紫光同芯

TONGXIN MICRO

TMCOS 4.0 0.3 on THD89 1.0.3 Secure Element

Java Card System Platform

Security Target Lite

Common Criteria EAL5+

Revision History

Rev	Date	Description
1.0	2025-1-14	ST Lite Public Release
1.1	2025-1-22	Chapter 1: Update the information in ST Reference and TOE Reference
1.2	2024-1-24	Chapter 1.1: Update the information in ST Reference Chapter 1.5.3: Update the site name

Content

1 ST INTRODUCTION

1.1 ST REFERENCE

Title:	TMCOS 4.0 0.3 on THD89 1.0.3 Secure Element Java Card System Platform Security Target Lite
Version:	1.2
Author:	TONGXIN MICROELECTRONICS Co., Ltd
Publication Date:	2025/1/24

1.2 TOE REFERENCE

Product Type:	Embedded Secure Element OS
TOE Name:	TMCOS 4.0 0.3 on THD89 1.0.3 Secure Element Java Card System Platform
TOE Version:	v4.0 0.3
Developer:	TSINGTENG Microsystem Co., Ltd

1.3 TOE IDENTIFICATION

The information for identification is listed in the table below:

Type	Information
TOE Version	v4.0 0.3
COS Name	TMCOS 4.0 0.3
Hardware	THD89 1.0.3

1.4 TOE OVERVIEW

1.4.1 TOE TYPE

The TOE type is an Embedded Secure Element that implements Java Card System and is deployed on a certified IC hardware. This TOE affords the security services including loading, installation and instantiation of applets before and after issuance, and secure execution of Java Card applets that are verified off-card.

The TOE includes the following components:

- The Java Card System Platform 3.1 – Open Configuration
- The Card Manager according to Global Platform 2.3 [GP]
- The Native Operating System which is the native part of the TMCOS
- The underlying IC Hardware THD89 Secure Element

This TOE communicates with external entities over SPI/ISO7816 interfaces. The SPI/ISO7816 interfaces are connected to the consumer devices so that this TOE is capable of providing security services to the application running in REE hosted in unsecure world.

1.4.2 TOE USAGE AND MAJOR SECURITY FEATURES

The intended usage of the TOE is to provide a secure platform of Java Card System on which applets carrying sensitive data can be operated securely over their lifecycle. Sensitive data that are contained in applets consist of cryptographic keys and PINs, and balance of an electronic wallet for various use cases as listed below for instance:

- Financial applications, like Credit/Debit ones, stored value purse, or electronic commerce, among others.
- Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.
- Personal identification, for granting access to secured sites or providing identification credentials to participants of an event.
- Electronic passports and identity cards.
- Secure information storage, like health records, or health insurance cards.
- Loyalty programs.

These applets can be secured against forgery and tampering with the security features provided by the TOE. The security features encompass encryption, decryption, signature generation, signature verification, key generation, secure management of PINs and secure storage of confidential data. In addition, the certified IC hardware also provides security features in hardware level to prevent physical attacks.

The security features provided by the TOE cover the following services of the TOE:

- Applet Loading
- Applet Installation
- Applet Personalization
- Applet Deletion
- Applet Extradition
- Applet Selection and deselection
- Applet Processing

- The TOE includes the following features:

  - 3DES for encryption/decryption (CBC and ECB) and MAC generation and verification (2-key 3DES, 3-key 3DES, Retail-MAC, CMAC and CBC-MAC).

  - AES for encryption/decryption with key length 128, 192 and 256 bits.

  - MAC generation and verification (CMAC, CBC-MAC).

- RSA CRT for decryption and signature generation.
- ECC for signature, verification, point multiplication and key pair generation operation with key sizes 224, 256, 384 and 521 bits. The supported curves include ansix9p224r1, ansix9p256r1, ansix9p384r1 and ansix9p521r1 from ANSI X9.62-2005, brainpoolP224r1, brainpoolP256r1 and brainpoolP384r1 curves from RFC 5639
- Random number generation.

- Java Card 3.1.0 functionality:
  - Executing Java Card bytecodes.
  - Managing memory allocation of code and data of applets.
  - Enforcing access rules between applets and the JCRE
  - Mapping of Java method calls to native implementations of e.g. cryptographic operation.
  - Garbage Collection fully implemented with complete memory reclamation including compactification.
  - Persistent Memory Management and Transaction Mechanism.
  - API extensibility using virtual method tables.
  - Array view object types. Array view objects are type of arrays with elements mapped to the elements of another array.
  - Static Resources in CAP files.
- Global Platform 2.3 functionality including Amendments A (scenario 1 and 3), D, E, F and H and is compliant with the Common Implementation Configuration.
  - Loading of Java Card packages.
  - Instantiating applet instances.
  - Java package deletion.
  - Java applet instance deletion.
  - Creating Supplementary Security Domains.
  - Associating applets to Security Domains.

- Installation of keys.
- Verification of signatures of signed applets
- CVM Management (Global PIN) fully implemented.
- Secure Channel Protocol SCP02 / SCP03 is supported.
- Delegated Management, DAP.
- Compliance to Secure Element configuration.

1.4.3 NON-TOE HARDWARE/SOFTWARE/FIRMWARE REQUIRED

A remote system is used to communicate with the TOE. The TOE implements Card Manager according to [GP] for card content management which manages the loading, installation and deletion of applets. The remote communication with the Card Manager on this TOE requires the establishment of a secure channel. In order to achieve this, a remote system is necessary to have the abilities to support secure channel protocol as defined in [GP], and securely manage the shared secrets with its on-card representative.

The way to create physical connection with the TOE is achieved over SPI interface. The SPI interface is directly connected to the user device as a contact interface. Communication with the TOE by this way necessitates the invocation of library APIs, for instance OM API, implemented outside the scope of the TOE.

The remote system and user device are off-card components which are out of the scope of this TOE.

Before loading an applet onto the TOE, the CAP files of the applet shall be verified off-card by the bytecode verifier in order to enforce all the rules imposed in [JCVM3] and the correctness of the format of the CAP files.

The bytecode verifier is a program that performs static checks on the bytecodes of the methods of a CAP file prior to the execution of the file on the card. Bytecode verification is a key component of security: applet isolation, for instance, depends on the file satisfying the properties a verifier checks to hold. A method of a CAP file that has been verified shall not contain, for instance, an instruction that allows forging a memory address or an instruction that makes improper use of a return address as if it were an object reference. In other words, bytecodes are verified to hold up to the intended use to which they are defined. Bytecode verification could be performed totally or partially dynamically. No standard procedure in that concern has yet been recognized. Furthermore, different approaches have been proposed for the implementation of bytecode verifiers, most notably data flow analysis, model checking and lightweight bytecode verification, this latter being an instance of what is known as proof carrying code. The actual set of checks performed by the verifier is implementation-dependent, but it is required that it should

at least enforce all the “must clauses” imposed in [JCVM3] on the bytecodes and the correctness of the CAP files’ format.

As for this Security Target, the bytecode verifier is an off-card component which is out of the scope of this TOE.

1.4.3.1 NON-EVALUATED SECURITY FUNCTIONALITY

The following security functionalities are supported by the TOE but not in the evaluation scope:

- SHA1, SHA 2 (224, 256, 384, 512)
- HMAC
- RSA for encryption/decryption and signature generation and verification
- RSA CRT for encryption and signature verification
- RSA and RSA CRT key generation
- Amendment C

1.5 TOE DESCRIPTION

1.5.1 PHYSICAL SCOPE

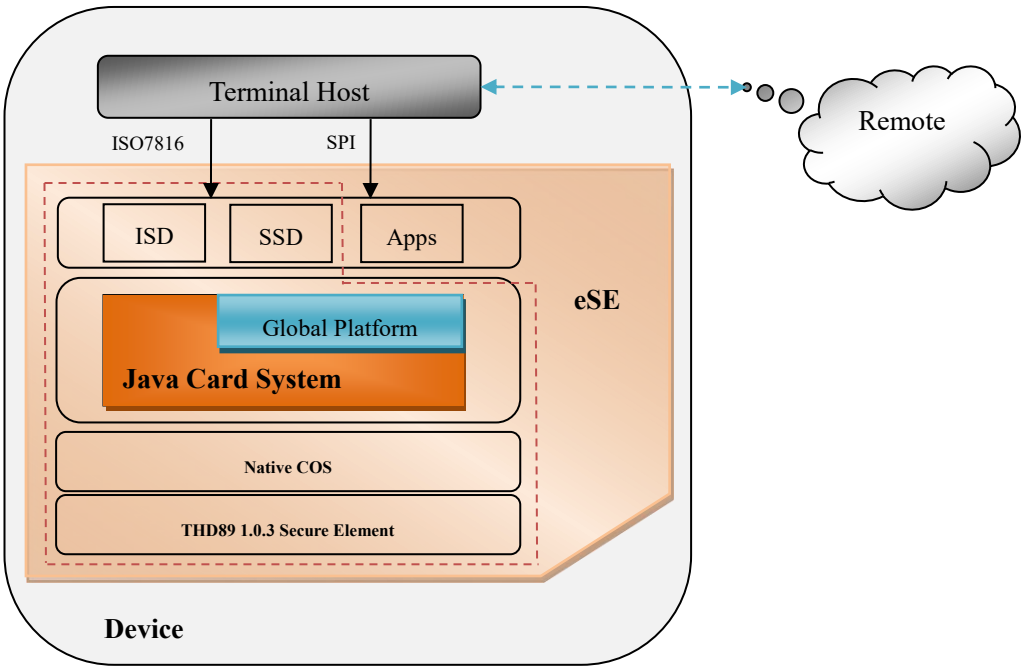


Figure 1: The TOE physical scope

The Target of Evaluation (TOE) is the Java Card System platform for embedded Secure Element. It covers the following components:

- The Java Card System Platform 3.1 – Open Configuration
- The Card Manager according to Global Platform 2.3 [GP]
- The Native Operating System which is the native part of the TMCOS
- The underlying IC Hardware THD89 Secure Element

Java Card RMI is not supported in the TOE

Category	Component	Version	Delivery form
HW	THD89 1.0.3	1.0.3	Module
FW	Crypto Library	2.1.0	Masked in ROM
FW	Crypto SU Library	2.2.0	Pre-Install in NVM
FW	CryptoECCSec Library	1.0.0	Pre-Install in NVM
SW	TMCOS	4.0 0.3	Binary in memory
DOC	[AGD_OPE]	[AGD_OPE]	Pdf file
DOC	[AGD_PRE]	[AGD_PRE]	Pdf file

1.5.2 LOGICAL SCOPE

The architecture of the TOE composed of various components is illustrated in the following diagram:

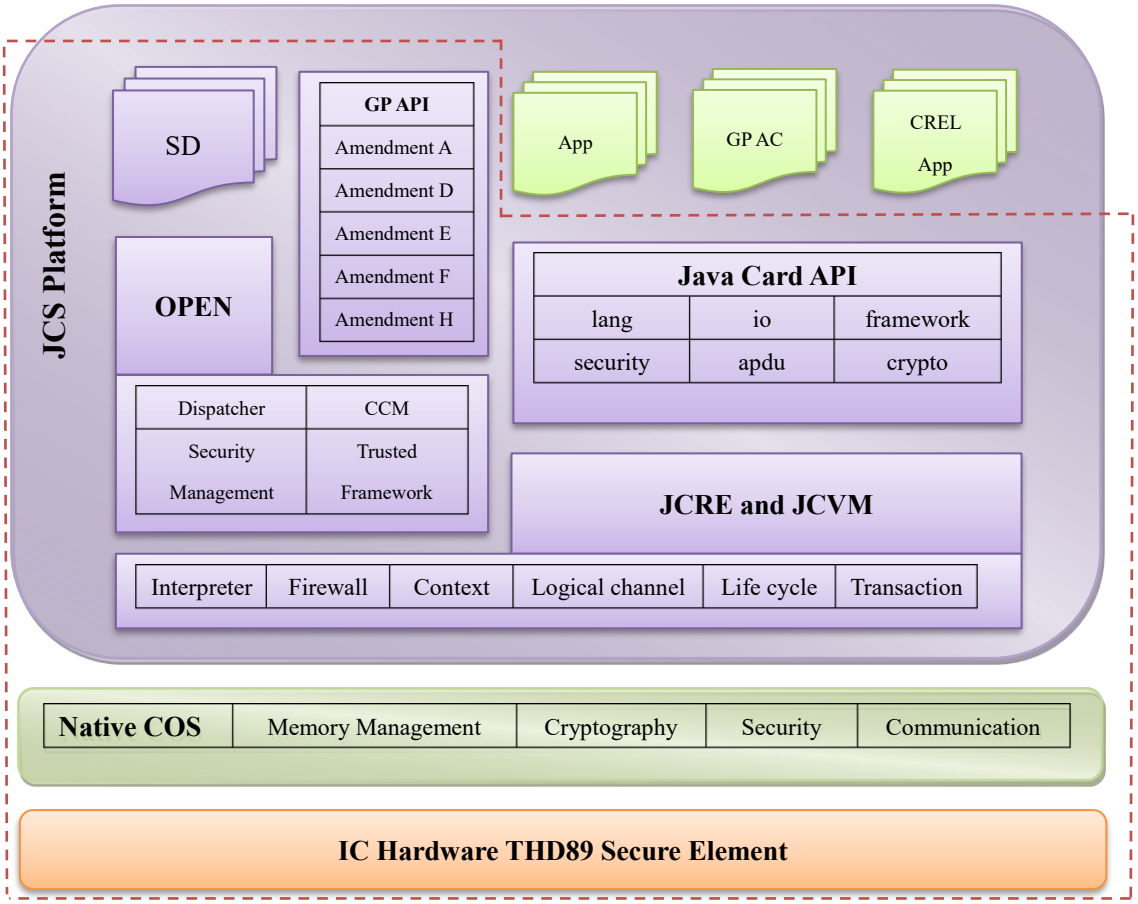


Figure 2: The TOE logical scope

As demonstrated in figure 2, the TOE is composed of three major components:

Hardware and IC dedicated software

The certified THD89 Secure Element is composed of hardware and IC dedicated software.

