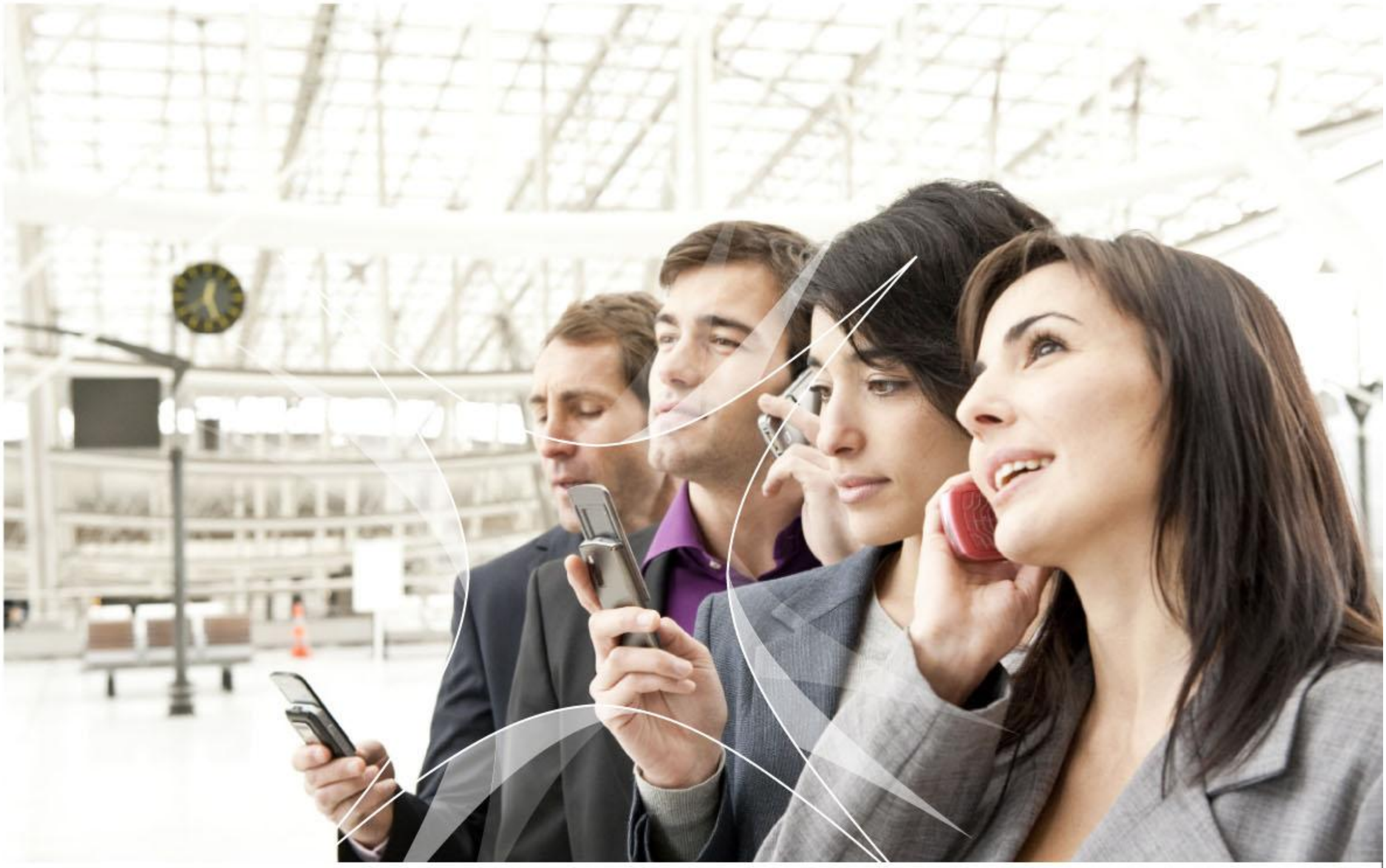
Security Target
LinqUs USIM 128k PK certified using
SC33F640



FINANCIAL SERVICES & RETAIL

ENTERPRISE

PUBLIC SECTOR

TELECOMMUNICATIONS

TRANSPORT

gemalto
security to be free

Table of Contents

Table of Figures

Table of Tables

1 ST Introduction

1.1 ST Reference

Security Target and associated evaluation are completely defined by information located in the following table.

Title:	Security Target : **LinqUs USIM 128k PK certified using SC33F640**
Reference:	D1172363
Version	1.7p
Origin:	GEMALTO
ITSEF:	THALES CEACI
Certification Body:	ANSSI
Evaluation scheme:	French

Table 1 ST References

This Security Target describes:

- The Target of Evaluation, the TOE components, the components in the TOE environment, the product type, the TOE environment and life cycle, the limits of the TOE.
- The assets to be protected, the threats to be countered by the TOE itself during the usage of the TOE
- The organizational security policies, and the assumptions,
- The security objectives for the TOE and its environment,
- The security functional requirements for the TOE and its IT environment,
- The TOE security assurance requirements
- The security functions and associated rationales.

1.2 TOE Reference

Product and TOE are completely defined by information located in the following table.

Product Name	LinqUs USIM 128k certified
Product Reference	T1017287
Product Version	Release A
TOE name	LinqUs USIM 128k PK certified platform using SC33F640
TOE Reference	S1092122
TOE Version	Release A
Commercial Name	LinqUs USIM 128k PK Certified

Table 2 TOE References

1.3 TOE overview

1.3.1 TOE Type

The product **LinqUs USIM 128k PK certified** (also named **LinqUs USIM 128k PK**) (U)SIM smart card defined to be used mainly in a mobile or a Smartphone, but can be used in any device with an interface conformant to [IS0 7816] specification. It is delivered using an ISO form factor including a plug-in form factor as defined in [TS 102 221] or 3FF form factor.

The product **LinqUs USIM 128k** implements the standard communication protocol (ISO 7816 T=0) and ETSI standard allowing communication between smartcard, mobile and server using OTA.

The Target of Evaluation (TOE) is the (U)SIM Java Card platform embedded in a (U)SIM card intended to be plugged in a mobile phone or other mobile devices to provide services to an end user.

The TOE is composed of the following bricks:

- A Java Card System according to [PP-JCS] which manages and executes applications called applets. It also provides APIs [JCAPI] to develop applets on top of it, in accordance with Java Card ^{TM} specifications.
- GlobalPlatform (GP) packages (partially evaluated), which provides a common and widely used interface to communicate with a smart card and manage applications in a secure way, in accordance with [GP] specifications,
- Platform APIs, which provides ways to specifically interact with (U)SIM applications, according to [TS131.130] specifications.
- Telecom environment including network authentication applications (not evaluated) and Telecom communication protocol.

The TOE configuration is defined using [PP-JCS] Javacard System protection profile Open Configuration with some proprietary extensions.

1.3.2 *TOE usage*

Figure 1: LinqUs USIM 128k card to be inserted in a mobile

The USIM defined in the [3GPP] standards as the Universal Subscriber Identity Module is an evolution of the SIM developed to ensure compliance within UMTS networks. A Subscriber Identity Module (SIM) is a removable module to plug within GSM mobile equipment that contains the International Mobile Subscriber Identity (IMSI) which unambiguously identifies a subscriber. It also stores other subscriber-related information or applications such as SIM Toolkit, and other application (as an E-sign application). In the rest of the document, the term of (U)SIM is used to refer to SIM or USIM as there are considered in the same way regarding security.

The primary services of the (U)SIM (when it is plugged in handset) are the user authentication by PIN capture and the SIM authentication on the MNO network, giving access to MNO services through the mobile. It also stores other subscriber-related information or applications such as SIM Toolkit applications as specified in [TS102.223] and [TS131.111].

The **LinqUs USIM 128k** Platform implements major industry standards:

- Java Card 2.2.2,
- Global Platform 2.2,
- Full ETSI release 6,
- 3GPP Release 6.

It supports **multiple networks (2G, 3G …)** and it implies that several Network Access Applications (NAA) working together, requiring for dynamic switching from networks (3G to 2G, 2G to 3G). Each application is designed like a plug-in.

1.3.3 TOE Boundaries

The following figures illustrate the TOE physical and logical boundaries.

The product is a smartcard including a plastic card and a module performing the interface between reader and the mobile and the embedded chip. The Target of Evaluation (TOE) is the Smart Card Integrated Circuit with Embedded Software in operation and in accordance to its functional specifications. Other smart card product items (such as plastic, module, bounding, printing…) are outside the scope of this evaluation.

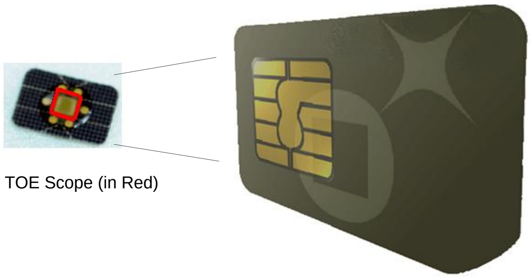


Figure 2: TOE Physical Boundaries

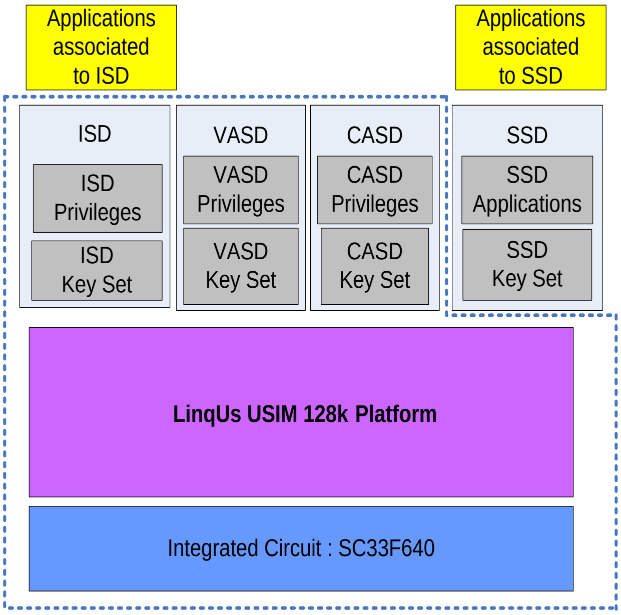


Figure 3: TOE Logical Boundaries

[GP] defines several types of security domains used for domain and application management:

- Issuer Security Domain (ISD),
- Supplementary Security Domain (SSD),
- Controlling Authority Security Domain (CASD),

and [PP-USIM] provides also

- Verification Authority Security Domain (VASD).

1.3.4 TOE Description

The TOE contains the following components:

Component	Reference/Version	Supplier
LinqUs USIM 128k Platform (Configuration with Native secure API)	S1092122 / Rel A	Gemalto
Micro-controller SC33F640 and part of its dedicated software (DS)	Rev E	STMicroelectronics

Table 3 TOE components

The IC (Micro-controller SC33F640) included in the TOE is compliant with the [PP-BSI-0035]. It includes a crypto library and a Flash loader used for software loading but inactivated for end user usage.

The TOE is compliant with the version of the Java card TM platform specified in [JCVM222], [JCRE222] and [JCAPI222]. It includes the Java card TM Virtual Machine (JCVM), the Java card TM Runtime Environment (JCRE) and the Java card TM Application Programming Interface (JCAPI). This set is also named Java Card System.
As the product is an open platform, the isolation mechanism between applications loaded on the TOE will be studied.

The TOE is compliant with platform APIs, which provides ways to interact with (U)SIM, SIM, UICC applications, according to [TS131.130] specifications.

The TOE provides thanks to UICC API [TS102.241] the means for the applications to access the smart card file system, to subscribe in order to receive the events of the common application toolkit framework, to handle information received and to send proactive commands.

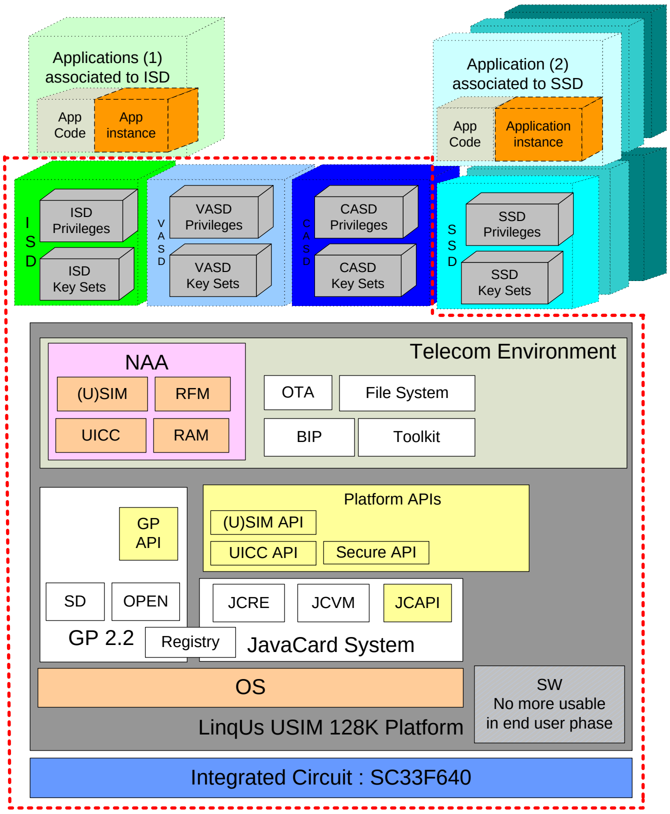
The TOE provides the (U)SIM API [TS131.130] extending the UICC API to provide features related to the 3G: it provides the means for applets to get access to the files of the (U)SIM, to register to the events defined in the USAT specification.

The BIP technology is an Over-The-Air (OTA) technology to exchange data between a (U)SIM card on a mobile phone and remote servers. It will replace the SMS technology as a data bearer for mobile phones. It is specified in 3GPP specifications as [TS102.223], [TS102.225] and [TS131.111]. Note that the BIP technology does not offer any security function for the TOE.

The TOE is compliant with the Global PlatformTM standard [GP22] which provides a set of APIs and technologies to perform in secure way, the operations involved in the management of the security domain and applications hosted by the card.
Most of the GP functionalities are present within the TOE but only a subset of services are TSF enforcing [secure loading including DAP verification, Secure channel management, Security domain management, SD (ISD, SSD, VASD) functions and privileges] as described in the Protection Profile [PP-JCS] and [ST]. Authorized Management privilege only supported for ISD.

The TOE includes the Telecom Environment with Network Authentication Application, Over The Air and BIP communication, File System management, and Toolkit services.

The following figure describes the major items included in the TOE.



(2) : Standard applet

(1) : (2) + Telecom applet +Toolkit applet
USIM, SIM,UICC, RAM, RFM linked to ISD

Figure 4: Major TOE Items and TOE scope

Configuration supplies a secure API in native language available to provide security to applets.

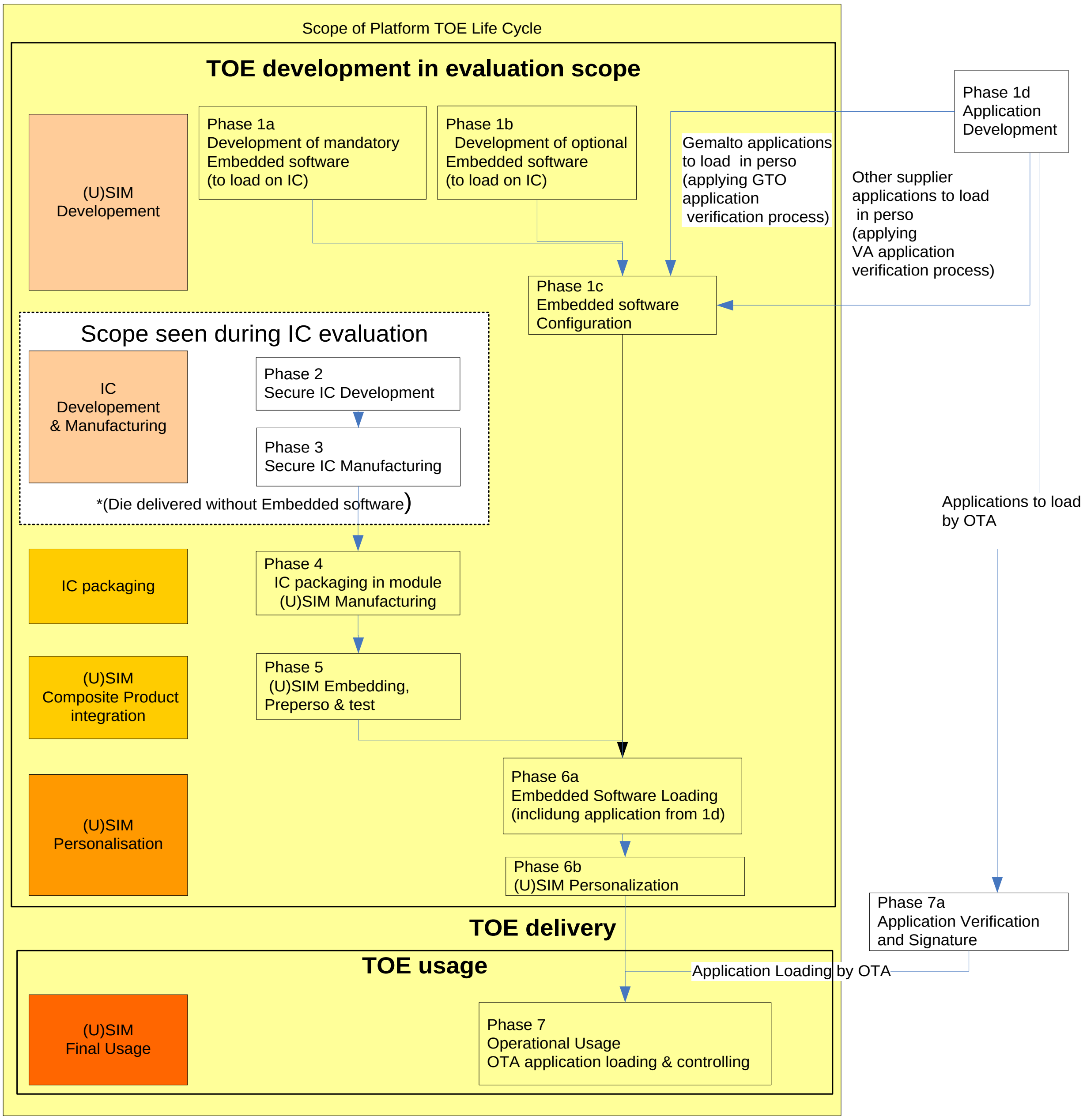
1.3.5 TOE Life Cycle

Product life cycle is described in the following picture using [PP-USIM] description refined with Gemalto specific environment due to embedded software loading in flash in phase 6.

The (U)SIM platform life cycle is composed of four stages (as defined in PP (U)SIM figure 3):

- Development (embedded software and IC separately),
- Storage, pre-personalization and test,
- Loading, Personalization and test,
- Final usage.

Refined life cycle based on [PP-BSI-0035] with Gemalto product constraints is described in the following figure.



