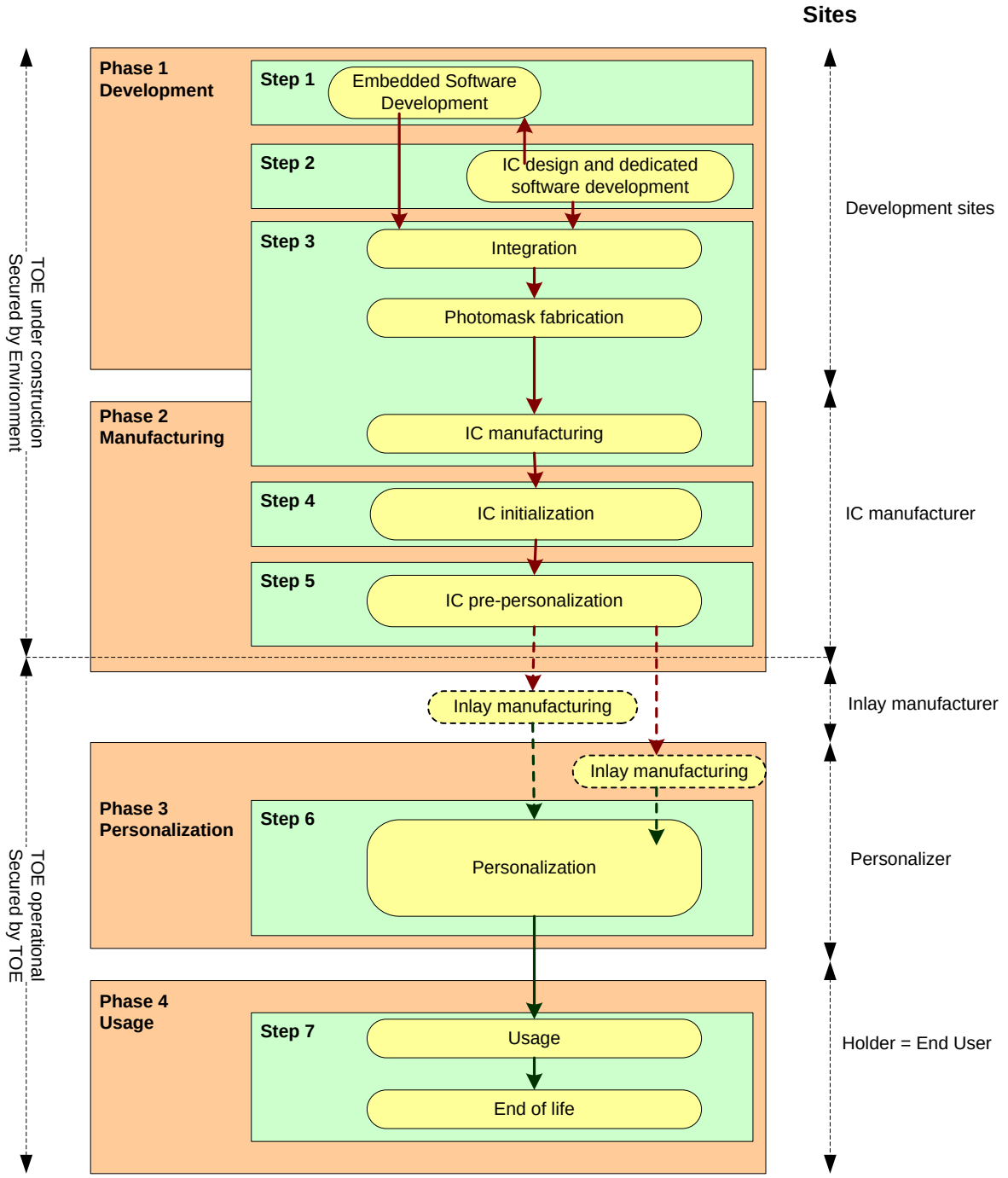

Figure 5: LC2 Init on module at Founder site

LC2 is an alternative to LC1. Figure 5: LC2 Init on module at Founder site~~Figure 5: LC2 Init on module at Founder site~~ describes the Life Cycle when the customer whishes to receive wafers directly from the founder. In this case, initialization and pre-personalization, which include sensitive operations such as the loading of patches, take place at the founder site.
During the shipment from the founder to the Personalizer, the module is protected by a diversified key.
Inlay manufacturing step is optional.

2.4.1.4 *Initialization on inlay at Gemalto site*

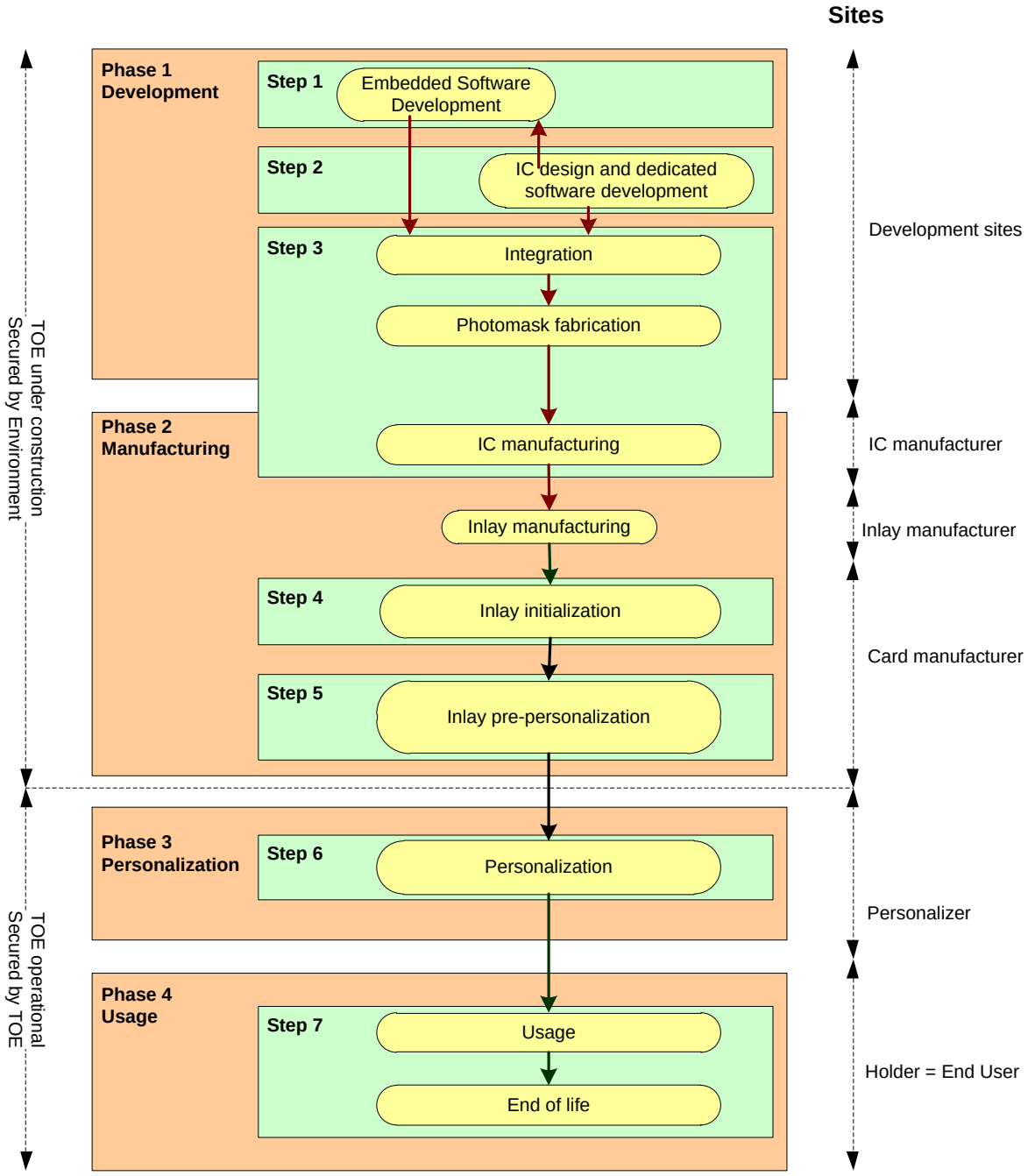

Figure 6: LC3: Init on inlay at Gemalto site

LC3 is another alternative to LC1. Figure 6: LC3: Init on inlay at Gemalto site~~Figure 6: LC3: Init on inlay at Gemalto site~~ describes the Life Cycle when Gemalto whishes to receive inlays instead of modules from the founder. In this case, the founder ships the module to the Inlay manufacturer.
During the shipment from the founder to Gemalto, the module is protected by a diversified key.

2.4.2 TOE Life-cycle

The Java Card System (the TOE) life cycle is part of the product life cycle, i.e. the Java Card platform with applications, which goes from product development to its usage by the final user.

The Java Card System (i.e. the TOE) life-cycle itself can be decomposed in four stages:

- Development
- Storage, pre-personalization and testing
- Personalization and testing
- Final usage

The JCS storage is not necessarily a single step in the life cycle since it can be stored in parts. The JCS delivery occurs before storage and may take place more than once if the TOE is delivered in parts.
These four stages map to the product life cycle phases as shown in Figure 6.

