CENTAUR JAVACARD OPEN PLATFORM

ICitizen Open v2

Cyberflex Access 64k v3

Common Criteria / ISO 15408

EAL4+

Security Target **Public version**

Table of contents

1 ST INTRODUCTION 3
1.1 PUBLIC ST IDENTIFICATION 3
1.2 ST OVERVIEW 3
1.3 CC CONFORMANCE 4
1.4 CURRENT ST VS. [ST/INFINEON] 4
1.5 REFERENCES 4
2 TOE DESCRIPTION 6
2.1 SCOPE OF THE TOE 6
2.2 TOE USAGE 7
2.3 SMART CARD PRODUCTS LIFE-CYCLE 8
2.4 TOE ENVIRONMENT 11
2.5 TOE INTENDED USAGE 12
3 TOE SECURITY ENVIRONMENT 13
3.1 ASSETS 13
3.2 SUBJECTS 15
3.3 ASSUMPTIONS 16
3.4 THREATS 17
3.5 ORGANISATIONAL SECURITY POLICIES 19
4 SECURITY OBJECTIVES 20
4.1 SECURITY OBJECTIVES FOR THE TOE 20
4.2 SECURITY OBJECTIVES FOR THE ENVIRONMENT 22
5 IT SECURITY REQUIREMENTS 23
5.1 TOE SECURITY FUNCTIONAL REQUIREMENTS 23
5.2 TOE SECURITY ASSURANCE REQUIREMENTS 45
5.3 SECURITY REQUIREMENTS FOR THE IT ENVIRONMENT 46
6 TOE SUMMARY SPECIFICATION 49
6.1 TOE SECURITY FUNCTIONS 49
7 PP CLAIMS 52
7.1 PP REFERENCE 52

Table of figures

Figure 1 TOE and its boundaries 6
Figure 2 TOE operations 7
Figure 3: TOE Life Cycle 10

1 ST introduction

1.1 Public ST Identification

Title: CENTAUR Platform Security Target
Version: V 1.1 Issued May 05 2006
Registration: Ref. ST_D1018903
Origin: AXALTO

TOE reference: T100921
Commercial names:
ICitizen Open v2
Cyberflex Access 64k v3

The TOE is composed with:

Component	Version number	Supplier
Micro-controller SLE66CX680PE	A13	Infineon
RMS library	2.5	Infineon
RSA2048library	1.4	Infineon
ROM MASK	SB 147 (Infineon)	Axalto
SOFT MASK	No	Axalto
CENTAUR Software	v1.0	Axalto

TOE function type and options: JavaCard Open Platform.

This ST claims two Protection Profiles:
[PP/JCS] for the Open Platform;
[PP/SSVG] for the IC.

The IC is evaluated under the German scheme for Common Criteria. The certification body is the 'Bundesamt für Sicherheit in der Informationstechnik' (BSI).

This Security Target deals with the evaluation of the Open Platform, as well as the composition with the evaluation of the IC. This evaluation is done under the French scheme for Common Criteria. The certification body is the 'Direction Centrale de la Sécurité des Systèmes d'Information' (DCSSI).

1.2 ST overview

Context

This TOE provides the security of an EAL4+ 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 EAL4+ evaluated on this platform.
The applications using only EAL4+ applets will BE EAL4+ even if NOT EAL4+ applets are loaded on the platform.
The applications using a NOT EAL4+ applets will NOT BE EAL4+.

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 [JCVM22]. 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 security target are:

To describe the Target of Evaluation (TOE). This ST focuses on the Java Card Open Platform, embedded in a Smart card integrated circuit.
To describe the security environment of the TOE including the assets to be protected and the threats to be countered by the TOE and by its environment.
To describe the security objectives of the TOE and its supporting environment.
To specify the security requirements which includes the TOE security functional requirements and the TOE security assurance requirements.
To specify the TOE summary specification, which includes the TOE security functions specifications and the assurance measures.

The assurance level for this product and its documentation is EAL4 augmented with:
ADV_IMP.2: Implementation of the TSF,
ALC_DVS.2: Sufficiency of security measures.
AVA_MSU.3: Analysis of insecure states,
AVA_VLA.4: Highly resistant,

ADV_IMP.2, ALC_DVS.2, AVA_MSU.3 and AVA_VLA.4 are required in [PP/SSVG].
The strength level for the TOE security functional requirements is “SOF high” (Strength Of Functions high).

1.3 CC conformance

The compliance is assumed with CC version V2.2 (ISO 15408) (see reference in 1.5.1).
This ST is built on [PP/JCS] and [PP/SSVG] and is conformant to these PP.
This ST is CC V2.2 conformant with Part2.
This ST is CC V2.2 conformant with Part3 and EAL4 augmented as stated in [PP/JCS], [PP/SSVG] and in present document.

1.4 Current ST vs. [ST/Infineon]

This ST is defined for the whole TOE, including the IC and the ES. However, the IC has its own ST, [ST/Infineon] and the assets, threats, objectives, SFR and Security functions specific to the IC that are described in this ST are not described in the current ST.
[ST/Infineon] refines the assets, threats, objectives and SFR of [PP/SSVG].
The current ST refines the assets, threats, objectives and SFR of [PP/JCS]. It also states how the SF of the current ST relies on the SF of [ST/Infineon].

1.5 References

1.5.1 External References [ER]

[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model CCIMB-2004-01-001, version 2.2, January 2004 (conform to ISO 15408)
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements CCIMB-2004-01-002, version 2.2, January 2004 (conform to ISO 15408)
[CC-3]	Common Criteria for Information Technology security Evaluation Part 3: Security Assurance Requirements CCIMB-2004-01-003, version 2.2, January 2004 (conform to ISO 5408)
[CEM]	Common Methodology for Information Technology Security Evaluation CCIMB-2004-01-004, version 2.2, January 2004.
[JCRE22]	Java Card 2.2.1 Runtime Environment (JCRE) Specification. Published by Sun Microsystems, Inc.
[JCVM22]	Java Card 2.2.1 Virtual Machine (JCVM) Specification.. Published by Sun Microsystems, Inc.