Scope of Platform TOE Life Cycle
TOE development in evaluation scope
(U)SIM Developement
Phase 1a Development of mandatory Embedded software (to load on IC)
Phase 1b Development of optional Embedded software (to load on IC)
Gemalto applications to load in perso (applying GTO application verification process)
Phase 1d Application Development
Other supplier applications to load in perso (applying VA application verification process)
Phase 1c Embedded software Configuration
Scope seen during IC evaluation
IC Developement & Manufacturing
Phase 2 Secure IC Development
Phase 3 Secure IC Manufacturing
*(Die delivered without Embedded software)
Applications to load by OTA
IC packaging
Phase 4 IC packaging in module (U)SIM Manufacturing
(U)SIM Composite Product integration
Phase 5 (U)SIM Embedding, Preperso & test
(U)SIM Personalisation
Phase 6a Embedded Software Loading (inclidung application from 1d)
Phase 6b (U)SIM Personalization
Phase 7a Application Verification and Signature
TOE delivery
TOE usage
Application Loading by OTA
(U)SIM Final Usage
Phase 7 Operational Usage OTA application loading & controlling

Figure 5: Refined TOE Life Cycle

The following phases corresponding to the one previously described are:

- Phases 1(a, b) correspond to the development of the TOE embedded software and its configuration (1c) with applications to be loaded in phase 6.
- Phase 2, 3 and 4 correspond to IC development, manufacturing and packaging in module, respectively.
- Phase 5 concerns the composite product integration with the module and other smart card items,
- Phase 6 (a, b) is dedicated to the TOE embedded software loading and product personalization prior to TOE delivery.
- Phase 7 is the product operational phase including application loading and controlling by VA authority.

Note: The Gemalto application will be verified using evaluated Gemalto verification process prior to be loaded in Pre-Issuance by Gemalto. In the same way, but to protect supplier intellectual property, the application provided by third party supplier must be verified and signed by verification authority prior to be loaded in Pre-Issuance by Gemalto. Gemalto will check application signature prior to load this application in Pre-Issuance.

Note: The IC used in the current life cycle does not contain any embedded software prior to phase 6. It is under protection of software security function of IC dedicated software. As generic product, the ICs are stored in personalization environment but there are not dedicated to the TOE. After loading in phase 6, IC loading services is locked and no more available after phase 6. (ref to FMT_LIM from ST_IC).

The TOE is delivered at the end of phase 6 as shown in previous figure. It is the operational **LinqUs USIM 128k** product, as a personalized smart card.

As far as the EAL4+ evaluation scope is concerned, phases 1 to 6 are considered as development and manufacturing phases of the product but the TOE is the result of these phases that can consequently be seen as phases of the TOE generation.

The TOE delivery is performed at end of phase 6 and phase 7 is the operational phase of the TOE.

Out of the TOE evaluation scope, there are also the following operations linked to the TOE:

- in phase 1(d), the application development,
- in phase 7(a), the application verification and signature by verification authority prior to application loading.

1.3.6 TOE Environment

Considering the TOE, the environment is defined as follows:

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

1.3.6.1 TOE Development Environment & Roles

The TOE described in this ST is developed in different places under the control of a defined administrator as indicated below:

Phase	Administrator and Location
IC design and Dedicated Software development	STMicroelectronics Sites are defined in [ST/IC]
Embedded software Development	Gemalto (Meudon, La Ciotat, Singapore)
Embedded software Configuration	Gemalto (Tczew, Gemenos)

1.3.6.2 TOE Manufacturing Environment

The TOE described in this ST is produced in different places under the control of a defined administrator as indicated below:

Phase	Administrator and Location
IC manufacturing and Testing	STMicroelectronics Sites are defined in [ST/IC]
IC packaging	Gemalto (Pont Audemer, Tczew)
Composite Product integration	Gemalto (Pont Audemer, Tczew)

1.3.6.3 TOE Personalization Environment

The TOE described in this ST is personalized in different places under the control of a defined administrator as indicated below:

Phase	Administrator and Location
Personalization	Gemalto (Pont Audemer, Tczew)
Delivery to Final user (MNO)	From Personalization site to MNO site

1.3.6.4 TOE User Environment

Smart Cards are used in a wide range of applications to assure authorized conditional access. This specific product is to be used on terminals such as GSM and UMTS handsets or smart card readers.
The end-user environment therefore covers an unprotected environment, thus making it difficult to avoid any abuse of the TOE. The product is prepared accordingly to mitigate such attacks in this environment.
The TOE is nevertheless under the control of the MNO administration using the OTA channel. The TOE can be blocked by GP administrative commands under administrator control.

1.3.7 Actors of the TOE

One of the characteristics of the (U)SIM Java Card platforms is that several entities are represented inside these platforms:

- The Mobile Network Operator (MNO or mobile operator), issuer of the (U)SIM Java Card platform and proprietary of the TOE. The TOE guarantees that the issuer, once authenticated, could manage the loading, instantiation or deletion of applications.
- The Application Provider (AP), entity or institution responsible for the applications and their associated services. It is a financial institution (a bank), a transport operator or a third party operator.
- The Controlling Authority (CA), entity independent from the MNO represented on the (U)SIM card and responsible for securing the keys creation and personalization of the Application Provider Security Domain (APSD) (Push and Pull personalization model of [GP-UICC]).
- The Verification Authority (VA), trusted third party represented on the (U)SIM card, acting on behalf of the MNO and responsible for the verification of applications signatures (mandated DAP) during the loading process. These applications shall be validated for the standard applications or certified for the secure ones.

1.3.8 TOE Security Features

The TOE can manage secure or standard applets. These applets can be loaded and instantiated onto the TOE either before card issuance or over-the-air (OTA) in post-issuance through the mobile network, without physical manipulation of the TOE and in a connected environment. Other administrative operations can also be done using OTA.

The main security feature of the TOE is the correct and secure execution of sensitive applications, in a connected environment and with the presence on the TOE of other standard applications.

1.3.8.1 Security services to applications

The TOE offers to applications a panel of security services in order to protect application data and assets:

- Confidentiality and integrity of cryptographic keys and associated operations. Cryptographic operations are protected, including protection against observation or perturbation attacks. Confidentiality and integrity of cryptographic keys, application data are guaranteed at all time during execution of cryptographic operations.

- Confidentiality and integrity of authentication data. Authentication data are protected, including protection against observation or perturbation attacks. Confidentiality and integrity of authentication data, application data are guaranteed at all time during execution of authentication operations.

- Confidentiality and integrity of application data among applications. Applications belonging to different contexts are isolated from each other. Application data are not accessible by a normal or abnormal execution of another standard or secure application.

- Application code execution integrity. The Java Card VM and the "applications isolation" property guarantee that the application code is operating as specified in absence of perturbations. In case of perturbation, this TOE security feature must also be valid.

1.3.8.2 Application Management

The TOE offers additional security services for applications management, relying on the GlobalPlatform framework:

- The MNO as Card issuer is initially the only entity authorized to manage applications (loading, instantiation, deletion) through a secure communication channel with the card, based on SMS or BIP technology. However, the MNO can grant these privileges to the AP through the delegated management functionality of GP.

- Before loading, all applications are verified by a validation laboratory for the standard applications, or by an ITSEF for the secure applications. All loaded applications are associated at load time to a Verification Authority (VA) signature (Mandated DAP) that is verified on card by the on-card representative of the VA prior to the completion of the application loading operation and prior to the instantiation of any applet defined in the loaded application.

- Application Providers personalize their applications and Security Domains (APSD) in a confidential manner. Application Providers have Security Domain keysets enabling them to be authenticated to the corresponding Security Domain and to establish a trusted channel between the TOE and an external trusted device. These Security Domains keysets are not known by the Card issuer.

Standard and Secure applets (as defined below) are loaded in different Java Card packages.

1.3.9 Non-TOE HW/SW/FW Available to the TOE

The non TOE HW/SW/FW are those defined in [PP JCS] and [PP-USIM] as:

Byte Code verifier, Verification tool and Application DAP creation tool available to Verification authority, mobile handset, Terminal in point of sale, Remote server for administration and Trusted network and IT system for communication.

2 Conformance claims

2.1 CC Conformance Claims

This Security Target has been written using CC version V3.1 release 3.
This Security Target is CC part 2 extended with the FCS_RND.1 family. All the other security requirements have been drawn from the catalogue of requirements in Part 2 [CC-2].

This Security Target is conformant with CC part 3 [CC-3].

The evaluation is performed according [CEM] and supporting documents [JIL].

The assurance requirement of this security target is **EAL4 augmented**.
Augmentation results from compliance to [PP-JCS] are the selection of:

- **ALC_DVS.2** Sufficiency of security measures,
- **AVA_VAN.5** Advanced methodical vulnerability analysis.

2.2 PP Conformance claims

This Security Target has a "demonstrable" conformance to [PP-JCS] "open configuration".

This Security Target does not claim any conformance to the Protection Profile referenced [PP-SSCD] but it supplies the items for a composite evaluation with a signature application.

The TOE includes an Integrated Circuit certified with CC EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

The SC33F640 Security Target [ST/IC] claims strict conformance to the Security IC Platform Protection Profile [BSI-PP-2007-0035], as required by this Protection Profile.

Refinements of [BSI-PP-2007-0035] are described in [ST/IC] and are not repeated here.

2.3 Conformance rationale

The differences between this Security Target and the [PP-JCS] are described here to justify the claimed conformance to the PP.

The TOE type consistency is assumed as TOE is a (U)SIM card conformant to referenced Javacard, GP and ETSI standards.

The SPD statement consistency is assumed because assets, threats, OSP and assumption defined in the ST are a copy of those supplied in the PP.
There are extra OSP (OSP.RNG, OSP.JCAPI-Services, OSP.SecureAPI) to provide additional services to applications. OSP.BASIC-APPS-VALIDATION is added to add verification on basic applet loaded on the card. Such extension has no impact on PP coverage.

A.VERIFICATION-AUTHORITY, A.CONTROLLING-AUTHORITY, A.MOBILE-OPERATOR, A.OTA-ADMIN, A.APPS-PROVIDER and OSP.KEY-ESCROW are added to clarify conditions of product administration as defined in [PP-USIM].

The security objectives statement consistency is assumed because TOE objectives and objectives for environment are aligned between the ST and the PP.
There are extra TOE objectives (O.RND, O.JCAPI-Services, O.SecureAPI) to provide additional services to applications. Such extension has no impact on PP coverage.

OE.TRUSTED-APPS-DEVELOPER and OE.TRUSTED-APPS-PRE-ISSUANCE LOADING are added to manage pre-issuance loading covering the associated organizational security policies.

Moreover objectives for environment in [PP-JCS] become objectives for the TOE in ST due to inclusion of IC and OS in the ST scope. It is the case for OE.SCP.IC, OE.SCP.RECOVERY, and OE.SCP.SUPPORT becoming respectively O.SCP.IC, O.SCP.RECOVERY, and O.SCP.SUPPORT.

O.APPLI-AUTH is added due to presence of VASD in the TOE. OE.BASIC-APPS-VALIDATION, OE.CONTROLLING-AUTHORITY and OE.VERIFICATION-AUTHORITY are added from [PP-USIM].

OE.MOBILE-OPERATOR, OE.OTA-ADMIN, and OE.APPS-PROVIDER are added from [PP-USIM] covering the associated organizational security policies.
OE.KEY-ESCROW is added from [PP-USIM] covering the associated organizational security policy.

The list of SFRs used in the security target includes all the SFR described in the PP JCS. The following table explains refinements performed between PP JCS and ST.

Note: Items from PP JCS are listed here.

Functional requirement	Refined in [PP-JCS]	Refined in this ST
FDP_ACC.2/FIREWALL	PP	NO
FDP_ACF.1/FIREWALL	PP	NO
FDP_IFC.1/JCVM	PP	NO
FDP_IFF.1/JCVM	PP	ST
FDP_RIP.1/OBJECTS	PP	NO
FMT_MSA.1/JCRE	PP	NO
FMT_MSA.1/JCVM	PP	NO
FMT_MSA.2/FIREWALL_JCVM	PP	NO
FMT_MSA.3/FIREWALL	PP	NO
FMT_MSA.3/JCVM	PP	NO
FMT_SMF.1	PP	NO
FMT_SMR.1	PP	NO
FCS_CKM.1	PP	ST
FCS_CKM.2	PP	ST
FCS_CKM.3	PP	ST
FCS_CKM.4	PP	ST
FCS_COP.1	PP	ST
FDP_RIP.1/ABORT	PP	NO
FDP_RIP.1/APDU	PP	NO
FDP_RIP.1/bArray	PP	NO
FDP_RIP.1/KEYS	PP	NO
FDP_RIP.1/TRANSIENT	PP	NO
FDP_ROL.1/FIREWALL	PP	NO
FAU_ARP.1	PP	NO
FDP_SDI.2	PP	NO
FPR_UNO.1	PP	ST
FPT_FLS.1	PP	NO
FPT_TDC.1	PP	NO
FIA_ATD.1/AID	PP	NO
FIA_UID.2/AID	PP	NO
FIA_USB.1/AID	PP	ST
FMT_MTD.1/JCRE	PP	NO
FMT_MTD.3/JCRE	PP	NO

Functional requirement	Refined in [PP-JCS]	Refined in this ST
FDP_ITC.2/Installer	PP	NO
FMT_SMR.1/Installer	PP	NO
FPT_FLS.1/Installer	PP	NO
FPT_RCV.3/Installer	PP	ST
FDP_ACC.2/ADEL	PP	NO
FDP_ACF.1/ADEL	PP	NO
FDP_RIP.1/ADEL	PP	NO
FMT_MSA.1/ADEL	PP	NO
FMT_MSA.3/ADEL	PP	NO
FMT_SMF.1/ADEL	PP	NO
FMT_SMR.1/ADEL	PP	NO
FPT_FLS.1/ADEL	PP	NO
FDP_ACC.2/JCRMI	PP	NO
FDP_ACF.1/JCRMI	PP	NO
FDP_IFC.1/JCRMI	PP	NO
FDP_IFF.1/JCRMI	PP	NO
FMT_MSA.1/EXPORT	PP	NO
FMT_MSA.1/REM_REFS	PP	NO
FMT_MSA.3/JCRMI	PP	NO
FMT_REV.1/JCRMI	PP	NO
FMT_SMF.1/JCRMI	PP	NO
FMT_SMR.1/JCRMI	PP	NO
FDP_RIP.1/ODEL	PP	NO
FPT_FLS.1/ODEL	PP	NO
FCO_NRO.2/CM	PP	NO
FDP_IFC.2/CM	PP	NO
FDP_IFF.1/CM	PP	ST
FDP_UIT.1/CM	PP	ST
FIA_UID.1/CM	PP	NO
FMT_MSA.1/CM	PP	ST
FMT_MSA.3/CM	PP	ST
FMT_SMF.1/CM	PP	ST
FMT_SMR.1/CM	PP	ST

Functional requirement	Refined in [PP-JCS]	Refined in this ST
FTP_ITC.1/CM	PP	NO

Table 4 Refinement of SFR of PP JCS

The functional requirements are both refined in the claimed PP and in this ST. This section demonstrates the compatibility of the refinements done in both documents.

No: No refinement in PP or ST

(PP): Refinement has been made in the PP.

(ST): Additional refinement has been made in the ST.

NA: the functional requirement requires no refinement.

	Addition in ST
Assets	NO
Threats	NO
Assumptions	YES
Organizational Security Policies	YES
Security objectives for the TOE	YES
Security objectives for the operational environment	YES
Security functional requirements	YES
Security assurance requirements	NO
Security Requirements for the IT Environment	NO

Table 5 Compatibility study

List of SFR added in ST versus PP JCS is given in the next table.

Functional requirement	Refined in this ST
FPT_RCV.3 /OS	YES
FPT_RCV.4 /OS	YES
FCS_COP.1/SHA	YES
FCS_COP.1/CRC	YES
FCS_RND.1	YES
FPT_FLS.1	YES
FPT_UNO.1	YES
FPT_ITT.1	YES

Table 6 List of SFR added in ST versus PP JCS

All PP requirements have been shown to be satisfied in the extended set of requirements whose completeness, consistency and soundness has been argued in the rationale sections of the present document.

The SARs statements consistency is assumed because the same assurance package is used.

3 Security problem definition

3.1 Assets

3.1.1 Java Card System Protection Profile - Open Configuration

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

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

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

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.APP_KEYs

Cryptographic keys owned by the applets.

To be protected from unauthorized disclosure and modification.

D.PIN

Any end-user's PIN.

To be protected from unauthorized disclosure and modification.

3.1.1.2 TSF data

D.API_DATA

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

To be protected from unauthorized disclosure and modification.

D.CRYPTO

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

To be protected from unauthorized disclosure and modification.

D.JCS_CODE

The code of the Java Card System.

To be protected from unauthorized disclosure and modification.

D.JCS_DATA

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

To be protected from unauthorized disclosure or modification.

D.SEC_DATA

The runtime security data of the Java Card RE, like, for instance, the AIDs used to identify the installed applets, the currently selected applet, the current context of execution and the owner of each object.

To be protected from unauthorized disclosure and modification.

3.2 Threats

3.2.1 Java Card System Protection Profile - Open Configuration

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. Several groups of threats are distinguished according to the configuration chosen for the TOE and the means used in the attack. The classification is also inspired by the components of the TOE that are supposed to counter each threat.

3.2.1.1 CONFIDENTIALITY

T.CONFID-APPLI-DATA

The attacker executes an application to disclose data belonging to another application. See #.CONFID-APPLI-DATA for details.

Directly threatened asset(s): D.APP_C_DATA, D.PIN, and D.APP_KEYs.

T.CONFID-JCS-CODE

The attacker executes an application to disclose the Java Card System code. See #.CONFID-JCS-CODE for details.

Directly threatened asset(s): D.JCS_CODE.

T.CONFID-JCS-DATA

The attacker executes an application to disclose data belonging to the Java Card System. See #.CONFID-JCS-DATA for details.

Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA and D.CRYPTO.

3.2.1.2 INTEGRITY

T.INTEG-APPLI-CODE

The attacker executes an application to alter (part of) its own code or another application's code. See #.INTEG-APPLI-CODE for details.

Directly threatened asset(s): D.APP_CODE

T.INTEG-APPLI-CODE.LOAD

The attacker modifies (part of) its own or another application code when an application package is transmitted to the card for installation. See #.INTEG-APPLI-CODE for details.

Directly threatened asset(s): D.APP_CODE.

T.INTEG-APPLI-DATA

The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA for details.

Directly threatened asset(s): D.APP_I_DATA, D.PIN, and D.APP_KEYs.

T.INTEG-APPLI-DATA.LOAD

The attacker modifies (part of) the initialization data contained in an application package when the package is transmitted to the card for installation. See #.INTEG-APPLI-DATA for details.

Directly threatened asset(s): D.APP_I_DATA and D_APP_KEYs.

T.INTEG-JCS-CODE

The attacker executes an application to alter (part of) the Java Card System code. See #.INTEG-JCS-CODE for details.

Directly threatened asset(s): D.JCS_CODE.

T.INTEG-JCS-DATA

The attacker executes an application to alter (part of) Java Card System or API data. See #.INTEG-JCS-DATA for details.

Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA and D.CRYPTO.

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

3.2.1.3 IDENTITY USURPATION

T.SID.1

An applet impersonates another application, or even the Java Card RE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal. See #.SID for details.

Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN and D.APP_KEYs.

T.SID.2

The attacker modifies the TOE's attribution of a privileged role (e.g. default applet and currently selected applet), which allows illegal impersonation of this role. See #.SID for further details.

Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

3.2.1.4 UNAUTHORIZED EXECUTION

T.EXE-CODE.1

An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.

Directly threatened asset(s): D.APP_CODE.

T.EXE-CODE.2

An applet performs an execution of a method fragment or arbitrary data. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.

Directly threatened asset(s): D.APP_CODE.

T.EXE-CODE-REMOTE

The attacker performs an unauthorized remote execution of a method from the CAD. See #.EXE-APPLI-CODE for details.

Directly threatened asset(s): D.APP_CODE.

Application note:

This threat concerns version 2.2.x of the Java Card RMI, which allow external users (that is, other than on-card applets) to trigger the execution of code belonging to an on-card applet. On the contrary, T.EXE-CODE.1 is restricted to the applets under the TSF.