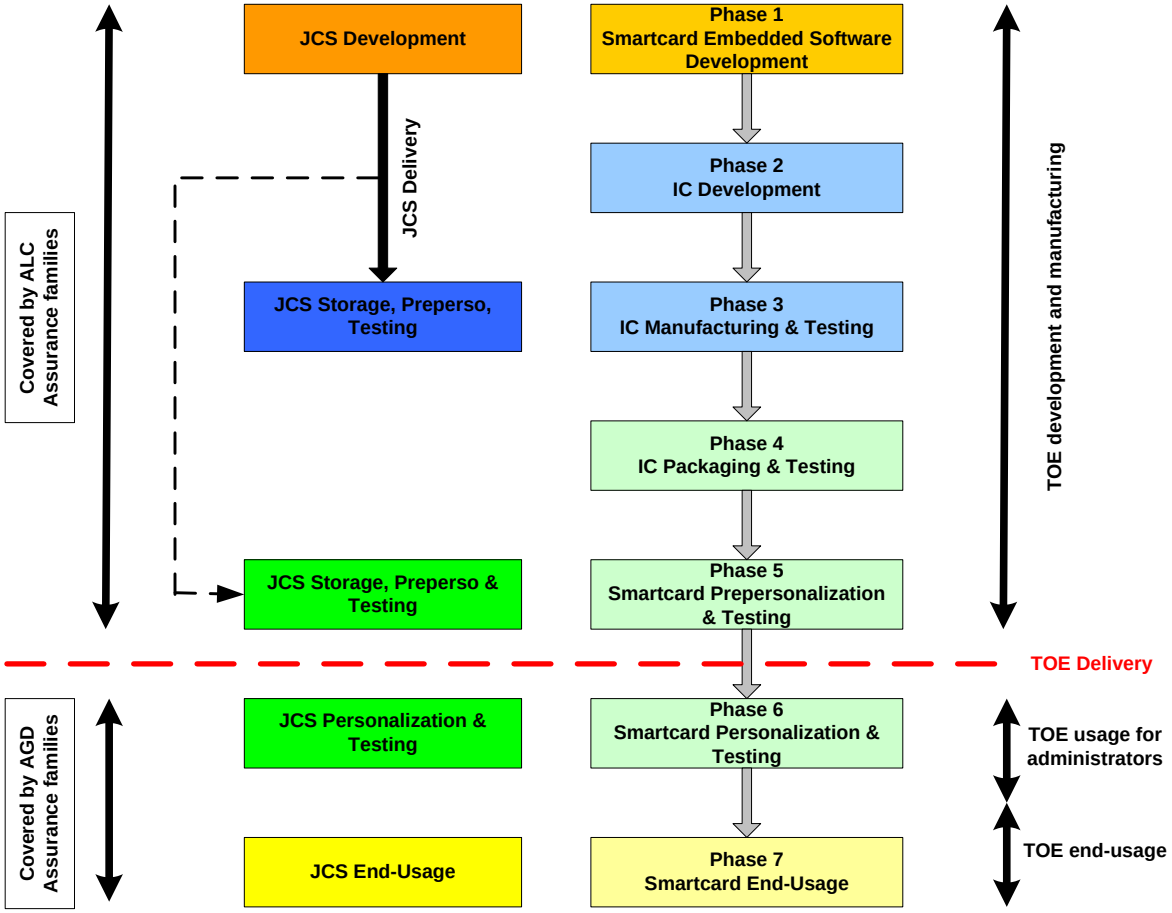

Figure 7: JCS (TOE) Life Cycle within Product Life Cycle

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

In Phase 3, the IC Manufacturer may store, initialize the JCS and potentially conduct tests on behalf of the JCS developer. The IC Manufacturing environment shall protect the integrity and confidentiality of the JCS and of any related material, for instance test suites. The present evaluation includes the whole IC

Manufacturing environment, in particular those locations where the JCS is accessible for installation or testing. As the Security IC has already been certified against [PP/0035] there is no need to perform the evaluation again.

In Phase 5, the SC Pre-Personalizer may store, pre-personalize the JCS and potentially conduct tests on behalf of the JCS developer. The SC Pre-Personalization environment shall protect the integrity and confidentiality of the JCS and of any related material, for instance test suites.

(Part of) JCS storage in Phase 5 implies a TOE delivery after Phase 5. Hence, the present evaluation includes the SC Pre-Personalization environment. The TOE delivery point is placed at the end of Phase 5, since the entire TOE is then built and embedded in the Security IC.

The JCS is personalized in Phase 6, if necessary. The SC Personalization environment is not included in the present evaluation. Appropriate security recommendations are provided to the SC Personalizer through the [AGD] documentation.

The JCS final usage environment is that of the product where the JCS is embedded in. It covers a wide spectrum of situations that cannot be covered by evaluations. The JCS and the product shall provide the full set of security functionalities to avoid abuse of the product by untrusted entities.

Note: Potential applications loaded in pre-issuance will be verified using dedicated evaluated verification process. Applications loaded in post-issuance will need to follow dedicated development rules.

2.4.3 GP Life-cycle

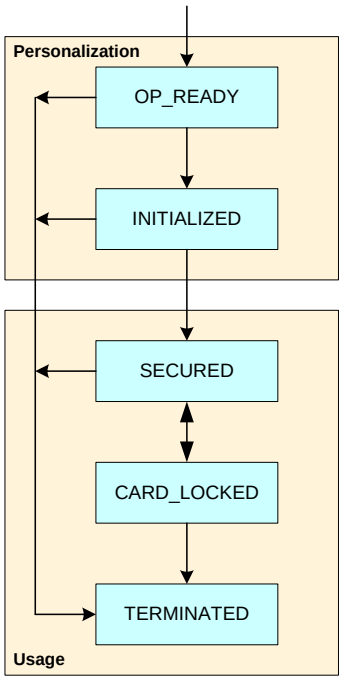

Figure 8: GP Life Cycle

2.5 TOE INTENDED USAGE

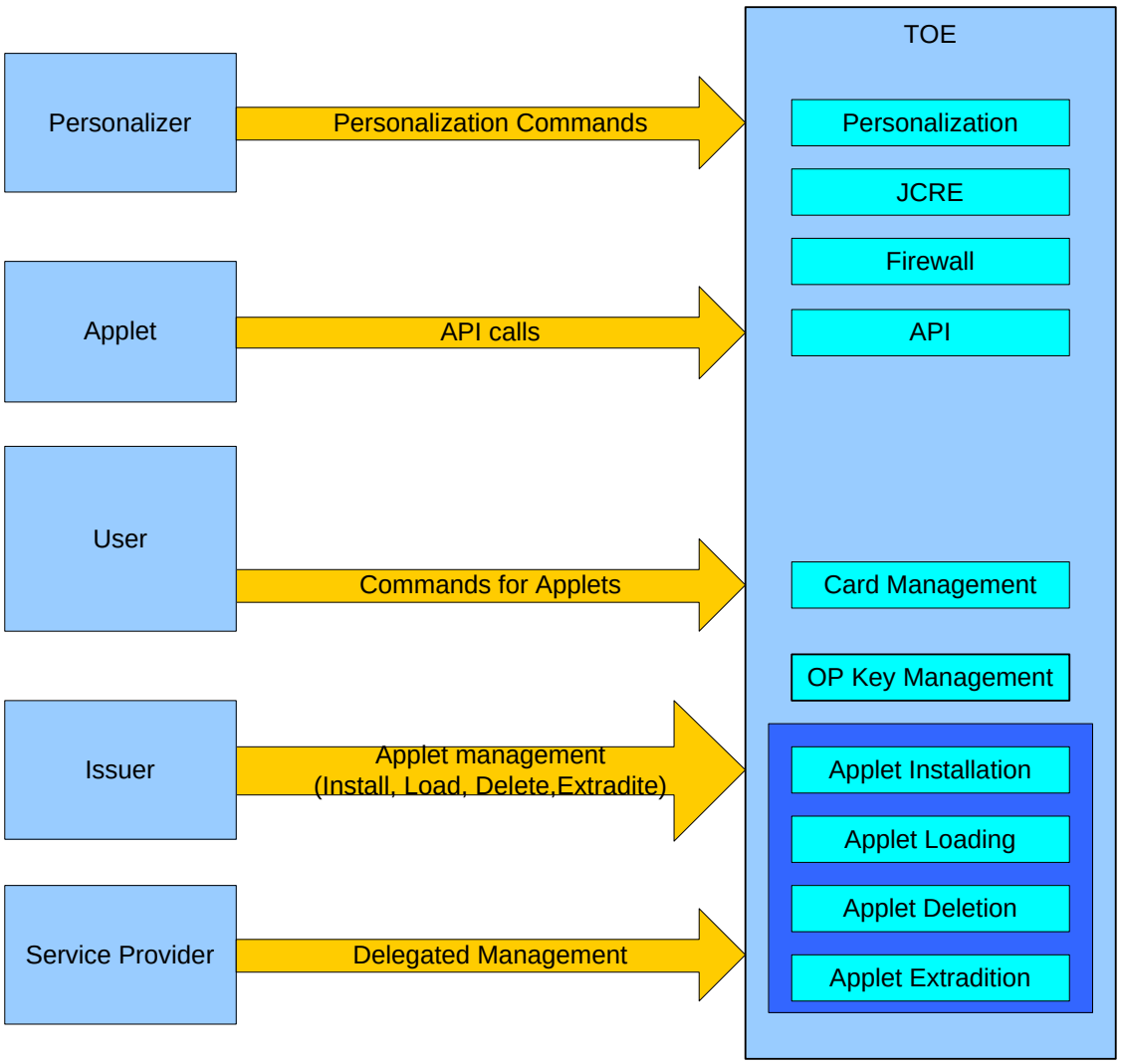

Table 3 TOE operations

2.5.1.1 Personalization Phase

During the Personalization Phase the following Administrative Services are available:

- Applet Load
- Applet Install
- Applet Delete
- Applet Extradite
- Applet Management Lock

All applet management operations require the authentication of the Issuer. By erasing the authentication keys with random numbers, the Issuer can prevent all subsequent applet management operations. This operation is not reversible.
In the Personalization phase, Applet Management Lock is optional.