The hardware is based on a 32-bit secure CPU with ROM (Non-Volatile Read-Only Memory), NVM (Non-Volatile Programmable Memory) and RAM (Volatile Memory). The Hardware also incorporates communication peripherals and cryptographic coprocessors for execution and acceleration of symmetric and asymmetric cryptographic algorithms.

The THD89 Secure Element supports the following communication interfaces:

- SPI interface,
- ISO7816

The features provided by the THD89 Secure Element are listed below:

- Hardware coprocessor for TDES
- Hardware coprocessor for AES
- True Random Number Generator
- Hardware for RSA-CRT decryption support
- Hardware for ECC support
- Protection against power analysis,
- Protection against physical attacks,
- Protection against perturbation attacks
- Software library with cryptographic services for TDES, AES, RSA-CRT and ECC.

The detailed information about the reference of the certified THD89 IC Hardware is shown in the table below:

Hardware Name:	THD89 1.0.3 Secure Element
Certified HW Version:	Version 1.0
Certification ID:	CCN-CC-30/2023
Security Target Reference:	THD89 1.0.3 Secure Element Version 1.0 Security Target, version 0.1 Jun. 2023.

Native COS

The Native COS contains the classic secure element services including memory management, I/O management and cryptographic primitives. The functionalities provided by Native COS to the upper layer operating system are listed below:

- 3DES (ECB, CBC)
- RSA-CRT decryption
- AES
- ECC
- Pseudo-Random Number Generation (PRNG)

JCS platform

The implementation is compliant with the following technical standards:

- The Java Card 3.1 Virtual Machine [JCVM3]

  The JCVM provides bytecode interpreter service and a defensive execution environment against type confusion.

- The Java Card 3.1 Runtime Environment [JCRE3]

  The JCRE provides the execution of applets a secure runtime with mechanisms consisting of firewall, context management, logical channel management, applet lifecycle management and transaction mechanism.

- The Java Card 3.1 Application Programming Interface [JCAPI3]

  The JCAPI provides standardized programming routines to applets with java packages including lang, io, framework, security, apdu and crypto.

- The Global Platform Card Specification version 2.3 [GP]

  The implementation of Global Platform consists of SDs, APIs and OPEN.

  Specifically, the SDs include ISD, SSD and CASD which are on-card representatives of card issuer, applet provider and controlling authority. Together with GP API, the API and functionalities defined in Amendment A / D / E / F / H are supported by this TOE as well. The OPEN is responsible for command dispatch, card content management, security management and trusted framework.

  The Card Manager within the scope of this TOE is achieved by the functionalities of ISD, OPEN and Cardholder Verification Method Services which are included in the GP API.

1.5.3 LIFE CYCLE

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

The Java Card System life cycle is composed of four stages:

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

JCS storage is not necessarily a single step in the life cycle since it can be stored in parts. JCS delivery occurs before storage and may take place more than once if the TOE is delivered in parts.

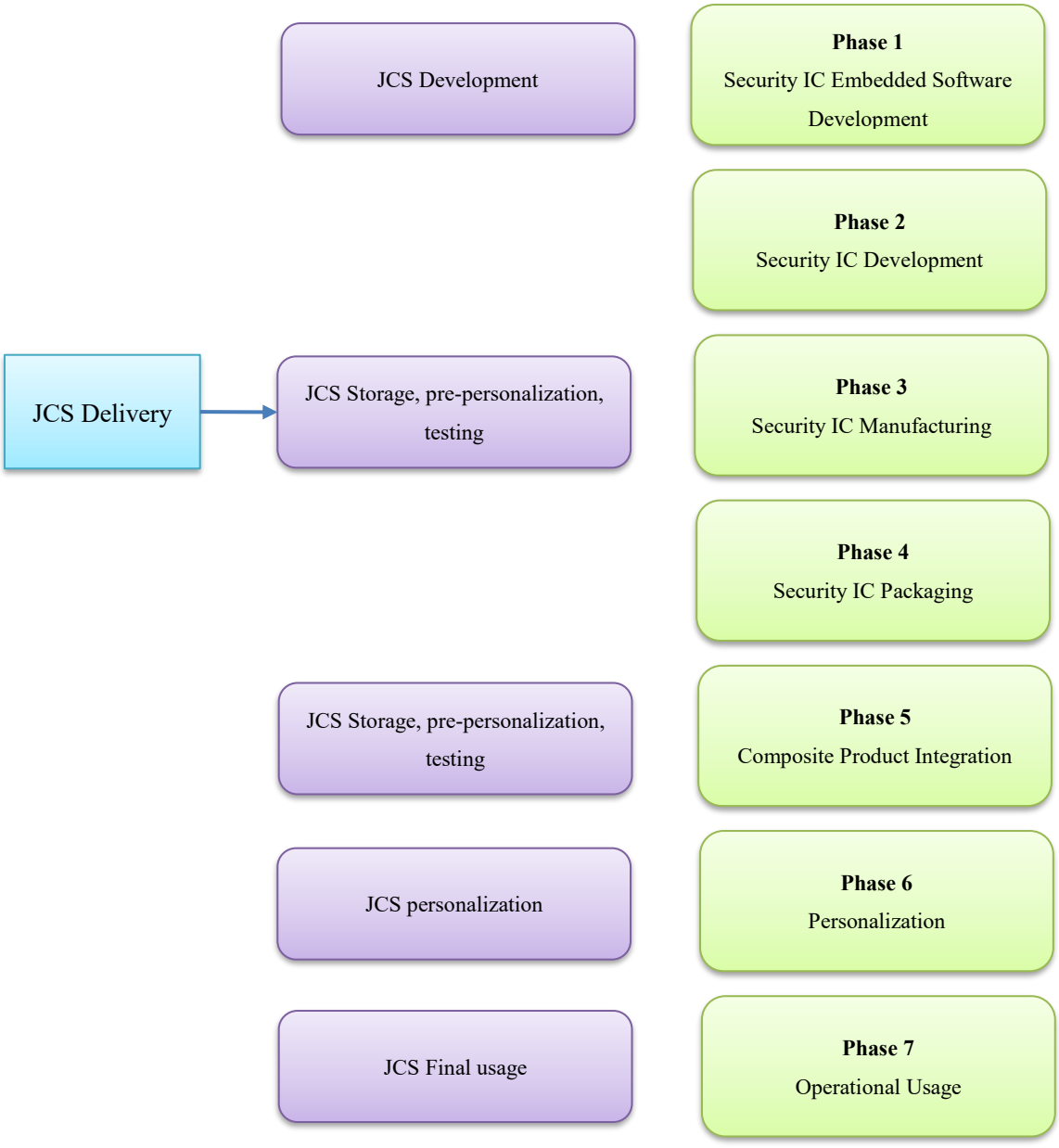


Figure 3: JCS (TOE) Life Cycle within Product Life Cycle

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

The delivery of the JCS occurs during Security IC Manufacturing (Phase 3). Delivery and acceptance procedures guarantees the authenticity, the confidentiality and integrity of the exchanged pieces by a means of involving encrypted signed sending. The evaluation of this TOE includes the delivery process.

In Phase 3, the Security IC Manufacturer stores personalizes the JCS and conducts tests on behalf of the JCS developer. The Security IC Manufacturing environment protects the integrity and confidentiality of the JCS and of any related material, for instance test suites. The evaluation of this TOE includes the whole Security IC Manufacturing environment, in particular those locations where the JCS is accessible for installation or testing...

In Phase 4, the IC Packaging Manufacturer is responsible for the IC packaging and testing.

The evaluation scope covers the activities across phase 1 to 4. After phase 4, the TOE is delivered to end customer as module form factor with its security functions fully enabled. The end customer shall follow the processes and procedures as defined in [AGD_PRE] for secure acceptance and installation to conduct activities that are covered in phase 5 to 6 as defined in [PP JCS].

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

The following table illustrates that the lifecycle of the Composite Product covers different steps in different sites.

Site	**Location**	**Activity**	**Phase**
Beijing Tsingteng MicroSystem Co., Ltd.	Bejing, China.	HW and SW development	1, 2
IC Manufacturing	Singapore	Wafer manufacturing& Wafer testing& JCS pre-Personalization	3
IC Packaging	Zibo Shandong, China	IC packaging & testing	4

1.5.4 TOE DELIVERY

The following table lists the items and methods for the TOE delivery:

Type	Name	Version	Delivery method
Composite Product	TMCOS 4.0 0.3 on THD89 1.0.3	V4.0 0.3	Module

Document	[AGD_OPE]	[AGD_OPE]	Encrypted e-mail
	[AGD_PRE]	[AGD_PRE]	Encrypted e-mail
Key	Root keys-TEK	v1.0	Courier deliver
	Root keys-IMK	v1.0	Encrypted e-mail

2 CONFORMANCE CLAIMS

2.1 CC CONFORMANCE CLAIM

Common Criteria Version:

This Security Target claims conformance to the Common Criteria version 3.1 revision 5 [CC1] [CC2] [CC3]:

- Common Criteria for Information Technology Security Evaluation, Part 1 [CC1]: Introduction and general model, Version 3.1, Revision 5, CCMB-2017-04-001, April 2017.
- Common Criteria for Information Technology Security Evaluation, Part 2 [CC2]: Security functional components, Version 3.1, Revision 5, CCMB-2017-04-002, April 2017.
- Common Criteria for Information Technology Security Evaluation, Part 3 [CC3]: Security assurance components, Version 3.1, Revision 5, CCMB-2017-04-003, April 2017.

The methodology used in the evaluation is [CEM]:

- Common Methodology for Information Technology Security Evaluation, Evaluation methodology, Version 3.1, Revision 5, CCMB-2017-04-004, April 2017.

Conformance to CC part 2 and 3:

This Security Target is CC Part 2 extended and CC Part 3 conformant of Common Criteria version 3.1, revision 5. The extended Security Functional Requirements are defined in Chapter Section 6.

2.2 CC PACKAGE CLAIM

This Security Target claims conformance to the assurance package EAL5 augmented with AVA_VAN.5 “Advanced methodical vulnerability analysis” and ALC_DVS.2 “Sufficiency of security measures”.

2.3 PP CLAIM

This Security Target claims demonstrable conformance to the Protection Profile “Java Card System – Open configuration” [PP JCS].

2.4 RATIONALE

2.4.1 PROTECTION PROFILE CONSISTENCY

This Security Target is conformant with the Protection Profile “Java Card System, Open configuration” [PP JCS].

Since the Card Manager is included in the TOE, OE.CARD-MANAGEMENT is changed to O.CARD-MANAGEMENT.

As the OE.CARD-MANAGEMENT is changed to O.CARD-MANAGEMENT, the assumption A.DELETION which is directly upheld by OE.CARD-MANAGEMENT is changed to T.DELETION, and then T.DELETION is countered by O.CARD-MANAGEMENT.

As the SCP is included in the TOE, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC are changed into the following Objectives on the TOE: O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC.

As no other modification was done, we can conclude that the conformance is demonstrated

3 SECURITY ASPECTS

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

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

3.1 CONFIDENTIALITY

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

3.2 INTEGRITY

#.INTEG-APPLI-CODE	Application code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to the memory zone where executable code is stored. In post-issuance application loading, this threat also concerns the modification of application code in transit to the card.

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

3.3 UNAUTHORIZED EXECUTIONS

#.EXE-APPLI-CODE	Application (byte)code must be protected against unauthorized execution. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC], §6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code; (3) unauthorized execution of a remote method from the CAD (if the TOE provides JCRMI functionality).
#.EXE-JCS-CODE	Java Card System bytecode must be protected against unauthorized execution. Java Card System bytecode includes any code of the Java Card RE or API. These concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC], §6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.
#.FIREWALL	The Firewall shall ensure controlled sharing of class instances7, and isolation of their data and code between CAP files (that is,

	controlled execution contexts) as well as between CAP files and the JCRE context. An applet shall not read, write, compare a piece of data belonging to an applet that is not in the same context, or execute one of the methods of an applet in another context without its authorization.
#.NATIVE	Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside those TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

3.4 BYTECODE VERIFICATION

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

3.4.1 CAP FILE VERIFICATION

Bytecode verification includes checking at least the following properties: (3) bytecode instructions represent a legal set of instructions used on the Java Card platform; (4) adequacy of bytecode operands to bytecode semantics; (5) absence of operand stack overflow/underflow; (6) control flow confinement to the current method (that is, no control jumps to outside the method); (7) absence of illegal data conversion and reference forging; (8) enforcement of the private/public access modifiers for class and class members; (9) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (10) enforcement of rules for binary compatibility (full details are given in [JCVM3], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [JCVM3] on the bytecodes and the correctness of the CAP files’ format.