T.NATIVE

An applet executes a native method to bypass a TOE Security Function such as the firewall. See #.NATIVE for details.

Directly threatened asset(s): D.JCS_DATA.

3.2.1.5 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. See #.RESOURCES for details.

Directly threatened asset(s): D.JCS_DATA.

3.2.1.6 CARD MANAGEMENT

T.DELETION

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

Directly threatened asset(s): D.SEC_DATA and D.APP_CODE.

T.INSTALL

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

Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

3.2.1.7 SERVICES

T.OBJ-DELETION

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

Directly threatened asset(s): D.APP_C_DATA, D.APP_I_DATA and D.APP_KEYs.

3.2.1.8 MISCELLANEOUS

T.PHYSICAL

The attacker discloses or modifies the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means. This threat includes IC failure analysis, electrical probing, unexpected tearing, and DPA. That also includes the modification of the runtime execution of Java Card System or SCP software through alteration of the intended execution order of (set of) instructions through physical tampering techniques.

This threatens all the identified assets.

This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

3.3 Organisational Security Policies

3.3.1 Java Card System Protection Profile - Open Configuration

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

OSP.VERIFICATION

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

3.3.2 (U)SIM

This section describes the organizational security policies to be enforced with respect to the TOE (U)SIM environment.

OSP.BASIC-APPS-VALIDATION

Basic applications shall be associated to a digital signature which will be checked by a VA during the loading into the TOE.

In addition to the rules stated by the Java Card specification, the validation process must enforce that basic applications:

must follow the extra-rules stated in the user manual of the considered (U)SIM Java Card Platform,

cannot be libraries,

must not use RMI,

must not use proprietary libraries which are not certified (except system libraries),

access control to certified proprietary libraries is controlled by the secure application which has defined the library,

must be associated to an identifier and this identifier has to be used in parameter of the function calls.

OSP.RNG

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

OSP.JCAPI-Services

This policy shall ensure that hashing and checksum security services defined in [JCAPI] provided by the TOE to applet is secure. Thus attacker is not able to predict or obtain information on manipulated data.

OSP.SecureAPI

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

OSP.TRUSTED-APPS-DEVELOPER

There are application developers (as Gemalto) considered as trusted by platform issuer and application providers. The confidence in these actors has been obtained by audit of development process and development environment performed by ITSEF during private scheme evaluation or Common Criteria composite evaluation process.

Application note: As a consequence, the development process applied by a trusted developer provides confidence that applications developed by such actors are considered as not agressive versus the platform and other applications loaded on the platform.

OSP.TRUSTED-APPS-PRE-ISSUANCE LOADING

For Pre-Issuance loading of trusted* applications, the audited process during Platform evaluation must be used.

[* Application notes: An application is considered as trusted if it has been developed or verified by a trusted actor (as Gemalto). An application developed by a third party can be considered as a trusted application only it has been verified and signed by verification authority. The application and associated signature will be verified by Gemalto prior authorizing loading in pre-issuance.

As a consequence, the loading process applied by a trusted personalizer provides confidence that applications developed by trusted actors are considered as not agressive versus the platform and other applications loaded on the platform.]

OSP.KEY-ESCROW

The key escrow is a trusted actor in charge of the secure storage of the initial AP keys generated by the TOE personalizer during initial personalization. He ensures the security of the keys.

3.4 Assumptions

3.4.1 Java Card System Protection Profile - Open Configuration

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

A.APPLET

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

A.DELETION

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

A.VERIFICATION

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

3.4.2 (U)SIM

This section introduces the assumptions made on the environment of the TOE(U)SIM.

A.CONTROLLING-AUTHORITY

The CA is a trusted actor responsible for securing the APSD keys creation and personalization. He is responsible for his security domain keys (CASD keys).

A.VERIFICATION-AUTHORITY

The VA is a trusted actor who is able to guarantee and check the digital signature attached to a basic or secure application.

A.MOBILE-OPERATOR

The mobile operator is a trusted actor responsible for the mobile network and the associated OTA servers.

The mobile operator as Card issuer cannot get access or change the application data which belongs to the AP.

A.OTA-ADMIN

Administrators of the mobile operator OTA servers are trusted people. They are trained to use and administrate securely those servers. They have the means and the equipments to perform their tasks.

They are aware of the sensitivity of the assets they managed and the responsibilities associated to the administration of OTA servers.

A.APPS-PROVIDER

The AP is a trusted actor that provides standard or secure applications. He is responsible for his security domain keys (APSD keys).

4 Security Objectives

4.1 Security Objectives for the TOE

4.1.1 Java Card System Protection Profile - Open Configuration

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

4.1.1.1 IDENTIFICATION

O.SID

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

4.1.1.2 EXECUTION

O.FIREWALL

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

O.GLOBAL_ARRAYS_CONFID

The TOE shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection.

The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

O.GLOBAL_ARRAYS_INTEG

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

O.NATIVE

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

O.OPERATE

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

O.REALLOCATION

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

Application note:

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

O.RESOURCES

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

4.1.1.3 SERVICES

O.ALARM

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

O.CIPHER

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

O.KEY-MNGT

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

Application note:

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

O.PIN-MNGT

The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT for details.

Application note:

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

O.REMOTE

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

O.TRANSACTION

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

4.1.1.4 OBJECT DELETION

O.OBJ-DELETION

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

4.1.1.5 APPLET MANAGEMENT

O.DELETION

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

O.LOAD

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

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.INSTALL

The TOE shall ensure that the installation of an applet performs as expected (See #.INSTALL for details).

4.1.1.6 SCP

O.SCP.SUPPORT

The TOE OS shall support the following functionalities:

(1) It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System.

(2) It provides secure low-level cryptographic processing to the Java Card System, GlobalPlatform.

(3) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism.

(4) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection).

O.SCP.RECOVERY

If there is a loss of power, or if the smart card is withdrawn from the CAD while an operation is in progress, the SCP must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state.

This security objective refers to the security aspect #.SCP(1): The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state.

O.SCP.IC

The SCP shall provide all IC security features against physical attacks.

This security objective refers to the point (7) of the security aspect #.SCP:

> It is required that the IC is designed in accordance with a well-defined set of policies and Standards (likely specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

4.1.2 (U)SIM

This section defines the security objectives to be achieved by the TOE(U)SIM.

O.RND

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

O.APPLI-AUTH

The card manager shall enforce the application security policies established by the card issuer by requiring application authentication during application loading on the card.

O.JCAPI-Services

The TOE must contribute to ensure that data manipulated during SHA and CRC services as defined in [JCAPI] shall not be observed.

O.Secure_API

The TOE shall provide to application a secure_API means to optimize control on sensitive operations performed by application.

TOE shall provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

4.2 Security objectives for the Operational Environment

4.2.1 Java Card System Protection Profile - Open Configuration

This section introduces the security objectives to be achieved by the environment.

4.2.1.1 (U)SIM

OE.VERIFICATION-AUTHORITY

The VA should be a trusted actor who is able to guarantee and check the digital signature attached to an application.

OE.BASIC-APPS-VALIDATION

Basic applications must be analysed during the validation process in order to ensure that the rules for correct usage of the TOE are still enforced.

OE.CONTROLLING-AUTHORITY

The CA shall be a trusted actor responsible for securing the APSD keys creation and personalisation. He must be responsible for his security domain keys (CASD keys).

OE.TRUSTED-APPS-DEVELOPER

The trusted application developer shall be a trusted actor that provides basic or secure application where correct usage of the TOE has been verified applying a secure development process in secure development environment.

OE.TRUSTED-APPS-PRE-ISSUANCE LOADING

The trusted pre-issuance loading on the platform must be done only using verified applet applying an audited process in a secure environment.

OE.MOBILE-OPERATOR

The mobile operator shall be a trusted actor responsible for the mobile network and the associated OTA servers.

OE.OTA-ADMIN

Administrators of the mobile operator OTA servers shall be trusted people. They shall be trained to use and administrate those servers. They have the means and the equipments to perform their tasks.

They must be aware of the sensitivity of the assets they manage and the responsibilities associated to the administration of OTA servers.

OE.APPS-PROVIDER

The AP shall be a trusted actor that provides standard or secure application. He must be responsible of his security domain keys.

OE.KEY-ESCROW

The key escrow shall be a trusted actor in charge of the secure storage of the AP initial keys generated by the personalizer.

4.2.1.2 Miscellaneous

OE.APPLET

No applet loaded post-issuance shall contain native methods.

OE.CARD-MANAGEMENT

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

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

OE.VERIFICATION

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

4.2.1.3 Conclusion

This section introduces the security objectives to be achieved by the environment.

4.3 Security Objectives Rationale

4.3.1 Threats

4.3.1.1 Java Card System Protection Profile - Open Configuration

CONFIDENTIALITY

T.CONFID-APPLI-DATA This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION)and OE.BASIC-APPS-VALIDATION. It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

As applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys, PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.

Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The disclosure of such data is prevented by the (O.GLOBAL_ARRAYS_CONFID) security objective.

Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

T.CONFID-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of those instructions enables reading a piece of code, no Java Card applet can therefore be executed to disclose a piece of code. Native applications are also harmless because of the objectives (O.NATIVE) and OE.BASIC-APPS-VALIDATION, so no application can be run to disclose a piece of code.

The (#.VERIFICATION) security aspect is addressed in this PP by the objective for the environment OE.VERIFICATION.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.CONFID-JCS-DATA This threat is covered by bytecode verification (OE.VERIFICATION and OE.BASIC-APPS-VALIDATION) and the isolation commitments stated in the (O.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

INTEGRITY

T.INTEG-APPLI-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objectives (O.NATIVE) and OE.BASIC-APPS-VALIDATION, so no application can be run to modify a piece of code.

The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.INTEG-APPLI-CODE.LOAD This threat is countered by the security objective O.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of packages code.

By controlling the access to card management functions such as the installation, update or deletion of applets the objective OE.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-APPLI-DATA This threat is countered by bytecode verification (OE.VERIFICATION and OE.BASIC-APPS-VALIDATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.

Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the (O.GLOBAL_ARRAYS_INTEG) objective.

Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

T.INTEG-APPLI-DATA.LOAD This threat is countered by the security objective O.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of applications data.

By controlling the access to card management functions such as the installation, update or deletion of applets the objective OE.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objectives (O.NATIVE) and OE.BASIC-APPS-VALIDATION, so no application can be run to disclose or modify a piece of code.

The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.INTEG-JCS-DATA This threat is countered by bytecode verification (OE.VERIFICATION and OE.BASIC-APPS-VALIDATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

IDENTITY USURPATION

T.SID.1 As impersonation is usually the result of successfully disclosing and modifying some assets, this threat is mainly countered by the objectives concerning the isolation of application data (like PINs), ensured by the (O.FIREWALL). Uniqueness of subject-identity (O.SID) also participates to face this threat. It should be noticed that the AIDs, which are used for applet identification, are TSF data.

In this configuration, usurpation of identity resulting from a malicious installation of an applet on the card is covered by the objective O.INSTALL.

The installation parameters of an applet (like its name) are loaded into a global array that is also shared by all the applications. The disclosure of those parameters (which could be used to impersonate the applet) is countered by the objective (O.GLOBAL_ARRAYS_CONFID) and (O.GLOBAL_ARRAYS_INTEG).

The objective OE.CARD-MANAGEMENT contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter this threat.

T.SID.2 This is covered by integrity of TSF data, subject-identification (O.SID), the firewall (O.FIREWALL) and its good working order (O.OPERATE).

The objective O.INSTALL contributes to counter this threat by ensuring that installing an applet has no effect on the state of other applets and thus can't change the TOE's attribution of privileged roles.

The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE objective of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

UNAUTHORIZED EXECUTION

T.EXE-CODE.1 Unauthorized execution of a method is prevented by the objectives OE.VERIFICATION and OE.BASIC-APPS-VALIDATION. This threat particularly concerns the point (8) of the security aspect #VERIFICATION (access modifiers and scope of accessibility for classes, fields and methods). The O.FIREWALL objective is also concerned, because it prevents the execution of non-shareable methods of a class instance by any subject apart from the class instance owner.

T.EXE-CODE.2 Unauthorized execution of a method fragment or arbitrary data is prevented by the objectives OE.VERIFICATION and OE.BASIC-APPS-VALIDATION. This threat particularly concerns those points of the security aspect related to control flow confinement and the validity of the method references used in the bytecodes.

T.EXE-CODE-REMOTE The O.REMOTE security objective contributes to prevent the invocation of a method that is not supposed to be accessible from outside the card.

T.NATIVE This threat is countered by O.NATIVE which ensures that a Java Card applet can only access native methods indirectly that is, through an API which is assumed to be secure thanks to OE.BASIC-APPS-VALIDATION. OE.APPLET also covers this threat by ensuring that no native applets shall be loaded in post-issuance. In addition to this, the bytecode verifier also prevents the program counter of an applet to jump into a piece of native code by confining the control flow to the currently executed methods (OE.VERIFICATION) and OE.BASIC-APPS-VALIDATION.

DENIAL OF SERVICE

T.RESOURCES This threat is directly countered by objectives on resource-management (O.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner.

Consumption of resources during installation and other card management operations are covered, in case of failure, by O.INSTALL.

It should be noticed that, for what relates to CPU usage, the Java Card platform is single-threaded and it is possible for an ill-formed application (either native or not) to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevent them from being blocked should a card fails to answer. That point is out of scope of this Protection Profile, though.

Finally, the objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.RESOURCES objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

CARD MANAGEMENT

T.DELETION This threat is covered by the O.DELETION security objective which ensures that both applet and package deletion perform as expected.

The objective OE.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

T.INSTALL This threat is covered by the security objective O.INSTALL which ensures that the installation of an applet performs as expected and the security objectives O.LOAD which ensures that the loading of a package into the card is safe.

The objective OE.CARD-MANAGEMENT and O.APPLI-AUTH controls the access to card management functions and thus contributes to cover this threat.

SERVICES

T.OBJ-DELETION This threat is covered by the O.OBJ-DELETION security objective which ensures that object deletion shall not break references to objects.

MISCELLANEOUS

T.PHYSICAL This threat is countered by physical protections which rely on the underlying platform and are therefore an environmental issue.

The security objectives O.SCP.SUPPORT and O.SCP.IC protect sensitive assets of the platform against loss of integrity and confidentiality and especially ensure the TSFs cannot be bypassed or altered. Physical protections rely on the underlying platform and are therefore an environmental issue.

4.3.2 Organisational Security Policies

4.3.2.1 Java Card System Protection Profile - Open Configuration

OSP.VERIFICATION This policy is upheld by the security objectives of the environment OE.VERIFICATION and OE.BASIC-APPS-VALIDATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time.

4.3.2.2 (U)SIM

OSP.BASIC-APPS-VALIDATION This OSP is enforced by the security objective for the operational environment of the TOE OE.BASIC-APPS-VALIDATION.

OSP.RNG This OSP is enforced by the TOE security objective O.RND.

OSP.JCAPI-Services This OSP is enforced by the TOE security objective O.JCAPI-Services.

OSP.SecureAPI This OSP is enforced by the TOE security objective O.Secure_API.

OSP.TRUSTED-APPS-DEVELOPER This OSP is enforced by the security objective OE.TRUSTED-APPS-DEVELOPER.

OSP.TRUSTED-APPS-PRE-ISSUANCE LOADING This OSP is enforced by the security objective OE.TRUSTED-APPS-PRE-ISSUANCE LOADING.

OSP.KEY-ESCROW This Security policy is directly upheld by OE.KEY-ESCROW.

4.3.3 Assumptions

4.3.3.1 Java Card System Protection Profile - Open Configuration

A.APPLET This assumption is upheld by the security objective for the operational environment OE.APPLET which ensures that no applet loaded post-issuance shall contain native methods.

A.DELETION The assumption A.DELETION is upheld by the environmental objective OE.CARD-MANAGEMENT which controls the access to card management functions such as deletion of applets.

A.VERIFICATION This assumption is upheld by the security objective on the operational environment OE.BASIC-APPS-VALIDATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time.

4.3.3.2 (U)SIM

A.CONTROLLING-AUTHORITY This assumption is directly upheld by OE.CONTROLLING-AUTHORITY.

A.VERIFICATION-AUTHORITY This assumption is directly upheld by OE.VERIFICATION-AUTHORITY.

A.MOBILE-OPERATOR This assumption is directly upheld by OE.MOBILE-OPERATOR.

A.OTA-ADMIN This assumption is directly upheld by OE.OTA-ADMIN.

A.APPS-PROVIDER This assumption is directly upheld by OE.APPS-PROVIDER.

4.3.4 SPD and Security Objectives

Threats	Security Objectives	Rationale
T.CONFID-APPLI-DATA	OE.CARD-MANAGEMENT, O.SID, O.OPERATE, O.FIREWALL, O.REALLOCATION, O.GLOBAL_ARRAYS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.SCP.SUPPORT, O.SCP.RECOVERY, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.CONFID-JCS-CODE	OE.CARD-MANAGEMENT, O.NATIVE, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.CONFID-JCS-DATA	OE.CARD-MANAGEMENT, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.SCP.SUPPORT, O.SCP.RECOVERY, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.INTEG-APPLI-CODE	OE.CARD-MANAGEMENT, O.NATIVE, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.INTEG-APPLI-CODE.LOAD	O.LOAD, OE.CARD-MANAGEMENT	Section 2.3.1
T.INTEG-APPLI-DATA	OE.CARD-MANAGEMENT, O.SID, O.OPERATE, O.FIREWALL, O.REALLOCATION, O.GLOBAL_ARRAYS_INTEG, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.SCP.SUPPORT, O.SCP.RECOVERY, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.INTEG-APPLI-DATA.LOAD	O.LOAD, OE.CARD-MANAGEMENT	Section 2.3.1
T.INTEG-JCS-CODE	OE.CARD-MANAGEMENT, O.NATIVE, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1

Threats	Security Objectives	Rationale
T.INTEG-JCS-DATA	OE.CARD-MANAGEMENT, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.SCP.SUPPORT, O.SCP.RECOVERY, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.SID.1	OE.CARD-MANAGEMENT, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID	Section 2.3.1
T.SID.2	O.SID, O.OPERATE, O.FIREWALL, O.INSTALL, O.SCP.SUPPORT, O.SCP.RECOVERY	Section 2.3.1
T.EXE-CODE.1	O.FIREWALL, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.EXE-CODE.2	OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.EXE-CODE-REMOTE	O.REMOTE	Section 2.3.1
T.NATIVE	OE.APPLET, O.NATIVE, OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.1
T.RESOURCES	O.INSTALL, O.OPERATE, O.RESOURCES, O.SCP.SUPPORT, O.SCP.RECOVERY	Section 2.3.1
T.DELETION	O.DELETION, OE.CARD-MANAGEMENT	Section 2.3.1
T.INSTALL	O.INSTALL, O.LOAD, OE.CARD-MANAGEMENT, O.APPLI-AUTH	Section 2.3.1
T.OBJ-DELETION	O.OBJ-DELETION	Section 2.3.1
T.PHYSICAL	O.SCP.IC, O.SCP.SUPPORT	Section 2.3.1

Table 7 Threats and Security Objectives - Coverage

Security Objectives	Threats
O.SID	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2
O.FIREWALL	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2, T.EXE-CODE.1
O.GLOBAL_ARRAYS_CONFID	T.CONFID-APPLI-DATA, T.SID.1
O.GLOBAL_ARRAYS_INTEG	T.INTEG-APPLI-DATA, T.SID.1
O.NATIVE	T.CONFID-JCS-CODE, T.INTEG-APPLI-CODE, T.INTEG-JCS-CODE, T.NATIVE
O.OPERATE	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.REALLOCATION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.RESOURCES	T.RESOURCES
O.ALARM	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA
O.CIPHER	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.KEY-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.PIN-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.REMOTE	T.EXE-CODE-REMOTE
O.TRANSACTION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.OBJ-DELETION	T.OBJ-DELETION
O.DELETION	T.DELETION
O.LOAD	T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA.LOAD, T.INSTALL
O.INSTALL	T.SID.1, T.SID.2, T.RESOURCES, T.INSTALL
O.SCP.SUPPORT	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES, T.PHYSICAL