[PP/JCS]	PP/0305 JavaCard System Standard 2.2 Configuration Protection Profile version 1.0b
[PP/SSVG]	Smartcard IC Platform Protection Profile
[PP/Type2]	Secure Signature-Creation device Protection Profile Type 2 v1.05, EAL4+
[PP/Type3]	Secure Signature-Creation device Protection Profile Type 3 v1.05, EAL4+
[OP211]	Open Platform – Card Specification version 2.1.1
[ST/Infineon]	Security Target of SLE66CX680PE Integrated Circuit
[FIPS 197]	FIPS-197: Advanced Encryption Standard (AES)
[FIPS 46-3]	FIPS-46-3: Data Encryption Standard (DES)
[SP 800-38 A]	NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of operation
[FIPS 180-2]	FIPS-46-3: Secure Hash Standard (SHA)
[RSA PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard
[ISO 9796-2]	ISO/IEC 9796-2

2 TOE Description

This part of the ST describes the TOE as an aid to the understanding of its security requirements.
It addresses the product type, the smart card product life cycle, the TOE environment along the smart card life cycle and the general IT features of the TOE.

2.1 Scope of the TOE

The Target of Evaluation (TOE) is the CENTAUR Platform defined by:

- The Java Platform 2.2.1 based on GEOS (Generic Operating System);
- The underlying Integrated Circuit and its libraries;
- The Figure below gives a description of the TOE and its boundaries. The dark blue parts are the limits of the TOE.

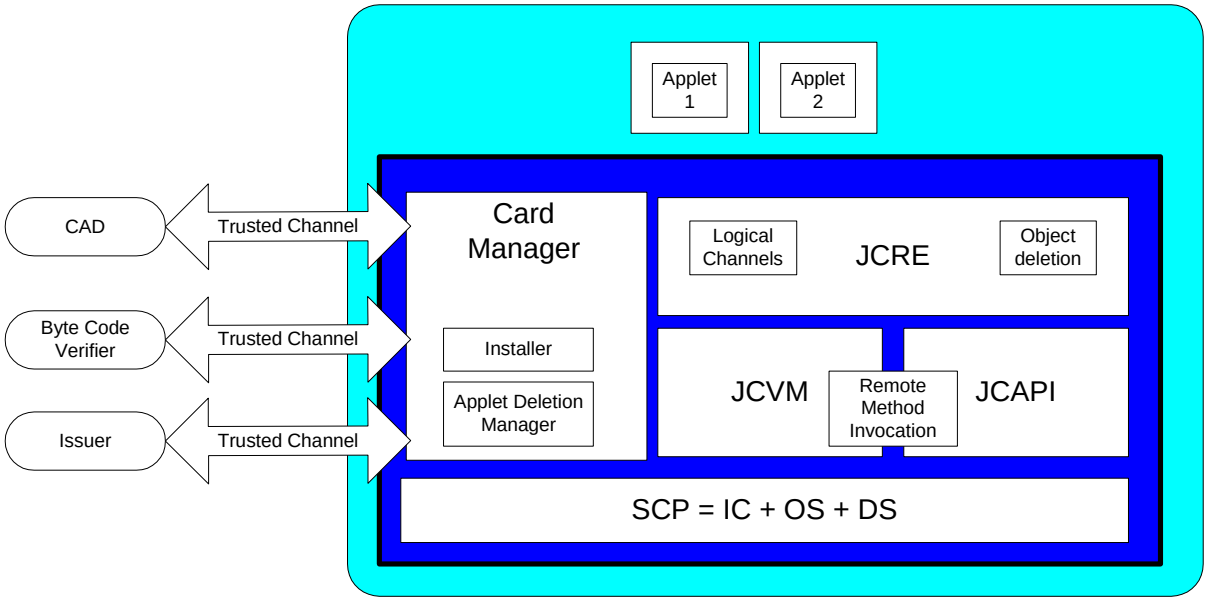

Figure 1 TOE and its boundaries

The TOE is the embedded software (ES), the Integrated Circuit (IC) and the plastic card. The ES comprises:

- The Java Platform including JCRE, JCVM and JCAPI
- The Card Manager
- The Operating System (OS)
- The Dedicated System (DS), which provides an interface with the Integrated Circuit (IC)

The Applets loaded pre issuance or post issuance are outside the TOE, but inside the TSC.
Other smart card product elements, (such as holograms, magnetic stripes, security printing,…) are outside the scope of this Security Target.

Terminology
This document uses the terminology of [PP/JCS].

The TOE of this ST is the TOE of [PP/JCS], enlarged to include the whole OS including the Card Manager, the DS and the IC.