2.5.1.2 Usage Phase

During the Usage Phase, if the Applet Management lock has not been put, the Administrative Services are available as during the Personalization phase:

- Applet Load
- Applet Install
- Applet Delete
- Applet Extradite
- Applet Management Lock

In addition, the following User services are available:

- Applet Selection
- Applet Interface

3 CONFORMANCE CLAIMS

3.1 CC CONFORMANCE CLAIM

Common criteria Version:

This ST conforms to CC Version 3.1 [CC-1] [CC-2] [CC-3]

Conformance to CC part 2 and 3:

- CC part 2 extended with the FCS_RND.1 family. All the other security requirements have been drawn from the catalogue of requirements in Part 2 [CC-2].
- CC part 3 conformant

The Common Methodology for Information Technology Security Evaluation, Evaluation Methodology; [CEM] has to be taken into account.

3.2 PP CLAIM

The MultiApp V3 JCS security target claims strict conformance to the Protection Profile “JavaCard System – Open configuration”, ([PP-JCS-Open]).

The MultiApp V3 JCS security target is a composite security target, including the IC security target [ST-IC]. However the security problem definition, the objectives, and the SFR of the IC are not described in this document.

The TOE also claims conformance to other Protection Profiles. This is described in other Security Targets:
The MultiApp V3 EAC security target claims strict conformance to the Protection Profile “Machine Readable Travel Document with ICAO Application, Extended Access Control” ([PP-MRTD-EAC]).
The MultiApp V3 BAC security target claims strict conformance to the Protection Profile “Machine Readable Travel Document with ICAO Application, Basic Access Control” ([PP-MRTD-BAC]).
The MultiApp V3 PACE security target claims strict conformance to the Protection Profile “Machine Readable Travel Document using Standard Inspection Procedure with PACE” ([PP-MRTD-PACE]).
The MultiApp V3 IAS Classic security target claims demonstrable conformance to the Protection Profiles related to “Secure Signature-creation Device” : [PP-SSCD-T2],[PP-SSCD-T3], [PP-SSCD-KG] and [PP-SSCD-KI].

3.3 PACKAGE CLAIM

This ST is conforming to assurance package EAL5 augmented with ALC_DVS.2 and AVA_VAN.5 defined in CC part 3 [CC-3].

3.4 CONFORMANCE STATEMENT

This ST strictly conforms to [PP-JCS-Open]. The conformance is explained in the rationale.

4 SECURITY ASPECTS

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

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

TSFs must be continuously active in one way or another; this is called “OPERATE”.

4.1 CONFIDENTIALITY

#.CONFID-APPLI-DATA	Application data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain read access to other application’s data.
#.CONFID-JCS-CODE	Java Card System code must be protected against unauthorized disclosure. Knowledge of the Java Card System code may allow bypassing the TSF. This concerns logical attacks at runtime in order to gain a read access to executable code, typically by executing an application that tries to read the memory area where a piece of Java Card System code is stored.
#.CONFID-JCS-DATA	Java Card System data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain a read access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card platform API classes as well.

4.2 INTEGRITY

#.INTEG-APPLI-CODE	Application code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to the memory zone where executable code is stored. In post-issuance application loading, this threat also concerns the modification of application code in transit to the card.
#.INTEG-APPLI-DATA	Application data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain unauthorized write access to application data. In post-issuance application loading, this threat also concerns the modification of application data contained in a package in transit to the card. For instance, a package contains the values to be used for initializing the static fields of the package.
#.INTEG-JCS-CODE	Java Card System code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to executable code.
#.INTEG-JCS-DATA	Java Card System data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card API classes as

well.

4.3 UNAUTHORIZED EXECUTIONS

#.EXE-APPLI-CODE	Application (byte)code must be protected against unauthorized execution. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code.;
#.EXE-JCS-CODE	Java Card System bytecode must be protected against unauthorized execution. Java Card System bytecode includes any code of the Java Card RE or API. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.
#.FIREWALL	The Firewall shall ensure controlled sharing of class instances, and isolation of their data and code between packages (that is, controlled execution contexts) as well as between packages and the JCRE context. An applet shall neither read, write nor compare a piece of data belonging to an applet that is not in the same context, nor execute one of the methods of an applet in another context without its authorization.
#.NATIVE	Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside the TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

4.4 BYTECODE VERIFICATION

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

4.4.1 CAP file Verification

Bytecode verification includes checking at least the following properties: (1) bytecode instructions represent a legal set of instructions used on the Java Card platform; (2) adequacy of bytecode operands to bytecode semantics; (3) absence of operand stack overflow/underflow; (4) control flow confinement to the current method (that is, no control jumps to outside the method); (5) absence of illegal data conversion and reference forging; (6) enforcement of the private/public access modifiers for class and class members; (7) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (8) enforcement of rules for binary compatibility (full details are given in [JCVM222], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [JCVM222] on the bytecodes and the correctness of the CAP files’ format.

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

Another important aspect to be considered about bytecode verification and application downloading is, first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM222] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

4.4.2 Integrity and Authentication

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

Once a verification authority has verified the package, it signs it and sends it to the card. Prior to the installation of the package, the card verifies the signature of the package, which authenticates the fact that it has been successfully verified. In addition to this, a secured communication channel is used to communicate it to the card, ensuring that no modification has been performed on it.

Alternatively, the card itself may include a verifier and perform the checks prior to the effective installation of the applet or provide means for the bytecodes to be verified dynamically. On-card bytecode verifier is out of the scope of this Security Target.

4.4.3 Linking and Verification

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

4.5 CARD MANAGEMENT

#.CARD-MANAGEMENT	(1) The card manager (CM) shall control the access to card management functions such as the installation, update or deletion of applets. (2) The card manager shall implement the card issuer's policy on the card.
#.INSTALL	(1) The TOE must be able to return to a safe and consistent state should the installation of a package or an applet fail or be cancelled (whatever the reasons). (2) Installing an applet must have no effect on the code and data of already installed applets. The installation procedure should not be used to bypass the TSFs. In short, it is an atomic operation, free of harmful effects on the state of the other applets. (3) The procedure of loading and installing a package shall ensure its integrity and authenticity.
#.SID	(1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security

Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a package or any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.

#OBJ-DELETION

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

#DELETION

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

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

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

4.6 SERVICES

#.ALARM

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

#.OPERATE

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

#.RESOURCES

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

#.CIPHER	The TOE shall provide a means to the applications for ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.
#.KEY-MNGT	The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified cryptographic key destruction methods.
#.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.
#.SCP	The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well defined set of policies and standards (for instance, those specified in [PP-IC-0035]), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.
#.TRANSACTION	The TOE must provide a means to execute a set of operations atomically. This mechanism must not jeopardise the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

5 SECURITY PROBLEM DEFINITION

5.1 ASSETS

The assets of the TOE are those defined in [PP-JCS-Open]. The assets of [PP-SC] are studied in [ST-IC].

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

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

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

5.1.1 User data

D.APP_CODE

The code of the applets and libraries loaded on the card.
To be protected from unauthorized modification.

D.APP_C_DATA

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

D.APP_I_DATA

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

D.APP_KEYs

Cryptographic keys owned by the applets.
To be protected from unauthorized disclosure and modification.

D.PIN

Any end-user's PIN.
To be protected from unauthorized disclosure and modification.

5.1.2 TSF data

D.API_DATA

Private data of the API, like the contents of its private fields.
To be protected from unauthorized disclosure and modification.

D.CRYPTO

Cryptographic data used in runtime cryptographic computations, like a seed used to generate a key.
To be protected from unauthorized disclosure and modification.

D.JCS_CODE
The code of the Java Card System.
To be protected from unauthorized disclosure and modification.

D.JCS_DATA
The internal runtime data areas necessary for the execution of the Java Card VM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated for an array, any pointer used to chain data-structures.
To be protected from monopolization and unauthorized disclosure or modification.

D.SEC_DATA
The runtime security data of the Java Card RE, like, for instance, the AIDs used to identify the installed applets, the currently selected applet, the current context of execution and the owner of each object.
To be protected from unauthorized disclosure and modification.

5.2 THREATS

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. The threats are classified in several groups.

5.2.1 Confidentiality

T.CONFID-APPLI-DATA
The attacker executes an application to disclose data belonging to another application. See #.CONFID-APPLI-DATA for details.
Directly threatened asset(s): **D.APP_C_DATA, D.PIN,** and **D.APP_KEYs**.

T.CONFID-JCS-CODE
The attacker executes an application to disclose the Java Card System code. See #.CONFID-JCS-CODE for details.
Directly threatened asset(s): **D.JCS_CODE**.

T.CONFID-JCS-DATA
The attacker executes an application to disclose data belonging to the Java Card System. See #.CONFID-JCS-DATA for details.
Directly threatened asset(s): **D.API_DATA, D.SEC_DATA, D.JCS_DATA,** and **D.CRYPTO**.

5.2.2 Integrity

T.INTEG-APPLI-CODE
The attacker executes an application to alter (part of) its own code or another application's code. See #.INTEG-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**

T.INTEG-APPLI-CODE.LOAD
The attacker modifies (part of) its own or another application code when an application package is transmitted to the card for installation. See #.INTEG-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**.

T.INTEG-APPLI-DATA
The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): **D.APP_I_DATA, D.PIN,** and **D.APP_KEYs**.

T.INTEG-APPLI-DATA.LOAD
The attacker modifies (part of) the initialization data contained in an application package when the package is transmitted to the card for installation. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): **D.APP_I_DATA** and **D_APP_KEYs**.

T.INTEG-JCS-CODE
The attacker executes an application to alter (part of) the Java Card System code. See #.INTEG-JCS-CODE for details.
Directly threatened asset(s): **D.JCS_CODE**.

T.INTEG-JCS-DATA
The attacker executes an application to alter (part of) Java Card System or API data. See #.INTEG-JCS-DATA for details.
Directly threatened asset(s): **D.API_DATA, D.SEC_DATA, D.JCS_DATA,** and **D.CRYPTO**.

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

5.2.3 Identity usurpation

T.SID.1
An applet impersonates another application, or even the Java Card RE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal. See #.SID for details.
Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN and D.APP_KEYs.