Security Objectives	Threats
O.SCP.RECOVERY	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.SCP.IC	T.PHYSICAL
O.RND	
O.APPLI-AUTH	T.INSTALL
O.JCAPI-Services	
O.Secure_API	
OE.VERIFICATION-AUTHORITY	
OE.BASIC-APPS-VALIDATION	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE
OE.CONTROLLING-AUTHORITY	
OE.TRUSTED-APPS-DEVELOPER	
OE.TRUSTED-APPS-PRE-ISSUANCE LOADING	
OE.MOBILE-OPERATOR	
OE.OTA-ADMIN	
OE.APPS-PROVIDER	
OE.KEY-ESCROW	
OE.APPLET	T.NATIVE
OE.CARD-MANAGEMENT	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, T.DELETION, T.INSTALL
OE.VERIFICATION	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE

Table 8 Security Objectives and Threats - Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.VERIFICATION	OE.BASIC-APPS-VALIDATION, OE.VERIFICATION	Section 2.3.2
OSP.BASIC-APPS-VALIDATION	OE.BASIC-APPS-VALIDATION	Section 2.3.2
OSP.RNG	O.RND	Section 2.3.2
OSP.JCAPI-Services	O.JCAPI-Services	Section 2.3.2
OSP.SecureAPI	O.Secure_API	Section 2.3.2
OSP.TRUSTED-APPS-DEVELOPER	OE.TRUSTED-APPS-DEVELOPER	Section 2.3.2
OSP.TRUSTED-APPS-PRE-ISSUANCE LOADING	OE.TRUSTED-APPS-PRE-ISSUANCE LOADING	Section 2.3.2
OSP.KEY-ESCROW	OE.KEY-ESCROW	Section 2.3.2

Table 9 OSPs and Security Objectives - Coverage

Security Objectives	Organisational Security Policies
O.SID	
O.FIREWALL	
O.GLOBAL_ARRAYS_CONFID	
O.GLOBAL_ARRAYS_INTEG	
O.NATIVE	
O.OPERATE	
O.REALLOCATION	
O.RESOURCES	
O.ALARM	
O.CIPHER	
O.KEY-MNGT	
O.PIN-MNGT	
O.REMOTE	
O.TRANSACTION	
O.OBJ-DELETION	
O.DELETION	
O.LOAD	
O.INSTALL	
O.SCP.SUPPORT	
O.SCP.RECOVERY	
O.SCP.IC	
O.RND	OSP.RNG
O.APPLI-AUTH	
O.JCAPI-Services	OSP.JCAPI-Services
O.Secure_API	OSP.SecureAPI
OE.VERIFICATION-AUTHORITY	
OE.BASIC-APPS-VALIDATION	OSP.VERIFICATION, OSP.BASIC-APPS-VALIDATION
OE.CONTROLLING-AUTHORITY	
OE.TRUSTED-APPS-DEVELOPER	OSP.TRUSTED-APPS-DEVELOPER
OE.TRUSTED-APPS-PRE-ISSUANCE LOADING	OSP.TRUSTED-APPS-PRE-ISSUANCE LOADING

Security Objectives	Organisational Security Policies
OE.MOBILE-OPERATOR	
OE.OTA-ADMIN	
OE.APPS-PROVIDER	
OE.KEY-ESCROW	OSP.KEY-ESCROW
OE.APPLET	
OE.CARD-MANAGEMENT	
OE.VERIFICATION	OSP.VERIFICATION

Table 10 Security Objectives and OSPs - Coverage

Assumptions	Security objectives for the Operational Environment	Rationale
A.APPLET	OE.APPLET	Section 2.3.3
A.DELETION	OE.CARD-MANAGEMENT	Section 2.3.3
A.VERIFICATION	OE.BASIC-APPS-VALIDATION	Section 2.3.3
A.CONTROLLING-AUTHORITY	OE.CONTROLLING-AUTHORITY	Section 2.3.3
A.VERIFICATION-AUTHORITY	OE.VERIFICATION-AUTHORITY	Section 2.3.3
A.MOBILE-OPERATOR	OE.MOBILE-OPERATOR	Section 2.3.3
A.OTA-ADMIN	OE.OTA-ADMIN	Section 2.3.3
A.APPS-PROVIDER	OE.APPS-PROVIDER	Section 2.3.3

Table 11 Assumptions and Security Objectives for the Operational Environment - Coverage

Security objectives for the Operational Environment	Assumptions
OE.VERIFICATION-AUTHORITY	A.VERIFICATION-AUTHORITY
OE.BASIC-APPS-VALIDATION	A.VERIFICATION
OE.CONTROLLING-AUTHORITY	A.CONTROLLING-AUTHORITY
OE.TRUSTED-APPS-DEVELOPER	
OE.TRUSTED-APPS-PRE-ISSUANCE LOADING	
OE.MOBILE-OPERATOR	A.MOBILE-OPERATOR
OE.OTA-ADMIN	A.OTA-ADMIN
OE.APPS-PROVIDER	A.APPS-PROVIDER
OE.KEY-ESCROW	
OE.APPLET	A.APPLET
OE.CARD-MANAGEMENT	A.DELETION
OE.VERIFICATION	

Table 12 Security Objectives for the Operational Environment and Assumptions - Coverage

5 Extended requirements

5.1 Extended families

5.1.1 Extended family FCS_RND - Random Number Generation

5.1.1.1 Description

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

5.1.1.2 Extended components

Extended component FCS_RND.1

Description

The generation of random numbers requires that random numbers meet a defined quality metric.

Definition

FCS_RND.1 Random Number Generation

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

Dependencies: No dependencies.

Rationale

It was chosen to define FCS_RNG.1 explicitly, because Part 2 of the Common Criteria do not contain generic security functional requirements for Random Number generation.

5.1.1.3 Rationale

This family has been introduced initially by IC manufacturer to offer unpredictible random number generation. It is extended here to software platform.

6 Security Requirements

6.1 Security Functional Requirements

6.1.1 Java Card System Protection Profile - Open Configuration

This section states the security functional requirements for the Java Card System - Open configuration. For readability and for compatibility with the original Java Card System Protection Profile Collection - Standard 2.2 Configuration [PP/0305], requirements are arranged into groups. All the groups defined in the table below apply to this Protection Profile.

Group	Description
Core with Logical Channels (*CoreG_LC*)	The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature. This group is the union of requirements from the Core (*CoreG*) and the Logical channels (*LCG*) groups defined in [PP/0305] (cf. Java Card System Protection Profile Collection [PP JCS]).
Installation (*InstG*)	The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.
Applet deletion (*ADELG*)	The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 2.2.
Remote Method Invocation (RMI)	The RMIG contains the security requirements for the remote method invocation feature, which provides a new protocol of communication between the terminal and the applets. This was introduced in Java Card specification version 2.2.
Object deletion (*ODELG*)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.
Secure carrier (*CarG*)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.

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

Subjects (prefixed with an "S") are described in the following table:

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE22], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §7.1.3.1.
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §7.1.7.
S.CAD	The CAD represents the actor that requests, by issuing commands to the card, for RMI services. It also plays the role of the off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.
S.JCRE	The runtime environment under which Java programs in a smart card are executed.
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.

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

Object	Description
O.APPLET	Any installed applet, its code and data.
O.CODE_PKG	The code of a package, including all linking information. On the Java Card platform, a package is the installation unit.
O.JAVAOBJECT	Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.
O.REMOTE_MTHD	A method of a remote interface.
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.RMI_SERVICE	These are instances of the class javacardx.rmi.RMIService. They are the objects that actually process the RMI services.
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.

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

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

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

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected".
Applet's version number	The version number of an applet (package) indicated in the export file.
Class	Identifies the implementation class of the remote object.
Context	Package AID or "Java Card RE".
Currently Active Context	Package AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM22], §4.5.2).
ExportedInfo	Boolean (indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies the remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file.
Registered Applets	The set of AID of the applet instances registered on the card.
Remote	An object is Remote if it is an instance of a class that directly or indirectly implements the interface java.rmi.Remote.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Returned References	The set of remote object references that have been sent to the CAD during the applet selection session. This attribute is implementation dependent.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, Java Card RE entry point or global array.

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

(*) Transient objects of type CLEAR_ON_RESET behave like persistent objects in that they can be accessed only when the Currently Active Context is the object's context.

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

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.
OP.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.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.INVOKE(O.RMI_SERVICE,...)	Requests a remote method invocation on the remote object.
OP.JAVA(...)	Any access in the sense of [JCRE22], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.RET_RORD(S.JCRE,S.CAD,I.RORD)	Send a remote object reference descriptor to the CAD.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE22], §6.2.8.7).
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).

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

6.1.1.1 CoreG_LC Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall.

Firewall Policy

FDP_ACC.2/FIREWALL Complete access control

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

Refinement:

The operations involved in the policy are:

OP.CREATE,

OP.INVK_INTERFACE,

OP.INVK_VIRTUAL,

OP.JAVA,

OP.THROW,

OP.TYPE_ACCESS.

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

Application note:

It should be noticed 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_ACF.1/FIREWALL Security attribute based access control

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

Subject/Object	**Security attributes**
S.PACKAGE	**LC Selection Status**
S.JCVM	**Active Applets, Currently Active Context**
S.JCRE	**Selected Applet Context**
O.JAVAOBJECT	**Sharing, Context, LifeTime**

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

R.JAVA.1 ([JCRE22], §6.2.8): S.PACKAGE may freely perform 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): S.PACKAGE may freely perform 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): S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.

R.JAVA.4 ([JCRE22], §6.2.8.6): S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following conditions applies:

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

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

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

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

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

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

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

Application note:

FDP_ACF.1.4/FIREWALL:

The deletion of applets may render some O.JAVAOBJECT inaccessible, and the Java Card RE may be in charge of this aspect. This can be done, for instance, by ensuring that references to objects belonging to a deleted application are considered as a null reference. Such a mechanism is implementation-dependent.

In the case of an array type, fields are components of the array ([JVM], §2.14, §2.7.7), as well as the length; the only methods of an array object are those inherited from the Object class.

The Sharing attribute defines four categories of objects:

Standard ones, whose both fields and methods are under the firewall policy,

Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,

JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,

Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.

When a new object is created, it is associated with the Currently Active Context. But the object is owned by the applet instance within the Currently Active Context when the object is instantiated ([JCRE22], §6.1.3). An object is owned by an applet instance, by the JCRE or by the package library where it has been defined (these latter objects can only be arrays that initialize static fields of packages).

([JCRE22], Glossary) Selected Applet Context. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.

While the expression "Selected Applet Context" refers to a specific installed applet, the relevant aspect to the policy is the context (package AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected package.

([JCRE22], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

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

It should be noticed that the Java Card platform, version 2.2.x and version 3 Classic Edition, introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM22], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([JCRE22], §4).

FDP_IFC.1/JCVM Subset information flow control

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

Application note:

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

FDP_IFF.1/JCVM Simple security attributes

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

Subjects	Security attributes
S.JCVM	Currently Active Context

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

An operation OP.PUT(S1, S.MEMBER, I.DATA) is allowed if and only if the Currently Active Context is "Java Card RE";

other OP.PUT operations are allowed regardless of the Currently Active Context's value.

FDP_IFF.1.3/JCVM The TSF shall enforce the **[No additional rules]**.

FDP_IFF.1.4/JCVM The TSF shall explicitly authorise an information flow based on the following rules: **[No additional rules]**.

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **[No additional rules]**.

Application note:

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE22], §6.2.8.1-3).

It should be noticed that this policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT operation under this scheme.

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 initialized with default values when the resource is allocated [JVM], §2.5.1.

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP** to restrict the ability to **modify** the security attributes **Selected Applet Context** to **the Java Card RE**.

Application note:

The modification of the Selected Applet Context should be performed in accordance with the rules given in [JCRE22], §4 and [JCVM22], §3.4.

FMT_MSA.1/JCVM Management of security attributes

FMT_MSA.1.1/JCVM The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes **Currently Active Context and Active Applets** to **the Java Card VM (S.JCVM)**.

Application note:

The modification of the Currently Active Context should be performed in accordance with the rules given in [JCRE22], §4 and [JCVM22], §3.4.

FMT_MSA.2/FIREWALL_JCVM Secure security attributes

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

Application note:

The following rules are given as examples only. For instance, the last two rules are motivated by the fact that the Java Card API defines only transient arrays factory methods. Future versions may allow the creation of transient objects belonging to arbitrary classes; such evolution will naturally change the range of "secure values" for this component.

The Context attribute of an O.JAVAOBJECT must correspond to that of an installed applet or be "Java Card RE".

An O.JAVAOBJECT whose Sharing attribute is a Java Card RE entry point or a global array necessarily has "Java Card RE" as the value for its Context security attribute.

An O.JAVAOBJECT whose Sharing attribute value is a global array necessarily has "array of primitive type" as a JavaCardClass security attribute's value.

Any O.JAVAOBJECT whose Sharing attribute value is not "Standard" has a PERSISTENT-LifeTime attribute's value.

Any O.JAVAOBJECT whose LifeTime attribute value is not PERSISTENT has an array type as JavaCardClass attribute's value.

FMT_MSA.3/FIREWALL Static attribute initialisation

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

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

Application note:

FMT_MSA.3.1/FIREWALL

Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE22], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".

The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

The intent is that none of the identified roles has privileges with regard to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP. The operation shall fail anyway if the created object would have had security attributes whose value violates FMT_MSA.2.1/FIREWALL_JCVM.

FMT_MSA.3/JCVM Static attribute initialisation

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

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

FMT_SMF.1 Specification of Management Functions

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

modify the Currently Active Context, the Selected Applet Context and the Active Applets.

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles**:**

Java Card RE (JCRE),

Java Card VM (JCVM).

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

Application Programming Interface

The following SFRs are related to the Java Card API.

The whole set of cryptographic algorithms is generally not implemented because of limited memory resources and/or limitations due to exportation. Therefore, the following requirement should only apply to the implemented subset.

It should be noticed that the execution of native code is not within the TSF. Nevertheless, access to API native methods from the Java Card System is controlled by TSF because there is no difference between native and interpreted methods in their interface or invocation mechanism.

FCS_CKM.1/DES Cryptographic key generation

FCS_CKM.1.1/DES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **TDES Key generation** and specified cryptographic key sizes **112 bits for TDES 2 keys, 168 bits for TDES 3 keys** that meet the following: **none (random numbers generation)**.

Application note:

The keys can be generated and diversified in accordance with [JCAPI22] specification in classes KeyBuilder.

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 for AES key** that meet the following: **none (random numbers generation)**.

Application note:

The keys can be generated and diversified in accordance with [JCAPI22] specification in classes KeyBuilder.

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 **[see application note]** and specified cryptographic key sizes **[1536 to 2048 bits]with CRT** that meet the following: **[see application note]**.

Application note:

The keys can be generated and diversified in accordance with [JCAPI22] specification in classes KeyBuilder and KeyPair (at least Session key generation).

FCS_CKM.2/DES Cryptographic key distribution

FCS_CKM.2.1/DES The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **(see application note)** that meets the following: **(see application note)**.

Application note:

Command SetKEY that meets [JCAPI22] Standards.

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 **(see application note)** that meets the following: **(see application note)**.

Application note:

Command SetKEY that meets [JCAPI22] Standards.

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 **(see application note)** that meets the following: **(see application note)**.

Application note:

Command SetKEY that meets [JCAPI22] Standards.

FCS_CKM.3/DES Cryptographic key access

FCS_CKM.3.1/DES The TSF shall perform **(see application note)** in accordance with a specified cryptographic key access method **(see application note)** that meets the following: **(see application note)**.

Application note:

The keys can be accessed in accordance with [JCAPI22] in class Key.

FCS_CKM.3/AES Cryptographic key access

FCS_CKM.3.1/AES The TSF shall perform **(see application note)** in accordance with a specified cryptographic key access method **(see application note)** that meets the following: **(see application note)**.

Application note:

The keys can be accessed in accordance with [JCAPI22] in class Key.

FCS_CKM.3/RSA Cryptographic key access

FCS_CKM.3.1/RSA The TSF shall perform **(see application note)** in accordance with a specified cryptographic key access method **(see application note)** that meets the following: **(see application note)**.

Application note:

The keys can be accessed in accordance with [JCAPI22] in class Key.

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **(see application note)** that meets the following: **(see application note)**.

Application note:

The keys are reset in accordance with [JCAPI22] 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/DES_CIPHER Cryptographic operation

FCS_COP.1.1/DES_CIPHER The TSF shall perform **[encryption and decryption of applet instance's data]** in accordance with a specified cryptographic algorithm **[Triple DES either in CBC or ECB mode and with padding schme (NOPAD,ISO9797 or PKCS#5)]** and cryptographic key sizes **[112 or 168 bits]** that meet the following: **[FIPS PUB 46-3, FIPS PUB 81, ISO 9797, according to JC API 22]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

FCS_COP.1/DES_MAC_COMP Cryptographic operation

FCS_COP.1.1/DES_MAC_COMP The TSF shall perform **[MAC generation or verification of applet instance's data]** in accordance with a specified cryptographic algorithm **[Triple DES in CBC mode and with or without padding generating MAC on 4-bytes or 8-bytes]** and cryptographic key sizes **[112 or 168 bits]** that meet the following: **[FIPS PUB 46-3, FIPS PUB 81, ISO 9797, according to JC API 22]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

FCS_COP.1/AES_CIPHER Cryptographic operation

FCS_COP.1.1/AES_CIPHER The TSF shall perform **[encryption and decryption of applet instance's data]** in accordance with a specified cryptographic algorithm **[AES (128 bits) either in CBC or ECB mode without padding]** and cryptographic key sizes **[128 bits]** that meet the following: **[FIPS PUB 197, FIPS PUB 81, ISO 9797, according to JC API 22]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

FCS_COP.1/AES_MAC_COMP Cryptographic operation

FCS_COP.1.1/AES_MAC_COMP The TSF shall perform **[MAC generation or verification of applet instance's data]** in accordance with a specified cryptographic algorithm **[AES (128bits) in CBC mode and with or without padding generating MAC on 4-bytes or 8-bytes]** and cryptographic key sizes **[112 or 168 bits]** that meet the following: **[FIPS PUB 197, FIPS PUB 81, ISO 9797, according to JC API 22]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

FCS_COP.1/RSA_SIGN Cryptographic operation

FCS_COP.1.1/RSA_SIGN The TSF shall perform **[signature generation or verification of applet instance's data]** in accordance with a specified cryptographic algorithm **[RSA with CRT in mode ISO 148888 with padding scheme (ISO9796 or PKCS #1)]** and cryptographic key sizes **[1536 to 2048 bits]** that meet the following: **[PKCS #1 Version 2.1]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

FCS_COP.1/RSA_CIPHER Cryptographic operation

FCS_COP.1.1/RSA_CIPHER The TSF shall perform **[encryption or decryption of applet instance's data]** in accordance with a specified cryptographic algorithm **[RSA with CRT]** and cryptographic key sizes **[1536 to 2048 bits]** that meet the following: **[PKCS #1 Version 2.1]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

FCS_COP.1/HMAC Cryptographic operation

FCS_COP.1.1/HMAC The TSF shall perform **[computation of a hash value for applet instance's data]** in accordance with a specified cryptographic algorithm **[HMAC, HMAC MD5, HMAC SHA-384 (48 bytes) or HMAC SHA-256 (32 bytes), HMAC SHA-224, HMAC SHA-1]** and cryptographic key sizes **[4-64 bytes]** that meet the following: **[rfc2104 & 2085]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

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 a transient object are described in [JCRE22], §5.1.