2.2 TOE Usage

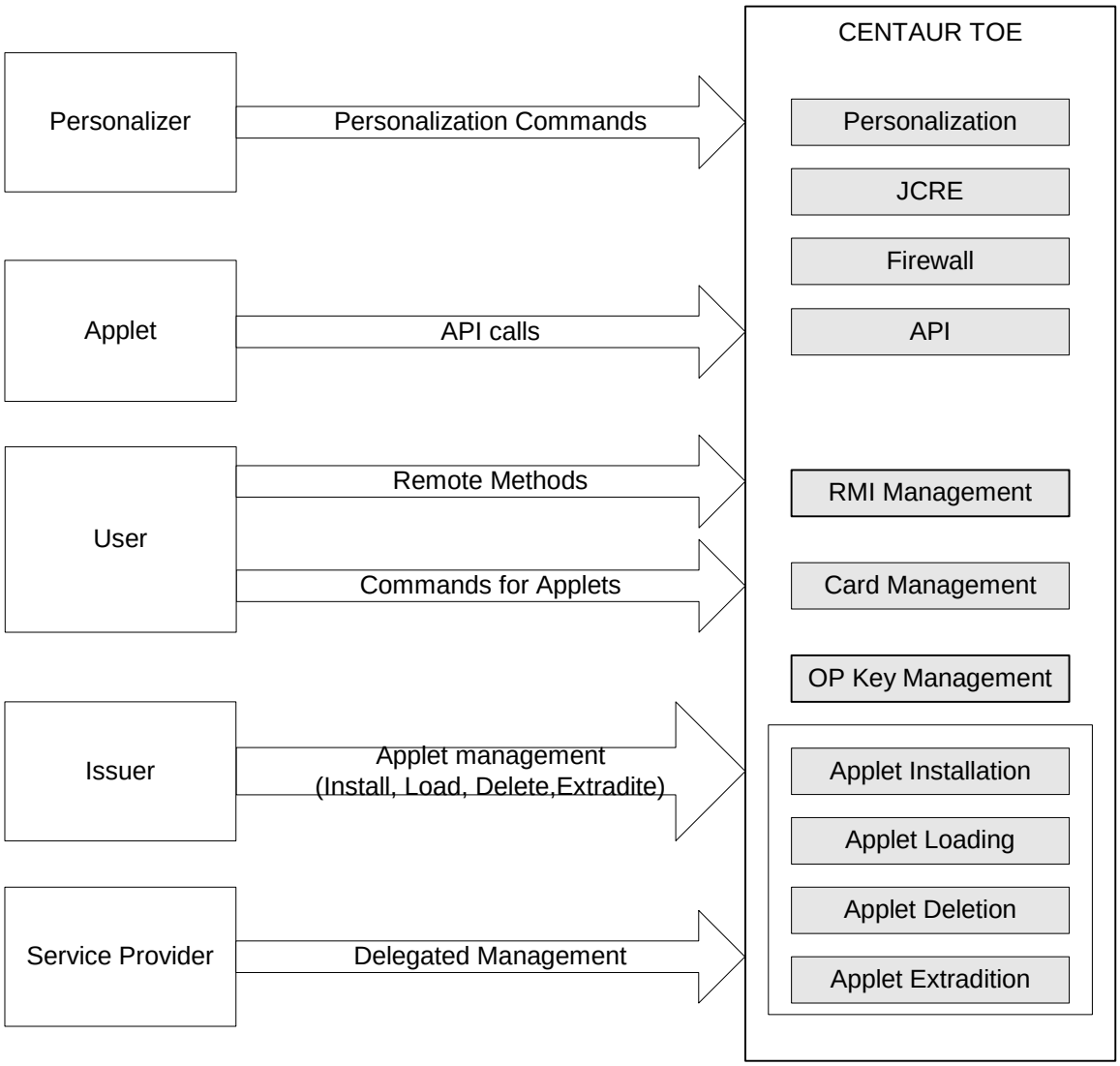

Figure 2 TOE operations

2.2.1.1 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.

2.2.1.2 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

2.3 Smart Card Products Life-cycle

The Smart card product life cycle, as defined in [PP/SSVG], is split up into 7 phases where the following authorities are involved:

Phase 1	**Smart card software development**	**The smart card embedded software developer** is in charge of the smart card embedded software development and the specification of IC pre-personalization requirements.
Phase 2	IC Development	**The IC designer** designs the integrated circuit, develops IC firmware if applicable, provides information, software or tools to the smart card software developer, and receives the software from the developer, through **trusted delivery and verification procedures**. From the IC design, IC firmware and smart card embedded software, he constructs the smart card IC database, necessary for the IC photomask fabrication.
Phase 3	**IC manufacturing and testing**	**The IC manufacturer** is responsible for producing the IC through three main steps: IC manufacturing, testing, and IC pre-personalization.
Phase 4	IC packaging and testing	**The IC packaging manufacturer** is responsible for the IC packaging and testing.
Phase 5	**Smart card product finishing process**	**The smart card product manufacturer** is responsible for the smart card product finishing process and testing, and the smart card pre-personalization
Phase 6	**Smart card personalization**	**The Personaliser** is responsible for the smart card personalization and final tests.
Phase 7	Smart card end-usage	**The smart card issuer** is responsible for the smart card product delivery to **the smart card end-user**, and for the end of life process.

The JCS Platform life Cycle as described in the standard smart cards life cycle can be matched as shown in Figure 3: TOE Life Cycle.

OS design and **JCVM design** correspond to life phase 1 “Smart card software development”.
Hardware design corresponds to life phase 2 “IC development”.
Hardware fabrication OS and JCVM embedding correspond to life phase 3 “IC manufacturing and testing”, phase 4 “IC packaging and testing”, phase 5 “Smart card product finishing process”.

The global security requirements of the TOE mandate to consider, during the development phase, the threats to security occurring in the other phases. This is why this ST addresses the functions used in phases 6 and 7 but developed during phases 1 to 5.
The limits of the evaluation process correspond to phases 1 to 3 including the TOE under development delivery from the party responsible of each phase to the parties responsible of the following phases.
Assumptions are made for phases 4 and 5.
These different phases may be performed at different sites. This implies that procedures on the delivery process of the TOE must exist and be applied for every delivery within a phase or between phases. This includes any kind of delivery performed from phase 1 to 5 to subsequent phases, including:
Intermediate delivery of the TOE or the TOE under construction within a phase,
Delivery of the TOE or the TOE under construction from one phase to the next.

These procedures must be compliant with the security assurance requirements developed later in this document.