T.SID.2
The attacker modifies the TOE's attribution of a privileged role (e.g. default applet and currently selected applet), which allows illegal impersonation of this role. See #.SID for further details.
Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

5.2.4 Unauthorized execution

T.EXE-CODE.1
An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**.

T.EXE-CODE.2
An applet performs an execution of a method fragment or arbitrary data. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**.

T.NATIVE
An applet executes a native method to bypass a security function such as the firewall. See #.NATIVE for details.
Directly threatened asset(s): **D.JCS_DATA**.

5.2.5 Denial of Service

T.RESOURCES

An attacker prevents correct operation of the Java Card System through consumption of some resources of the card: RAM or NVRAM. See #.RESOURCES for details.
Directly threatened asset(s): **D.JCS_DATA**.

5.2.6 Card management

T.DELETION

The attacker deletes an applet or a package already in use on the card, or uses the deletion functions to pave the way for further attacks (putting the TOE in an insecure state). See #.DELETION (p 343) for details).
Directly threatened asset(s): **D.SEC_DATA** and **D.APP_CODE**.

T.INSTALL

The attacker fraudulently installs post-issuance of an applet on the card. This concerns either the installation of an unverified applet or an attempt to induce a malfunction in the TOE through the installation process. See #.INSTALL for details.
Directly threatened asset(s): **D.SEC_DATA** (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

5.2.7 Services

T.OBJ-DELETION

The attacker keeps a reference to a garbage collected object in order to force the TOE to execute an unavailable method, to make it to crash, or to gain access to a memory containing data that is now being used by another application. See #.OBJ-DELETION for further details.
Directly threatened asset(s): **D.APP_C_DATA, D.APP_I_DATA** and **D.APP_KEYs**.

5.2.8 Miscellaneous

T.PHYSICAL

The attacker discloses or modifies the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means. This threat includes IC failure analysis, electrical probing, unexpected tearing, and DPA. That also includes the modification of the runtime execution of Java Card System or SCP software through alteration of the intended execution order of (set of) instructions through physical tampering techniques.
This threatens all the identified assets.
This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

5.3 ORGANIZATIONAL SECURITY POLICIES

5.3.1 Java Card System Protection Profile – Open Configuration

This section describes the organizational security policies to be enforced with respect to the TOE environment.

OSP.VERIFICATION

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

If the application development guidance provided by the platform developer contains recommandations related to the isolation property of the platform, this policy shall also ensure that the verification authority checks that these recommendations are applied in the application code.

5.3.2 TOE additional OSP

OSP.SpecificAPI

The TOE must contribute to ensure that application can optimize control on its sensitive operations using a dedicated API provided by TOE. TOE will provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

OSP.RNG

This policy shall ensure the entropy of the random numbers provided by the TOE to applet using [JCAPI222] is sufficient. Thus attacker is not able to predict or obtain information on generated numbers.

5.4 ASSUMPTIONS

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

A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM222], §3.3) outside the API.

A.DELETION

Deletion of applets, if available through the card manager, is secure. Refer to #.DELETION for details on this assumption.

The rationale for this latter assumption is that even a Java Card System 2.1.1 TOE could be installed on a product that includes applet deletion features. This assumes that these functions are secure with respect to the SFRs herein.

A.VERIFICATION

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

5.5 COMPATIBILITY BETWEEN SECURITY ENVIRONMENTS OF [ST-JCS] AND [ST-IC]

5.5.1 Compatibility between threats of [ST-JCS] and [ST-IC]

T.CONFID-JCS-CODE, T.CONFID-APPLI-DATA, and T.CONFID-JCS-DATA are included in T.Phys-Probing, T.Leak-Inherent and T.Leak-Forced.

T.SID.2 is partly included in T.Phys-Manipulation and T.Malfunction.

T.PHYSICAL is included in T.Phys-Probing, T.Leak-Inherent, T.Phys-Manipulation, T.Malfunction and T.Leak-Forced.

T.INTEG-APPLI-CODE T.INTEG-JCS-CODE T.INTEG-APPLI-DATA DATA T.INTEG-JCS-DATA T.INTEG-APPLI-CODE.LOAD T.INTEG-APPLI-DATA.LOAD T.SID.1 T.EXE-CODE.1 T.EXE-CODE.2 T.NATIVE

T.RESOURCES T.INSTALL T.DELETION T.OBJ-DELETION are threats specific to the Java Card platform and they do no conflict with the threats of [ST-IC].

5.5.2 Compatibility between OSP of [ST-JCS] and [ST-IC]

OSP.VERIFICATION is an OSP specific to the Java Card platform and it does no conflict with the OSP of [ST-IC].
OSP.SpecificAPI has been added to this [ST-JCS] in order to manage Specific API and it does no conflict with the OSP of [ST-IC].
OSP.RND has been added to this [ST-JCS] in order to manage RNG and it does no conflict with the OSP of [ST-IC].

We can therefore conclude that the OSP for the environment of [ST-JCS] and [ST-IC] are consistent.

5.5.3 Compatibility between assumptions of [ST-JCS] and [ST-IC]

A.VERIFICATION, A.DELETION, and A.APPLET are assumptions specific to the Java Card platform and they do no conflict with the assumptions of [ST-IC].
We can therefore conclude that the assumptions for the environment of [ST-JCS] and [ST-IC] are consistent.

6 SECURITY OBJECTIVES

6.1 SECURITY OBJECTIVES FOR THE TOE

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

6.1.1 Identification

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

6.1.2 Execution

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

O.GLOBAL_ARRAYS_CONFID
The TOE shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection.
The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

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

O.NATIVE
The only means that the Java Card VM shall provide for an application to execute native code is the invocation of a method of the Java Card API, or any additional API. See #.NATIVE (p 341) for details.

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

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

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 [JCVM222].

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

6.1.3 Services

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

O.CIPHER

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

O.KEY-MNGT
The TOE shall provide a means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys. See #.KEY-MNGT.

Application note:

O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION and O.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries can also be present on the card and made available to applets; those may be built on top of the Java Card API or independently. Depending on whether they contain native code or not, these proprietary libraries will need to be evaluated together with the TOE or not (see #.NATIVE). In any case, they are not included in the Java Card System for the purpose of the present document.

O.PIN-MNGT
The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT for details.
Application note:
PIN objects may play key roles in the security architecture of client applications. The way they are stored and managed in the memory of the smart card must be carefully considered, and this applies to the whole object rather than the sole value of the PIN. For instance, the try counter's value is as sensitive as that of the PIN.

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

6.1.4 Object deletion

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

6.1.5 Applet management

O.DELETION
The TOE shall ensure that both applet and package deletion perform as expected. (See #.DELETION for details).

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

O.INSTALL
The TOE shall ensure that the installation of an applet performs as expected. (See #.INSTALL for details).
Besides, for codes loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. If not performed during the loading process, this verification by the TOE shall occur during the install process.

6.1.6 SCP

The Objectives described in this section are Objectives for the Environment in [PP-JCS-Open]. They become Objectives for the TOE because the TOE in this ST includes the SCP.

O.SCP.RECOVERY
If there is a loss of power, or if the smart card is withdrawn from the CAD while an operation is in progress, the SCP must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state.
This security objective for the environment refers to the security aspect #.SCP.1: The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state.

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

O.SCP.IC
The SCP shall provide all IC security features against physical attacks.
This security objective for the environment refers to the point (7) of the security aspect #.SCP:
It is required that the IC is designed in accordance with a well-defined set of policies and Standards (likely specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

6.1.7 CMGR

The Objectives described in this section are Objectives for the Environment in [PP-JCS-Open]. They become Objectives for the TOE because the TOE in this ST includes the Card Manager.

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

6.1.8 Additional objectives

Objectives described in this section are additional objectives related to the TOE.

O.SpecificAPI
The TOE shall provide to application a specific API means to optimize control on sensitive operations performed by application.
TOE shall provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

O.RND
The TOE must contribute to ensure that random numbers shall not be predictable and shall have sufficient entropy.

6.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

6.2.1 Objectives extracted from [PP-JCS-Open]

This section introduces the security objectives to be achieved by the environment and extracted from [PP-JCS-Open].

OE.VERIFICATION
All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details.

Additionally the applet shall follow all recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.
Application Note:
Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

OE.APPLET
No applet loaded post-issuance shall contain native methods.

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

7 EXTENDED COMPONENTS DEFINITION

7.1 DEFINITION OF THE FAMILY FCS_RND

To define the IT security functional requirements of the TOE a sensitive family (FCS_RND) of the Class FCS (cryptographic support) is defined here. This family describes the functional requirements for random number generation used for cryptographic purposes. The component FCS_RND is not limited to generation of cryptographic keys unlike the component FCS_CKM.1. The similar component FIA_SOS.2 is intended for non-cryptographic use.

The family “Generation of random numbers (FCS_RND)” is specified as follows.

FCS_RND Generation of random numbers

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

Component levelling:

FCS_RND Generation of random numbers	1

FCS_RND.1 Generation of random numbers requires that random numbers meet a defined quality metric.

Management: FCS_RND.1
There are no management activities foreseen.

Audit: FCS_RND.1
There are no actions defined to be auditable.

FCS_RND.1 Quality metric for random numbers

Hierarchical to: No other components
Dependencies: No dependencies

FCS_RND.1.1 The TSF shall provide a mechanism to generate random numbers that meet [assignment: *a defined quality metric*].

8 SECURITY REQUIREMENTS

8.1 SECURITY FUNCTIONAL REQUIREMENTS

This section states the security functional requirements for the Java Card System – Open configuration.