As most of the actual Java Card VMs do not perform all the required checks at runtime, mainly because smart cards lack memory and CPU resources, CAP file verification prior to execution is mandatory. On the other hand, there is no requirement on the precise moment when the verification shall actually take place, as far as it can be ensured that the verified file is not

modified thereafter. Therefore, the bytecodes can be verified either before the loading of the file on to the card or before the installation of the file in the card or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. This Security Target assumes bytecode verification is performed off-card.

Another important aspect to be considered about bytecode verification and application downloading is, first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM3]), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

3.4.2 INTEGRITY AND AUTHENTICATION

Verification off-card is useless if the application CAP file is modified afterwards. The usage of cryptographic certifications coupled with the verifier in a secure module is a simple means to prevent any attempt of modification between CAP file verification and CAP file installation. Once a verification authority has verified the CAP file, it signs it and sends it to the card. Prior to the installation of the CAP file, the card verifies the signature of the CAP file, which authenticates the fact that it has been successfully verified. In addition to this, a secured communication channel is used to communicate it to the card, ensuring that no modification has been performed on it.

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

3.4.3 LINKING AND VERIFICATION

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

3.5 CARD MANAGEMENT

#.CARD-MANAGEMENT	(1) The card manager (CM) shall control the access to card management functions such as the installation, update or deletion of applets. (2) The card manager shall implement the card issuer’s policy on the card.
#.INSTALL	(1) The TOE must be able to return to a safe and consistent state when the installation of a CAP file or an applet fails or be cancelled (whatever the reasons). (2) Installing an applet must have no effect on the code and data of already installed applets. The installation procedure should not be used to

	bypass the TSFs. In short, it is an atomic operation, free of harmful effects on the state of the other applets. (3) The procedure of loading and installing a CAP file shall ensure its integrity and authenticity. In case of Extended CAP files, installation of a CAP shall ensure installation of all the packages in the CAP file.
#.SID	Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2.x). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a CAP file or any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.
#.OBJ-DELETION	(1) Deallocation of objects should not introduce security holes in the form of references pointing to memory zones that are no longer in use, or have been reused for other purposes. Deletion of collection of objects should not be maliciously used to circumvent the TSFs. (2) Erasure, if deemed successful, shall ensure that the deleted class instance is no longer accessible.
#.DELETION	(1) Deletion of installed applets (or CAP files) should not introduce security holes in the form of broken references to garbage collected code or data, nor should they alter integrity or confidentiality of remaining applets. The deletion procedure should not be maliciously used to bypass the TSFs. (2) Erasure, if deemed successful, shall ensure that any data owned by the deleted applet is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted applet cannot be selected or receive APDU commands. CAP file deletion shall make the code of the CAP file is no longer available for execution. In case of Extended CAP files, deletion of a CAP shall ensure that code and data

for all the packages in the CAP file is no longer available for execution. (3) Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the SFRs. This does not mandate, however, the process to be atomic. For instance, an interrupted deletion may result in the loss of user data, as long as it does not violate the SFRs.

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

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

3.6 SERVICES

#.ALARM	The TOE shall provide appropriate feedback upon detection of a potential security violation. This particularly concerns the type errors detected by the bytecode verifier, the security exceptions thrown by the Java Card VM, or any other security-related event occurring during the execution of a TSF.
#.OPERATE	(1) The TOE must ensure continued correct operation of its security functions. (2) In case of failure during its operation, the TOE must also return to a well-defined valid state before the next service request.
#.RESOURCES	The TOE controls the availability of resources for the applications in order to prevent unauthorized denial of service or malfunction of the TSFs. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of applications and CAP files.
#.CIPHER	The TOE shall provide a means to the applications for

	ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.
#.KEY-MNGT	The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified cryptographic key destruction methods.
#.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Protection of PIN's security attributes (state, try counter, try limit, …) in integrity.
#.SCP	The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by packages of Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is

	required that (7) the IC is designed in accordance with a well-defined set of policies and standards (for instance, those specified in [PP0084b]), 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 confidential application data such as cryptographic keys.
#.TRANSACTION	The TOE must provide a means to execute a set of operations atomically. This mechanism must not jeopardize the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

4 SECURITY PROBLEM DEFINITION

4.1 ASSETS

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.

4.1.1 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

Confidentiality - sensitive data of the applications, like the data contained in an object, an array view, a static field, 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, an array view and the PIN security attributes (PIN Try limit, PIN Try counter and State).

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.

4.1.2 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.

4.2 THREATS

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required.

4.2.1 CONFIDENTIALITY

T.CONFID-APPLI-DATA

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

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

T.CONFID-JCS-CODE

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

Directly threatened asset(s): D.JCS_CODE.

T.CONFID-JCS-DATA

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

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

4.2.2 INTEGRITY

T.INTEG-APPLI-CODE

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

Directly threatened asset(s): D.APP_CODE.

T.INTEG-APPLI-CODE.LOAD

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

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

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.

4.2.3 IDENTITY USURPATION

T.SID.1

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

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

T.SID.2

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

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

4.2.4 UNAUTHORIZED EXECUTION

T.EXE-CODE.1

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

Directly threatened asset(s): D.APP_CODE.

T.EXE-CODE.2

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

Directly threatened asset(s): D.APP_CODE.

T.NATIVE

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

Directly threatened asset(s): D.JCS_DATA.

4.2.5 DENIAL OF SERVICE

T.RESOURCES

An attacker prevents correct operation of the Java Card System through consumption of some resources of the card: RAM or NVRAM. See #.RESOURCES for details.

Directly threatened asset(s): D.JCS_DATA.

4.2.6 CARD MANAGEMENT

T.DELETION

The attacker deletes an applet or a CAP file 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).

4.2.7 SERVICES

T.OBJ-DELETION

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

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

4.2.8 MISCELLANEOUS

T.PHYSICAL

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

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

4.3 ORGANISATIONAL SECURITY POLICIES

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

OSP.VERIFICATION

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

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

4.4 ASSUMPTIONS

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

A.CAP_FILE

CAP Files 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.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.

Additional assumptions are taken from [PP0084b] which are listed below:

5 SECURITY OBJECTIVES

5.1 SECURITY OBJECTIVES FOR THE TOE

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

5.1.1 IDENTIFICATION

O.SID

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

5.1.2 EXECUTION

O.FIREWALL

The TOE shall ensure controlled sharing of data containers owned by applets of different CAP files 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 cleared 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 cleared after the return from the install method.

O.GLOBAL_ARRAYS_INTEG

The TOE shall ensure that no application can store a reference to the APDU buffer, a global byte array created by the user through makeGlobalArray method and the byte array used for invocation of the install method of the selected applet.

O.ARRAY_VIEWS_CONFID

The TOE shall ensure that no application can read elements of an array view not having array view security attribute ATTR_READABLE_VIEW.

The TOE shall ensure that an application can only read the elements of the array view within the bounds of the array view.

O. ARRAY_VIEWS_INTEG

The TOE shall ensure that no application can write to an array view not having array view security attribute ATTR_WRITABLE_VIEW.

The TOE shall ensure that an application can only write within the bounds of the array view.

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.

O.RESOURCES

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

5.1.3 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.RNG

The TOE shall ensure the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have sufficient entropy.

The TOE shall ensure that no information about the produced random numbers is available to an attacker since they might be used for instance to generate cryptographic keys.

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.

O.PIN-MNGT

The TOE shall provide a means to securely manage PIN objects (including the PIN try limit, PIN try counter and states). If the PIN try limit is reached, no further PIN authentication must be allowed.

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 limit and the try counter's value are as sensitive as that of the PIN and the TOE must restrict their modification only to authorized applications such as the card manager.

O.TRANSACTION

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

O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.RNG and O.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries can also be present on the card and made available to applets; those may be built on top of the Java Card API or independently. These proprietary libraries will be evaluated together with the TOE.

5.1.4 OBJECT DELETION

O.OBJ-DELETION

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

5.1.5 APPLET MANAGEMENT

O.DELETION

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

O.LOAD

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

Besides, for code loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application CAP file by the verification authority. This verification by the TOE shall occur during the loading or later during the install process.

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 CAP files 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).

Besides, for code loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application CAP file by the verification authority. If not performed during the loading process, this verification by the TOE shall occur during the install process.

5.1.6 CARD MANAGER

O.CARD-MANAGEMENT

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

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

5.1.7 SMART CARD PLATFORM

O.SCP.IC

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

This security objective for the TOE 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.

O.SCP.RECOVERY

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

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

the next power up to either complete the interrupted operation or revert to a secure state.

O.SCP.SUPPORT

The SCP shall support the TSFs of the TOE.

This security objective for the TOErefers to the security aspects 2, 3, 4 and 5 of #.SCP:

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

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

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

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

5.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

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

OE.CAP_FILE

No CAP file loaded post-issuance shall contain native methods.

OE.VERIFICATION

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

Additionally, the applet shall follow all the recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.

Application Note:

Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

OE.CODE-EVIDENCE

For application code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that loaded application has not been changed since the code verifications required in OE.VERIFICATION.

For application code loaded post-issuance and verified off-card according to the requirements of OE.VERIFICATION, the verification authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification.

For application code loaded post-issuance and partially or entirely verified on-card, technical measures must ensure that the verification required in OE.VERIFICATION are performed. On-card bytecode verifier is out of the scope of this Security Target.

Application Note:

For application code loaded post-issuance and verified off-card, the integrity and authenticity evidence can be achieved by electronic signature of the application code, after code verification, by the actor who performed verification.

Additional security objectives for operational environment are taken from [PP0084b] which are listed below:

5.3 SECURITY OBJECTIVES RATIONALE

5.3.1 THREATS

5.3.1.1 CONFIDENTIALITY

T.CONFID-APPLI-DATA

This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). 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 O.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, O.RNG). 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 security objective O.GLOBAL_ARRAYS_CONFID.

An applet might share data buffer with another applet using array views without the array view security attribute ATTR_READABLE_VIEW. The disclosure of data of the applet creating the array view is prevented by the security object O.ARRAY_VIEWS_CONFID.

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 objective O.NATIVE, so no application can be run to disclose a piece of code.

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

The objectives O.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 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 O.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.

5.3.1.2 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 objective O.NATIVE, so no application can 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 O.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 objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that integrity and authenticity evidences exist for the application code loaded into the platform.

T.INTEG-APPLI-CODE.LOAD

This threat is countered by the security objective O.LOAD which ensures that the loading of CAP files is done securely and thus preserves the integrity of CAP files' code.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective O.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-APPLI-DATA

This threat is countered by bytecode verification (OE.VERIFICATION) 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 O.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 objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. The objectives OE.SCP.RECOVERY and OE.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, O.RNG). 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 objective O.GLOBAL_ARRAYS_INTEG.

An applet might share data buffer with another applet using array views without the array view security attribute ATTR_WRITABLE_VIEW. The integrity of data of the applet creating the array view is ensured by the security objective O.ARRAY_VIEWS_INTEG.

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 CAP files is done securely and thus preserves the integrity of applications data.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the

application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective O.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 objective O.NATIVE, 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 O.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 objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.

T.INTEG-JCS-DATA

This threat is countered by bytecode verification (OE.VERIFICATION) 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 O.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 objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. 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.

5.3.1.3 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 objectives O.GLOBAL_ARRAYS_CONFID and O.GLOBAL_ARRAYS_INTEG.

The objective O.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.

5.3.1.4 UNAUTHORIZED EXECUTION

T.EXE-CODE.1

Unauthorized execution of a method is prevented by the objective OE.VERIFICATION. 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 objective OE.VERIFICATION. 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.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. OE.CAP_FILE also covers this threat by ensuring that no CAP files containing native code 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 method (OE.VERIFICATION).

5.3.1.5 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 Security Target, 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.

5.3.1.6 CARD MANAGEMENT

T.DELETION

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

The objective O.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 CAP file into the card is safe.

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

5.3.1.7 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.

5.3.1.8 MISCELLANEOUS

T.PHYSICAL

Covered by O.SCP.IC. Physical protections rely on the underlying platform and are therefore an environmental issue.

5.3.2 ORGANISATIONAL SECURITY POLICIES

OSP.VERIFICATION

This policy is upheld by the security objective of the environment OE.VERIFICATION 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.

This policy is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification, and by the security objective for the TOE O.LOAD which shall ensure that the loading of a CAP file into the card is safe.

5.3.3 ASSUMPTIONS

A.CAP_FILE

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

A.VERIFICATION

This assumption is upheld by the security objective on the operational environment OE.VERIFICATION 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.