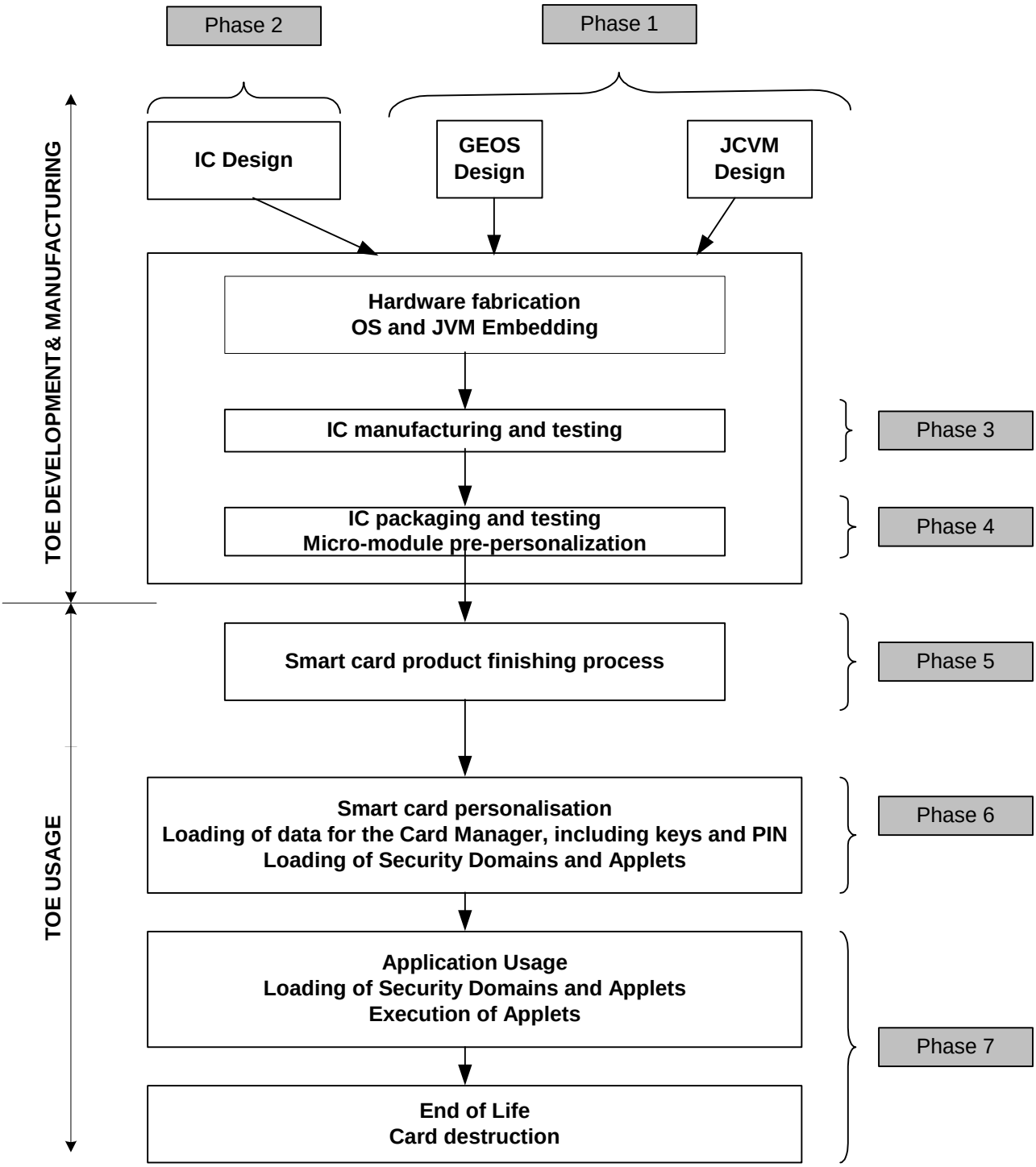

Figure 3: TOE Life Cycle

2.4 TOE Environment

Considering the TOE, four types of environment are defined:

- Development and fabrication environment (phase 1 to 4),
- Initialization environment corresponding to smart card pre-personalization (phase 5)
- Personalization environment, during which the card loads and installs applets (phase 6),
- User environment, during which the card owner uses the Applets loaded on the card. In this environment the card can also load, install and delete applets (phase 7),
- End of life environment, during which the TOE is made inapt for any operation (end of the phase 7).

2.4.1 TOE Development & Production Environment

The TOE described in this ST is developed in different places as indicated below:

IC design	Infineon München
Secure OS Design	Axalto Louveciennes
JavaCard Platform design	Axalto Louveciennes
IC manufacturing and Testing	Infineon München
IC packaging and testing	Axalto Orleans

In order to ensure security, the environment in which the development takes place must be made secure with access control tracing entries. Furthermore, it is important that all authorized personnel feels involved and fully understands the importance and the rigid implementation of the defined security procedures.
The development begins with the TOE specification. All parties in contact with sensitive information are required to abide by Non-disclosure Agreement.
Design and development of the ES then follows. The engineers use a secure computer system (preventing unauthorized access) to make the conception, design, implementation and test performances.
Storage of sensitive documents, databases on tapes, CDs, and printed circuit layout information are in appropriately locked cupboards/safe. Of paramount importance also is the disposal of unwanted data (complete electronic erasures) and documents (e.g. shredding).
Testing, programming and deliveries of the TOE then take place. When these are done offsite, they must be transported and worked on in a secure environment with accountability and tractability of all (good and bad) products.
During the electronic transfer of sensitive data, procedures must be established to ensure that the data arrive, only at the destination and is not accessible at intermediate stages (e.g. stored on a buffer server where system administrators make backup copies). It must also be ensured that transfer is done without modification or alteration.
During fabrication, phases 3, and 4, all the persons involved in storage and transportation operations should fully understand the importance of the defined security procedures.
Moreover, the environment in which these operations take place must be secured.
The TOE Initialization is performed in [Infineon München phase 3 ; Orleans phase 4 & 5].
In the initialization environment of the TOE, smart card pre-personalization takes place (phase 5).
Initialization requires a secure environment, which guarantees the integrity and confidentiality of operations.

2.4.2 Usage of the TOE

2.4.2.1 Personalization Environment

The personalization (phase 6) is done when the TOE is constructed. However it should take place in a protected environment. During the personalization, applets and data may be loaded into the card.
After the personalization, the TOE is issued to the end User.

2.4.2.2 Usage Environment

Once delivered to the end user (phase 7), the TOE can activate the applets. The TOE can also perform management operations on applets (Load, Install, delete, extradite).
The TOE is owned by the end user and the environment is not secure. Therefore the TOE itself has to ensure that the security requirements are met.

2.4.3 TOE End of life

End of life must be considered for several reasons:
The Keys can be compromised.
The TOE can be stolen.
The TOE physical support can come to the end of its useful life.
In all these cases, it must be ensured that the TOE cannot be used any more.

2.4.4 The actors and roles

The users of the TOE include
People like the card issuer and the card owner,
Hardware like the CAD where the card is inserted
Software components like the application packages installed on the card.

The Issuer can Load, Install, Delete Applets. He can also Lock Applet management.
The Card Owner can activate Applets and functions inside the Applets.
The CAD is involved in the JCRMI.
The application packages respond to the Card Owner. They can use functions of the Card by activating the Java API.

2.5 TOE intended usage

The TOE intended usage is the management and the use of applets in a Secure Environment.

3 TOE security environment

3.1 Assets

The assets of the TOE are those defined in [PP/JCS]. The assets of [PP/SSVG] are studied in [ST/INFINEON].

3.1.1 Java Card 2.2 Standard configuration

3.1.1.1 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.PIN

Any end-user's PIN.

To be protected from unauthorized disclosure and modification.

D.APP_KEYs

Cryptographic keys owned by the applets.

To be protected from unauthorized disclosure and modification.

3.1.1.2 TSF Data

D.JCS_CODE

The code of the Java Card System.

To be protected from unauthorized disclosure and modification.

D.JCS_DATA

The internal runtime data areas necessary for the execution of the JCVM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated 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 JCRE, like, for instance, the AIDs used to identify the installed applets, the Currently selected applet, the current context of execution and the owner of each object.