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. There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism (FDP_ROL.1.2/FIREWALL): the scope of the rollback does not extend outside the execution of the install() method, and the de-allocation occurs precisely right after the return of it.

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_ROL.1/FIREWALL Basic rollback

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

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

Application note:

FDP_ROL.1.2/FIREWALL 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 is either implemented in Java Card platform or relies on the transaction mechanism offered by the underlying platform. Some operations of the API are not conditionally updated, as documented in [JCAPI22] (see for instance, PIN-blocking, PIN-checking, update of Transient objects). It should be noticed that the rollback within the scope of the uninstall() method only applies to Java Card platform, version 2.2.1 compliant TOEs.

FDP_RIP.1/TRANSIENT Subset residual information protection

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

Application note:

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

The clearing of CLEAR_ON_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable applet instances, CLEAR_ON_DESELECT memory segments may be attached to 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.

Card Security Management

The following SFRs are related to the security requirements at the level of the whole card, in contrast to the previous ones that are somewhat restricted to the TOE alone. For instance, a potential security violation detected by the virtual machine may require a reaction that concerns more than the virtual machine, such as blocking the card (or a request for the appropriate security module with the power to block the card to perform the operation).

FAU_ARP.1 Security alarms

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

throw an exception,

or lock the card session

or reinitialize the Java Card System and its data

upon detection of a potential security violation.

Refinement:

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

FDP_SDI.2 Stored data integrity monitoring and action

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

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **shall increase a counter of integrity error event and mute the card if counter is greater than max value**.

Application note:

The following data persistently stored by TOE have a integrity check data security attribute:

* PIN (objects instance of class OwnerPin), * Key (i.e. objects instance of classes implemented the interface Key), * package.

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **any user** are unable to observe the operation **read, write, cryptographic operations** on **PIN, KEY** by **any other user or subject**.

Application note:

Although it is not required in [JCRE22] specifications, the non-observability of operations on sensitive information such as keys appears as impossible to circumvent in the smart card world.

FPT_FLS.1 Failure with preservation of secure state

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

Application note:

The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE22], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE222]). Behavior of the TOE on power loss and reset is described in [JCRE22], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE222], §3.6.1.

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

FPT_TDC.1.2 The TSF shall use

The rules defined in [JCVM22] specification;

The API tokens defined in the export files of reference implementation

when interpreting the TSF data from another trusted IT product.

Application note:

FPT_TDC.1.1:

Concerning the interpretation of data between the TOE and the underlying Java Card platform, it is assumed that the TOE is developed consistently with the SCP functions, namely concerning memory management, I/O functions, cryptographic functions, and so on.

AID Management

FIA_ATD.1/AID User attribute definition

FIA_ATD.1.1/AID The TSF shall maintain the following list of security attributes belonging to individual users:

Package AID,

Applet's version number,

Registered applet AID,

Applet Selection Status ([JCVM22], §6.5).

Refinement:

"Individual users" stand for applets.

FIA_UID.2/AID User identification before any action

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

Application note:

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

The role Java Card RE defined in FMT_SMR.1 is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself with respect to the TOE, but it is a part of it.

FIA_USB.1/AID User-subject binding

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

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **Package AID are defined with associated value during loading and with context identifier**.

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

Application note:

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

FMT_MTD.1/JCRE Management of TSF data

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

Application note:

The installer and the Java Card RE manage other TSF data such as the applet life cycle or CAP files, but this management is implementation specific. Objects in the Java programming language may also try to query AIDs of installed applets through the lookupAID(...) API method.

The installer, applet deletion manager or even the card manager may be granted the right to modify the list of registered applets' AIDs in specific implementations (possibly needed for installation and deletion; see #.DELETION and #.INSTALL).

FMT_MTD.3/JCRE Secure TSF data

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

6.1.1.2 InstG Security Functional Requirements

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

See also FIA_ATD.1, FIA_USB.1, FMT_MTD.1, FMT_SMR.1 for various information about applet installation.

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

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

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 TOE:

Package 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 dependent package is lesser than or equal to the major (minor) Version attribute associated to the resident package ([JCVM22], §4.5.2)..

Application note:

FDP_ITC.2.1/Installer:

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

FDP_ITC.2.3/Installer:

The format of the CAP file is precisely defined in [JCVM22] specifications; it contains the user data (like applet's code and data) and the security attributes altogether. Therefore there is no association to be carried out elsewhere.

FDP_ITC.2.4/Installer:

Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM22], §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM22], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatible. However, package files do have "package Version Numbers" ([JCVM22]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

FDP_ITC.2.5/Installer:

A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs.

The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM22], §4.4).

The installation (the invocation of an applet's install method by the installer) is implementation dependent ([JCRE22], §11.2).

Other rules governing the installation of an applet, that is, its registration to make it SELECTable by giving it a unique AID, are also implementation dependent (see, for example, [JCRE22], §11).

FMT_SMR.1/Installer Security roles

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

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

FPT_FLS.1/Installer Failure with preservation of secure state

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

Application note:

The TOE may provide additional feedback information to the card manager in case of potential security violations (see FAU_ARP.1).

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **a failure or service discontinuity** 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/Installer For **[detection of a potential loss of integrity during the transmission of an Executable Load File to the card, abortion of the installation**

process of an Executable Load File, or any fatal error occurred during the linking of an Executable Load File to the Executable Files already installed on the card], the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/Installer The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[the loss of the Executable Load File being installed]** for loss of TSF data or objects under the control of the TSF.

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

Application note:

FPT_RCV.3.1/Installer:

This element is not within the scope of the Java Card specification, which only mandates the behavior of the Java Card System in good working order. Further details on the "maintenance mode" shall be provided in specific implementations. The following is an excerpt from [CC-2], p298: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorised users should be allowed access to this mode but the real details of who can access this mode is a function of FMT: Security management. If FMT: Security management does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the SFRs.

FPT_RCV.3.2/Installer:

Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE22], §11.1.5 for possible scenarios. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a package/applet. See ([JCRE22], 11.3.4) for possible scenarios. Precise behavior is left to implementers.

Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP0035]) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT_FLS.1.1, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3/Installer:

The quantification is implementation dependent, but some facts can be recalled here. First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (EEPROM). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSF scope should be limited to the same restrictions of the transaction mechanism.

6.1.1.3 ADELG Security Functional Requirements

This group combines the SFRs related to the deletion of applets and/or packages, enforcing the applet deletion manager (ADEL) policy on security aspects outside the runtime. Deletion is a critical phase and therefore requires specific treatment. This policy is better thought as a frame to be filled by ST implementers.

Applet Deletion Manager Policy

FDP_ACC.2/ADEL Complete access control

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

Refinement:

The operations involved in the policy are:

OP.DELETE_APPLET,

OP.DELETE_PCKG,

OP.DELETE_PCKG_APPLET.

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

FDP_ACF.1/ADEL Security attribute based access control

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

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

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

In the context of this policy, an object O is reachable if and only one of the following conditions hold:

(1) the owner of O is a registered applet instance A (O is reachable from A),

(2) a static field of a resident package P contains a reference to O (O is reachable from P),

(3) there exists a valid remote reference to O (O is remote reachable),

(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): S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,

(1) S.ADEL is currently selected,

(2) there is no instance in the context of O.APPLET that is active in any logical channel and

(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.15 ([JCRE22], §11.3.4.1, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,

(1) S.ADEL is currently selected,

(2) there is no instance of any of the O.APPLET being deleted that is active in any logical channel and

(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.16 ([JCRE22], §11.3.4.2, Applet/Library Package Deletion): S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PKG only if,

(1) S.ADEL is currently selected,

(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG that is an instance of a class that belongs to O.CODE_PKG, exists on the card and

(3) there is no resident package on the card that depends on O.CODE_PKG.

R.JAVA.17 ([JCRE22], §11.3.4.3, Applet Package and Contained Instances Deletion): S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PKG only if,

(1) S.ADEL is currently selected,

(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG, which is an instance of a class that belongs to O.CODE_PKG exists on the card,

(3) there is no package loaded on the card that depends on O.CODE_PKG, and

(4) for every O.APPLET of those being deleted it holds that: (i) there is no instance in the context of O.APPLET that is active in any logical channel and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([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 [Editorially Refined] The TSF shall explicitly deny access of **any subject but S.ADEL to O.CODE_PKG or O.APPLET for the purpose of deleting them from the card**.

Application note:

FDP_ACF.1.2/ADEL:

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

S.ADEL calls the "uninstall" method of the applet instance to be deleted, if implemented by the applet, to inform it of the deletion request. The order in which these calls and the dependencies checks are performed are out of the scope of this protection profile.

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:

Deleted freed resources (both code and data) may be reused, depending on the way they were deleted (logically or physically). 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.

FMT_MSA.1/ADEL Management of security attributes

FMT_MSA.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to restrict the ability to **modify** the security attributes **Registered Applets and Resident Packages** to **the Java Card RE**.

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 role(s): none,** to specify alternative initial values to override the default values when an object or information is created.

Application note:

No role may change the restrictive initial value set for these security attributes.

FMT_SMF.1/ADEL Specification of Management Functions

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

Application note:

The modification of the Active Applets security attribute should be performed in accordance with the rules given in [JCRE22], §4.

FMT_SMR.1/ADEL Security roles

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

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

FPT_FLS.1/ADEL Failure with preservation of secure state

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

Application note:

The TOE may provide additional feedback information to the card manager in case of a potential security violation (see FAU_ARP.1).

The Package/applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with Java Card specification ([JCRE22], §11.3.4.)

6.1.1.4 RMIG Security Functional Requirements

This group is mainly devoted to specifying the policies that control access to remote objects and the flow of information that takes place when the RMI service is used. There are specific control rules concerning the access to remote objects. The rules relate mainly to the lifetime of their corresponding remote references. Information concerning remote object references can be sent out of the card only if the corresponding remote object has been designated as exportable. Array parameters of remote method invocations must be allocated on the card as global arrays. Therefore, the storage of references to those arrays must be restricted as well.

JCRMI Policy

The JCRMI policy embodies both an access control and an information flow control policy.

FDP_ACC.2/JCRMI Complete access control

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

Refinement:

The operations involved in this policy are:

OP.GET_ROR,

OP.INVOKE.

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

FDP_ACF.1/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:

Subject/Object	**Attributes**
S.JCRE	**Selected Applet Context**
O.REMOTE_OBJ	**Owner, Class, Identifier, ExportedInfo**
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: S.CAD may perform OP.GET_ROR upon 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: 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 it belongs to the Returned References of O.RMI_SERVICE, and if the Identifier of O.REMOTE_MTHD matches one of the remote methods in the Class of the O.REMOTE_OBJ 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 [Editorially Refined] The TSF shall explicitly deny access of **any subject but S.JCRE to O.REMOTE_OBJ and O.REMOTE_MTHD for the purpose of performing a remote method invocation**.

Application note:

FDP_ACF.1.2/JCRMI:

The validity of a remote object reference is specified as a lifetime characterization. The security attributes involved in the rules for determining valid remote object references are the Returned References of the O.RMI_SERVICE and the Active Applets (see

FMT_REV.1.1/JCRMI and FMT_REV.1.2/JCRMI). The precise mechanism by which a remote method is invoked on a remote object is defined in detail in ([JCRE22], §8.5.2 and [JCAPI22]).

Note that the owner of an O.RMI_SERVICE is the applet instance that created the object. The attribute Returned References lists the remote object references that have been sent to the S.CAD during the applet selection session. This attribute is implementation dependent.

FDP_IFC.1/JCRMI Subset information flow control

FDP_IFC.1.1/JCRMI The TSF shall enforce the **JCRMI information flow control SFP** on **S.JCRE, S.CAD, I.RORD and OP.RET_RORD(S.JCRE,S.CAD,I.RORD)**.

Application note:

FDP_IFC.1.1/JCRMI:

Array parameters of remote method invocations must be allocated on the card as global arrays objects. References to global arrays cannot be stored in class variables, instance variables or array components. The control of the flow of that kind of information has already been specified in FDP_IFC.1.1/JCVM.

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

FDP_IFF.1/JCRMI Simple security attributes

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

Subjects/Information	**Security attributes**
I.RORD	**ExportedInfo**

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:

OP.RET_RORD(S.JCRE, S.CAD, I.RORD) is permitted only if the attribute ExportedInfo of I.RORD has the value "true" ([JCRE22], §8.5).

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

FDP_IFF.1.4/JCRMI The TSF shall explicitly authorise an information flow based on the following rules: **[No additional rules]**.

FDP_IFF.1.5/JCRMI The TSF shall explicitly deny an information flow based on the following rules: **[No additional rules]**.

Application note:

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

FMT_MSA.1/EXPORT Management of security attributes

FMT_MSA.1.1/EXPORT The TSF shall enforce the **JCRMI access control SFP** to restrict the ability to **modify** the security attributes**: ExportedInfo of O.REMOTE_OBJ** to **its owner applet**.

Application note:

The Exported status of a remote object can be modified by invoking its methods export() and unexport(), and only the owner of the object may perform the invocation without raising a SecurityException (javacard.framework.service.CardRemoteObject). However, even if the owner of the object may provoke the change of the security attribute value, the modification itself can be performed by the Java Card RE.

FMT_MSA.1/REM_REFS Management of security attributes

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

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 **following role(s): none,** to specify alternative initial values to override the default values when an object or information is created.

Application note:

FMT_MSA.3.1/JCRMI:

Remote objects' security attributes are created and initialized at the creation of the object, and except for the ExportedInfo attribute, the values of the attributes are not longer modifiable. The default value of the Exported attribute is true. There is one default value for the Selected Applet Context that is the default applet identifier's context, and one default value for the active context, that is "Java Card RE".

FMT_MSA.3.2/JCRMI:

The intent is to have none of the identified roles to have privileges with regards to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP.

FMT_REV.1/JCRMI Revocation

FMT_REV.1.1/JCRMI [Editorially Refined] The TSF shall restrict the ability to revoke **the Returned References of O.RMI_SERVICE** to **the Java Card RE**.

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

Application note:

The rules are described in [JCRE22], §8.5

FMT_SMF.1/JCRMI Specification of Management Functions

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

modify the security attribute ExportedInfo of O.REMOTE_OBJ,

modify the security attribute Returned References of O.RMI_SERVICE.

FMT_SMR.1/JCRMI Security roles

FMT_SMR.1.1/JCRMI The TSF shall maintain the roles**: applet**.

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

Application note:

Applets own remote interface objects and may choose to allow or forbid their exportation, which is managed through a security attribute.

6.1.1.5 ODELG Security Functional Requirements

The following requirements are concerned with the secure deletion of information provoked by the object deletion mechanism. This mechanism is triggered by the applet that owns the deleted objects by invoking a specific API method.

FPT_FLS.1/ODEL Failure with preservation of secure state

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

Application note:

The TOE may provide additional feedback information to the card manager in case of potential security violation (see FAU_ARP.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].

There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism: the execution of requestObjectDeletion() is not in the scope of the rollback because it must be performed in between APDU command processing, and therefore no transaction can be in progress.

6.1.1.6 CarG Security Functional Requirements

This group of requirements applies to those TOEs where the bytecode verifier is not part of them (BCV not embedded on the card). If this is the case, the TOE shall include requirements for preventing the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.

FCO_NRO.2/CM Enforced proof of origin

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

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

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **as assumption the key used is kept integer and confidential by origin**.

Application note:

FCO_NRO.2.1/CM:

Upon reception of a new application package for installation, the card manager shall first check that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification.

FCO_NRO.2.3/CM:

The exact limitations on the evidence of origin are implementation dependent. In most of the implementations, the card manager performs an immediate verification of the origin of the package using an electronic signature mechanism, and no evidence is kept on the card for future verifications.

FDP_IFC.2/CM Complete information flow control

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

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

Application note:

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

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

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

FDP_IFF.1/CM Simple security attributes

FDP_IFF.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** based on the following types of subject and information security attributes: **[the Command Security Level defined for the messages that the card receives through the secure channel]**.

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: **[assignment: the rules describing the communication protocol (as defined in GP22 standard §10) used by the CAD and the card for transmitting a new package]**.

FDP_IFF.1.3/CM The TSF shall enforce the **[possible security levels are: NO-SEC (clear text), C-AUTHENTICATED (authentication of the command's emitter), C-MAC (authentication of the emitter and integrity of the command), C-DEC (authentication of the emitter, integrity and confidentiality of the command)]**.

FDP_IFF.1.4/CM The TSF shall explicitly authorise an information flow based on the following rules: **[the SD may process:**

an (INITIALIZE-UPDATE) operation only if the key set specified in the command exist,

an (EXTERNAL-AUTHENTICATE) operation if the following conditions are fulfilled: 1) The cryptogram received from the off-card subject is equal to the cryptogram computed by the Security Domain. 2) The MAC attached to the message has been generated using the CMAC session key and the current value of the ICV.

a (GET-DATA) operation if the following condition are fulfilled: 1) If the command security level is at least C-MAC, 2) the MAC attached to the message has been generated from the command using the C-MAC session key and the current value of the ICV.

any received operation for any other command if the following conditions hold: 1) The current security level is at least AUTHENTICATED. 2) If the command security level is at least C-MAC, the MAC attached to the message has been generated from the clear-text command using the C-MAC session key and the current value of the ICV.

FDP_IFF.1.5/CM The TSF shall explicitly deny an information flow based on the following rules: **[A Security Domain may always process a (SELECT) operation or a (Get DATA) operation at the security level NO-SEC]**.

Application note:

FDP_IFF.1.1/CM:

The security attributes used to enforce the PACKAGE LOADING SFP are implementation dependent. More precisely, they depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that can be used are: (1) the keys used by the subjects to encrypt/decrypt their messages; (2) the number of pieces the application package has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the package, etc. See for example Appendix D of [GP].

FDP_IFF.1.2/CM:

The precise set of rules to be enforced by the function is implementation dependent. The whole exchange of messages shall verify at least the following two rules: (1) the subject S.INSTALLER shall accept a message only if it comes from the subject S.CAD; (2) the subject S.INSTALLER shall accept an application package only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD.

FDP_UIT.1/CM Data exchange integrity

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

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

Application note:

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

FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM The TSF shall allow **selection of a security domain and execution of Card Manager** on behalf of the user to be performed before the user is identified.

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

Application note:

Conditions for package installation are at least: Security Domain is selected, SD life cycle is consistent, Key set is identified, Keys in key set are integer, secure channel is opened.

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 **[Card Life cycle, Security Level]** to **[Card Manager]**.

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_SMF.1/CM Specification of Management Functions

FMT_SMF.1.1/CM The TSF shall be capable of performing the following management functions: **[modification of the Card life cycle inducing availability of management functions]**.

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles **[S.CAD, S.CARDMANAGER]**.

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

FTP_ITC.1.2/CM [Editorially Refined] The TSF shall permit **the CAD placed in the card issuer secured environment** to initiate communication via the trusted channel.

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

Application note:

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

6.1.2 SCP

This section states the security functional requirements for the Smart Card Platform.

6.1.2.1 Operating System

This section presents those requirements of the Smart Card Platform group that concern the Operating System. Due to enlargement in the scope of evaluation, the requirements related to OS are now assigned to the TOE and no more to the environment. Other internal security mechanisms are not addressed by SFR but ADV_ARC activities.