Group	Description
Core with Logical Channels (CoreG_LC)	The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature. This group is the union of requirements from the Core (CoreG) and the Logical channels (LCG) groups defined in [PP/0305]. (cf Java Card Sysytem Protection Profile Collection [PP JCS]).
Installation (InstG)	The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.
Applet deletion (ADELG)	The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 2.2.
Object deletion (ODELG)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.
Secure carrier (CarG)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecodes verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.
Smart Card Platform (SCPG)	The SCPG group contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon.
Card Manager (CMGRG)	The CMGRG group contains the security requirements for the card manager.
Additional SFR (ASFR)	The ASFR group contains security requirements related to specific API and to random generation

The SFRs refer to all potentially applicable subjects, objects, information, operations and security attributes.

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

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE222], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §8.1.3~~7.1.3~~.
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §8.1.6~~7.1.6~~.

Subject	Description
S.CAD	The CAD represents off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.
S.JCRE	The runtime environment un which Java programs in a smart card are executed.
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
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.
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.

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

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

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

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

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

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected"
Applet's version number	The version number of an applet (package) indicated in the export file
Class	Identifies the implementation class of the remote object.
Context	Package AID, or "Java Card RE"
Currently Active Context	Package AID, or "Java Card RE"
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM222], §4.5.2).
ExportedInfo	Boolean (Indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies a remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package). The owner of a remote object is the

Security attribute	Description/Value
	applet instance that created the object.
Package AID	The AID of each package indicated in the export file
Registered applets	The set of AID of the applet instance registered on the card
ResidentPackages	The set of AIDs of the packages already loaded on the card
Selected Applet Context	Package AID, or "None"
Sharing	Standards, SIO, Java Card RE entry point, or global array
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

(*) Transient objects of type CLEAR_ON_RESET behave like persistent objects in that they can be accessed only when the Currently Active Context is the object's context.
Operations (prefixed with "OP") are described in the following table. Each operation has 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.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.
OP.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.JAVA(...)	Any access in the sense of [JCRE222], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE222],§6.2.8.7)
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).

(*) For this operation, there is no accessed object. This rule enforces that shareable transient objects are not allowed. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

8.1.1 CoreG_LC Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall. This policy essentially concerns the security of installed applets. The policy focuses on the execution of bytecodes.

8.1.1.1 Firewall Policy

FDP_ACC.2/FIREWALL Complete access control

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

Refinement:
The operations involved in the policy are:

- OP.CREATE,
- OP.INVK_INTERFACE,
- OP.INVK_VIRTUAL,
- OP.JAVA,
- OP.THROW,
- OP.TYPE_ACCESS.

FDP_ACC.2.2/FIREWALL The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.
Application note:
Accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT.

FDP_ACF.1/FIREWALL Security attribute based access control

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

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

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

- R.JAVA.1 ([JCRE222]§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 ([JCRE222]§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 ([JCRE222]§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.4 ([JCRE222], §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 the invoked interface method extends the Shareable interface and one of the following applies:

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

  (b) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable», and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets.
- R.JAVA.5 An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".

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

1) **The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**
2) **The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method (through OP.INVK_INTERFACE or OP.INVK_VIRTUAL).**

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

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

Application note:

In the case of an array type, fields are components of the array ([JVM], §2.14, §2.7.7), as well as the length; the only methods of an array object are those inherited from the Object class.
The Sharing attribute defines four categories of objects:
• Standard ones, whose both fields and methods are under the firewall policy,
• Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
• JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,
• Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.

When a new object is created, it is associated with the Currently Active Context. But the object is owned by the applet instance within the Currently Active Context when the object is instantiated ([JCRE222], §6.1.3). An object is owned by an applet instance, by the JCRE or by the package library where it has been defined (these latter objects can only be arrays that initialize static fields of packages).

([JCRE222], Glossary) Selected Applet Context. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.
While the expression "Selected Applet Context" refers to a specific installed applet, the relevant aspect to the policy is the context (package AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected package.
([JCRE222], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

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

The Java Card platform, version 2.2.x introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM222], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([JCRE222], §4).

FDP_IFC.1/JCVM Subset information flow control

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

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

FDP_IFF.1/JCVM Simple security attributes

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

Subject / Information	Description
S.JCVM	Currently active context.

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

- **An operation OP.PUT (S1, S.MEMBER, I) is allowed if and only if the active context is "Java Card RE";**
- **Other OP.PUT operations are allowed regardless of the Currently Active Context's value**.

FDP_IFF.1.3/JCVM The TSF shall enforce **no additional information flow control SFP rules**.

FDP_IFF.1.4/JCVM The TSF shall explicitly authorize an information flow based on the following rules: **no additional information flow control SFP rules**.

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **no additional information flow control SFP rules**.

FDP_RIP.1/OBJECTS Subset residual information protection

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

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP** to restrict the ability to **modify** the security attributes **the selected applet Context security attribute** to **the Java Card RE (S.JCRE)**.
Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE222], §4 and [JCVM222], §3.4.

FMT_MSA.1/JCVM Management of security attributes

FMT_MSA.1.1/JCVM The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes **the currently active context and the Active Applets security attributes** to **the Java Card VM (S.JCVM)**.
Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE222], §4 and [JCVM222], §3.4.

FMT_MSA.2/FIREWALL_JCVM Secure security attributes

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

FMT_MSA.3/FIREWALL Static attribute initialization

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

FMT_MSA.3.2/FIREWALL The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

FMT_MSA.3/JCVM Static attribute initialization

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

FMT_MSA.3.2/JCVM The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMR.1/JCRE Security roles

FMT_SMR.1.1/JCRE The TSF shall maintain the roles**:**
- **the Java Card RE (JCRE)**.
- **the Java Card VM (JCVM)**.

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

FMT_SMF.1/CORE_LC Specification of Management Functions

FMT_SMF.1.1/Core_LC The TSF shall be capable of performing the following management functions:
- **Modify the Currently Active Context, the Selected Applet Context, and the Active Applets**

8.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.
The execution of the additional native code is not within the TSF. Nevertheless, access to API native methods from the Java Card System is controlled by TSF because there is no difference between native and interpreted methods in the interface or the invocation mechanism.

FCS_CKM.1 Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **[assignment: cryptographic key generation algorithm]** and specified cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

Iteration	Algorithm	Key size	Standards
/RSA Std	RSA standard key generation	1024, 1536, 2048,	ANSI X9.31
/RSA CRT	RSA CRT key generation	1024, 1536, 2048,	ANSI X9.31
/GP	GP session keys	112 (for SCP01, SCP02) 128,192, 256 (for SCP03)	[GP211] (for SCP01, SCP02) [GP221] (for SCP03)
/ECFP	ECC key generation	160, 192, 224, 256, 320, 384, 512, 521	ANSI X9.62
/ECDH	EC Diffie-Hellman	160, 192, 224, 256, 320, 384, 512, 521	[IEEE-P1363]
/DH	DH key generation	1024, 1536, 2048	ANSI X9.42
/DH	DH key exchange	1024, 1536, 2048	ANSI X9.42

Remark:
- Keys generated for PACE are ECDH keys.

Application note:

- The keys are generated and diversified in accordance with [JCAPI222] specification in classes KeyBuilder and KeyPair (at least Session key generation).

FCS_CKM.2 Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **[assignment: cryptographic key distribution method]** that meets the following: **[assignment: list of standards]**.

iteration	Distribution method	standards
/RSA	JC API setkey()	none
/TDES	JC API setkey()	none
/AES	JC API setkey()	none
/ECFP	JC API setkey()	none
/DH	JC API setkey()	none

FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **[assignment: type of cryptographic key access]** in accordance with a specified cryptographic key access method **[assignment: cryptographic key access method]** that meets the following: **[assignment: list of standards]**.

iteration	Key access method	standards
/RSA	JC API getkey()	none
/TDES	JC API getkey()	none
/AES	JC API getkey()	none
/ECFP	JC API getkey()	none
/DH	JC API getkey()	none

FCS_CKM.4 Cryptographic key destruction

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

- The keys are reset in accordance with [JCAPI222] in class Key with the method clearKey(). Any access to a cleared key attempting to use it for ciphering or signing shall throw an exception.

FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform **[assignment: list of cryptographic operations]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

Iteration	operation	algorithm	Key size	Standards

/RSA-SIGN	signature & verification	RSA (STD) RSA CRT	1024, 1152, 1280, 1536 and 2048 3072, 4096	[ISO9796-2] RSA SHA PKCS#1 RSA SHA PKCS#1 PSS
/RSA-CIPHER	Encryption & decryption	RSA (STD) RSA CRT	1024, 1152, 1280, 1536 and 2048 3072, 4096	[ISO9796-2] RSA SHA PKCS#1 RSA SHA PKCS#1 PSS
/ECC-SIGN	signature & verification	ECC	160, 192, 224, 256, 320, 384, 512, 521	[TR-03111] ECDSA SHA
/TDES-CIPHER	Encryption & decryption	TDES	112 168	[SP800-67] [ISO9797-1] DES NOPAD DES PKCS#5 DES 9797 M1 M2
/AES-CIPHER	Encryption & decryption	AES	128, 192, 256	[FIPS197] AES 128 NOPAD
/TDES-CIPHER FAST	Encryption & decryption	TDES	112 168	[SP800-67] [ISO9797-1] DES NOPAD DES PKCS#5 DES 9797 M1 M2
/AES-CIPHER FAST	Encryption & decryption	AES	128, 192, 256	[FIPS197] AES 128 NOPAD
/TDES-MAC	Signature, Verification	TDES	112 168	[SP800-67] [ISO9797-1] DES MAC ISO9797-1 M1 M2 Alog3 DES MAC NOPAD DES MAC PKCS#5
/TDES-MAC FAST	Signature, Verification	TDES	112 168	[SP800-67] [ISO9797-1] DES MAC ISO9797-1 M1 M2 Alog3 DES MAC NOPAD DES MAC PKCS#5
/AES-MAC	Signature, Verification	AES	128, 192, 256	[FIPS197] AES 128 NOPAD
/AES-CMAC FAST	Signature, Verification	AES	128, 192, 256	SP800-38B
/SHA	Hashing	Hashing	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	NA
/ECC-PACE	Integrited Mapping Generic Mapping	ECC	160, 192, 224, 256, 320, 384, 512, 521	ISO/IEC JTC1 SC17 WG3/TF5 ‘Supplemental Access Control for Machine Readable Travel Documents’

FDP_RIP.1/ABORT Subset residual information protection

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

FDP_RIP.1/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)**.