To be protected from unauthorized disclosure and modification.

D.API_DATA

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

To be protected from unauthorized disclosure and modification.

D.JCS_KEYs

Cryptographic keys used when loading a file into the card.

To be protected from unauthorized disclosure and modification.

D.CRYPTO

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

To be protected from unauthorized disclosure and modification.

3.2 Subjects

3.2.1 Java Card 2.2 Standard configuration

The main *subjects* of the TOE considered in this document are the following ones:

S.PACKAGE

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

S.JCRE

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

S.BCV

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

Application note:

The BCV is off-card

S.CRD

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

S.ADEL

The *applet deletion manager*, which also acts on behalf of the card issuer. This subject is involved in the ***ADEL security policy*** defined in applet Deletion Manager Policy.

S.CAD

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

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

S.SPY

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

3.3 Assumptions

The assumptions of the TOE are those defined in [PP/JCS]. The assumptions of [PP/SSVG] are studied in [ST/INFINEON].

3.3.1 Java Card 2.2 Standard configuration

These Assumptions are those described in [PP/JCS] except A.NATIVE.

The Assumption A. NATIVE deals with native code, which is not included in the TOE of the [PP/JCS]. This native code is included in the TOE of this ST. Therefore this assumptiom is changed into an OSP.

A.VERIFICATION

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

A.APPLET

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

3.4 Threats

The threats of the TOE are those defined in [PP/JCS]. The threats of [PP/SSVG] are studied in [ST/INFINEON].

3.4.1 Java Card 2.2 Standard configuration

3.4.1.1 Confidentiality

T.CONFID-JCS-CODE

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

T.CONFID-APPLI-DATA

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

T.CONFID-JCS-DATA

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

3.4.1.2 Integrity

T.INTEG-APPLI-CODE

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

T.INTEG-JCS-CODE

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

T.INTEG-APPLI-DATA

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

T.INTEG-JCS-DATA

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

T.INTEG-APPLI-CODE.2

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

T.INTEG-APPLI-DATA.2

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

3.4.1.3 Identity Usurpation

T.SID.1

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

T.SID.2

The attacker modifies the identity of the privileged roles.

3.4.1.4 Unauthorized Executions

T.EXE-CODE.1

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

T.EXE-CODE.2

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

T.NATIVE

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

T.EXE-CODE-REMOTE

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

3.4.1.5 Denial of Service

T.RESOURCES

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

3.4.1.6 Modifications of the Set of Applications

T.INSTALL

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

3.4.1.7 Card Management

T.DELETION

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

3.4.1.8 Services

T.OBJ-DELETION

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

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

3.5 Organisational security policies

3.5.1 Java Card 2.2 Standard configuration

The Organisational security policies of the TOE are based on those defined in [PP/JCS].

OSP.VERIFICATION

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

OSP.NATIVE

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

4 Security objectives

4.1 Security objectives for the TOE

The security objectives for the TOE are those defined in [PP/JCS]. The security objectives for the TOE of [PP/SSVG] are studied in [ST/INFINEON].

4.1.1 Java Card 2.2 Standard configuration

4.1.1.1 Identification

O.SID

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

4.1.1.2 Execution

O.OPERATE

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

O.RESOURCES

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

O.FIREWALL

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

O.NATIVE

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

O.REALLOCATION

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

Application note:

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

O.SHRD_VAR_CONFID

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

O.SHRD_VAR_INTEG

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

4.1.1.3 Services

O.ALARM

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

O.TRANSACTION

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

Application note:

Transactions are provided to *applets* as Java Card class libraries.

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.

O.PIN-MNGT

The TOE shall provide means to securely manage PIN objects.

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.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.

O.REMOTE

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

4.1.1.4 Applet Management

O.INSTALL

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

O.LOAD

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

Application note:

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

O.DELETION

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

4.1.1.5 Object Deletion

O.OBJ-DELETION

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

4.1.1.6 SCP

The Objectives described in this section are Objectives for the Environment in [PP/JCS]. They become Objectives for the TOE because the TOE in this ST includes the SCP. OE.NATIVE is renamed to O.NATIVE.2 in order to avoid a duplication of names

O.NATIVE.2

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

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.

O.SCP.SUPPORT

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

O.SCP.IC

The *SCP* shall possess IC security features.

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.

4.2 Security objectives for the environment

The security objectives for the environment are those defined in [PP/JCS]. The security objectives for the environment of [PP/SSVG] are studied in [ST/INFINEON].

4.2.1 Java Card 2.2 Standard configuration

OE.APPLET

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

OE.VERIFICATION

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

5 IT security requirements

The IT security functional requirements are those defined in [PP/JCS]. The IT security functional requirements of [PP/SSVG] are studied in [ST/INFINEON].

5.1 TOE security functional requirements

5.1.1 Java Card 2.2 Standard configuration

FAU_ARP.1/JCS Security alarms

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

Global refinement:

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

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

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

FCO_NRO.2/CM Enforced proof of origin

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

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

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **that the cryptographic keys used to verify the sender's identity have been loaded using a secure process**.

Application note: If a new application package is received by the card for installation, the card manager shall first check that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification.

FCS_CKM.1/AES Cryptographic key generation

FCS_CKM.1.1/AES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **AES Key generation** and specified cryptographic key sizes **128 bits** that meet the following: **none (generation of random numbers)**.

FCS_CKM.1/RSA Cryptographic key generation

FCS_CKM.1.1/RSA The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **RSA Key generation** and specified cryptographic key sizes **1536, 1792, 2048 bits** that meet the following: **none (generation of random numbers and Miller- Rabin primality testing)**.

FCS_CKM.1/TDES Cryptographic key generation

FCS_CKM.1.1/TDES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **TDES Key generation** and specified cryptographic key sizes **112 bits** that meet the following: **none (generation of random numbers)**.

Application note (for all FCS_CKM.1 SFRs): The keys can be generated and diversified in accordance with [JCAPI21] specification in classes KeyBuilder and KeyPair (at least Session key generation).

FCS_CKM.2/AES Cryptographic key distribution

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

FCS_CKM.2/RSA Cryptographic key distribution

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

FCS_CKM.2/TDES Cryptographic key distribution

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

Application note (for all FCS_CKM.2 SFRs): Command SetKEY that meets [JCAPI21] standard.