This assumption is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

5.3.4 SPD AND SECURITY OBJECTIVES

Threats	Security Objectives	Rationale
T.CONFID-APPLI-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.ARRAY_VIEWS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.RNG, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION	Section 5.3.1
T.CONFID-JCS-CODE	OE.VERIFICATION, O.CARD-MANAGEMENT, O.NATIVE	Section 5.3.1
T.CONFID-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM	Section 5.3.1
T.INTEG-APPLI-CODE	O.CARD-MANAGEMENT, OE.VERIFICATION, O.NATIVE, OE.CODE-EVIDENCE	Section 5.3.1
T.INTEG-APPLI-CODE.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 5.3.1
T.INTEG-APPLI-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_INTEG, O.ARRAY_VIEWS_INTEG, O.ALARM, O.TRANSACTION, O.CIPHER, O.RNG, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION, OE.CODE-EVIDENCE,	Section 5.3.1
T.INTEG-APPLI-DATA.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 5.3.1
T.INTEG-JCS-CODE	O.CARD-MANAGEMENT, OE.VERIFICATION, O.NATIVE, OE.CODE-EVIDENCE	Section 5.3.1
T.INTEG-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, OE.CODE-EVIDENCE	Section 5.3.1
T.SID.1	O.CARD-MANAGEMENT, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID,	Section 5.3.1

	O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID	
T.SID.2	O.SCP.RECOVERY, O.SCP.SUPPORT, O.SID, O.OPERATE, O.FIREWALL, O.INSTALL	Section 5.3.1
T.EXE-CODE.1	OE.VERIFICATION, O.FIREWALL	Section 5.3.1
T.EXE-CODE.2	OE.VERIFICATION	Section 5.3.1
T.NATIVE	OE.VERIFICATION, OE.CAP_FILE , O.NATIVE	Section 5.3.1
T.RESOURCES	O.INSTALL, O.OPERATE, O.RESOURCES, O.SCP.RECOVERY, O.SCP.SUPPORT	Section 5.3.1
T.DELETION	O.DELETION, O.CARD-MANAGEMENT	Section 5.3.1
T.INSTALL	O.INSTALL, O.LOAD, O.CARD-MANAGEMENT	Section 5.3.1
T.OBJ-DELETION	O.OBJ-DELETION	Section 5.3.1
T.PHYSICAL	O.SCP.IC	Section 5.3.1

Table 1 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.ARRAY_VIEWS_CONFID	T.CONFID-APPLI-DATA
O.ARRAY_VIEWS_INTEG	T.INTEG-APPLI-DATA
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.RNG	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.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
OE.CAP_FILE	T.NATIVE
O.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
O.SCP.IC	T.PHYSICAL
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.SUPPORT	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
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
OE.CODE-EVIDENCE	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

Table 2 Security Objectives and Threats - Coverage

Organizational Security Policies	Security Objectives	Rationale
OSP.VERIFICATION	OE.VERIFICATION, O.LOAD, OE.CODE-EVIDENCE	Section 5.3.2

Table 3 OSPs and Security Objectives - Coverage

Security Objectives	Organizational Security Policies
O.SID	

O.FIREWALL	
O.GLOBAL_ARRAYS_CONFID	
O.GLOBAL_ARRAYS_INTEG	
O.ARRAY_VIEWS_CONFID	
O.ARRAY_VIEWS_INTEG	
O.NATIVE	
O.OPERATE	
O.REALLOCATION	
O.RESOURCES	
O.ALARM	
O.CIPHER	
O.RNG	
O.KEY-MNGT	
O.PIN-MNGT	
O.TRANSACTION	
O.OBJ-DELETION	
O.DELETION	
O.LOAD	OSP.VERIFICATION
O.INSTALL	
OE.CAP_FILE	
O.CARD-MANAGEMENT	
O.SCP.IC	
O.SCP.RECOVERY	
O.SCP.SUPPORT	
OE.VERIFICATION	OSP.VERIFICATION
OE.CODE-EVIDENCE	OSP.VERIFICATION

Table 4 Security Objectives and OSPs - Coverage

Assumptions	**Security Objectives for the Operational Environment**	**Rationale**
A.CAP_FILE	OE.CAP_FILE	Section 5.3.3
A.VERIFICATION	OE.VERIFICATION, OE.CODE-EVIDENCE	Section 5.3.3

Table 5 Assumptions and Security Objectives for the Operational Environment - Coverage

Security Objectives for the Operational Environment	**Assumptions**
OE. CAP_FILE	A.CAP_FILE
OE.VERIFICATION	A.VERIFICATION
OE.CODE-EVIDENCE	A.VERIFICATION

Table 6 Security Objectives for the Operational Environment and Assumptions – Coverage

5.3.5 COMPATIBILITY BETWEEN OBJECTIVES OF TOE AND [THD89]

Objectives including O.SID, O.OPERATE, O.RESOURCES, O.FIREWALL, O.NATIVE, O.REALLOCATION, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ARRAY_VIEWS_CONFID, O. ARRAY_VIEWS_INTEG, O.ALARM, O.TRANSACTION, O.PIN-MNGT, O.KEY-MNGT, O.OBJ-DELETION, O.INSTALL, O.LOAD, O.DELETION, are defined for the Java Card System, hence they do not conflict with the objectives of [THD89].

O.CIPHER is consistent with O.RND, O.TDES, O.ECC, O.RSA-CRT and O.AES which are defined in [THD89].

O.SCP.IC is consistent with O.Phys-Manipulation, O.Phys-Probing, O.Malfunction, O.Leak-Inherent, O.Leak-Forced, O.Abuse-Func which are defined in [THD89].

Therefore the objectives for the TOE and [THD89] are consistent.

The following table shows the compatibility between the objectives for the operational environment of the TOE and [THD89], and they are separated into three groups:

- IrOE: The objectives for the environment being not relevant for the Composite-ST, e.g. the objectives for the environment about the developing and manufacturing phases of the platform.
- CfPOE: The objectives for the environment being fulfilled by the Composite-ST automatically. Such objectives of the environment of the Platform-ST can always be assigned to the TOE security objectives of the Composite-ST. Due to this fact they will be fulfilled either by the Composite-SFR or by the Composite-SAR automatically.
- SgOE: The remaining Objectives for the environment of the Platform-ST belonging neither to the group IrOE nor CfOE Exactly this group makes up the significant objectives for the environment for the Composite-ST, which shall be addressed in the Composite-ST.

Objectives from [THD89]	IrOE	CfPOE	SgOE
OE.Resp-Appl		X	
OE.Process-Sec-IC			X

Therefore the objectives of the operational environment for the TOE and [THD89] are consistent.

6 EXTENDED COMPONENTS DEFINITION

6.1 DEFINITION OF THE FAMILY FCS_RNG

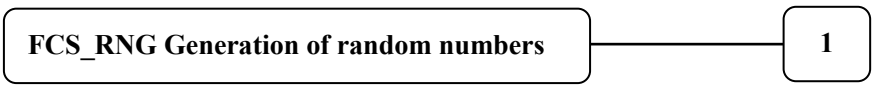
To define the IT security functional requirements of the TOE an additional family (FCS_RNG) of the Class FCS (cryptographic support) is defined here. This family describes the functional requirements for random number generation used for cryptographic purposes.

FCS_RNG Generation of random numbers

Family behavior

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

Component levelling:



FCS_RNG.1	Generation of random numbers requires that random numbers meet a defined quality metric.
Management:	FCS_RNG.1 There are no management activities foreseen.
Audit:	FCS_RNG.1 There are no actions defined to be auditable.
FCS_RNG.1	**Random number generation**
Hierarchical to:	No other components
Dependencies:	No dependencies
FCS_RNG.1.1	The TSF shall provide a [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic] random number generator [selection: DRG.2, DRG.3, DRG.4, PTG.2, PTG.3, NTG.1] [AIS20] [AIS31] that implements: [assignment: list of security capabilities].
FCS_RNG.1.2	The TSF shall provide random numbers that meet [assignment: a defined quality metric].

7 SECURITY REQUIREMENTS

7.1 SECURITY FUNCTIONAL REQUIREMENTS

This section states the security functional requirements for the [PP JCS].

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.
Installation (InstG)	The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.
Applet deletion (ADELG)	The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 2.2.
Object deletion (ODELG)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.
Secure carrier (CarG)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, the installation of a CAP file that has not been bytecode verified, or that has been modified after bytecode verification.
Smart Card Platform (SCPG)	The SCPG group contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon.
Card Manager (CMGRG)	The CMGRG group contains the security requirements for the card manager.

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

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

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

Subject	Description

S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE22], §11), but its role asks anyway for a specific treatment from the security viewpoint.
S.APPLET	Any applet instances.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the CAP files.
S.CAD	The CAD represents off-card entity that communicates with the S.INSTALLER. If the TOE provides JCRMI functionality, CAD can request RMI services by issuing commands to the card.
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 CAP files 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.CAP_FILE	A CAP file may contain multiple Java language packages. A package is a namespace within the Java programming language that may contain classes and interfaces. A CAP file may contain packages that define either user library, or one or several applets. A CAP file compliant with Java Card Specifications version 3.1 may contain multiple Java language packages. An EXTENDED CAP file as specified in Java Card Specifications version 3.1 may contain only applet packages, only library packages or a combination of library packages. A COMPACT CAP file as specified in Java Card Specifications version 3.1 or CAP files compliant to previous versions of Java Card Specification, MUST contain only a single package representing a library or one or more applets.
S.CM	Card Manager which is a generic term for the card management entities of a GlobalPlatform card as defined in [GP].

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

Object	**Description**
O.APPLET	Any installed applet, its code and data.
O.CODE_CAP_FILE	The code of a CAP file, including all linking information. On the Java Card platform, a CAP file 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.GP_KEY	Keys are used for token and DAP verification and generation as well as secure channel protocol as defined in [GP].

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

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

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

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 indicated in the export file.
CAP File AID	The AID of a CAP file.
Context	CAP file AID or "Java Card RE".
Currently Active Context	CAP file AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the package AID and Applet's version number ([JCVM22], §4.5.2).
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 CAP file (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the CAP file). 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.
Resident CAP files	The set of AIDs of the CAP files already loaded on the card.
Resident packages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	CAP file AID or "None".
Sharing	Standard, SIO, Array View, Java Card RE entry point or global array.
Static References	Static fields of a CAP file may contain references to objects. The

	Static References attribute record those references.
Security Level	Security level for secure messaging as defined in [GP].
Privilege	The privilege assigned to on card entity as defined in [GP].
Life Cycle Status	Life cycle status associated with on card entity as defined in [GP].
DAP	Data authentication pattern for cap file as defined in [GP].
MAC	Message authentication code as defined in [GP].
GP key	Keys as defined in [GP].
DM token	Delegated Management token as defined in [GP].
Key type	Key type identifier as defined in [GP].
Key checksum	Checksum used to verify the integrity of key values as defined in [GP].

(*) 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.ARRAY_LENGTH (O.JAVAOBJECT, field)	Get length of an array component.
OP.ARRAY_T_ALOAD(O.JAVAOBJECT, field)	Read from an array component
OP.ARRAY_T_ASTORE(O.JAVAOBJECT, field)	Write to an array component
OP.ARRAY_AASTORE(O.JAVAOBJECT, field)	Store into reference array component
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new, makeTransient or createArrayView call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_CAP_FILE(O.CODE_CAP_FILE,...)	Delete a CAP file, either logically or physically.
OP.DELETE_CAP_FILE_APPLET(O.CODE_CAP_FILE,...)	Delete a CAP file and its installed applets, either logically or physically.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an

	instance of a class in the Java programming language.
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.JAVA(...)	Any access in the sense of [JCRE3], §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.ARRAY_LENGTH
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE3], §6.2.8.7).
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instance of on an object in order to access to classes (standard or shareable interfaces objects).
OP.CCM(...)	The Card Content Management as defined in [GP]. It stands for one of the operations OP.LOAD, OP.INSTALL, OP.DELETE, OP.PUT_KEY.

(*) 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.

The following table lists all the security functional requirements of the TOE from [PP JCS]. They apply to this Security Target with some modification (i.e. operation performed or replaced).

The notation used is detailed below:

- Equivalent (E): The element in the ST is the same as in [PP JCS].
- Addition (A): The element is newly defined in the ST; it is not present in [PP JCS] and does not affect it.
- X The element is present in [PP JCS].