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**.

FDP_ROL.1/FIREWALL Basic rollback

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

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE222], §7.7, within the bounds of the Commit Capacity ([JCRE222], §7.8), and those described in [JCAPI222]**.

8.1.1.3 Card Security Management

FAU_ARP.1 Security alarms

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

- **throw an exception,**
- **or lock the card session**
- **or reinitialize the Java Card System and its data**

upon detection of a potential security violation.

Refinement:
The TOE detects the following potential security violation:

- CAP file inconsistency

- Applet life cycle inconsistency
- Card Manager 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(), [JCAPI222] and ([JCRE222], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Random trap detection

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **integrity-sensitive data**.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall
- **Prevent the use of modified data**
- **Raise an exception**

Application note:
Cryptographic keys, PINs, applets, and softmasks have the **integrity-sensitive data** attribute when they are stored in EEPROM.

FPR_UNO.1 Unobservability

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

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**.

Application note:
- The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE222], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE222]). Behavior of the TOE on power loss and reset is described in [JCRE222], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE222], §3.6.2.

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

FPT_TDC.1.2 The TSF shall use
- The rules defined in [JCVM222] specification;
- The API tokens defined in the export files of reference implementation
- **The rules defined in ISO 7816-6**
- **The rules defined in [GP211] specification**

when interpreting the TSF data from another trusted IT product.

8.1.1.4 AID Management

FIA_ATD.1/AID User attribute definition

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:

- **package AID**
- **Applet's version number**
- **registered applet's AID**
- **applet selection status ([JCVM222], §6.5)**.

Application note:

- "Individual users" stands for applets.

FIA_UID.2/AID User identification before any action

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

Application notes:

- By users here it must be understood the ones associated to the packages (or applets) that act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected applet or the package that is the subject's owner. Means of identification are provided during the loading procedure of the package and the registration of applet instances.
- The role Java Card RE defined in FMT_SMR.1/JCRE is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself with respect to the TOE, but it is a part of it.

FIA_USB.1/AID User-subject binding

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

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users:

- **Initial applet selection is performed as described in [JCRE222]§4**
- **The default applet depends on personalization.**

FIA_USB.1.3/AID The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users:

- **Applet selection is performed after a successful SELECT FILE command as described in [JCRE222]§4**.

Application note:

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

FMT_MTD.1/JCRE Management of TSF data

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

FMT_MTD.3/JCRE Secure TSF data

FMT_MTD.3.1/JCRE The TSF shall ensure that only secure values are accepted for **the AIDs of registered applets**.

8.1.2 INSTG Security Functional Requirements

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

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

FDP_ITC.2.1/Installer The TSF shall enforce the **PACKAGE LOADING information flow control SFP** when importing user data, controlled under the SFP, from outside of the TOE.
Application note:

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

FDP_ITC.2.2/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.
Application note:

- The format of the CAP file is precisely defined in Sun's specification ([JCVM222]); it contains the user data (like applet's code and data) and the security attribute altogether. Therefore there is no association to be carried out elsewhere.

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.
Application note:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM222], §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM222], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications.. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatibles. However, package files do have "package Version Numbers" ([JCVM222]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

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

A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs. The loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM222],§4.5.2).

FMT_SMR.1/Installer Security roles

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

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

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [JCRE222] §11.1.4**.

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **[none]** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.
Application note:

- The TOE has no maintenance mode.

FPT_RCV.3.2/Installer For **[Failure during applet loading, installation and deletion; sensitive data loading]**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/Installer The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[none]** for loss of TSF data or objects under the control of the TSF.

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

8.1.3 ADELG Security Functional Requirements

This group consists of the SFRs related to the deletion of applets and/or packages, enforcing the applet deletion manager (ADEL) policy on security aspects outside the runtime. Deletion is a critical phase and therefore requires specific treatment.

FDP_ACC.2/ADEL Complete access control

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

Refinement:
The operations involved in the policy are:
- OP.DELETE_APPLET,
- OP.DELETE_PCKG,
- OP.DELETE_PCKG_APPLET.

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

FDP_ACF.1/ADEL Security attribute based access control

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

Subject/Object	Attributes
S.JCVM	Active Applets
S.JCRE	Selected Applet Context, Registered Applets, Resident Packages
O.CODE_PKG	Package AID, Dependent Package AID, Static References
O.APPLET	Applet Selection Status
O.JAVAOBJECT	Owner, Remote

FDP_ACF.1.2/ADEL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:
In the context of this policy, an object O is reachable if and only if one of the following conditions holds:

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

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

R.JAVA.14 ([JCRE222], §11.3.4.1, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,

(1) S.ADEL is currently selected,
(2) There is no instance in the context of O.APPLET that is active in any logical channel and
(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE222], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.15 ([JCRE222], §11.3.4.1, Multiple Applet Instance Deletion). S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,

(1) S.ADEL is currently selected,
(2) There is no instance in the context of O.APPLET that is active in any logical channel and
(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE222], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.16 ([JCRE222], §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 resident package on the card that depends on O.CODE_PCKG.

R.JAVA.17 ([JCRE222], §11.3.4.3, Applet Package and Contained Instances Deletion). 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) There is no instance in the context of O.APPLET that is active in any logical channel and
(ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE222],§8.5) O.JAVAOBJECT is remote reachable.

Application notes:

- This policy introduces the notion of reachability, which provides a general means to describe objects that are referenced from a certain applet instance or package.
- S.ADEL calls the "uninstall" method of the applet instance to be deleted, if implemented by the applet, to inform it of the deletion request. The order in which these calls and the dependencies checks are performed are out of the scope of this security target.

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

FDP_ACF.1.4/ADEL] The TSF shall explicitly deny access of **any subject but the S.ADEL to O.CODE_PKG or O.APPLET for the purpose of deleting it from the card**.

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 [JCRE222], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

FMT_MSA.1/ADEL Management of security attributes

FMT_MSA.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to restrict the ability to **modify** the security attributes: **Registered Applets** and **Resident Packages** to **the Java Card RE (S.JCRE)**.
Application note:
The modification of the ActiveApplets security attribute should be performed in accordance with the rules given in [JCRE222], §4.

FMT_MSA.3/ADEL Static attribute initialization

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

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

FMT_SMF.1/ADEL Specification of Management Functions

FMT_SMF.1.1/ADEL The TSF shall be capable of performing the following management functions: **Modify the list of registered applets' AIDs and the Resident Packages**.

FMT_SMR.1/ADEL Security roles

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

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

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 [JCRE222], §11.3.4**.
Application note:

- The applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with the Java Card specifications. That is, if a reset or power fail occurs during the deletion process, then before any applet is selected in card, either the applet instance deletion is completed or the applet shall be selectable and all objects owned by the applet remain unchanged (that is, the functionality of all applet instances on the card remains the same as prior to the unsuccessful deletion attempt) [JCRE222], §11.3.4.

8.1.4 ODELG Security Functional Requirements

The following requirements concern the object deletion mechanism. This mechanism is triggered by the applet that owns the deleted objects by invoking a specific API method.

FDP_RIP.1/ODEL Subset residual information protection

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

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**.

8.1.5 CarG Security Functional Requirements

This group includes requirements for preventing the installation of packages that have not been bytecode verified, or that has been modified after bytecode verification.

FCO_NRO.2/CM Enforced proof of origin

FCO_NRO.2.1/CM The TSF shall enforce the generation of evidence of origin for transmitted **application packages** at all times.
Application note:
Upon reception of a new application package for installation, the card manager shall first check that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification.

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

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **no limitation**.

FDP_IFC.2/CM Complete information flow control

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

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

Application note:

The subjects covered by this policy are those involved in the loading of an application package by the card through a potentially unsafe communication channel:

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

The information controlled by the policy is the APDUs exchanged by the subjects through the communication channel linking the card and the CAD. Each of those messages contain part of an application package that is required to be loaded on the card, as well as any control information used by the subjects in the communication protocol.

FDP_IFF.1/CM Simple security attributes

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

Subject / Information	Attribute	value
User	**role**	**Operator, Owner, Issuer, None**
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 or Issuer can load an applet.**
- **Only applets with the security attribute Checked set to YES can be transferred.**

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

FDP_IFF.1.4/CM The TSF shall explicitly authorize an information flow based on the following rules:

- **The Issuer, behaving as the BCV, can load it through a secure channel, after having verified the applet.**
- **The Issuer can load an applet with a DAP specifying that it has been verified by the BCV.**
- **The Operator, having checked the applet can load it through a secure channel.**

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

- The TOE fails to verify the integrity and authenticity evidences of the application package
- **An applet, not verified by a BCV cannot be loaded.**

FDP_UIT.1/CM Data exchange integrity

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

FDP_UIT.1.2/CM The TSF shall be able to determine on receipt of user data, whether **modification, deletion, insertion, replay of some of the pieces of the application sent by the CAD** has occurred.

Application note:

Modification errors should be understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the application package to be installed on the card to be different from the one sent by the CAD.

FIA_UAU.1/CM Timing of authentication

FIA_UAI.1.1/CM The TSF shall allow

- **JCAPI with already installed applets**
- **APDUs for Applets**

on behalf of the user to be performed before the user is authenticated.

Application note:

This authentication of the card manager is a strong authentication as soon as the TOE leaves the protected environment of audited facilities. For this purpose, keys are diversified.

FIA_UAU.1.2/CM The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM The TSF shall allow

- **JCAPI with already installed applets**
- **APDUs for Applets**

on behalf of the user to be performed before the user is identified.