FCS_CKM.3/AES Cryptographic key access

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

FCS_CKM.3/RSA Cryptographic key access

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

FCS_CKM.3/TDES Cryptographic key access

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

Application note (for all FCS_CKM.3 SFRs): The keys can be accessed in accordance with [JCAPI21] in class Key.

FCS_CKM.4/AES Cryptographic key destruction

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

FCS_CKM.4/RSA Cryptographic key destruction

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

FCS_CKM.4/TDES Cryptographic key destruction

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

Application note (for all FCS_CKM.4 SFRs): The keys are reset in accordance with [JCAPI21] 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/AES Cryptographic operation

FCS_COP.1.1/AES The TSF shall perform **AES encryption** in accordance with a specified cryptographic algorithm **AES** and cryptographic key sizes **128 bits** that meet the following: **[FIPS 197]**.

FCS_COP.1/RSA Cryptographic operation

FCS_COP.1.1/RSA The TSF shall perform **RSA encryption and decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **1024, 1536, and 2048 bits** that meet the following: **[RSA PKCS#1]**.

FCS_COP.1/TDES Cryptographic operation

FCS_COP.1.1/TDES The TSF shall perform **DES / TDES encryption and decryption** in accordance with a specified cryptographic algorithm **DES / TDES-CBC, TDES-EBC** and cryptographic key sizes **56 bits (DES) / 112 bits (TDES)** that meet the following: **[FIPS 46-3] and [SP 800-38 A]**.

FCS_COP.1/Signature Cryptographic operation

FCS_COP.1.1/Signature The TSF shall perform **Signature and verification of signature** in accordance with a specified cryptographic algorithm **RSA_SHA_PKCS#1, RSA_SHA_ISO9796_2** and cryptographic key sizes **1024, 1536, and 2048 bits** that meet the following: **[RSA PKCS#1], [ISO 9796_2]**.

FCS_COP.1/SHA-1 Cryptographic operation

FCS_COP.1.1/SHA-1 The TSF shall perform **secure hashing** in accordance with a specified cryptographic algorithm **SHA-1** and cryptographic key sizes **none** that meet the following: **[FIPS 180-2]**.

Application note: This cryptographic algorithm SHA-1 doesn't use a key.

FCS_COP.1/SHA-256 Cryptographic operation

FCS_COP.1.1/SHA-256 The TSF shall perform **secure hashing** in accordance with a specified cryptographic algorithm **SHA-256** and cryptographic key sizes **none** that meet the following: **[FIPS 180-2]**.

Application note: This cryptographic algorithm SHA-256 doesn't use a key.

Application note: (for all FCS_COP.1 SFRs): The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] in accordance to [JCAPI22] specification (see javacardx.crypto.Cipher and javacardx.security packages).

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_ACC.2/ADEL Complete access control

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

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

Global refinement:

For *FDP_ACC.2.1/ADEL*

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

S.ADEL The applet Deletion Manager. *This subject is unique.*

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

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

O.JAVAOBJECT Java class instance or array.

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

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

FDP_ACC.2/FIREWALL Complete access control

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

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

Global refinement:

For *FDP_ACC.2.1/FIREWALL:* Subjects (prefixed with an " *S* ") and objects (prefixed with an " *O* ") covered by this policy are:

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

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

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

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

FDP_ACC.2/JCRMI Complete access control

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

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

Global refinement:

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

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

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

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

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

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

O.REMOTE_MTHD A method of a remote interface.

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

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

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

FDP_ACF.1/ADEL Security attribute based access control

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

Non editorial refinement:

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

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

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

When an export file is used during preparation of a CAP file, the version numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM22], §4.5.2): the dependent packages AIDs attribute allows the retrieval of those identifications.

Static fields of a package may contain references to objects. The Static References attribute records those references. An applet instance can be in two different selection states: selected or deselected. If the applet is selected (in some logical channel), then in turn it could either be currently selected or just active. At any time there could be up to four active applet instances, but only one currently selected. This latter is the one that is processing the current command ([JCRE22], §4).The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialise static fields of the package).

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

Finally, there are needed security attributes that are not attached to any object or subject of the TSP: (1) the ResidentPackages Versions (or Resident Image, [JCVM22],§4.5) and AIDs. They describe the packages that are already on the card, (2) the list of registered applet instances and (3) the ActiveApplets security attribute. They are all attributes internal to the VM, that is, not attached to any specific object or subject of the SPM. These attributes are TSF data that play a role in the SPM.

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

Non editorial refinement:

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

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

In the context of this policy, a Java Object O is reachable if and only if either: (1) the Owner of O is a registered applet instance A (O is reachable from A), (2) a static field of a loaded package P contains a reference to O (O is reachable from P), (3) there exists a valid remote reference to O (O is remote reachable), and (4) there exists

an object O' that is reachable according to either (1) or (2) or (3) above and O' contains a reference to O (the reachability status of O is that of O').

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

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

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

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

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

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

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

FDP_ACF.1/CMGR Security attribute based access control

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

Subjects: Byte Code Verifier, Operator

Objects: applets 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 OP keys into the cards on behalf of the Operator.

.

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

FDP_ACF.1.4/CMGR The TSF shall explicitly deny access of subjects to objects based on the **No code but Java packages can be loaded or deleted**.

FDP_ACF.1/FIREWALL Security attribute based access control

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

Non editorial refinement:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

FDP_ACF.1/JCRMI Security attribute based access control

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

Non editorial refinement:

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

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

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

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

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

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

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

FDP_IFC.1/JCRMI Subset information flow control

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

Non editorial refinement:

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

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

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

There is a unique operation in this policy:

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

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

FDP_IFC.1/JCVM Subset information flow control

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

Non editorial refinement:

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

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

There is a unique operation in this policy:

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

FDP_IFC.2/CM Complete information flow control

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

Non editorial refinement:

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

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

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

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

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

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

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

FDP_IFF.1/CM Simple security attributes

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

Subject / Object	Attribute	value
user	role	Operator, Owner, 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 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_IFF.1.3/CM The TSF shall enforce the **None**.

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

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

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

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

FDP_IFF.1/JCRMI Simple security attributes

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

Non editorial refinement:

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

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

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

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

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

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

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

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

FDP_IFF.1/JCVM Simple security attributes

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

FDP_IFF.1.2/JCVM The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **An operation *OP.PUT(S 1, S.MEMBER, I)* is allowed if and only if the active context is "JCRE"; other *OP.PUT* operations are allowed regardless of the active context's value.**.