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 function either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

FPT_RCV.3/SCP Automated recovery without undue loss

FPT_RCV.3.1/SCP When automated recovery from **security policy violation** 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/SCP For **execution access to a memory zone reserved for TSF data, writing access to a memory zone reserved for TSF's code, and any**

segmentation fault performed by a Java Card applet, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/SCP The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **o the contents of Java Card static fields, instance fields, and array positions that fall under the scope of an open transaction; o the Java Card objects that were allocated into the scope of an open transaction; o the contents of Java Card transient objects; o any possible Executable Load File being loaded when the failure occurred** for loss of TSF data or objects under the control of the TSF.

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

6.1.2.2 Integrated Circuit

The section should contain the requirements of the Smart Card Platform group introduced in [JCSPP] concerning the Integrated Circuit. Due to enlargement in the scope of evaluation, the requirements related to IC are now assigned to the TOE and no more to the environment.

Those requirements are fulfilled in the [ICST] and are covered by the IC certificate reused in the composite evaluation process. There are not repeated here.

They mainly concern protecting the smart card's chip against physical tampering, preventing the disclosure of information when it is transferred from different physical parts of the chip, providing the basic DES operation, and keeping a secure state when a malfunction is detected and providing an independent security domain for the hardware.

6.1.3 (U)SIM

6.1.3.1 Card Content Management

FCS_COP.1/DAP Cryptographic operation

FCS_COP.1.1/DAP The TSF shall perform **verification of the DAP signature attached to Executable Load Applications** in accordance with a specified cryptographic algorithm

DES Scheme: Single DES plus final Triple DES MAC (Retail MAC)

and cryptographic key sizes

DES Scheme: DES key of mimimum length 16 bytes

that meet the following:

Sections C.1.2 and C.6 of [GP]

DES Scheme: ISO 9797-1 as MAC Algorithm 3 with output transformation 3, without truncation, and with DES taking the place of the block cipher.

6.1.3.2 Crypto JCAPI

FCS_COP.1/SHA2 Cryptographic operation

FCS_COP.1.1/SHA2 The TSF shall perform **[computation of a hash value for applet instance's data]** in accordance with a specified cryptographic algorithm **[SHA-384 (48 bytes) or SHA-256 (32 bytes)]** and cryptographic key sizes **[None]** that meet the following: **[FIPS 180-2]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI222] (see javacardx.crypto.Cipher and javacardx.security packages).

This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI222]).

FCS_COP.1/CRC Cryptographic operation

FCS_COP.1.1/CRC The TSF shall perform **[Computation of checksum CRC16 or CRC32 of applet instance's data]** in accordance with a specified cryptographic algorithm **[CRC16 or CRC32]** and cryptographic key sizes **[none]** that meet the following: **[ISO3309]**.

Application note:

The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).

This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

FCS_RND.1 Random Number Generation

FCS_RND.1.1 The TSF shall provide a mechanism to generate random numbers that meet **the STANDARD level specified in [DCSSI2741]**.

6.1.3.3 SecureAPI

FPT_FLS.1/SecureAPI Failure with preservation of secure state

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

FPT_ITT.1/SecureAPI Basic internal TSF data transfer protection

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

FPR_UNO.1/SecureAPI Unobservability

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

6.2 Security Assurance Requirements

The security assurance requirement level is EAL4 augmented with AVA_VAN.5 and ALC_DVS.2.

6.3 Security Requirements Rationale

6.3.1 Objectives

6.3.1.1 Security Objectives for the TOE

<u>**Java Card System Protection Profile - Open Configuration**</u>

IDENTIFICATION

O.SID Subjects' identity is AID-based (applets, packages), and is met by the following SFRs: FDP_ITC.2/Installer, FIA_ATD.1/AID, (FMT_MSA.1/JCRE, FMT_MSA.1/JCVM or FMT_MSA.1/JCVM_LC), FMT_MSA.1/REM_REFS, FMT_MSA.1/JCRMI, FMT_MSA.1/EXPORT, FMT_MSA.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/JCRMI, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/JCRE, FMT_MSA.3/CM, FMT_SMF.1/CM, FMT_SMR.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, FMT_MTD.1/JCRE and FMT_MTD.3/JCRE.

Lastly, installation procedures ensure protection against forgery (the AID of an applet is under the control of the TSFs) or re-use of identities (FIA_UID.2/AID, FIA_USB.1/AID).

EXECUTION

O.FIREWALL This objective is met by the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), the JCRMI access control policy (FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI) and the functional requirement FDP_ITC.2/Installer. The functional requirements of the class FMT (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1, FMT_SMR.1/ADEL, FMT_SMR.1/JCRMI, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, FMT_SMF.1/CM, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_REV.1/JCRMI) also indirectly contribute to meet this objective.

O.GLOBAL_ARRAYS_CONFID Only arrays can be designated as global, and the only global arrays required in the Java Card API are the APDU buffer and the global byte array input parameter (bArray) to an applet's install method. The clearing requirement of these arrays is met by (FDP_RIP.1/APDU and FDP_RIP.1/bArray respectively). The JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from keeping a pointer to a shared buffer, which could be used to read its contents when the buffer is being used by another application.

Protection of the array parameters of remotely invoked methods, which are global as well, is covered by the general initialization of method parameters (FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT).

O.GLOBAL_ARRAYS_INTEG This objective is met by the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), which prevents an application from keeping a pointer to the APDU buffer of the card or to the global byte array of the applet's install method. Such a pointer could be used to access and modify it when the buffer is being used by another application.

O.NATIVE This security objective is covered by FDP_ACF.1/FIREWALL: the only means to execute native code is the invocation of a Java Card API method. This objective mainly relies on the environmental objective OE.APPLET, which uphold the assumption A.APPLET.

O.OPERATE The TOE is protected in various ways against applets' actions (FPT_TDC.1), the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FAU_ARP.1). Its security-critical parts and procedures are also protected: safe recovery from failure is ensured (FPT_RCV.3/Installer), applets' installation may be cleanly aborted (FDP_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FDP_ITC.2/Installer, FIA_ATD.1/AID, FIA_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).

Almost every objective and/or functional requirement indirectly contributes to this one too.

Application note: Startup of the TOE (TSF-testing) can be covered by FPT_TST.1. This SFR component is not mandatory in [JCRE22], but appears in most of security requirements documents for masked applications. Testing could also occur randomly. Self-tests may become mandatory in order to comply to FIPS certification [FIPS 140-2].

O.REALLOCATION This security objective is satisfied by the following SFRs: FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ADEL, which imposes that the contents of the re-allocated block shall always be cleared before delivering the block.

O.RESOURCES The TSFs detects stack/memory overflows during execution of applications (FAU_ARP.1, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer). Failed installations are not to create memory leaks (FDP_ROL.1/FIREWALL, FPT_RCV.3/Installer) as well. Memory management is controlled by the TSF (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1 FMT_SMR.1/ADEL, FMT_SMR.1/JCRMI, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, FMT_SMF.1/CM and FMT_SMR.1/CM).

SERVICES

O.ALARM This security objective is met by FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL which guarantee that a secure state is preserved by the TSF when failures occur, and FAU_ARP.1 which defines TSF reaction upon detection of a potential security violation.

O.CIPHER This security objective is directly covered by FCS_CKM.1/RSA,FCS_CKM.1/DES, FCS_CKM.1/AES, FCS_CKM.2/DES, FCS_CKM.2/AES, FCS_CKM.2/RSA, FCS_CKM.3/AES, FCS_CKM.3/RSA, FCS_CKM.3/DES, FCS_CKM.4, FCS_COP.1/DES_CIPHER, FCS_COP.1/AES_CIPHER, FCS_COP.1/RSA_CIPHER. The SFR FPR_UNO.1 contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys.

O.KEY-MNGT This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1 and FDP_SDI.2 as well. Precisely it is met by the following components: FCS_CKM.1/RSA,FCS_CKM.1/DES, FCS_CKM.1/AES, FCS_CKM.2/DES, FCS_CKM.2/AES, FCS_CKM.2/RSA, FCS_CKM.3/AES, FCS_CKM.3/RSA, FCS_CKM.3/DES, FCS_CKM.4, FCS_COP.1/DES_MAC_COMP, FCS_COP.1/AES_MAC_COMP, FCS_COP.1/RSA_SIGN, FCS_COP.1/HMAC, FCS_COP.1/DES_CIPHER, FCS_COP.1/RSA_CIPHER, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT.

O.PIN-MNGT This security objective is ensured by FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_ROL.1/FIREWALL and FDP_SDI.2 security functional requirements. The TSFs behind these are implemented by API classes. The firewall security functions FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL shall protect the access to private and internal data of the objects.

O.REMOTE The access to the TOE's internal data and the flow of information from the card to the CAD required by the JCRMI service is under control of the JCRMI access control policy (FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI) and the JCRMI information flow control policy (FDP_IFC.1/JCRMI, FDP_IFF.1/JCRMI). The security functional requirements of the class FMT (FMT_MSA.1/JCRMI, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_REV.1/JCRMI and FMT_SMR.1/JCRMI, FMT_SMF.1/JCRMI) included in the group RMIG also contribute to meet this objective.

O.TRANSACTION Directly met by FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT and FDP_RIP.1/OBJECTS (more precisely, by the element FDP_RIP.1.1/ABORT).

OBJECT DELETION

O.OBJ-DELETION This security objective specifies that deletion of objects is secure. The security objective is met by the security functional requirements FDP_RIP.1/ODEL and FPT_FLS.1/ODEL.

APPLET MANAGEMENT

O.DELETION This security objective specifies that applet and package deletion must be secure. The non-introduction of security holes is ensured by the ADEL access control policy (FDP_ACC.2/ADEL, FDP_ACF.1/ADEL). The integrity and confidentiality of data that does not belong to the deleted applet or package is a by-product of this policy as well. Non-accessibility of deleted data is met by FDP_RIP.1/ADEL and the TSFs are protected against possible failures of the deletion procedures (FPT_FLS.1/ADEL, FPT_RCV.3/Installer). The security functional requirements of the class FMT (FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL) included in the group ADELG also contribute to meet this objective.

O.LOAD This security objective specifies that the loading of a package into the card must be secure. Evidence of the origin of the package is enforced (FCO_NRO.2/CM) and the integrity of the corresponding data is under the control of the PACKAGE LOADING information flow policy (FDP_IFC.2/CM, FDP_IFF.1/CM) and FDP_UIT.1/CM. Appropriate identification (FIA_UID.1/CM) and transmission mechanisms are also enforced (FTP_ITC.1/CM).

O.INSTALL This security objective specifies that installation of applets must be secure. Security attributes of installed data are under the control of the FIREWALL access control policy (FDP_ITC.2/Installer), and the TSFs are protected against possible failures of the installer (FPT_FLS.1/Installer, FPT_RCV.3/Installer).

SCP

O.SCP.SUPPORT The SCP is a part of the TOE supporting TSFs of the upper layer of the TOE, especially for recovery operations as FPT_RCV.4/SCP.

O.SCP.RECOVERY The SCP is a part of the TOE supporting TSFs of the upper layer of the TOE, especially for recovery operations as FPT_RCV.3/SCP.

O.SCP.IC The SCP.IC is a part of the TOE supporting TSFs of the upper layer of the TOE and more specially FPT_FLS.1.

(U)SIM

O.RND The security objective O.RND is met by the following SFR FCS_RND.1.

O.APPLI-AUTH The security objective O.APPLI-AUTH is met by the following SFRs:

FCS_COP.1/DAP compute DAP to be compared with input and ensures that the loaded Executable Application is legitimate by specifying the algorithm to be used in order to verify the DAP signature of the Verification Authority.

O.JCAPI-Services The security objective is met by the following SFR FCS_COP.1/SHA2 and FCS_COP.1/CRC.

O.Secure_API The security objective is met by the following SFR FPT_FLS.1/SecureAPI, FPT_ITT.1/SecureAPI and FPR_UNO.1/SecureAPI.

6.3.2 Rationale tables of Security Objectives and SFRs

Security Objectives	Security Functional Requirements	Rationale
O.SID	FIA_ATD.1/AID, FIA_UID.2/AID, FMT_MSA.1/JCRE, FMT_MSA.3/JCRMI, FMT_MSA.1/REM_REFS, FMT_MSA.1/EXPORT, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.3/JCVM, FMT_MSA.1/CM, FMT_MSA.3/CM, FDP_ITC.2/Installer, FMT_MSA.1/JCVM, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FIA_USB.1/AID, FMT_SMF.1/ADEL, FMT_SMR.1/CM, FMT_SMF.1/CM, FMT_SMF.1/JCRMI, FMT_MSA.3/FIREWALL	Section 4.3.1
O.FIREWALL	FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FMT_MSA.1/JCVM, FMT_MSA.1/JCRE, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/JCVM, FMT_SMF.1, FMT_SMR.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FDP_ITC.2/Installer, FMT_SMR.1/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_REV.1/JCRMI, FMT_SMF.1/JCRMI, FMT_SMR.1/JCRMI, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_SMF.1/CM, FMT_MSA.3/FIREWALL	Section 4.3.1
O.GLOBAL_ARRAYS_CONFID	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/TRANSIENT	Section 4.3.1
O.GLOBAL_ARRAYS_INTEG	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM	Section 4.3.1
O.NATIVE	FDP_ACF.1/FIREWALL	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.OPERATE	FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FDP_ROL.1/FIREWALL, FAU_ARP.1, FPT_TDC.1, FPT_FLS.1, FIA_ATD.1/AID, FIA_USB.1/AID, FDP_ITC.2/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL	Section 4.3.1
O.REALLOCATION	FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/TRANSIENT	Section 4.3.1
O.RESOURCES	FMT_SMF.1, FMT_SMR.1, FDP_ROL.1/FIREWALL, FAU_ARP.1, FPT_FLS.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FPT_FLS.1/ADEL, FMT_SMF.1/JCRMI, FMT_SMR.1/JCRMI, FPT_FLS.1/ODEL, FMT_SMF.1/CM, FMT_SMR.1/CM	Section 4.3.1
O.ALARM	FAU_ARP.1, FPT_FLS.1, FPT_FLS.1/Installer, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL	Section 4.3.1
O.CIPHER	FCS_CKM.1/RSA, FCS_CKM.2/DES, FCS_CKM.3/DES, FCS_CKM.4, FCS_COP.1/DES_CIPHER, FCS_COP.1/AES_CIPHER, FCS_COP.1/RSA_CIPHER, FPR_UNO.1, FCS_CKM.2/AES, FCS_CKM.2/RSA, FCS_CKM.3/AES, FCS_CKM.3/RSA, FCS_CKM.1/DES, FCS_CKM.1/AES	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.KEY-MNGT	FDP_RIP.1/OBJECTS, FCS_CKM.1/RSA, FCS_CKM.2/DES, FCS_CKM.3/DES, FCS_CKM.4, FCS_COP.1/DES_CIPHER, FCS_COP.1/RSA_CIPHER, FCS_COP.1/DES_MAC_COMP, FCS_COP.1/AES_MAC_COMP, FCS_COP.1/RSA_SIGN, FCS_COP.1/HMAC, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_SDI.2, FCS_CKM.2/AES, FCS_CKM.2/RSA, FCS_CKM.3/AES, FCS_CKM.3/RSA, FCS_CKM.1/DES, FCS_CKM.1/AES, FDP_RIP.1/TRANSIENT	Section 4.3.1
O.PIN-MNGT	FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_ROL.1/FIREWALL, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_SDI.2, FDP_RIP.1/TRANSIENT, FDP_RIP.1/KEYS	Section 4.3.1
O.REMOTE	FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI, FDP_IFC.1/JCRMI, FDP_IFF.1/JCRMI, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_REV.1/JCRMI, FMT_SMF.1/JCRMI, FMT_SMR.1/JCRMI	Section 4.3.1
O.TRANSACTION	FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_ROL.1/FIREWALL, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/TRANSIENT	Section 4.3.1
O.OBJ-DELETION	FDP_RIP.1/ODEL, FPT_FLS.1/ODEL	Section 4.3.1
O.DELETION	FPT_RCV.3/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL	Section 4.3.1
O.LOAD	FCO_NRO.2/CM, FIA_UID.1/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FTP_ITC.1/CM	Section 4.3.1
O.INSTALL	FDP_ITC.2/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.SCP.SUPPORT	FPT_RCV.4/SCP	Section 4.3.1
O.SCP.RECOVERY	FPT_RCV.3/SCP	Section 4.3.1
O.SCP.IC	FPT_FLS.1	Section 4.3.1
O.RND	FCS_RND.1	Section 4.3.1
O.APPLI-AUTH	FCS_COP.1/DAP	Section 4.3.1
O.JCAPI-Services	FCS_COP.1/SHA2, FCS_COP.1/CRC	Section 4.3.1
O.Secure_API	FPT_FLS.1/SecureAPI, FPT_ITT.1/SecureAPI, FPR_UNO.1/SecureAPI	Section 4.3.1

Table 13 Security Objectives and SFRs - Coverage

Security Functional Requirements	Security Objectives
FDP_ACC.2/FIREWALL	O.FIREWALL, O.OPERATE, O.PIN-MNGT
FDP_ACF.1/FIREWALL	O.FIREWALL, O.NATIVE, O.OPERATE, O.PIN-MNGT
FDP_IFC.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG
FDP_IFF.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG
FDP_RIP.1/OBJECTS	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION
FMT_MSA.1/JCRE	O.SID, O.FIREWALL
FMT_MSA.1/JCVM	O.SID, O.FIREWALL
FMT_MSA.2/FIREWALL_JCVM	O.FIREWALL
FMT_MSA.3/FIREWALL	O.SID, O.FIREWALL
FMT_MSA.3/JCVM	O.SID, O.FIREWALL
FMT_SMF.1	O.FIREWALL, O.RESOURCES
FMT_SMR.1	O.FIREWALL, O.RESOURCES
FCS_CKM.1/DES	O.CIPHER, O.KEY-MNGT
FCS_CKM.1/AES	O.CIPHER, O.KEY-MNGT
FCS_CKM.1/RSA	O.CIPHER, O.KEY-MNGT
FCS_CKM.2/DES	O.CIPHER, O.KEY-MNGT
FCS_CKM.2/AES	O.CIPHER, O.KEY-MNGT
FCS_CKM.2/RSA	O.CIPHER, O.KEY-MNGT
FCS_CKM.3/DES	O.CIPHER, O.KEY-MNGT
FCS_CKM.3/AES	O.CIPHER, O.KEY-MNGT
FCS_CKM.3/RSA	O.CIPHER, O.KEY-MNGT
FCS_CKM.4	O.CIPHER, O.KEY-MNGT
FCS_COP.1/DES_CIPHER	O.CIPHER, O.KEY-MNGT
FCS_COP.1/DES_MAC_COMP	O.KEY-MNGT
FCS_COP.1/AES_CIPHER	O.CIPHER
FCS_COP.1/AES_MAC_COMP	O.KEY-MNGT
FCS_COP.1/RSA_SIGN	O.KEY-MNGT
FCS_COP.1/RSA_CIPHER	O.CIPHER, O.KEY-MNGT