Security Functional requirements	[PP JCS]	Security Target
FDP_ACC.2/FIREWALL	X	(E)
FDP_ACF.1/FIREWALL	X	(E)
FDP_IFC.1/JCVM	X	(E)
FDP_IFF.1/JCVM	X	(E)
FDP_RIP.1/OBJECTS	X	(E)
FMT_MSA.1/JCRE	X	(E)
FMT_MSA.1/JCVM	X	(E)
FMT_MSA.2/FIREWALL_JCVM	X	(E)
FMT_MSA.3/FIREWALL	X	(E)
FMT_MSA.3/JCVM	X	(E)
FMT_SMF.1	X	(E)
FMT_SMR.1	X	(E)
FCS_CKM.1	X	(E)
FCS_CKM.4	X	(E)
FCS_COP.1	X	(E)
FCS_RNG.1	X	(E)
FDP_RIP.1/ABORT	X	(E)
FDP_RIP.1/APDU	X	(E)
FDP_RIP.1/bArray	X	(E)
FDP_RIP.1/GlobalArray	X	(E)
FDP_RIP.1/KEYS	X	(E)
FDP_RIP.1/TRANSIENT	X	(E)
FDP_ROL.1/FIREWALL	X	(E)
FAU_ARP.1	X	(E)
FDP_SDI.2/DATA	X	(E)
FPR_UNO.1	X	(E)
FPT_FLS.1	X	(E)
FPT_TDC.1	X	(E)
FIA_ATD.1/AID	X	(E)
FIA_UID.2/AID	X	(E)
FIA_USB.1/AID	X	(E)
FMT_MTD.1/JCRE	X	(E)
FMT_MTD.3/JCRE	X	(E)
FDP_ITC.2/Installer	X	(E)

FMT_SMR.1/Installer	X	(E)
FPT_FLS.1/Installer	X	(E)
FPT_RCV.3/Installer	X	(E)
FDP_ACC.2/ADEL	X	(E)
FDP_ACF.1/ADEL	X	(E)
FDP_RIP.1/ADEL	X	(E)
FMT_MSA.1/ADEL	X	(E)
FMT_MSA.3/ADEL	X	(E)
FMT_SMF.1/ADEL	X	(E)
FMT_SMR.1/ADEL	X	(E)
FPT_FLS.1/ADEL	X	(E)
FDP_RIP.1/ODEL	X	(E)
FPT_FLS.1/ODEL	X	(E)
FCO_NRO.2/CM	X	(E)
FDP_IFC.2/CM	X	(E)
FDP_IFF.1/CM	X	(E)
FDP_UIT.1/CM	X	(E)
FIA_UID.1/CM	X	(E)
FMT_MSA.1/CM	X	(E)
FMT_MSA.3/CM	X	(E)
FMT_SMF.1/CM	X	(E)
FMT_SMR.1/CM	X	(E)
FTP_ITC.1/CM	X	(E)
FDP_ACC.1/CMGR	X	(A)
FDP_ACF.1/CMGR	X	(A)
FMT_MSA.1/CMGR	X	(A)
FMT_MSA.3/CMGR	X	(A)
FPT_RCV.4	X	(A)
FPT_TST.1/SCP	X	(A)
FPT_PHP.3/SCP	X	(A)

7.1.1 COREG_LC SECURITY FUNCTIONAL REQUIREMENTS

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

7.1.1.1 FIREWALL POLICY

FDP_ACC.2/FIREWALL Complete access control

FDP_ACC.2.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** on **S.CAP_FILE, 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.
- OP.ARRAY_LENGTH
- OP.ARRAY_T_ALOAD
- OP.ARRAY_T_ASTORE
- OP.ARRAY_AASTORE

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.CAP_FILE	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 ([JCRE3], §6.2.8): S.CAP_FILE may freely perform, 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 ([JCRE3], §6.2.8): S.CAP_FILE 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 ([JCRE3], §6.2.8.10): S. CAP_FILE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT with Context attribute different from the currently active context, 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 ([JCRE3], §6.2.8.6): S.CAP_FILE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT with Context attribute different from the currently active context, whose Sharing attribute has the value "SIO", and whose Context attribute has the value "CAP File 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 CAP file whose AID is "CAP File AID" is "Multiselectable",
  b) The value of the attribute Selection Status of the CAP file whose AID is "CAP File AID" is "Non-multiselectable", and either "CAP File AID" is the value of the currently selected applet or otherwise "CAP File AID" does not occur in the attribute Active Applets.
- R.JAVA.5: S.CAP_FILE may perform OP.CREATE upon O.JAVAOBJECT only if the value of the Sharing parameter is "Standard" or “SIO”.
- R.JAVA.6 ([JCRE3], §6.2.8): S.CAP_FILE may freely perform OP.ARRAY_ACCESS or OP.ARRAY_LENGTH upon any O.JAVAOBJECT whose Sharing attribute has value "global array".

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

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

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

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

3) S.CAP_FILE performing OP.ARRAY_AASTORE of the reference of an O.JAVAOBJECT whose sharing attribute has value “global array” or “Temporary”.
4) S.CAP_FILE performing OP.PUTFIELD or OP.PUTSTATIC of the reference of an O.JAVAOBJECT whose sharing attribute has value “global array” or “Temporary”
5) R.JAVA.7 ([JCRE3], §6.2.8.2): S.CAP_FILE performing OP.ARRAY_T_ASTORE into an array view without ATTR_WRITABLE_VIEW access attribute.
6) R.JAVA.8 ([JCRE3], §6.2.8.2):S.CAP_FILE performing OP.ARRAY_T_ALOAD into an array view without ATTR_READABLE_VIEW access attribute.

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 five 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.
- Array Views, having fields/elements access controlled by access control attributes, ATTR_READABLE_VIEW and ATTR_WRITABLE_VIEW 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 ([JCRE3], §6.1.3). An object is owned by an applet instance, by the JCRE or by the library where it has been defined (these latter objects can only be arrays that initialize static fields of CAP files).

([JCRE3], 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 (CAP file AID) of the selected applet. In this policy,

the "Selected Applet Context" is the AID of the selected CAP file.

([JCRE3], §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 CAP File" 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.x.x 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 CAP file being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same CAP file are either all multiselectable or not ([JCVM3], §2.2.5). Therefore, the selection mode can be regarded as an attribute of CAP files. No selection mode is defined for a library CAP file.

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 ([JCRE3], §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 information flow control SFP rules**.

FDP_IFF.1.4/JCVM The TSF shall explicitly authorize an information flow based on the following rules: **none**.

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **none**.

Application Note:
The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE3], §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 [JCRE3], §4 and [JCVM3], §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 [JCRE3], §4 and [JCVM3], §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.
- 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 initialization

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

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

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 ([JCRE3], §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 initialization

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

FMT_MSA.3.2/JCVM [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.

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.

7.1.1.2 APPLICATION PROGRAMMING INTERFACE

The following SFRs are related to the Java Card API.

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

FCS_CKM.1 Cryptographic key generation

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

Iteration	Algorithm	Key size	Standards
ECC Key Pair	ECC	ansix9p224r1, ansix9p256r1, ansix9p384r1, ansix9p521r1, brainpoolP224r1, brainpoolP256r1 and brainpoolP384r1 curves	ANSI X9.62-2005, RFC 5639
SCP02 session key	TDES in CBC mode	128 bits	[GP]
SCP03 session key	KDF in counter mode	128 bits	[GP]

Application Note:
The keys can be generated and diversified in accordance with [JCAPI3] specification in classes KeyBuilder, KeyPair (at least Session key generation) and RandomData

FCS_CKM.4 Cryptographic key destruction

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

Application Note:

- The keys are reset as specified in [JCAPI3] Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing shall throw an exception.

FCS_COP.1 Cryptographic operation

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

Iteration	Operation	Algorithm	Key Size	Standards
TDES	Encryption and Decryption	ALG_DES_CBC_ISO9797_M1 ALG_DES_CBC_ISO9797_M2 ALG_DES_CBC_NOPAD ALG_DES_ECB_ISO9797_M1 ALG_DES_ECB_ISO9797_M2 ALG_DES_ECB_NOPAD ALG_DES_CBC_PKCS5 ALG_DES_ECB_PKCS5	LENGTH_DES3_2KEY,	NIST-SP800-67 ISO9797-1 DES NOPAD DES PKCS#5 DES 9797 M1 M2
DESMAC	Generation and Verification	ALG_DES_MAC4_ISO9797_1_M1_ALG3 ALG_DES_MAC4_ISO9797_1_M2_ALG3 ALG_DES_MAC4_ISO9797_M1 ALG_DES_MAC4_ISO9797_M2 ALG_DES_MAC8_ISO9797_1_M1_ALG3 ALG_DES_MAC8_ISO9797_1_M2_ALG3 ALG_DES_MAC8_ISO9797_M1 ALG_DES_MAC8_ISO9797_M2 ALG_DES_MAC8_NOPAD ALG_DES_MAC4_PKCS5 ALG_DES_MAC8_PKCS5	LENGTH_DES3_2KEY,	[JCAPI3]
RSA-CRT Signature PKCS1	Generation	ALG_RSA_SHA_224_PKCS1 ALG_RSA_SHA_224_PKCS1_PSS ALG_RSA_SHA_256_PKCS1 ALG_RSA_SHA_256_PKCS1_PSS ALG_RSA_SHA_384_PKCS1 ALG_RSA_SHA_384_PKCS1 ALG_RSA_SHA_512_PKCS1 ALG_RSA_SHA_512_PKCS1_PSS ALG_RSA_SHA_ISO9796 ALG_RSA_SHA_ISO9796_MR ALG_RSA_SHA_PKCS1 ALG_RSA_SHA_PKCS1_PSS	1024, 2048, 4096	[JCAPI3]
ECDSA	Signature and Verification	ALG_ECDSA_SHA ALG_ECDSA_SHA_224 ALG_ECDSA_SHA_256 ALG_ECDSA_SHA_384 ALG_ECDSA_SHA_512	224, 256, 384 and 521 bits	[JCAPI3]

AES CBC/ECB	Encryption and Decryption	ALG_AES_BLOCK_128_CBC_NOPAD ALG_AES_BLOCK_128_ECB_NOPAD ALG_AES_CBC_ISO9797_M1 ALG_AES_CBC_ISO9797_M2 ALG_AES_CBC_PKCS5 ALG_AES_ECB_ISO9797_M1 ALG_AES_ECB_ISO9797_M2 ALG_AES_ECB_PKCS5	128, 192, 256 bits	[JCAPI3]
AES MAC	Signature and Verification	ALG_AES_CMAC_128	128, 192, 256 bits	[JCAPI3]
AES CMAC	Signature and Verification	ALG_AES_MAC_128_NOPAD	128, 192, 256 bits	[JCAPI3]

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 [JCRE3], §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/GlobalArray Subset residual information protection

FDP_RIP.1.1/GlobalArray [Refined] The TSF shall ensure that any previous information content of a resource is made unavailable upon **deallocation of the resource from** the applet as a result of returning from the process method to the following objects: **a user Global Array**.

Application Note:
An array resource is allocated when a call to the API method JCSystem.makeGlobalArray is performed. The Global Array is created as a transient JCRE Entry Point Object ensuring that reference to it cannot be retained by any application. On return from the method which called

JCSystem.makeGlobalArray, the array is no longer available to any applet and is deleted and the memory in use by the array is cleared and reclaimed in the next object deletion cycle.

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 [JCAPI3].

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 [JCRE3], §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 CAP file must share the transient memory segment if they are concurrently active ([JCRE3], §4.3.

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 **object O.JAVAOBJECT**.

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 [JCRE3], §7.7, within the bounds of the Commit Capacity ([JCRE3], §7.8), and those described in [JCAPI3]**.

Application Note:
Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism 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 [JCAPI3] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

7.1.1.3 CARD SECURITY MANAGEMENT

FAU_ARP.1 Security alarms

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

- **throw an exception,**
- **lock the card session,**
- **reinitialize the Java Card System and its data.**
- **none**

upon detection of a potential security violation.

Refinement:
The "potential security violation" stands for one of the following events:

- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card manager life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context, (see abortTransaction(), [JCAPI3] and ([JCRE3], §7.6.2)
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow.

Application Note:

- The developer shall provide the exhaustive list of actual potential security violations the TOE reacts to. For instance, other runtime errors related to applet's failure like uncaught exceptions.

- The bytecode verification defines a large set of rules used to detect a "potential security violation". The actual monitoring of these "events" within the TOE only makes sense when the bytecode verification is performed on-card.

- Depending on the context of use and the required security level, there are cases where the card manager and the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behavior must be described in this component. It shall detail the nature of the feedback information provided to the card manager (like the identity of the offending application) and the conditions under which the feedback will occur (any occurrence of the java.lang.SecurityException exception).

- The "locking of the card session" may not appear in the policy of the card manager. Such measure should only be taken in case of severe violation detection; the same holds for the re-initialization of the Java Card System. Moreover, the locking should occur when "clean" re-initialization seems to be impossible.

- The locking may be implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alerts in this case should also be carefully considered herein.

FDP_SDI.2/DATA Stored data integrity monitoring and action

FDP_SDI.2.1/DATA The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **symmetric keys, RSA private key, RSA-CRT key, ECC private key, PIN.**

FDP_SDI.2.2/DATA Upon detection of a data integrity error, the TSF shall **throw an exception**.

Application Note:

- Although no such requirement is mandatory in the Java Card specification, at least an exception shall be raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects shall be considered with particular attention by ST authors as they play a key role in the overall security.
- It is also recommended to monitor integrity errors in the code of the native applications and Java Card applets.

For integrity sensitive application, their data shall be monitored (D.APP_I_DATA):

applications may need to protect information against unexpected modifications, and explicitly control whether a piece of information has been changed between two accesses. For example, maintaining the integrity of an electronic purse's balance is extremely important because this value represents real money. Its modification must be controlled, for illegal ones would denote an important failure of the payment system.

- A dedicated library could be implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs, for instance.

FPR_UNO.1 Unobservability

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

Application Note:
The non-observability of operations on sensitive information such as keys appears as impossible to circumvent in the smart card world. The precise list of operations and objects is left unspecified, but should at least concern secret keys and PIN values when they exist on the card, as well as the cryptographic operations and comparisons performed on them.