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

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

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

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

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

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

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

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

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

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

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

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**.

Application note: The events that provoke the de-allocation of the previously mentioned references are described in [JCRE22], §7.6.3.

FDP_RIP.1/ADEL Subset residual information protection

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

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

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**.

Application note: The allocation of a resource to the APDU buffer is typically performed as the result of a call to the `process()` method of an applet.

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**.

Application note: A resource is allocated to the bArray object when a call to *an* applet's install() method is performed.

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)**.

Application note: The javacard.security & javacardx.crypto packages do provide secure interfaces to the *cryptographic buffer* in a transparent way. See javacard.security.KeyBuilder and Key interface of [JCAPI22].

FDP_RIP.1/OBJECTS Subset residual information protection

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

*Application note:*The semantics of the Java programming language requires for any object field and array position to be initializsed with default values when the resource is allocated [JVM22], §2.5.1.

FDP_RIP.1/ODEL Subset residual information protection

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

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

FDP_RIP.1/TRANSIENT Subset residual information protection

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

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

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

FDP_ROL.1/FIREWALL Basic rollback

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

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

Application note: Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism relies on the transaction mechanism offered by the platform. Some operations of the API are not conditionally updated, as documented in [JCAPI22] (see for instance, PIN-blocking, PINchecking, update of Transient objects).

Application note: The loading and linking of applet packages (the installation or registration is covered by *FDP_ROL.1.1/FIREWALL*) is subject to some kind of rollback mechanism (see *FPT_RCV.3.1/Installer*), described in [JCRE22], §11.1.5, but is implementation-dependent.

FDP_SDI.2/JCS Stored data integrity monitoring and action

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

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

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

1. PINs
2. keys
3. application sensitive data
4. applet code

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.

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.

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

Application note:
modification, deletion, insertion, or replay apply to some pieces of the application sent by the CAD

FIA_ATD.1/AID User attribute definition

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

FIA_UID.1/JCS Timing of identification

FIA_UID.1.1/JCS The TSF shall allow

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

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

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

Application note: The User in this SFR is the Issuer

Application note: This SFR combines FIA_UID.1/CM and FIA_UID.1/CMGR from [PP/JCS].

FIA_UID.2/AID User identification before any action

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

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

FIA_USB.1 User-subject binding

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

FMT_MSA.1/ADEL Management of security attributes

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

FMT_MSA.1/CM Management of security attributes

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

FMT_MSA.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.1/EXPORT Management of security attributes

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

FMT_MSA.1/JCRE Management of security attributes

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

Application note:

FMT_MSA.1/JCRE includes FMT_MSA.1/JCRMI, which is not repeated in this ST.

FMT_MSA.1/REM_REFS Management of security attributes

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

FMT_MSA.2/JCRE Secure security attributes

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

Application note:

Secure values conform to the following rules:

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

FMT_MSA.3/ADEL Static attribute initialisation

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

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

FMT_MSA.3/CM Static attribute initialisation

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

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

FMT_MSA.3/CMGR Static attribute initialisation

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

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

FMT_MSA.3/FIREWALL Static attribute initialisation

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

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

FMT_MSA.3/JCRMI Static attribute initialisation

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

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

FMT_MTD.1/JCRE Management of TSF data

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

FMT_MTD.3 Secure TSF data

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

FMT_REV.1/JCRMI Revocation

FMT_REV.1.1/JCRMI The TSF shall restrict the ability to revoke security attributes associated with the **O.RMI_SERVICE object** within the TSC to **the JCRE**.

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

Global refinement:

The security attributes are the Retuned References

Application note:

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

FMT_SMF.1/JCS Specification of management functions

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

- The selection of applets and opening of logical channels
- **The loading** and the installing **of the applets, with their DAP and AID by the Operator**.

Application note:

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

FMT_SMR.1/JCS Security roles

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

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

Application note:

1 This SFR combines FMT_SMR.1/ADEL, FMT_SMR.1/CM, FMT_SMR.1/CMGR, FMT_SMR.1/JCRE, FMT_SMR.1/Installer, FMT_SMR.1/JCRMI, from [PP/JCS]

2 The Operator includes Applet Deletion Manager and Bytecode verifier. The Card Manager includes the Installer.

FPR_UNO.1 Unobservability

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

FPT_AMT.1/JCS Abstract machine testing

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

Global refinement:

In this document, the underlying abstract machine test is the IC and its library.

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:

1 those associated to the potential security violations described in FAU_ARP.1/JCS.

2 the Installer fails to load/install a package/applet as described in [JCRE22]§11.1.5

3 the applet deletion manager fails to delete a package/applet as described in [JCRE22], §11.3.4.

4 the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method.

Application note:

This SFR combines FPT_FLS.1/ADEL, FPT_FLS.1/Installer, FPT_FLS.1/JCS, FPT_FLS.1/ODEL, and FPT_FLS.1/SCP from [PP/JCS].

FPT_PHP.3/JCS Resistance to physical attack

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

FPT_RCV.3/JCS Automated recovery without undue loss

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

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

FPT_RCV.3.3/JCS The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **None** for loss of TSF data or objects within the TSC.

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

Application note:

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

2 The TOE has no maintenance mode.

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.

FPT_RVM.1/SCP Non-bypassability of the TSP

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

Application note: This SFR combines FPT_RVM.1.1 (CoreG) and FPT_RVM.1.1/SCP (SCPG) from [PP/JCS].

Application note: Access to native methods from the Java Card System is subject to TSF control, as there is no difference in the interface or the invocation mechanism between native and interpreted methods.

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

FPT_SEP.1/SCP TSF domain separation

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

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

Application note:

By security domain here it is intended "execution space", which should not be confused with other meanings of "security domains".

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

Application note: This TOE includes the Card Manager. So this SFR is modified and the interpretation consistency is required between the BCV and the TOE instead of the CM and the TOE.

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

- **The [JCVM22] specification;**
- **Reference export files;**
- **The ISO 7816-6 rules;**
- **The [OP21] specification**

when interpreting the TSF data from another trusted IT product.

FPT_TST.1/SCP TSF testing

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

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

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

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

FRU_FLT.1/SCP Degraded fault tolerance

FRU_FLT.1.1/SCP The TSF shall ensure the operation of **all operations but applet loading** when the following failures occur: **not enough memory left**.