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

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 **AID** to **None**.

FMT_MSA.3/CM Static attribute initialization

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

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

Subject / Information	Attribute	value	default
User	role	Operator, Owner, None	None
Applet	checked	Boolean	No
DAP Key	OK	Boolean	No

FMT_SMF.1/CM Specification of Management Functions

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

- The loading of the applets, with their AID by the Card Manager.

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles **Card Manager**.

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

FTP_ITC.1.2/CM The TSF shall permit **the CAD placed in the card issuer secured environment** to initiate communication via the trusted channel.
FTP_ITC.1.3/CM The TSF shall initiate communication via the trusted channel for **loading and 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 loaded and installed on the card only on demand of the card issuer.

8.1.6 SCPG Security Functional Requirements

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

FPT_TST.1/SCP TSF Testing

FPT_TST.1.1/SCP The TSF shall run a suite of self tests **periodically during normal operation** to demonstrate the correct operation of **security mechanisms of the IC**.

FPT_TST.1.2/SCP The TSF shall provide authorized users with the capability to verify the integrity of **Keys**.

FPT_TST.1.3/SCP The TSF shall provide authorized users with the capability to verify the integrity of **Applets, user PIN, user Keys**.

FPT_PHP.3/SCP Resistance to physical attacks

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

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.

8.1.7 CMGR Group Security Functional Requirements

This group includes requirements for Card Manager.

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, Issuer, Card Manager
Objects: applets and keys
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 Card Manager loads applets into the card on behalf of the Byte Code Verifier.
The Card Manager extradites applets in the card on behalf of the Operator.
The Card Manager locks the loading of applets on the card on behalf of the Issuer.
The Card Manager loads GP keys into the cards on behalf of the Operator.
.
FDP_ACF.1.3/CMGR The TSF shall explicitly authorize 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 **No code but Java packages can be loaded or deleted**.

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 initialization

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.

8.1.8 ASFR Group Security Functional Requirements

This group includes specific requirements for the TOE.

FPT_FLS.1/SpecificAPI Failure with preservation of secure state

FPT_FLS.1.1/SpecificAPI The TSF shall preserve a secure state when the following types of failures occur: **the application fails to perform a specific execution flow control protected by the Specific API**.

FPT_ITT.1/SpecificAPI Basic internal TSF data transfer protection

FPT_ITT.1.1/SpecificAPI The TSF shall protect TSF data from **disclosure and modification** when it is transmitted between separate parts of the TOE.

FPR_UNO.1/SpecificAPI Unobservability

FPR_UNO.1.1/SpecificAPI The TSF shall ensure that **external attacker** are unable to observe the operation **as sensitive comparison or copy** on **sensitive objects defined** by **the application using the Specific API**.

FCS_RND.1 Quality metric for random numbers

FCS_RND.1.1 The TSF shall provide a mechanism to generate random numbers that meet **RGS [RGS-B1]**

8.2 SECURITY ASSURANCE REQUIREMENTS

The security assurance requirement level is EAL5 augmented with AVA_VAN.5 and ALC_DVS.2.

9 TOE SUMMARY SPECIFICATION

9.1 TOE SECURITY FUNCTIONS

TOE Security Functions are provided by the TOE embedded software (including the optional NVM ES) and by the chip.

9.1.1 SF provided by MultiApp v3 platform

9.1.1.1 SF.FW: 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* 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*.

An operation OP.PUT (S1, S.MEMBER, I) is allowed if and only if the active context is "JCRE"; other OP.PUT operations are allowed regardless of the active context's value.	FDP_IFC.1/JCVM FDP_IFF.1/JCVM
Upon allocation of a resource to class instances and arrays, any previous information content of the resource is made unavailable	FDP_RIP.1/OBJECTS
Only the S.JCRE can modify the security attributes the active context, the selected applet context security attributes.	FMT_MSA.1/JCRE
Only the S.JCVM can modify the security attributes the active context, the currently active Context and the Active Applets security attributes.	FMT_MSA.1/JCVM
only secure values are accepted for all the security attributes of subjects and objects defined in the FIREWALL access control SFP and the JCVM information flow control SFP.	FMT_MSA.2/FIREWALL_ JCVM
provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/FIREWALL
The TSF shall maintain the roles: the Java Card RE, the Java Card VM. The TSF shall be able to associate users with roles.	FMT_SMR.1/JCRE
The TSF shall be capable of performing the following management functions: • Modify the active context and the SELECTed applet Context. • Modify the list of registered applets' AID	FMT_SMF.1/CORE_LC
([JCRE222]§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".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE222]§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	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL

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.	
([JCRE222]§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.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
• ([JCRE222], §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: (c) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable", (d) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets, and in either of the cases above the invoked interface method extends the Shareable interface	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the following two exceptions: 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.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
Upon deallocation of the resource from any transient object, any previous information content of the resource is made unavailable.	FDP_RIP.1/TRANSIENT
The TSF shall permit the rollback of the operations OP.JAVA and OP.CREATE on the O.JAVAOBJECTs.	FDP_ROL.1/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 [JCRE222], §7.7, within the bounds of the Commit Capacity ([JCRE222], §7.8), and those described in [JCAPI222].	FDP_ROL.1/FIREWALL
Only updates to persistent objects participate in the transaction. Updates to transient objects and global arrays are never undone, regardless of whether or not they were “inside a transaction.” [JCRE222], §7.7	FDP_ROL.1/FIREWALL
A TransactionException is thrown if the commit capacity is exceeded during a transaction. [JCRE222], §7.8	FDP_ROL.1/FIREWALL
Transaction & PIN: When comparing a PIN, even if a transaction is in progress, update of internal state - the try counter, the validated flag, and the blocking state, do not participate in the transaction. [JCAPI222]	FDP_ROL.1/FIREWALL

9.1.1.2 SF.API: Application Programming Interface

This security function provides the cryptographic algorithm and functions used by the TSF:

- TDES algorithm support 112-bit key and 168-bit key
- RSA algorithm supports up to 4096 bit keys (CRT method), 2048 bit keys (Std method).
- AES algorithm with 128, 192 and 256 bit keys.
- ECC algorithm with160, 192, 224, 256, 320, 384, 512, and 521 bit keys.

- Random generator uses the certified Hardware Random Generator that fulfils the requirements of AIS31 (see [ST_IC]).
- SHA-1, SHA224, SHA-256, SHA-384, and SHA-512 algorithms
- Diffie-Hellman based on EC algorithm.
- PACE based on EC algorithm (integrated mapping and generic mapping)

This security function controls all the operations relative to the card keys management.

- Key generation: The TOE provides the following:
  - RSA key generation manages 1024 to 2048-bits long keys. The RSA key generation is SW and does not use the IC cryptographic library.
  - The TDES key generation (for session keys) uses the random generator.
  - AES key generation
  - DH key generation
- Key destruction: the TOE provides a specified cryptographic key destruction method that makes Key unavailable.

This security function ensures the confidentiality of keys during manipulation and ensures the de-allocation of memory after use.
This security function is supported by the IC security function SF.CS (Cryptographic support) for Random Number Generator (see [ST_IC]).

RSA standard Key generation Algorithm - 1024,1536,2048	FCS_CKM.1
RSA CRT Key generation Algorithm - 1024,1536,2048)	FCS_CKM.1
TDES Key generation Algorithm - 112, 168	FCS_CKM.1
AES Key generation Algorithm - 128, 192, 256	FCS_CKM.1
ECC Key generation Algorithm - 160, 192, 224, 256, 320, 384, 512, 521	FCS_CKM.1
EC Diffie-Hellman Key agreement Algorithm 160, 192, 224, 256, 320, 384, 512, 521	FCS_CKM.1
DH Key agreement Algorithm 1024, 1536, 2048	FCS_CKM.1
Key distribution with JC API setkey()	FCS_CKM.2
Key access with JC API getkey()	FCS_CKM.3
Key deletion with JC API clearkey()	FCS_CKM.4
RSA standard Signature & Verification – RSA SHA PKCS#1, RSA SHA PKCS#1 PSS – 1024,1152,1280,1536,2048	FCS_COP.1
RSA CRT Signature & Verification – RSA SHA PKCS#1, RSA SHA PKCS#1 PSS 1024,1152,1280,1536,2048, 3072, 4096	FCS_COP.1
RSA standard Encryption & Decryption – 1536, 1792, 2048	FCS_COP.1
RSA CRT Encryption & Decryption – 1024,1152,1280,1536,2048, 3072, 4096	FCS_COP.1
TDES Encryption & Decryption – DES NOPAD, DES PKCS#5, DES 9797 M1 M2 – 112, 168	FCS_COP.1
TDES Signature & Verification – DES MAC ISO9797-1 M1 M2, DES MAC NOPAD, DES MAC PKCS#5- 112, 168	FCS_COP.1
AES Encryption & Decryption – AES 128 NOPAD – 128, 192, 256	FCS_COP.1
AES Signature & Verification – AES MAC 128 NOPAD – 128, 192, 256	FCS_COP.1
ECDSA Signature & Verification – ECDSA SHA – 160, 192, 224, 256, 320, 384, 512, 521	FCS_COP.1
SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Message digest	FCS_COP.1
ECC for PACE Integrited Mapping & Generic Mapping 160, 192, 224, 256, 320, 384, 512, 521	FCS_COP.1