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 ([JCRE3], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE3]). Behavior of the TOE on power loss and reset is described in [JCRE3], §3.6 and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE3], §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 [JCVM3] specification,**
- **the API tokens defined in the export files of reference implementation,**
- **The rules defined in [GP] specification**

when interpreting the TSF data from another trusted IT product.

Application Note:
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, including memory management, I/O functions and cryptographic functions.

7.1.1.4 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:

- **CAP File AID,**
- **Package AID,**
- **Applet's version number,**
- **Registered applet AID,**
- **Applet Selection Status.**

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 CAP files (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 CAP file that is the subject's owner. Means of identification are provided during the loading procedure of the CAP file 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 to the TOE, but it is 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: **CAP file AID**.

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user

security attributes with subjects acting on the behalf of users: **rules for the initial selection of applet defined in [JCRE3]§4**.

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: **rules for the applet selection defined in [JCRE3]§4**.

Application Note:
The user is the applet and the subject is the S.CAP_FILE. The subject security attribute "Context" shall hold the user security attribute "CAP file 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**.

7.1.2 INSTG SECURITY FUNCTIONAL REQUIREMENTS

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

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

FDP_ITC.2.1/Installer The TSF shall enforce **the CAP FILE 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:

CAP file loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major version attribute associated to the dependent package file is equal to the major version attribute of the resident package and the minor version attribute is equal to or less than the minor version attribute associated to the resident package ([JCVM3], §4.5.2).

Application Note:

FDP_ITC.2.1/Installer:

- The most common importation of user data is CAP file 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 CAP file and the package or packages contained within the CAP file, 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 [JCVM3] 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 CAP file and all the packages contained within a CAP file contain a Version attribute, which is a pair of major and minor version numbers ([JCVM3], §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 version numbers and AIDs of imported packages indicated in the export file are recorded in the CAP files ([JCVM3], §4.5.2): the dependent packages' Version and AID attributes allow the retrieval of these identifications. Implementation-dependent checks may occur on a case-by-case basis to check that packages are binary compatible. Packages have "package Version Numbers" ([JCVM3]) that indicate binary compatibility or incompatibility between successive implementations of a package, which 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 ([JCVM3], §4.4).
- The installation (the invocation of an applet's install method by the installer) is implementation dependent ([JCRE3], §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, [JCRE3], §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 CAP file/applet as described in [JCRE3] §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 **none** 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 **failures during load/installation of a package/applet and deletion of a package/applet/object**, 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 capacity of transaction buffer** 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 [CC2], p298: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorized 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 CAP file/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE3], §11.1.5 for possible scenarios. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a CAP file/applet. See ([JCRE3], 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 [PP0084b]) 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.

7.1.3 ADELG SECURITY FUNCTIONAL REQUIREMENTS

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

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_CAP_FILE** and all operations among subjects and objects covered by the SFP.

Refinement:
The operations involved in the policy are:

- OP.DELETE_APPLET,
- OP.DELETE_CAP_FILE,
- OP.DELETE_CAP_FILE_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 CAP files
O.CODE_CAP_FILE	CAP file AID, AIDs of packages within a CAP file, 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:

- ✓ **the owner of O is a registered applet instance A (O is reachable from A),**
- ✓ **a static field of a resident package P contains a reference to O (O is reachable from P),**
- ✓ **there exists a valid remote reference to O (O is remote reachable),**
- ✓ **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 ([JCRE3], §11.3.4.29, Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,**

  - ✓ **S.ADEL is currently selected,**
  - ✓ **there is no instance in the context of O.APPLET that is active in any logical channel and**
  - ✓ **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 ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.**

- **R.JAVA.15 ([JCRE3], §11.3.4.2.110, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,**

  - ✓ **S.ADEL is currently selected,**
  - ✓ **there is no instance of any of the O.APPLET being deleted that is active in any logical channel and**
  - ✓ **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 CAP file P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.**

- **R.JAVA.16 ([JCRE3], §11.3.4.311, Applet/Library CAP file Deletion): S.ADEL may perform OP.DELETE_CAP_FILE upon an O.CODE_CAP_FILE only if,**

  - ✓ **S.ADEL is currently selected,**
  - ✓ **no reachable O.JAVAOBJECT, from a CAP file distinct from O.CODE_CAP_FILE that is an instance of a class that belongs to O.CODE_CAP_FILE, exists on the card and**
  - ✓ **there is no resident package on the card that depends on O.CODE_CAP_FILE.**

- **R.JAVA.17 ([JCRE3], §11.3.4.412, Applet CAP file and Contained Instances Deletion): S.ADEL may perform OP.DELETE_CAP_FILE_APPLET upon an O.CODE_CAP_FILE only if,**

  - ✓ **S.ADEL is currently selected,**
  - ✓ **no reachable O.JAVAOBJECT, from a CAP file distinct from O.CODE_CAP_FILE, which is an instance of a class that belongs to O.CODE_CAP_FILE exists on the card,**
  - ✓ **there is no CAP file loaded on the card that depends on O.CODE_CAP_FILE, and**
  - ✓ **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 CAP file not being deleted, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.

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

FDP_ACF.1.4/ADEL The TSF shall explicitly deny access of subjects to objects based on the following additional rules:
any subject but S.ADEL to O.CODE_CAP_FILE 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 CAP file.
- S.ADEL calls the "uninstall" method of the applet instance to be deleted, if implemented by the applet, to inform it of the deletion request. The order in which these calls and the dependencies checks are performed are out of the scope of this Security Target.

FDP_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 CAP files 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/CAP file deletion are described in [JCRE3], §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 CAP files** to **the Java Card RE**.

FMT_MSA.3/ADEL Static attribute initialization

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

FMT_MSA.3.2/ADEL The TSF shall allow the **following role(s): none,** to specify alternative

initial values to override the default values when an object or information is created.

FMT_SMF.1/ADEL Specification of Management Functions

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

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 CAP file/applet as described in [JCRE3], §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 CAP file/applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with Java Card specification ([JCRE3], §11.3.4.)

7.1.4 ODELG SECURITY FUNCTIONAL REQUIREMENTS

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

FDP_RIP.1/ODEL Subset residual information protection

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

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 [JCAPI3].
- 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.

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).

7.1.5 CARG SECURITY FUNCTIONAL REQUIREMENTS

This group includes requirements for preventing the installation of CAP files 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 CAP files** at all times.

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

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **limitations that no evidence is retained on the card for future verifications once the evidence is verified**.

Application Note:
FCO_NRO.2.1/CM:

- Upon reception of a new application CAP file for installation, the card manager shall first check that it actually comes from the verification authority and represented by the subject S.BCV. The verification authority is indeed 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 CAP file 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 **CAP FILE 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 CAP file 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 CAP file 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 **CAP FILE LOADING information flow control SFP** based on the following types of subject and information security attributes:

Subject / Information	Security Attribute
S.CAD	Secure channel keys, DAP key.
S.INSTALLER	Secure channel keys, DAP key, security level.
I.APDU	MAC, DAP block, sequence number.

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

- the subject S.INSTALLER shall accept a message conveyed in I.APDU only if the subject S.CAD is authenticated during secure channel establishment,
- the subject S.INSTALLER shall accept an application CAP file only if the MAC and DAP block associated with I.APDU is verified and the sequence number in all the I.APDUs sent by the subject S.CAD is checked in order.

FDP_IFF.1.3/CM The TSF shall enforce the **[assignment: none]**.

FDP_IFF.1.4/CM The TSF shall explicitly authorize an information flow based on the following rules: **none**.

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

- **The TOE fails to verify the integrity and authenticity evidences of the application CAP file.**
- **CAP file which is not verified by off-card S.BCV**.

Application Note:

FDP_IFF.1.1/CM:

- The security attributes used to enforce the CAP FILE 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 CAP file has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the CAP file, 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 CAP file only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD.

FDP_IFF.1.5/CM:

- The verification of the integrity and authenticity evidences can be performed either during loading or during the first installation of an application of the CAP file.

FDP_UIT.1/CM Data exchange integrity

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

FDP_UIT.1.2/CM [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 CAP file 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 **applet selection, APDU dispatch and initiation of secure channel** 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:
The list of TSF-mediated actions is implementation-dependent, but CAP file installation requires the user to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/CM.

FMT_MSA.1/CM Management of security attributes

FMT_MSA.1.1/CM The TSF shall enforce the **CAP FILE LOADING information flow control SFP** to restrict the ability to **change, modify and delete** the security attributes **CAP file AID** to **none**.

FMT_MSA.3/CM Static attribute initialization

FMT_MSA.3.1/CM The TSF shall enforce the **CAP FILE 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: **Card Content Management consisting of loading, installing, extradition and deletion by Card Manager**.

FMT_SMR.1/CM Security roles

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

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

FTP_ITC.1.2/CM [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 CAP file on the card**.

Application Note:
There is no dynamic CAP file loading on the Java Card platform. New CAP files can be installed on the card only on demand of the card issuer.

7.1.6 SCPG SECURITY FUNCTIONAL REQUIREMENTS

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

FPT_TST.1/SCP TSF Testing

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

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

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

FPT_PHP.3/SCP Resistance to physical attacks

FPT_PHP.3.1/SCP The TSF shall resist **physical manipulation and physical probing** to the **all TOE components implementing the TSF** by responding automatically such that the SFRs are always enforced.

FPT_RCV.4/SCP Function recovery

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

FCS_RNG.1 Random number generation

FCS_RNG.1.1 The TSF shall provide a **physical** random number generator **PTG.2** [AIS20] [AIS31] that implements:

- **A total failure test detects a total failure of entropy source immediately when the RNG has started. When a total failure is detected, no random numbers will be output.**
- **If a total failure of the entropy source occurs while the RNG is being operated, the RNG prevents the output of any internal random number that depends on some raw random numbers that have been generated after the total failure of the entropy source.**
- **The online test shall detect non-tolerable statistical defects of the raw random number sequence (i) immediately when the RNG has started. And (ii) while the RNG is being operated. The TSF must not output any random numbers before the power-up online test has finished successfully or when a defect has been detected.**
- **The online test procedure shall be effective to detect non-tolerable weakness of the random numbers soon.**
- **The online test procedure checks the quality of the raw random number sequence. It is triggered applied upon specified internal events. The online test is suitable for detecting non-tolerable statistical defects of the statistical properties of the raw random numbers within an acceptable period of time.**

FCS_RNG.1.2 The TSF shall provide **random numbers** that meet:

- **32 bits random number words.**
- **Test procedure A and no other test suites does not distinguish the internal random numbers from output sequences of an ideal RNG.**
- **The average Shannon entropy per internal random bit exceeds 0.997.**

7.1.7 CMGRG SECURITY FUNCTIONAL REQUIREMENTS

This group includes requirements for Card Manager.

FDP_ACC.1/CMGR Subset access control

FDP_ACC.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP on S.CM, S.CAD, O.APPLET, O.CODE_CAP_FILE, O.GP_KEY and operations among subjects and objects covered by the SFP**.

Refinement:
The operations involved in the policy are:

- OP.LOAD,
- OP.INSTALL,
- OP.DELETE,
- OP.PUT_KEY.

FDP_ACF.1/CMGR Security attribute-based access control

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

Subject/Object	Security attributes
S.CM	Card Life Cycle, Life Cycle Status, Privileges, Registered Applets, Active Applets, Applet Selection Status, Security Level, GP key.
S.CAD	Security Level, DM token, GP key.
O.CODE_CAP_FILE	DAP, MAC.
O.GP_KEY	Key types and key checksum.

FDP_ACF.1.2/CMGR The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: the runtime behaviors defined for the following operations as defined in [GP]:

- The S.CM load O.CODE_CAP_FILE into the card on behalf of S.CAD for OP.LOAD operation based on the attributes defined in [GP];
- install O.CODE_CAP_FILE on the card on behalf of S.CAD for OP.INSTALL operation based on the attributes defined in [GP];
- delete O.CODE_CAP_FILE from the card on behalf of S.CAD for OP.DELETE operation based on the attributes defined in [GP];
- load O.GP_KEY into the card on behalf of S.CAD for OP.PUT_KEY operation based on the attributes defined in [GP].

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

FDP_ACF.1.4/CMGR The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **Only O.CODE_CAP_FILE can be loaded or deleted by S.CM**.

FMT_MSA.1/CMGR Management of security attributes

FMT_MSA.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to restrict the ability to **modify** the security attributes **Privileges, Life-cycle Status, Security Level, GP key** to **S.CM.**

FMT_MSA.3/CMGR Static attribute initialization

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

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

7.2 SECURITY ASSURANCE REQUIREMENTS

The Evaluation Assurance Level is EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

7.3 SECURITY REQUIREMENTS RATIONALE

7.3.1 SECURITY OBJECTIVES FOR THE TOE

7.3.1.1 IDENTIFICATION

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

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).

7.3.1.2 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) 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_SMF.1/ADEL, 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/JCRE, FMT_MSA.1/JCVM) 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, the global byte array input parameter (bArray) to an applet's install method and the global arrays created by the JCSystem.makeGlobalArray(…) method. The clearing requirement of these arrays is met by (FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray 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.

If the TOE provides JCRMI functionality, 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, to the global byte array of the applet's install method or to the global arrays created by the JCSystem.makeGlobalArray(…) method. Such a pointer could be used to access and modify it when the buffer is being used by another application.