FRU_RSA.1/Installer Maximum quotas

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

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

Application note:

The remote trusted IT product, mentioned in this SFR, is the CAD placed in the card issuer secured environment

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

5.2 TOE security assurance requirements

The security assurance requirement level is EAL4. The EAL is augmented with ADV_IMP.2, ALC_DVS.2, AVA_VLA.4 and AVA_MSU.3.

5.3 Security requirements for the IT environment

5.3.1 IT environment functional requirements

FDP_IFC.2/BCV Complete information flow control

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

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

Global refinement:

cf [PP/JCS]

FDP_IFF.2/BCV Hierarchical security attributes

FDP_IFF.2.1/BCV The TSF shall enforce the **TYPING information flow control SFP** based on the following types of subject and information security attributes:

(1) type attribute of the information,

(2) type attribute of the storage units of the JCVM,

(3) class attribute of the fields and methods,

(4) bounds attributes of the methods..

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

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

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

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

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

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

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

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

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

FDP_IFF.2.3/BCV The TSF shall enforce the **none**.

FDP_IFF.2.4/BCV The TSF shall provide the following **none**

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

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

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

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

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

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

FMT_MSA.1/BCV.1 Management of security attributes

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

FMT_MSA.1/BCV.2 Management of security attributes

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

FMT_MSA.2/BCV Secure security attributes

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

FMT_MSA.3/BCV Static attribute initialisation

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

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

FMT_SMF.1/BCV Specification of management functions

FMT_SMF.1.1/BCV The TSF shall be capable of performing the following security management functions: **Verification of the bytecode by the Bytecode Verifier**.

FMT_SMR.1/BCV Security roles

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

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

FRU_RSA.1/BCV Maximum quotas

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

Global refinement:

The Individual user is a method

6 TOE summary specification

6.1 TOE security functions

The TOE security functions are those of both the hardware and the software. The Security Functions that reply to the SFR of the IC are described in [ST/Infineon].

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

SF.SmartCardPlatform

The TOE runs tests at power on to check it has not been corrupted.

The TOE provides the ability to check the integrity of sensitive data stored in the card.

The TOE hides sensitive data transfers and operations from outside.

The TOE is protected against SPA, DPA, DFA & timing attacks.

Detects physical tampering and provides automatic response

SF.TrustedChannel

This SF establishes a trusted channel between the card and a remote IT. The trusted channel is OP2.1.1 compliant. It protects the integrity and/or the confidentiality of data loading into the card.

Additionally, a DAP mechanism protects the integrity of package loading and the Non repudiation of origin.

Shis SF protects the integrity of packages during the loading.

Shis SF protects the integrity and the confidentiality of keys during the transfers.

This function has no strength.

SF.CardManager

The Card Manager controls the flow of APDUs between off card applications and the current applet and it controls the access to the APDU buffer.

The Card Manager controls the Card Life Cycle and the Applet Life Cycle.

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 OP keys into the cards on behalf of the Operator.

This function has no strength.

SF.Attributes

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

This function has no strength.

SF.Crypto

This SF provides the crypto functions and their Java API. This includes:

- Key generation, Import, Export, and Deletion;
- Crypto Operations

  for the following algorithms: TDES, RSA 1536 - 2048, AES 128, SHA-1, SHA-256, Signature RSA_SHA_PKCS#1 & RSA_SHA_ISO9796-2. RSA is used in the CRT mode

The strength of this function is SOF-high.

Application note:

The TOE also provides other algorithms like DES, RSA 1024, and CRC. These algorithms are not strong enough for SOF high and therefore no FCS_COP functional requirement was created for any of them. These algorithms are considered as services and their usage requires special care.

SF.Erase

This SF ensures that sensitive data cannot be accessed upon and after some types of operations. This SF may be split in:

- Logical deletion of data upon package and/or applet(s) deletion,
- Physical deletion of data upon and after object(s) deletion.

  This SF erases the APDU buffer and the cryptographic buffer.

The strength of this function is SOF-high.

SF.Firewall

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

This SF participates to information confidentiality.

This function has no strength.

SF.Install

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

It also ensures a safe *applet* installation process.

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

This SF ensures that Java Card objects such as classes, *package* s and *applet* instances use limited resources.

This function has no strength.

SF.JCRE

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

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

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

The *JCRE* has access to any objects and methods owned by any context. But it only invokes the methods select(), process(), deselect(), getShareableInterfaceObject() defined in Applet class and MultiSelectable interface.

The *JCRE* is the currently active context when the *JCVM* starts running after a card reset.

It is the only context allowed to register applets.

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

This function has no strength.

SF.Applet

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

This SF realises applet code interpretation. From the interpretation point of view, the *applet* 's code is considered as data to read. There is no way for an *applet* to be executed independently or to access platform resources. Thus the *JCRE* decides to start and to stop the applet execution.

This function has no strength.

SF.Domain

This SF consists in memory management with the *JCRE* scope. It is realized thanks to OS services.

The *applet* needs some different types of storage areas to be executed. As *applet* has no direct access to internal resources in particular to memory, the *JCRE* manages the memory for the applets. For this purpose, some services have been defined. On its side, the *JCRE* has its own execution space in memory that is not accessible to the *applet* s.

This SF includes the File System and its API.

This function has no strength.

SF.PIN

This SF supplies to applet a means to authenticate a user with a PIN. It provides a global PIN accessible by the org.GlobalPatform.CVM class and user PINs accessible by the javacard.framework.OwnerPin class.

The authentication is done through a secure comparison between a reference PIN stored in the persistent memory and data received from the CAD. This SF checks the integrity of the PINs and their attributes.

This SF participates to information confidentiality.

This SF uses probabilistic mechanisms.

The strength of this function is SOF-high.

SF.JCRMI

This SF handles RMI security features: access control, information flow control, the related security attributes and their management.

This function has no strength.

SF.Transaction

This SF deals with transactions in the JCVM.

It enables to create Java Objects within a transaction. Transaction management includes the rollback of operations according to [JCRE22].

The TSF preserves a secure state when failures occur.

This SF manages transactions in the OS.

This function has no strength.

7 PP claims

7.1 PP reference

[PP/SSVG] and [PP/JCS] are claimed.
The TOE of [PP/JCS] is extended to cover the whole OS and the IC.