Security Functional Requirements	Security Objectives
FCS_COP.1/HMAC	O.KEY-MNGT
FDP_RIP.1/ABORT	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION
FDP_RIP.1/APDU	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION
FDP_RIP.1/bArray	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION
FDP_RIP.1/KEYS	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION
FDP_ROL.1/FIREWALL	O.OPERATE, O.RESOURCES, O.PIN-MNGT, O.TRANSACTION
FDP_RIP.1/TRANSIENT	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION
FAU_ARP.1	O.OPERATE, O.RESOURCES, O.ALARM
FDP_SDI.2	O.KEY-MNGT, O.PIN-MNGT
FPR_UNO.1	O.CIPHER, O.KEY-MNGT, O.PIN-MNGT
FPT_FLS.1	O.OPERATE, O.RESOURCES, O.ALARM, O.SCP.IC
FPT_TDC.1	O.OPERATE
FIA_ATD.1/AID	O.SID, O.OPERATE
FIA_UID.2/AID	O.SID
FIA_USB.1/AID	O.SID, O.OPERATE
FMT_MTD.1/JCRE	O.SID, O.FIREWALL, O.RESOURCES
FMT_MTD.3/JCRE	O.SID, O.FIREWALL, O.RESOURCES
FDP_ITC.2/Installer	O.SID, O.FIREWALL, O.OPERATE, O.INSTALL
FMT_SMR.1/Installer	O.FIREWALL, O.RESOURCES
FPT_FLS.1/Installer	O.OPERATE, O.RESOURCES, O.ALARM, O.INSTALL
FPT_RCV.3/Installer	O.OPERATE, O.RESOURCES, O.DELETION, O.INSTALL

Security Functional Requirements	Security Objectives
FDP_ACC.2/ADEL	O.DELETION
FDP_ACF.1/ADEL	O.DELETION
FDP_RIP.1/ADEL	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.DELETION
FMT_MSA.1/ADEL	O.SID, O.FIREWALL, O.DELETION
FMT_MSA.3/ADEL	O.SID, O.FIREWALL, O.DELETION
FMT_SMF.1/ADEL	O.SID, O.FIREWALL, O.RESOURCES
FMT_SMR.1/ADEL	O.FIREWALL, O.RESOURCES, O.DELETION
FPT_FLS.1/ADEL	O.OPERATE, O.RESOURCES, O.ALARM, O.DELETION
FDP_ACC.2/JCRMI	O.FIREWALL, O.REMOTE
FDP_ACF.1/JCRMI	O.FIREWALL, O.REMOTE
FDP_IFC.1/JCRMI	O.REMOTE
FDP_IFF.1/JCRMI	O.REMOTE
FMT_MSA.1/EXPORT	O.SID, O.FIREWALL, O.REMOTE
FMT_MSA.1/REM_REFS	O.SID, O.FIREWALL, O.REMOTE
FMT_MSA.3/JCRMI	O.SID, O.FIREWALL, O.REMOTE
FMT_REV.1/JCRMI	O.FIREWALL, O.REMOTE
FMT_SMF.1/JCRMI	O.SID, O.FIREWALL, O.RESOURCES, O.REMOTE
FMT_SMR.1/JCRMI	O.FIREWALL, O.RESOURCES, O.REMOTE
FPT_FLS.1/ODEL	O.OPERATE, O.RESOURCES, O.ALARM, O.OBJ-DELETION
FDP_RIP.1/ODEL	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.OBJ-DELETION
FCO_NRO.2/CM	O.LOAD
FDP_IFC.2/CM	O.LOAD
FDP_IFF.1/CM	O.LOAD
FDP_UIT.1/CM	O.LOAD
FIA_UID.1/CM	O.LOAD
FMT_MSA.1/CM	O.SID, O.FIREWALL

Security Functional Requirements	Security Objectives
FMT_MSA.3/CM	O.SID, O.FIREWALL
FMT_SMF.1/CM	O.SID, O.FIREWALL, O.RESOURCES
FMT_SMR.1/CM	O.SID, O.FIREWALL, O.RESOURCES
FTP_ITC.1/CM	O.LOAD
FPT_RCV.4/SCP	O.SCP.SUPPORT
FPT_RCV.3/SCP	O.SCP.RECOVERY
FCS_COP.1/DAP	O.APPLI-AUTH
FCS_COP.1/SHA2	O.JCAPI-Services
FCS_COP.1/CRC	O.JCAPI-Services
FCS_RND.1	O.RND
FPT_FLS.1/SecureAPI	O.Secure_API
FPT_ITT.1/SecureAPI	O.Secure_API
FPR_UNO.1/SecureAPI	O.Secure_API

Table 14 SFRs and Security Objectives

6.3.3 Dependencies

6.3.3.1 SFRs dependencies

Requirements	CC Dependencies	Satisfied Dependencies
FDP_ITC.2/Installer	(FDP_ACC.1 or FDP_IFC.1) and (FPT_TDC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FMT_SMR.1/Installer	(FIA_UID.1)	
FPT_FLS.1/Installer	No dependencies	
FPT_RCV.3/Installer	(AGD_OPE.1)	AGD_OPE.1
FPT_FLS.1/ODEL	No dependencies	
FDP_RIP.1/ODEL	No dependencies	
FCO_NRO.2/CM	(FIA_UID.1)	FIA_UID.1/CM
FDP_IFC.2/CM	(FDP_IFF.1)	FDP_IFF.1/CM
FDP_IFF.1/CM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/CM, FMT_MSA.3/CM
FDP_UIT.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM
FIA_UID.1/CM	No dependencies	
FMT_MSA.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_IFC.2/CM, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_SMF.1/CM	No dependencies	
FMT_SMR.1/CM	(FIA_UID.1)	FIA_UID.1/CM
FTP_ITC.1/CM	No dependencies	
FPT_RCV.4/SCP	No dependencies	
FPT_RCV.3/SCP	(AGD_OPE.1)	AGD_OPE.1
FCS_COP.1/DAP	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/DES, FCS_CKM.4

Requirements	CC Dependencies	Satisfied Dependencies
FCS_COP.1/SHA2	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FCS_COP.1/CRC	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FCS_RND.1	No dependencies	
FPT_FLS.1/SecureAPI	No dependencies	
FPT_ITT.1/SecureAPI	No dependencies	
FPR_UNO.1/SecureAPI	No dependencies	
FDP_ACC.2/FIREWALL	(FDP_ACF.1)	FDP_ACF.1/FIREWALL
FDP_ACF.1/FIREWALL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/FIREWALL, FMT_MSA.3/JCVM
FDP_IFC.1/JCVM	(FDP_IFF.1)	FDP_IFF.1/JCVM
FDP_IFF.1/JCVM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_RIP.1/OBJECTS	No dependencies	
FMT_MSA.1/JCRE	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FMT_SMR.1
FMT_MSA.1/JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_SMF.1, FMT_SMR.1
FMT_MSA.2/FIREWALL_JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/FIREWALL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/JCVM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_SMF.1	No dependencies	
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2/AID

Requirements	CC Dependencies	Satisfied Dependencies
FCS_CKM.1/DES	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.2/DES, FCS_CKM.4, FCS_COP.1/DES_CIPHER, FCS_COP.1/DES_MAC_COMP
FCS_CKM.1/AES	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.2/AES, FCS_CKM.4, FCS_COP.1/AES_CIPHER, FCS_COP.1/AES_MAC_COMP
FCS_CKM.1/RSA	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.2/RSA, FCS_CKM.4, FCS_COP.1/RSA_SIGN, FCS_COP.1/RSA_CIPHER
FCS_CKM.2/DES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/DES, FCS_CKM.4
FCS_CKM.2/AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/AES, FCS_CKM.4
FCS_CKM.2/RSA	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/RSA, FCS_CKM.4
FCS_CKM.3/DES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/DES, FCS_CKM.4
FCS_CKM.3/AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/AES, FCS_CKM.4
FCS_CKM.3/RSA	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/RSA, FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1/DES, FCS_CKM.1/AES, FCS_CKM.1/RSA
FCS_COP.1/DES_CIPHER	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/DES, FCS_CKM.4

Requirements	CC Dependencies	Satisfied Dependencies
FCS_COP.1/DES_MAC_COMP	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/DES, FCS_CKM.4
FCS_COP.1/AES_CIPHER	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/AES, FCS_CKM.4
FCS_COP.1/AES_MAC_COMP	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/AES, FCS_CKM.4
FCS_COP.1/RSA_SIGN	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/RSA, FCS_CKM.4
FCS_COP.1/RSA_CIPHER	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/RSA, FCS_CKM.4
FCS_COP.1/HMAC	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/DES, FCS_CKM.1/AES, FCS_CKM.4
FDP_RIP.1/ABORT	No dependencies	
FDP_RIP.1/APDU	No dependencies	
FDP_RIP.1/bArray	No dependencies	
FDP_RIP.1/KEYS	No dependencies	
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FDP_RIP.1/TRANSIENT	No dependencies	
FAU_ARP.1	(FAU_SAA.1)	
FDP_SDI.2	No dependencies	
FPR_UNO.1	No dependencies	
FPT_FLS.1	No dependencies	
FPT_TDC.1	No dependencies	
FIA_ATD.1/AID	No dependencies	
FIA_UID.2/AID	No dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
FIA_USB.1/AID	(FIA_ATD.1)	FIA_ATD.1/AID
FMT_MTD.1/JCRE	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1, FMT_SMR.1
FMT_MTD.3/JCRE	(FMT_MTD.1)	FMT_MTD.1/JCRE
FDP_ACC.2/ADEL	(FDP_ACF.1)	FDP_ACF.1/ADEL
FDP_ACF.1/ADEL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/ADEL, FMT_MSA.3/ADEL
FDP_RIP.1/ADEL	No dependencies	
FMT_MSA.1/ADEL	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/ADEL, FMT_SMF.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.3/ADEL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_SMF.1/ADEL	No dependencies	
FMT_SMR.1/ADEL	(FIA_UID.1)	
FPT_FLS.1/ADEL	No dependencies	
FDP_ACC.2/JCRMI	(FDP_ACF.1)	FDP_ACF.1/JCRMI
FDP_ACF.1/JCRMI	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/JCRMI, FMT_MSA.3/JCRMI
FDP_IFC.1/JCRMI	(FDP_IFF.1)	FDP_IFF.1/JCRMI
FDP_IFF.1/JCRMI	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCRMI, FMT_MSA.3/JCRMI
FMT_MSA.1/EXPORT	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/JCRMI, FMT_SMF.1/JCRMI, FMT_SMR.1/JCRMI
FMT_MSA.1/REM_REFS	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/JCRMI, FMT_SMF.1/JCRMI, FMT_SMR.1/JCRMI
FMT_MSA.3/JCRMI	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_SMR.1/JCRMI
FMT_REV.1/JCRMI	(FMT_SMR.1)	FMT_SMR.1/JCRMI
FMT_SMF.1/JCRMI	No dependencies	
FMT_SMR.1/JCRMI	(FIA_UID.1)	FIA_UID.2/AID

Table 15 SFRs dependencies

Rationale for the exclusion of dependencies

The dependency FIA_UID.1 of FMT_SMR.1/Installer is unsupported. This PP does not require the identification of the "installer" since it can be considered as part of the TSF.

The dependency FCS_CKM.4 of FCS_COP.1/SHA2 is unsupported. Hash operation according to SHA2 do not require key.

The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1/SHA2 is unsupported. Hash operation according to SHA2 do not require key.

The dependency FCS_CKM.4 of FCS_COP.1/CRC is unsupported. CRC operation do not require any key.

The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1/CRC is unsupported. CRC operation do not require any key.

The dependency FMT_SMF.1 of FMT_MSA.1/JCRE is unsupported. The dependency between FMT_MSA.1/JCRE and FMT_SMF.1 is not satisfied because no management functions are required for the Java Card RE.

The dependency FAU_SAA.1 of FAU_ARP.1 is unsupported. Potential violation analysis is used to specify the set of auditable events whose occurrence or accumulated occurrence held to indicate a potential violation of the SFRs, and any rules to be used to perform the violation analysis. The dependency of FAU_ARP.1 on this functional requirement assumes that a "potential security violation" is an audit event indicated by the FAU_SAA.1 component. The events listed in FAU_ARP.1 are, on the contrary, merely self-contained ones (arithmetic exception, ill-formed bytecodes, access failure) and ask for a straightforward reaction of the TSFs on their occurrence at runtime. The JCVM or other components of the TOE detect these events during their usual working order. Thus, in principle there would be no applicable audit recording in this framework. Moreover, no specification of one such recording is provided elsewhere. Therefore no set of auditable events could possibly be defined.

The dependency FIA_UID.1 of FMT_SMR.1/ADEL is unsupported. This PP does not require the identification of the "deletion manager" since it can be considered as part of the TSF.

6.3.3.2 SARs dependencies

Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.4, ADV_TDS.3
ADV_FSP.4	(ADV_TDS.1)	ADV_TDS.3
ADV_IMP.1	(ADV_TDS.3) and (ALC_TAT.1)	ADV_TDS.3, ALC_TAT.1
ADV_TDS.3	(ADV_FSP.4)	ADV_FSP.4

Requirements	CC Dependencies	Satisfied Dependencies
AGD_OPE.1	(ADV_FSP.1)	ADV_FSP.4
AGD_PRE.1	No dependencies	
ALC_CMC.4	(ALC_CMS.1) and (ALC_DVS.1) and (ALC_LCD.1)	ALC_CMS.4, ALC_DVS.2, ALC_LCD.1
ALC_CMS.4	No dependencies	
ALC_DEL.1	No dependencies	
ALC_DVS.2	No dependencies	
ALC_LCD.1	No dependencies	
ALC_TAT.1	(ADV_IMP.1)	ADV_IMP.1
ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No dependencies	
ASE_INT.1	No dependencies	
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.4, ASE_INT.1, ASE_REQ.2
ATE_COV.2	(ADV_FSP.2) and (ATE_FUN.1)	ADV_FSP.4, ATE_FUN.1
ATE_DPT.1	(ADV_ARC.1) and (ADV_TDS.2) and (ATE_FUN.1)	ADV_ARC.1, ADV_TDS.3, ATE_FUN.1
ATE_FUN.1	(ATE_COV.1)	ATE_COV.2
ATE_IND.2	(ADV_FSP.2) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_COV.1) and (ATE_FUN.1)	ADV_FSP.4, AGD_OPE.1, AGD_PRE.1, ATE_COV.2, ATE_FUN.1
AVA_VAN.5	(ADV_ARC.1) and (ADV_FSP.4) and (ADV_IMP.1) and (ADV_TDS.3) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_DPT.1)	ADV_ARC.1, ADV_FSP.4, ADV_IMP.1, ADV_TDS.3, AGD_OPE.1, AGD_PRE.1, ATE_DPT.1

Table 16 SARs dependencies

6.3.4 Rationale for the Security Assurance Requirements

EAL4 is required for this type of TOE and product since it is intended to defend against sophisticated attacks. This evaluation assurance level allows a developer to gain maximum assurance from positive security engineering based on good practices. EAL4 represents the highest practical level of assurance expected for a commercial grade product. In order to provide a meaningful level of assurance that the TOE and its embedding product provide an adequate level of defense against such attacks: the evaluators should have access to the low level design and source code. The lowest for which such access is required is EAL4.

6.3.5 AVA_VAN.5 Advanced methodical vulnerability analysis

The TOE is intended to operate in hostile environments. AVA_VAN.5 "Advanced methodical vulnerability analysis" is considered as the expected level for Java Card technology-based products hosting sensitive applications. AVA_VAN.5 has dependencies on ADV_ARC.1, ADV_FSP.1, ADV_TDS.3, ADV_IMP.1, AGD_PRE.1, AGD_OPE.1. All of them are satisfied by EAL4.

6.3.6 ALC_DVS.2 Sufficiency of security measures

Development security is concerned with physical, procedural, personnel and other technical measures that may be used in the development environment to protect the TOE and the embedding product. The standard ALC_DVS.1 requirement mandated by EAL4 is not enough. Due to the nature of the TOE and embedding product, it is necessary to justify the sufficiency of these procedures to protect their confidentiality and integrity. ALC_DVS.2 has no dependencies.

7 TOE Summary Specification

7.1 TOE Summary Specification

7.1.1 Java Card System Protection Profile - Open Configuration

7.1.1.1 JCS

This section defines the security functions to be achieved by the JCS part of the TOE.

JCS.APDUBuffer

The security function maintains a byte array buffer accessible from any applet context. This buffer is used to transfer incoming APDU header and data bytes as well as outgoing data according to [JCAPI]. The APDU class API is designed to be transport protocol independent T=0, T=1, T=CL (as defined in ISO 7816-3).

Application note:

ADPU buffer is a JCRE temporary entry point object where no associated reference can be stored in a variable or an array component.

JCS.ByteCodeExecution

This security function realizes applet bytecode execution according to JCVM rules [JCVM].

The JCVM execution may be summarized in JCVM interpreter start-up, bytecode execution and JCVM interpreter loop. The applet bytecode execution consists in:

fetching the next bytecode to execute according to the applet's code flow control,

decoding the next bycode,

executing the fetched bytecode.

The JCVM manages 5 types of objects: persistent objects, transient objects, persistent arrays (boolean, byte, short, int or reference), transient arrays (boolean, byte, short, int or reference) and static field images. For each type of object, different types of control are performed [see JCVM §4].

JCS.Crypto

The security function offers the following services to applets thanks to JavaCard API:

Generation of random number as defined in [JCAPI] and conformant to ANSSI standard to be used for key values or challenges during external exchanges,

Computation of checksum CRC16 and CRC32 conformant with ISO3309,

Ciphering and deciphering operation using DES algorithm in ECB and CBC mode with padding scheme (NOPAD, ISO9797 or PKCS #5),

Ciphering and deciphering operation using AES (128 bits) algorithm in ECB and CBC mode with padding scheme (NOPAD),

Ciphering and deciphering operation using RSA with CRT algorithm in mode ISO14888, with padding scheme (ISO9796 or PKCS #1),

Data Hash operation for message digests using SHA-2 algorithm (SHA2-256, SHA2-384), Generation of an signature of a byte array, and verifying an signature stored in a byte array using a generation of 20-byte SHA-2 message digest using RSA algorithm with PKCS#1-PSS padding scheme,

Generation of 4-byte or 8-byte MAC using DES (112 or 168 bits key) algorithm according to ISO9797-1,

Generation of 16-byte MAC using AES algorithm in CBC mode (MAC_128) with padding scheme (NOPAD).

These operations are performed in a way to avoid revealing the key values. If the applet specifies an algorithm that the platform does not support, the JCRE refuses to perform the cryptographic operation and generates an exception. Even if [JCAPI] specifies some other algorithms or parameters for cryptographic operations, the use of these other values are not advised; and clearly out of scope of the TOE. See [USR] for details.

JCS.EraseResidualData

The security function ensures that sensitive data are locked upon the following operations as defined in [JCRE22]:

Deletion of package and/or applications,

Deletion of objects.

There are erased when space needs to be reused for allocation of new object.

This security function also ensures that the sensitive temporary buffers (transient object, bArray object, APDU buffer, Cryptographic buffer) are securely cleared after their usage with respect to their life-cycle as defined in [JCRE22], transient object at reset or allocation and persistent object are erased at allocation for new object.

JCS.Exception

This security function manages throwing of an instance of Exception class in the following cases:

a SecurityException when an illegal access to an object is detected,

a SystemException with an error code describing the error condition,

a RemoteException when a communication-related exception has occurred during the execution of a remote method call,

a CryptoException in case of algorithm error or illegal use,

any exception decided by the applet or the JCRE handled as temporary JCRE entry point object with associated JC API. It also offers a means to applet to handle exception and to JCRE to handle uncaught exception by applets.

JCS.Firewall

This security function enforces the Firewall access control policy and the JCVM information flow control policy at runtime. It defines how accessing the following items: Static Class Fields, Array Objects, Class Instance Object Fields, Class Instance Object Methods, Standard Interface Methods, Shareable Interface Methods, Classes, Standard Interfaces, Shareable Interfaces, Array Object Methods.

Based on security attributes [Sharing, Context, Lifetime], it performs access control to object fields between objects and thows security exception when access is denied. Thus, it enforces applet isolation located in different packages and controls the access to global data containers shared by all applet instances.