O.ARRAY_VIEWS_CONFID Array views have security attributes of temporary objects where the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from storing a reference to the array view. Furthermore, array views may not have ATTR_READABLE_VIEW security attribute which ensures that no application can read the contents of the array view.

O.ARRAY_VIEWS_INTEG Array views have security attributes of temporary objects where the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from storing a reference to the array view. Furthermore, array views may not have ATTR_WRITABLE_VIEW security attribute which ensures that no application can alter the contents of the array view.

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.**CAP_FILE**, which uphold the assumption A.CAP_FILE.

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 [JCRE3], 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 with FIPS certification [FIPS 140-2].

O.REALLOCATION This security objective is satisfied by the following SFRs: FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, 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_SMF.1/ADEL, FMT_SMF.1/CM and FMT_SMR.1/CM).

Additionally, if the TOE provides JCRMI functionality, memory management is controlled by the TSF FMT_SMR.1/JCRMI, and FMT_SMF.1/JCRMI.

7.3.1.3 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, , FCS_CKM.4 and FCS_COP.1. 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.RNG This security objective is directly covered by FCS_RNG.1 which ensures the cryptographic quality of random number generation.

O.KEY-MNGT This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1 and FDP_SDI.2/DATA as well. Precisely it is met by the following components: FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray 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/GlobalArray 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/DATA 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.TRANSACTION Directly met by FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray 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).

7.3.1.4 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.

7.3.1.5 APPLET MANAGEMENT

O.DELETION This security objective specifies that applet and CAP file 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 CAP file 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 CAP file into the card must be secure. Evidence of the origin of the CAP file is enforced (FCO_NRO.2/CM) and the integrity of the corresponding data is under the control of the CAP FILE 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).

7.3.1.6 CARD MANAGER

O.CARD_MANAGEMENT This security objective specifies that the access control to card management functions. This is enforced by FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_SMF.1/CM.

7.3.1.7 SMART CARD PLATFORM

O.SCP.RECOVERY This security objective specifies that the platform must behave securely if an unexpected loss of power occurs. This is covered by FPT_RCV.4 which specifies the recovery after unexpected power failure.

O.SCP.SUPPORT This security objective specifies that the SCP provides security features to the JCS. This is provided by FPT_TST.1/SCP. This is also provided by requirements of the IC, which are described in [THD89].

O.SCP.IC This security objective specifies that the IC must provide mechanisms to protect itself against physical attacks. This is provided by FPT_PHP.3/SCP. This is also provided by requirements of the IC, which are described in [THD89].

7.3.2 RATIONALE TABLES OF SECURITY OBJECTIVES AND SFRS

Security Objectives	Security Functional requirements
O.SID	FIA_ATD.1/AID, FIA_UID.2/AID, FMT_MSA.1/JCRE, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.1/CM, FMT_MSA.3/CM, FDP_ITC.2/Installer, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FIA_USB.1/AID, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM
O.FIREWALL	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FMT_SMR.1/Installer, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.3/FIREWALL, FMT_SMR.1, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_MSA.1/JCRE, FDP_ITC.2/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MSA.2/FIREWALL_JCVM, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM
O.GLOBAL_ARRAYS_CONFID	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_RIP.1/bArray, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT

O.GLOBAL_ARRAYS_INTEG	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM
O.ARRAY_VIEWS_CONFID	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_ACC.2/Firewall, FDP_ACF.1/Firewall
O.ARRAY_VIEWS_INTEG	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_ACC.2/Firewall, FDP_ACF.1/Firewall
O.NATIVE	FDP_ACF.1/FIREWALL
O.OPERATE	FAU_ARP.1, FDP_ROL.1/FIREWALL, FIA_ATD.1/AID, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FDP_ITC.2/Installer, FPT_RCV.3/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FPT_TDC.1, FIA_USB.1/AID
O.REALLOCATION	FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArrayFDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS
O.RESOURCES	FAU_ARP.1, FDP_ROL.1/FIREWALL, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMR.1/ADEL,FPT_FLS.1/Installer, FPT_FLS.1/ODEL, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_SMR.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE
O.ALARM	FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL, FAU_ARP.1
O.CIPHER	FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FPR_UNO.1
O.RNG	FCS_RNG.1
O.KEY-MNGT	FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_SDI.2/DATA, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT
O.PIN-MNGT	FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_ROL.1/FIREWALL, FDP_SDI.2/DATA, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL
O.TRANSACTION	FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT,

	FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_RIP.1/OBJECTS
O.OBJ-DELETION	FDP_RIP.1/ODEL, FPT_FLS.1/ODEL
O.DELETION	FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL
O.LOAD	FCO_NRO.2/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FIA_UID.1/CM, FTP_ITC.1/CM
O.INSTALL	FDP_ITC.2/Installer, FPT_RCV.3/Installer, FPT_FLS.1/Installer
O.CARD-MANAGEMENT	FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_SMF.1/CM
O.SCP.IC	FPT_RCV.4
O.SCP.RECOVERY	FPT_TST.1/SCP
O.SCP.SUPPORT	FPT_PHP.3/SCP

Table 7 Security Objectives and SFRs - Coverage

Security Functional Requirements	**Security Objectives**
FDP_ACC.2/FIREWALL	O.FIREWALL, O.OPERATE, O.PIN-MNGT, O.ARRAY_VIEWS_CONFID, O.ARRAY_VIEWS_INTEG
FDP_ACF.1/FIREWALL	O.FIREWALL, O.NATIVE, O.OPERATE, O.PIN-MNGT, O.ARRAY_VIEWS_CONFID, O.ARRAY_VIEWS_INTEG
FDP_IFC.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ARRAY_VIEWS_CONFID, O.ARRAY_VIEWS_INTEG
FDP_IFF.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ARRAY_VIEWS_CONFID, O.ARRAY_VIEWS_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	O.CIPHER, O.KEY-MNGT
FCS_CKM.4	O.CIPHER, O.KEY-MNGT
FCS_COP.1	O.CIPHER, O.KEY-MNGT
FCS_RNG.1	O_RNG
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/GlobalArray	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_RIP.1/TRANSIENT	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
FAU_ARP.1	O.OPERATE, O.RESOURCES, O.ALARM
FDP_SDI.2/DATA	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
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
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_RIP.1/ODEL	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.OBJ-DELETION
FPT_FLS.1/ODEL	O.OPERATE, O.RESOURCES, O.ALARM, 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
FMT_MSA.3/CM	O.SID, O.FIREWALL
FMT_SMF.1/CM	O.SID, O.FIREWALL, O.RESOURCES, O.CARD-MANAGEMENT
FMT_SMR.1/CM	O.FIREWALL, O.RESOURCES
FTP_ITC.1/CM	O.LOAD
FDP_ACC.1/CMGR	O.CARD-MANAGEMENT
FDP_ACF.1/CMGR	O.CARD-MANAGEMENT
FMT_MSA.1/CMGR	O.CARD-MANAGEMENT
FMT_MSA.3/CMGR	O.CARD-MANAGEMENT
FPT_RCV.4	O.SCP.IC
FPT_TST.1/SCP	O.SCP.RECOVERY
FPT_PHP.3/SCP	O.SCP.SUPPORT

Table 8 SFRs and Security Objectives

7.3.3 DEPENDENCIES

7.3.3.1 SFRS DEPENDENCIES

Requirements	CC Dependencies	Satisfied 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/FIREWALL
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/JCVM, FMT_SMR.1
FMT_SMF.1	No Dependencies	
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2/AID
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_COP.1, FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4，FDP_ITC.2/Installer
FCS_RNG.1	No Dependencies	
FDP_RIP.1/ABORT	No Dependencies	
FDP_RIP.1/APDU	No Dependencies	
FDP_RIP.1/bArray	No Dependencies	

FDP_RIP.1/GlobalArray	No Dependencies	
FDP_RIP.1/KEYS	No Dependencies	
FDP_RIP.1/TRANSIENT	No Dependencies	
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FAU_ARP.1	(FAU_SAA.1)	
FDP_SDI.2/DATA	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	
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_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
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_RIP.1/ODEL	No Dependencies	
FPT_FLS.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	FDP_IFC.2/CM,

	(FMT_MSA.3)	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	
FDP_ACC.1/CMGR	FDP_ACF.1	FDP_ACF.1/CMGR
FDP_ACF.1/CMGR	FDP_ACC.1, FMT_MSA.3	FDP_ACC.1/CMGR, FMT_MSA.3/CMGR
FMT_MSA.1/CMGR	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_ACC.1/CMGR, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CMGR	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/CMGR, FMT_SMR.1/CM
FPT_RCV.4	No Dependencies	
FPT_TST.1/SCP	No Dependencies	
FPT_PHP.3/SCP	No Dependencies	

Table 9 SFRs Dependencies

7.3.3.2 RATIONALE FOR THE EXCLUSION OF SFRS DEPENDENCIES

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

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

The dependency FMT_SMF.1 of FMT_MSA.1/JCRE is discarded. 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 discarded. The dependency of FAU_ARP.1 on FAU_SAA.1 assumes that a "potential security violation" generates an audit event. On

the contrary, the events listed in FAU_ARP.1 are self-contained (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, there is no mandatory audit recording in this ST.

7.3.3.3 SARS DEPENDENCIES

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

Table 10 SARs Dependencies

7.3.4 COMPATIBILITY BETWEEN SFR OF TOE AND [THD89]

The table below lists the SFR defined by THD89 IC hardware on which the SFR of this TOE is relying and they are separated in three groups:

- IP_SFR: Irrelevant Platform-SFRs not being used by the Composite-ST.
- RP_SFR-SERV: Relevant Platform-SFRs being used by the Composite-ST to implement a security service with associated TSFI.
- RP_SFR-MECH: Relevant Platform-SFRs being used by the Composite-ST because of its security properties providing protection against attacks to the TOE as a whole and are addressed in ADV_ARC. These required security properties are a result of the security mechanisms and services that are implemented in the Platform TOE.

SFR of THD89	IP_SFR	RP_SFR-SERV	RP_SFR-MECH
FDP_ITT.1			FCS_RNG.1
FDP_IFC.1			FDP_IFC.1/JCVM
FPT_ITT.1	X		
FDP_SDC.1	X		
FPT_PHP.3			FPT_PHP.3/SCP
FRU_FLT.2			FPT_FLS.1, FPT_PHP.3/SCP
FPT_FLS.1			FPT_FLS.1, FPT_FLS.1/Installer, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL
FDP_SDI.2			FDP_SDI.2/DATA
FMT_LIM.1	X		
FMT_LIM.2	X		
FAU_SAS.1	X		
FCS_RNG.1[PTG.2]		FCS_RNG.1	
FCS_COP.1[TDES]		FCS_COP.1	
FCS_COP.1[RSA-CRT]		FCS_COP.1	
FCS_COP.1[AES]		FCS_COP.1	
FCS_COP.1[ECC]		FCS_COP.1	

7.3.5 RATIONALE FOR THE SECURITY ASSURANCE REQUIREMENTS

EAL5 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. EAL5 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 EAL5.

7.3.6 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 EAL5 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.

7.3.7 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, in particular in payment and identity areas. AVA_VAN.5 has dependencies on ADV_ARC.1, ADV_FSP.1, ADV_TDS.3, ADV_IMP.1, AGD_PRE.1 and AGD_OPE.1. All of them are satisfied by EAL5.

8 TOE SUMMARY SPECIFICATION

8.1 TOE SECURITY FUNCTIONALITY

8.1.1 SF.FW: FIREWALL POLICY

The SF.FW provides the Firewall Security Function which enforces firewall policy to ensure applet isolation and object sharing. Isolation means that one applet cannot access the fields or objects of an applet in another context unless the other applet explicitly provides an interface for access. To be more specific, firewalls essentially partition the Java Card platform's object system into separate protected object spaces called contexts. The firewall is the boundary between one context and another. The Java Card RE shall allocate and manage a context for each Java Card CAP File containing applets. All applet instances within a single Java Card CAP File shares the same context. There is no firewall between individual applet instances within the same CAP File. That is, an applet instance can freely access objects belonging to another applet instance that resides in any package in the same Java Card CAP File.

The following SFR are fulfilled by SF.FW:

- FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL
- FDP_IFC.1/JCVM and FDP_IFF.1/JCVM
- FDP_RIP.1/OBJECTS
- FMT_MSA.1/JCRE
- FMT_MSA.1/JCVM
- FMT_MSA.2/FIREWALL_JCVM
- FMT_MSA.3/FIREWALL
- FMT_MSA.3/JCVM
- FMT_SMF.1
- FMT_SMR.1

8.1.2 SF.API: APPLICATION PROGRAMMING INTERFACE

The SF.API Security Function provides cryptographic operations and key management according to [JCAPI3] and [GP]. It also ensures that the resource is made unavailable upon the deallocation of that resource according to [JCRE3].