9.1.1.3 SF.CSM: Card Security Management

Upon allocation of a resource to the APDU buffer, any previous information content of the resource is made unavailable.	FDP_RIP.1/APDU
Upon deallocation of a resource from the bArray object, any previous information content of the resource is made unavailable.	FDP_RIP.1/bArray
Upon deallocation of a resource from any reference to an object instance created during an aborted transaction, any previous information content of the resource is made unavailable.	FDP_RIP.1/ABORT
Upon deallocation of a resource from the cryptographic buffer (D.CRYPTO), any previous information content of the resource is made unavailable.	FDP_RIP.1/KEYS
The TSF shall take the following actions: • throw an exception, • or lock the card session • or reinitialize the Java Card System and its data upon detection of a potential security violation.	FAU_ARP.1
The TOE detects the following potential security violation: • CAP file inconsistency • Applet life cycle inconsistency • Card Manager 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(), [JCAPI222] and ([JCRE222], §7.6.2) • Violation of the Firewall or JCVM SFPs • Unavailability of resources • Array overflow • Random trap detection	FAU_ARP.1
The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all the following objects: Cryptographic keys, PINs, applets, and softmasks when they are stored in EEPROM. Upon detection of a data integrity error, the TSF shall: • Prevent the use of modified data • Raise an exception	FDP_SDI.2
In order to consistently interpret **the CAP files, the bytecode and its data argument,** when shared between the TSF and another trusted IT product, the TSF shall use: • The rules defined in [JCVM222] specification; • The API tokens defined in the export files of reference implementation • **The rules defined in ISO 7816-6** • **The rules defined in [GP211] specification**	FPT_TDC.1
The TSF shall preserve a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU_ARP.1.** The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE222], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE222] §4.1.2). Behavior of the TOE on power loss and reset is described in [JCRE222], §3.6, and §7.1. Behavior of the TOE on RF signal	FPT_FLS.1/JCS

loss is described in [JCRE222], §3.6.2	
No one can observe the operation **cryptographic operations / comparisons operations** on **Key values / PIN values** by **S.JCRE, S.Applet.**	FPR_UNO.1

9.1.1.4 SF.AID: AID Management

Only **the JCRE** can **modify** the **list of registered applets' AIDs**.	FMT_MTD.1/JCRE
Only secure values are accepted for **the AIDs of registered applets**.	FMT_MTD.3/JCRE
The TSF shall maintain the following list of security attributes belonging to individual users: • **package AID** • **Applet's version number** • **registered applet's AID** • **applet selection status ([JCVM222], §6.5)**	FIA_ATD.1/AID
The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.	FIA_UID.2/AID
Initial applet selection is performed as described in [JCRE222]§4 **Applet selection is performed after a successful SELECT FILE command as described in [JCRE222]§4**.	FIA_USB.1/AID

9.1.1.5 SF.INST: Installer

the protocol used provides for the unambiguous association between the security attributes and the user data received: The format of the CAP file is precisely defined in Sun's specification ([JCVM222]); it contains the user data (like applet's code and data) and the security attribute altogether.	FDP_ITC.2/Installer
Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM222], §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM222], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications.. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatibles. However, package files do have "package Version Numbers" ([JCVM222]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.	FDP_ITC.2/Installer
A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs. The 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 ([JCVM222],§4.5.2).	FDP_ITC.2/Installer
The TSF shall maintain the roles**: the installer**	FMT_SMR.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 [JCRE222] §11.1.4**	FPT_FLS.1/Installer
After **Failure during applet loading, installation and deletion; sensitive data**	FPT_RCV.3/Installer

loading, the TSF ensures the return of the TOE to a secure state using automated procedures. The TSF provides the capability to determine the objects that were or were not capable of being recovered.	

9.1.1.6 SF.ADEL: Applet Deletion

Only **the Java Card RE (S.JCRE)** can **modify** the security attributes: **ActiveApplets**. The modification of the ActiveApplets security attribute should be performed in accordance with the rules given in [JCRE222], §4.	FMT_MSA.1/ADEL
Provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/ADEL
The TSF shall maintain the roles: **the applet deletion manager**.	FMT_SMR.1/ADEL
The TSF shall be able to **Modify the ActiveApplets security attribute**.	FMT_SMF.1/ADEL
([JCRE222], §11.3.4.1, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if, (1) S.ADEL is currently selected, (2) O.APPLET is deselected and (3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE222], §8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE222], §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 ([JCRE222], §8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE222], §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.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE222], §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	FDP_ACC.2/ADEL FDP_ACF.1/ADEL

(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 ([JCRE222],§8.5) O.JAVAOBJECT is remote reachable.	
However, the S.ADEL may be granted privileges ([JCRE222], §11.3.5) to bypass the preceding policies. For instance, the logical deletion of an applet renders it un-selectable; this has implications on the management of the associated TSF data (see application note of FMT_MTD.1.1/JCRE).	FDP_ACF.1/ADEL
Only the S.ADEL can delete O.CODE_PKG or O.APPLET from the card.	FDP_ACF.1/ADEL
Upon deallocation of a resource from the **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**, any previous information content of the resource is made unavailable.	FDP_RIP.1/ADEL
Requirements on de-allocation during applet/package deletion are described in [JCRE222], §11.3.4.1, §11.3.4.2 and §11.3.4.3.	FDP_RIP.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 [JCRE222], §11.3.4**.	FPT_FLS.1/ADEL

9.1.1.7 SF.ODEL: Object Deletion

Upon deallocation of the resource from the objects owned by the context of an applet instance which triggered the execution of the method javacard.framework.JCSystem.requestObjectDeletion(), any previous information content of the resource is made unavailable.	FDP_RIP.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.	FPT_FLS.1/ODEL

9.1.1.8 SF.CAR: Secure Carrier

No one can **modify** the security attributes **AID**	FMT_MSA.1/CM
Default values for security attributes are: • User role: none • Applet checked: No • DAP Key OK: No	FMT_MSA.3/CM
The TSF shall maintain the roles**: Card Manager**	FMT_SMR.1/CM
The Card Manager loads applets with their AID.	FMT_SMF.1/CM
The TOE enforces the generation of evidence of origin for transmitted **application packages** at all times.	FCO_NRO.2/CM
The TOE allows: • JCAPI with already installed applets • APDUs for Applets on behalf of the user to be performed before the user is authenticated.	FIA_UAU.1/CM
The TOE allows: • JCAPI with already installed applets • APDUs for Applets on behalf of the user to be performed before the user is identified.	FIA_UID.1/CM

• Only the user with the security attribute role set to Operator can load an applet. • 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_IFC.2/CM FDP_IFF.1/CM
Package loading is protected against **modification, deletion, insertion, and replay** errors. If such an error occurs, it is detected at receiption .	FDP_UIT.1/CM
New packages can be loaded and installed on the card only on demand of the card issuer. This is done through a GP Secure Channel.	FTP_ITC.1/CM

9.1.1.9 SF.SCP: Smart Card Platform

The TSF periodically tests the security mechanisms of the IC. It also checks the integrity of sensitive assets: Applets, PIN and Keys.	FPT_TST.1/SCP
The TSF resists physical attacks	FPT_PHP.3/SCP
The TSF offers transaction mechanisms	FPT_RCV.4/SCP

9.1.1.10 SF.CMG: Card Manager

The Card Manager loads and extradites applets. It also loads GP key.	FDP_ACC.1/CMGR FDP_ACF.1/CMGR
No one can modify the security attribute code category	FMT_MSA.1/CMGR
Only restrictive default values can be used for the code category	FMT_MSA.3/CMGR

9.1.1.11 SF.API: Specific API

Provides means to application to control execution flow, to detect any failure and to react if required	FPT_FLS.1/SpecificAPI
Provides means to application to execute securely data transfer and comparison, to detect any failure during operation and to react if required..	FPT_ITT.1/SpecificAPI
Provides means to introduce dummy operations leading to unobservability of sensitive operation	FPR_UNO.1/SpecificAPI

9.1.1.12 SF.RND: RNG

Provide a random value	FCS_RND.1

9.1.2 TSFs provided by the SLE78

The evaluation is a composite evaluation and uses the results of the CC evaluation provided by [CR-IC]. The IC and its primary embedded software have been evaluated at level EAL 5+. These SF are the same for the IC considered in this ST, SLE78C(L)X1600P.

SF	Description
SF_DPM	Device Phase Management
SF_PS	Protection against Snooping
SF_PMA	Protection against Modification Attacks
SF_PLA	Protection against Logical Attacks
SF_CS	Cryptographic Support

Table 45: Security Functions provided by the Infineon SLE78CLX1600P chip

These SF are described in [ST-IC].

9.2 ASSURANCE MEASURES

Assurance Measure	**Document title**
AM_ASE	MultiApp V3 JCS Security Target
AM_ADV_Spec	Functional Specifications - MultiApp V3
AM_ADV_Design	Design – MultiApp V3
AM_ADV_Int	Internals – MultiApp V3
AM_ALC	Class ALC – MultiApp V3
AM_AGD	Guidance – MultiApp V3
AM_ATE	Class ATE – MultiApp V3
AM_CODE	Source Code – MultiApp V3
AM_Samples	Samples – MultiApp V3

Table 56: Assurance Measures.

The development team uses a configuration management system that supports the generation of the TOE. The configuration management system is well documented and identifies all different configuration items. The configuration management tracks the implementation representation, design documentation, test documentation, guidance documentation. The security of the configuration management is described in detail in a separate document.
The delivery process of the TOE is well defined and follows strict procedures. Several measures prevent the modification of the TOE based on the developer's master copy and the user's version. The Administrator and the User are provided with necessary documentation for initialization and start-up of the TOE.
The implementation is based on an informal design of the components of the TOE. The description is sufficient to generate the TOE without other design requirements.
The correspondence of the Security Functional Requirements (SFR) with less abstract representations will be demonstrated in a separate document. This addresses ADV_ARC, ADV_FSP, ADV_IMP, and ADV_TDS.
The tools used in the development environment are appropriate to protect the confidentiality and integrity of the TOE design and implementation. The development is controlled by a life cycle model of the TOE. The development tools are well defined and documented.
The Gemalto R&E organization is equipped with organizational and personnel means that are necessary to develop the TOE.
As the evaluation is identified as a composite evaluation based on the CC evaluation of the hardware, the assurance measures related to the hardware (IC) will be provided by documents of the IC manufacturer

END OF DOCUMENT