The JCRE shall allocate and manage a context for each Java API package containing applets. The JCRE maintains its own context a special system privileges so that it can perform operations that are denied to contexts of applets.

JCS.KeyManagement

This security function enforces key management for the different associated operations: key building, key agreement, key generation, key importation, key exportation, key masking, key destruction using standard API defined in [JCAPI].

Key generation support generation of RSA key pairs using a secure random number generator compliant with ANSSI's Standard security level for cryptography operations.

Key agreement enables an applet to agree on a shared secret with the external, with a method conformant to [JCAPI]. It is built to avoid disclosure of this secret to third parties observing exchange done for key agreement.

Key masking protects the confidentiality of cryptographic keys from being read out from the memory. It ensures the service of accessing and modifying them.

Key destruction disables the use of a key both logically and physically. Reuse is only possible after erase.

Key importation and exportation is done using method protecting confidentiality and integrity of key.

JCS.OutOfLifeDataUndisclosure

This security function ensures that sensitive data are locked until postponed erasure on the following operations:

Deletion of persistent and transient objects according to [JCRE22].

JCS.OwnerPIN

This security function supplies to applet a mean to assume a user identification and authentication with the OwnerPin class conformant to [JCAPI].

It offers to create a PIN and store it securely in the persistent memory. It allow access to PIN value only to perform a secure comparison between a PIN stored in the persistent memory and a data received as parameter.

A method returns a positive result if a valid Pin has been presented during current session. If the PIN is not blocked and the comparison is successful, the validated flag is set to and the try counter is set to its maximum, otherwise the authentication fails and the associated try counter is decremented. When the validated flas is set, it is assumed that the user is authenticated.

When the try counter reaches zero, the PIN is blocked and the authentication is no more possible until the PIN is unblocked.

JCS.Package

Thsi security functions manages packages. Package is a structural item defined for naming, loading, storing, execution context definition. There are rules for identification of package, for structure check and access rules definition. If inconsistent items are found during checks, an error message is sent.

JCS.RMI

This security function enforces the RMI access control policy and the RMI information flow control policy between CAD and Remote Object located in card using dedicated security attributes and their management as defined in [JCRE].

The Java Card RMI message is encapsulated within the APDU object passed into the RMIService methods. RMIService class implements the Java Card RMI protocol and processes the RMI access commands

JCS.RNG

This security function provides random value using a given algorithm with or without a seed as defined in [JCAPI].

JCS.RunTimeExecution

This security function provides a secure run time environment and deals with:

Instance registration or deletion,

Application selection,

Applet opcode execution,

JCS API methods execution,

Logical channel management,

APDU flow control, dispatch and buffer management,

JCRE memory and context management,

JCRE reference deletion,

JCRE access rights,

JCRE throw exception,

JCRE security reaction.

7.1.1.2 OS

This section defines the security functions to be achieved by the OS part of the TOE.

OS.Atomicity

The security function performs write operations atomically on complex type or object in order to avoid incomplete update. Prior to be written, the data are stored in an atomic back-up area. In case on writing interrupt, the only two possible values are: initial value if writing is not started or final value if writing is started. At next start-up, the atomic back-up area is check to finalize interrupted writing.

OS.Memory Management

The security function allocates memory areas and performs access control to memory areas to avoid unauthorized access. It manages circular writing to avoid instable memory state. It assumes memory recovery in case of error detection. It offers (when required) confidentiality services for data storage: Ciphering / Deciphering of Data in RAM or in FLASH, Scrambling / Unscrambling of Data in RAM or in FLASH.

7.1.1.3 IC

IC.Limited FaultTolerance

The TSF manages a certain number of faults or errors that may happen, related to memory content, CPU, Random generation and cryptographic operation, thus preventing risk of malfunction. It is related to FRU_FLT.2 from [ST/IC].

IC.Secure State

The TSF provides preservation of secure state managing security violation resulting in an immediate reset. It is related to FPT_FLS.1 from [ST/IC].

IC.LIM.Capability (TEST)

The TSF ensures that test capability is unavailable in USER configuration. It is related to FMT_LIM.1 [TEST] from [ST/IC].

IC.LIM.Capability (ISSUER)

The TSF ensures that secure flash loader and test capability are unavailable in USER configuration. It is related to FMT_LIM.1 [ISSUER] from [ST/IC].

IC.ModeControl

The TSF ensures that only defined modes are available: TEST, ISSUER, USER configuration. It is related to FMT_LIM.2 [TEST] & ISSUER] from [ST/IC].

IC.Audit Storage

The TSF provides command to store data for audit purpose using commands only available to authorized process. It is related to FAU_SAS.1 from [ST/IC].

IC.Resistance to Physical Attack

The TSF ensures resistance to physical tampering using features against probing and an active shield detecting integrity violation. It is related to FPT_PHP.3 from [ST/IC].

IC.Internal Data Transfer Protection

The TSF prevents disclosure of internal and user data thanks to memory scrambling and encryption, bus encryption... It is related to FDP_ITT.1, FPT_ITT.1, FDP_IFC.1 from [ST/IC].

IC.Random Number Generation

The TSF produces AIS31-qualified random numbers that can be directly used in embedded software. It is related to FCS_RNG.1 from [ST/IC].

IC.Cryptoaccelerator

The TSF provides EDES accelerator to perform DES and TDES encryption and decryption conformant to FIPS PUB 46-3. It is related to FCS_COP.1 from [ST/IC].

It also uses RNG, arithmethic primitives of Nescrypt.But there are no usage of NesLib.

IC.Memory Protection

The TSF enforces a default memory protection policy when none other is programmed by the embedded software. It is related to FMT_MSA.3 from [ST/IC].

IC.MPU

The TSF provides a dynamic Memory protection unit (MPU) that can be configured by the ES. It is related to FMT_MSA.1, FMT_SMF.1 from [ST/IC].

IC.Loading Access Control

The TSF provides an access control to loading. The Standard Loader instructions and/or Advanced Loader instructions can be executed only if valid passwords have been presented. It is related to FDP_ACC.2, FDP_ACF.1 from [ST/IC].

7.1.2 GP

GP.CardContentManagement

This security function management operations dedicated to application management and associated rules in [GP22, §9.3]: Application loading, Application installation, Application registration, Application extradition, Application selection, Application removal.

According to context application can be executable files or application instances.

It also performs (when required) DAP verification of executable file prior registration as described in [GP22 §6 and 9.2].

GP.SecurityDomain

This security function provides security domain management: as SD creation, SD selection, SD privileges setting, SD deletion in SD hierarchy. It provides means to associate or extradite an application to a security domain in order to provide services (as secure channel) to the dedicated application without sharing the related keys stored in SD. It also provides Keyset Management in SD, with Key Set creation, Key set deletion, key importation, replacement, or deletion in Key Set.

Security Domains are privileged Applications as defined in [GP2.2.2 § 7], holding cryptographic keys to be used to support Secure Channel Protocol operations and/or to authorize card content management functions. There are different types of security domain with dedicated privileges and associated operations: ISD Security domain, Supplementary Security domains, and Controlling Authority Security domains.

ISD Security domain as defined in [GP2.2.2 §7.1], is the mandatory Security Domain, implicitly selected if the Application implicitly selectable on the same logical channel of the same card I/O interface is removed. It inherits of the Final Application privilege if the Application with that privilege is removed.

Supplementary Security Domains are privileged Applications with dedicated privileges:

Token Verification Privilege as described in [GP22 §9.1.3.1]

Delegated Management Privilege as described in [GP22 §9.1.3.3]

Global Delete Privilege as described in [GP22 §9.1.3.4]

Global Lock Privilege as described in [GP22 §9.1.3.5]

Receipt Generation Privilege as described in [GP22 §9.1.3.6]

GP.SCP

This security function manages Secure Channel protocol according to [GP22] annex D,E and [GP22-A].

GP.CASD

This security function manages supplementary Controlling Authority Security Domain with associated functions to confidential Card Content Management as defined in [GP22-A].

Controlling Authority Security Domain is a supplementary Security Domain dedicated to the Controlling Authority with dedicated privileges. It contains Security Domains cryptographic keys needed to confidentially personalize an initial set of Secure Channel Keys of an APSD.

GP.VASD

This security function manages supplementary Verification Authority Security Domain with associated functions to mandated DAP as defined in [GP22 §9.2].

7.1.3 SecureAPI

SA.FlowControl

The security function provides means to application to control execution flow, to detect any failure and to react if required.

SA.SecureOperation

The security function provides means to application to execute securily data transfer and comparison, to detect any failure during operation and to react if required.

SA.RandomDelay

The security function provides means to introduce dummy operations leading to unobservability of sensitive operation.

7.2 SFRs and TSS

7.2.1 SFRs and TSS – Rationale

Chapter content has been removed in Public version.

7.2.2 Association tables of SFRs and TSS

Chapter content has been removed in Public version.

8 Notice

This document has been generated with TL SET version 2.3.7 (for CC3). For more information about the security editor tool of Trusted Labs visit our website at www.trusted-labs.com.

9 References, Glossary and Abbreviations

9.1 External References

Reference	Title
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model CCIMB-2009-07-001, version 3.1 Release 3, July 2009.
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements CCIMB-2009-07-002, version 3.1 Release 3, July 2009.
[CC-3]	Common Criteria for Information Technology security Evaluation Part 3: Security Assurance Requirements CCIMB-2009-07-003, version 3.1 Release 3, July 2009.
[CEM]	Common Methodology for Information Technology Security Evaluation CCIMB-2009-07-004, version 3.1 Release 3, July 2009.
[Comp]	CCDB, Composite product evaluation for Smart Cards and similar devices, September 2007, Version 1.0 - Revision 1, September 2007, CCDB-2007-09-001
[DCSSI2741]	Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques de niveau de robustesse standard N° 2741/SGDN/DCSSI/SDS/LCR Version 1.10
[FIPS 46-3]	FIPS 46-3: DES Data Encryption Standard (DES and TDES). National Institute of Standards and Technology
[FIPS 197]	FIPS 197: AES Advanced Encryption Standard. National Institute of Standards and Technology.
[FIPS 180-2]	FIPS-46-3: Secure Hash Standard (SHA). National Institute of Standards and Technology.
[GP22]	Global Platform 2.2.2, Specification GP
[GP-CCCM]	GlobalPlatform, Card Confidential Card Content Management, Card specification v2.2 – Amendment A,
[GP-UICC]	GlobalPlatform Card UICC Configuration Version 1.0
[ISO 7816-4]	Identification cards - Integrated circuit(s) cards with contacts, Part 4: Interindustry commands for interchange.
[ISO 7816-6]	Identification cards - Integrated circuit(s) cards with contacts, Part 6: Interindustry data elements.
[ISO 7816-9]	Identification cards - Integrated circuit(s) cards with contacts, Part 9: Additional Inter industry commands and security attributes.
[ISO 9796-2]	ISO/IEC 9796-2 Information technology -- Security techniques -- Digital signature schemes giving message recovery -- Part 2: Integer factorization based mechanisms
[JCAPI222]	Java Card^{TM} APIs specification version 2.2.2, Sun Microsystems, Inc, March 2006.
[JCRE222]	Java Card^{TM} Runtime Environment Specification version 2.2.2, Sun Microsystems, Inc, March 2006
[JCVM222]	Java Card 2.2.2 Virtual Machine Specification, Sun Microsystems, March 2006
[JIL]	Joint Interpretation Library Composite product evaluation for Smart Cards and similar devices Version 1.0 September 2007
[PP-BSI-0035]	Security IC Platform Protection Profile Version 1.0 15.06.2007

Reference	Title
[PP-JCS]	Java Card™ System Protection Profile "Open Configuration" Version 2.6
[PP-USIM]	(U)SIM Java Card Platform Protection Profile Basic Configuration V2.0.2, June 2010
[PP-USIMb]	(U)SIM Java Card Platform Protection Profile in SCWS Configuration V2.0.2, June 2010
[PP-SSCD]	Protection profiles for Secure signature creation device — Part 2: Device with key generation
[RFC2085]	HMAC-MD5 IP Authentication with Replay Prevention
[RFC2104]	HMAC: Keyed-Hashing for Message Authentication
[RSA PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard
[ST/IC]	Security Target of SC33F640 SMD_SC33F640_ST_10_001 Rev 01.00
[STM_SG]	SC33F640 Security Guidance
[TS03.19]	ETSI 3GPP TS 03.19, Subscriber Identity Module Application Programming Interface (SIM API) for Java Card™; Stage 2
[TS 51.011]	ETSI 3GPP TS 51.011, 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (Release 4)
[TS 51.013]	ETSI 3GPP TS 51.013, 3rd Generation Partnership Project; Technical Specification Group Terminals; Test specification for SIM API for Java Card™ (Release 4)
[TS 51.014]	ETSI 3GPP 51.014, 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (Release 4)
[TS 51.013]	ETSI 3GPP TS 102.225, 3rd Generation Partnership Project; Technical Specification Group Terminals; Test specification for SIM API for Java Card™ (Release 4) – v 2.0.0 (2002-09).
[TS 102.221]	ETSI 3GPP TS 102.221, Smart cards; UICC-Terminal interface; Physical and logical characteristics (Release 6)
[TS 102.222]	ETSI 3GPP TS 102.222, Integrated Circuit Cards (ICC); Administrative commands for telecommunications applications (Release 6) – v 6.0.0 (2003-02).
[TS 102.223]	ETSI 3GPP TS 102.223, Smart Cards; Card Application Toolkit (CAT) (Release 6)
[TS 102.224]	ETSI 3GPP TS 102.224, Smart cards; Security mechanisms for UICC based Applications - Functional requirements (Release 6)
[TS 102.225]	ETSI 3GPP TS 102.225, Smart cards; Secured packet structure for UICC based applications (Release 6) –
[TS 102.226]	ETSI 3GPP TS 102.226, Smart Cards; Remote APDU structure for UICC based applications (Release 6)
[TS 102.240]	ETSI 3GPP TS 102.240, Smart Cards; UICC Application Programming Interface and Loader Requirements; Service description; (Release 6)
[TS 102.241]	ETSI 3GPP TS 102.241, UICC API, 3GPP Release 6
[TS 102.613]	ETSI 3GPP TS 102.613, UICC - Contactless Front-end (CLF) Interface; Part 1: Physical and data link layer characteristics (Release 7)
[TS131.111]	ETSI 3GPP TS 131.111, Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) (Release 6)

Reference	Title
[T131.130]	ETSI TS 131.130, Digital cellular telecommunications system (Phase 2+);Universal Mobile Telecommunications System (UMTS);(U)SIM Application Programming Interface (API);(U)SIM API for Java Card (3GPP TS 31.130 version 6.6.0 Release 6)

9.2 Internal References

Reference	Title
[ST2_PLF]	Security Target : LinqUs USIM 128K PK certified using SC33F640 ST_D1172363
[FSP_PLF]	PHENIX Platform Functional Specification D1145952 (FSP_D1145952)
[TDS_PLF]	PHENIX Platform TOE Design Specification D1145953 (TDS_D1145953)
[ARC_PLF]	PHENIX Platform TOE Security Architecture D1145954 (ARC_D1145954)
[IMP_PLF]	PHENIX Platform Implementation representation D1145955 (IMP_D1145955)
[PRE_PLF]	PHENIX Platform Preparation Guidance D1185540 (PRE_D1185540)
[OPE_PLF]	PHENIX Platform Operational Guidance D1185542 (OPE_D1185542)
[COV_PLF]	PHENIX Platform Analysis of test coverage D1145959 (COV_D1145959)
[DPT_PLF]	PHENIX Platform Analysis of the depth of testing D1145960 (DPT_D1145960)
[FUN_PLF]	PHENIX Platform Functional Test Documentation D1145961 (FUN_D1145961)
[CMC_PLF] [CMS_PLF]	PHENIX Platform Configuration Management Plan D1145963 (CMC_D1145963) PHENIX Platform Configuration Management Scope D1145965 (CMS_D1145965)
[LCD_PLF]	PHENIX Platform Life cycle Support D1145997 (LCD_D1145997)
[FLR_PLF]	PHENIX Platform Problem tracking Plan D1145968 (PTP_D1145968) PHENIX Platform Problem Flaw Remediation Plan D1145970 (FLR_D1145970)
[DVS_PLF]	PHENIX Platform Development Security Documentation D1145975 (DVS_D1145975)
[DEL_PLF]	PHENIX Platform Delivery Documentation D1145978 (DEL_D1145978)
[TAT_PLF]	PHENIX Platform Documentation of development tools D1145971 (TAT_D1145971)
[COMP_PLF]	PHENIX Platform Composition with Hardware D1145972 (COM_D1145972)

9.3 ABBREVIATIONS

Abbreviation	Description
AES	Advanced Encryption Standard
AID	Applet Identifier
APDU	Application Protocol Data Unit
API	Application Programmer Interface
CBC	Cipher Block Chaining
CC	Common Criteria
CM	Card Manager
CPLC	Card Production Life Cycle
DAP	Data Authentication Pattern
DES	Data Encryption Standard
DS	Dedicated Software
EAL	Evaluation Assurance Level
GP	Global Platform
HMAC	Keyed-Hash Message Authentication Code
IC	Integrated Circuit
JCRE	Java Card Runtime Environment
JCS	Java Card System
JCVM	Java Card Virtual Machine
MAC	Message Authentication Code
OSP	Organizational Security Policy
PP	Protection Profile
RNG	Random Number Generation
RSA	Cryptographic module "Rivest, Shamir, Adleman"
SHA-2	Cryptographic module "Secure hash standard"
ST	Security Target
TOE	Target of Evaluation.

9.4 Glossary

Term	Definition
Application	Instance of an Executable Module after it has been installed and made selectable
APDU	Standard communication messaging protocol between a card accepting device and a smart card
Card Administrator	The card administrator is an external entity issuing the card and performing main functions of card administration (as Card life cycle Management). It is usually the Card Issuer or MNO.
Controlling Authority	A Controlling Authority is entity independent from the MNO represented on the (U)SIM card and responsible for securing the keys creation and personalization of the Supplementary Security Domains.
DAP Block	Part of the Load File used for ensuring Load File Data Block verification
DAP Verification	A mechanism used by a Security Domain to verify that a Load File Data Block is authentic
Delegated Management	Pre-authorized Card Content changes performed by an approved Application Provider
Executable Load File	Actual on-card container of one or more application's executable code. It may reside in Immutable Persistent Memory or may be created in Mutable Persistent Memory as the resulting image of a Load File Data Block.
Executable Module	Contains the on-card executable code of a single application present within an Executable Load File
Issuer Security Domain	The primary on-card entity providing support for the control, security, and communication requirements of the card administrator (typically the Card Issuer or MNO)
Load File	A file transferred to a GlobalPlatform card that contains a Load File Data Block and possibly one or more DAP Blocks
Load File Data Block	Part of the Load File that contains one or more application(s) or libraries and support information for the application(s) as required by the specific platform
Load File Data Block Hash	A value providing integrity for the Load File Data Block
Message Authentication Code	A symmetric cryptographic transformation of data that provides data origin authentication and data integrity
Secure Channel	A communication mechanism between an off-card entity and a card that provides a level of assurance, to one or both entities
Secure Channel Protocol	A secure communication protocol and set of security services
Security Domain	On-card entity providing support for the control, security, and communication requirements of an off-card entity (e.g. the Card Issuer, an Application Provider or a Controlling Authority)
Supplementary Security Domain	A Security Domain other than the Issuer Security Domain dedicated to Application provider.
Token	A cryptographic value provided by a Card Issuer as proof that a Delegated Management operation has been authorized
Verification Authority	The Verification Authority (VA), is a trusted third party represented on the (U)SIM card, acting on behalf of the MNO and responsible for the verification of application signatures (mandated DAP) during the loading process.