The following key generation algorithms, key sizes and standards are supported:

Iteration	Algorithm	Key size	Standards
ECC Key Pair	ECC	ansix9p224r1, ansix9p256r1, ansix9p384r1, ansix9p521r1,	ANSI X9.62-2005, RFC 5639

		brainpoolP224r1, brainpoolP256r1 and brainpoolP384r1 curves	
SCP02 session key	TDES in CBC mode	128 bits	[GP]
SCP03 session key	KDF in counter mode	128 bits	[GP]

The following cryptographic operations with supported algorithms, key sizes and standards are listed below:

Iteration	Operation	Algorithm	Key Size	Standards
TDES	Encryption and Decryption	ALG_DES_CBC_ISO9797_M1 ALG_DES_CBC_ISO9797_M2 ALG_DES_CBC_NOPAD ALG_DES_ECB_ISO9797_M1 ALG_DES_ECB_ISO9797_M2 ALG_DES_ECB_NOPAD ALG_DES_CBC_PKCS5 ALG_DES_ECB_PKCS5	LENGTH_DES3_2KEY,	NIST-SP800-67 ISO9797-1 DES NOPAD DES PKCS#5 DES 9797 M1 M2
DESMAC	Generation and Verification	ALG_DES_MAC4_ISO9797_1_M1_ALG3 ALG_DES_MAC4_ISO9797_1_M2_ALG3 ALG_DES_MAC4_ISO9797_M1 ALG_DES_MAC4_ISO9797_M2 ALG_DES_MAC8_ISO9797_1_M1_ALG3 ALG_DES_MAC8_ISO9797_1_M2_ALG3 ALG_DES_MAC8_ISO9797_M1 ALG_DES_MAC8_ISO9797_M2 ALG_DES_MAC8_NOPAD ALG_DES_MAC4_PKCS5 ALG_DES_MAC8_PKCS5	LENGTH_DES3_2KEY,	[JCAPI3]
RSA-CRT Signature PKCS1	Generation	ALG_RSA_SHA_224_PKCS1 ALG_RSA_SHA_224_PKCS1_PSS ALG_RSA_SHA_256_PKCS1 ALG_RSA_SHA_256_PKCS1_PSS ALG_RSA_SHA_384_PKCS1 ALG_RSA_SHA_384_PKCS1 ALG_RSA_SHA_512_PKCS1 ALG_RSA_SHA_512_PKCS1_PSS ALG_RSA_SHA_ISO9796 ALG_RSA_SHA_ISO9796_MR ALG_RSA_SHA_PKCS1 ALG_RSA_SHA_PKCS1_PSS	1024, 2048, 4096	[JCAPI3]
ECDSA	Signature and Verification	ALG_ECDSA_SHA ALG_ECDSA_SHA_224	224, 256, 384 and 521 bits	[JCAPI3]

		ALG_ECDSA_SHA_256 ALG_ECDSA_SHA_384 ALG_ECDSA_SHA_512		
AES CBC/ECB	Encryption and Decryption	ALG_AES_BLOCK_128_CBC_NOPAD ALG_AES_BLOCK_128_ECB_NOPAD ALG_AES_CBC_ISO9797_M1 ALG_AES_CBC_ISO9797_M2 ALG_AES_CBC_PKCS5 ALG_AES_ECB_ISO9797_M1 ALG_AES_ECB_ISO9797_M2 ALG_AES_ECB_PKCS5	128, 192, 256 bits	[JCAPI3]
AES MAC	Signature and Verification	ALG_AES_CMAC_128	128, 192, 256 bits	[JCAPI3]
AES CMAC	Signature and Verification	ALG_AES_MAC_128_NOPAD	128, 192, 256 bits	[JCAPI3]

The following SFR are fulfilled by SF.API:

- FCS_CKM.1
- FCS_CKM.4
- FCS_COP.1
- FDP_RIP.1/ABORT
- FDP_RIP.1/APDU
- FDP_RIP.1/GlobalArray
- FDP_RIP.1/bArray
- FDP_RIP.1/KEYS
- FDP_RIP.1/TRANSIENT
- FDP_ROL.1/FIREWALL

8.1.3 SF.CSM: CARD SECURITY MANAGEMENT

The Card Security Management Security Function ensures the security policy for card content management, lifecycle management and the related cryptographic operations according to [JCVM3] and [GP].

Upon detection of a potential security violation, one of the following actions is invoked:

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

The "potential security violation" stands for one of the following events:

- CAP file inconsistency,
- typing error in the operands of a bytecode,

- applet life cycle inconsistency,
- card manager life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context,
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,
- integrity error on sensitive data including cryptographic key and PIN.

The THD89 IC provides hardware security mechanisms including Bus Masking, Data Masking and dummy operation to ensure that it is unable to observe the cryptographic key values and PIN values in terms of cryptographic operation and comparison.

The following SFR are fulfilled by SF.CSM:

- FAU_ARP.1
- FDP_SDI.2/DATA
- FPR_UNO.1
- FPT_FLS.1
- FPT_TDC.1

8.1.4 SF.AID: AID MANAGEMENT

The AID Security Function enforces rules for the AID as defined in [JCRE3] with respect to the following security attributes:

- CAP File AID,
- Package AID,
- Applet's version number,
- Registered applet AID,
- Applet Selection Status.

Each user is identified before allowing any other actions on behalf of that user. Upon applet selection, the rules for initial selection and applet selection are enforced according to [JCRE3] §4. Furthermore, the modification on the AID list that is maintained by JCRE is restricted.

The following SFR are fulfilled by SF.AID:

- FIA_ATD.1/AID
- FIA_UID.2/AID
- FIA_USB.1/AID
- FMT_MTD.1/JCRE
- FMT_MTD.3/JCRE

8.1.5 SF.INST: INSTALLER

The Installer Security Function enforces rules for loading and installation of a package and applet respectively as defined in [JCVM3].

CAP file loading is allowed only if, for each dependent package, its AID is equal to a resident package AID, the major version associated to the dependent package file is equal to the major version of the resident package and the minor version is equal to or less than the minor version associated to the resident package ([JCVM3], §4.5.2).

For failures during load / installation of a package/applet and deletion of a package / applet / object, SF.INST ensures the return of the TOE to a secure state using automated procedures.

The SF.INST fulfils the following SFR:

- FDP_ITC.2/Installer
- FMT_SMR.1/Installer
- FPT_FLS.1/Installer
- FPT_RCV.3/Installer

8.1.6 SF.ADEL: APPLET DELETION

The ADEL Security Function enforces rules for applet deletion and resource management after the applet deletion as defined in [JCVM3].

The Cap file and applet can be deleted only if the operation meets the rules according to [JCVM3]. Once deleted, any previous information content of a resource is made unavailable upon the deallocation of the resource from the deleted objects.

A secure state is preserved when the applet deletion manager fails to delete a CAP file/applet as described in [JCRE3], §11.3.4.

The following SFR are fulfilled by SF.ADEL:

- FDP_ACC.2/ADEL and FDP_ACF.1.1/ADEL
- FDP_RIP.1/ADEL
- FMT_MSA.1/ADEL
- FMT_MSA.3/ADEL
- FMT_SMF.1/ADEL
- FMT_SMR.1/ADEL
- FPT_FLS.1/ADEL

8.1.7 SF.ODEL: OBJECT DELETION

The ODEL Security Function enforces rules for object deletion and resource management after the object deletion as defined in [JCVM3].

Any previous information content of a resource is made unavailable upon the deallocation of the resource from the deleted objects.

A secure state is preserved when the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the deletion.

The following SFR are fulfilled by SF.ODEL:

- **FDP_RIP.1/ODEL**
- **FPT_FLS.1/ODEL**

8.1.8 SF.CAR: SECURE CARRIER

The Secure Carrier Security Function ensures the confidentiality, integrity and authenticity for applet package loading between on-card entity and off-card entity. This is achieved by the supporting of DAP verification and secure channel protocols including SCP02 and SCP03.

The following SFR are fulfilled by SF.CAR:

- FCO_NRO.2/CM
- FDP_IFC.2/CM and FDP_IFF.1/CM
- FDP_UIT.1/CM
- FIA_UID.1/CM
- FMT_MSA.1/CM
- FMT_MSA.3/CM
- FMT_SMF.1/CM
- FMT_SMR.1/CM
- FTP_ITC.1/CM

8.1.9 SF.SCP: SMART CARD PLATFORM

The SCP Security Function provides capabilities for self-protection and data integrity based on the security services provided by THD89 IC hardware and Native COS.

The THD89 IC hardware is tampering-resistant with a set of security mechanisms to detect and react upon malicious operations. Moreover, the hardware is capable of verifying the integrity of sensitive data like Keys and PIN. Lastly, a secure random number generator is provided as well.

The Native COS provides transaction mechanism to ensure that reading from and writing to static and objects' fields interrupted by power loss have the property that the operation either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

The following SFR are fulfilled by SF.SCP:

- FPT_TST.1/SCP

- FPT_PHP.3/SCP
- FPT_RCV.4/SCP
- FCS_RNG.1

8.1.10 SF.CM: CARD MANAGER

The Card Manager Security Function enforce the CARD CONTENT MANAGEMENT rules as defined in [GP]. The CARD CONTENT MANAGEMENT rules include the runtime behaviors of loading, installation, extradition, personalization, and deletion according to [GP].

The following SFR are fulfilled by SF.CM:

- FDP_ACC.1/CMGR and FDP_ACF.1/CMGR
- FMT_MSA.1/CMGR
- FMT_MSA.3/CMGR

8.2 PROTECTION AGAINST INTERFERENCE AND LOGICAL TAMPERING

The combination of logical and physical counter measures that are implemented in this TOE provides protection against interference and logical tampering.

A non-exhaustive list of the software security mechanisms is depicted below:

- Complex patterned values instead simple true or false are applied for security sensitive checking to prevent the toggle of the test result.
- The security attributes related to access control is managed in a redundant manner. The values of the security attributes are double checked before they are accessed.
- Buffers that contain sensitive information is processed in random order.
- Buffers that contain sensitive information is processed in time invariant manner.
- The sensitive operations in critical path are protected by a flow control counter by which the circumvention of program flow can be detected.
- The validity of byte-code is checked in runtime and a security exception is thrown upon the invocation of undefined byte-code.
- Sensitive data like PIN and Key are stored in confidential format.

More detailed information about the software security mechanism can be found in ADV_ARC.

In addition to the software countermeasures, the certified IC hardware also provides a set of security mechanisms that can be utilized to protect the TOE from interference and logical

tampering. Those hardware-based features are configured during initialization phase. The configuration process is protected with software countermeasure consisting of read after write and flow control counter so that a correct and secure operational state can be ensured and perturbation during initialization phase can be detected.

- A set of environmental sensors to monitor the correct operation environmental conditions. These sensors include voltage, light, frequency, and temperature.
- Bus masking mechanism to encrypt the sensitive data transferred over system bus, which can prevent the extracting of internal signal.
- Security mechanism to guarantee the integrity of data stored in volatile or non-volatile memories.
- random clock jitter mechanism to vary the central frequency of the oscillator, which can increase the difficulty of Side-Channel Attack.
- Security library to process cryptographic operation.

Upon detection of interference and logical tampering, the TOE takes two kinds of reactions. If the attack is detected by the software security mechanism, a proper security exception is thrown, whilst if the threat is detected by the hardware security mechanism, the reaction is card mute and a reset of card is required to restore the TOE to the secure operational state.

8.3 PROTECTION AGAINST BYPASS

The implementation of the TOE is compliant with the following technical standards in order to prevent bypass:

- The Java Card 3.1 Virtual Machine [JCVM3];
- The Java Card 3.1 Runtime Environment [JCRE3];
- The Java Card 3.1 Application Programming Interface [JCAPI3];
- The Global Platform Card Specification version 2.3 [GP].

All the APDU commands supported by the TOE are well defined and strictly checked upon reception according to the APDU encoding rules defined in the above specifications. In addition to the standard APDUs, there are TMS proprietary APDU commands which are well documented in AGD_OPE document. Thus, attempts to access the TOE security functions by incorrect and undefined APDU can be detected and a propriate exception is thrown.

The Firewall Security Function enforces firewall access control policy to ensure applet isolation and object sharing. Isolation means that one applet cannot access the fields or objects of an applet in another context unless the other applet explicitly provides an interface for

access. To be more specific, firewalls essentially partition the Java Card platform's object system into separate protected object spaces called contexts. The firewall is the boundary between one context and another. The Java Card RE allocates and manages a context for each Java Card CAP File containing applets. All applet instances within a single Java Card CAP File shares the same context. There is no firewall between individual applet instances within the same CAP File. That is, an applet instance can freely access objects belonging to another applet instance that resides in any package in the same Java Card CAP File.

The TOE provides secure messaging service to ensure the confidentiality, integrity and authenticity when data is transferred between on-card and off-card. The secure messaging protocol in its design requires mutual authentication of the on-card representative and off-card entity at the communication initialization phase. Once the secure messaging session is established, the following communication is based on an ordinal sequence with chaining C-MAC to prevent replay attack and to detect incorrect command sequence.

9 Appendix

9.1 Glossary of vocabulary

Term	Definition
AID	Application identifier, an ISO-7816 data format used for unique identification of Java Card applets (and certain kinds of files in card file systems). The Java Card platform uses the AID data format to identify applets, packages and CAP files. AIDs are administered by the International Opens Organization (ISO), so they can be used as unique identifiers. AIDs are also used in the security policies (see “Context” below): applets’ AIDs are related to the selection mechanisms; CAP files’ AIDs are used in the enforcement of the firewall. Note: although they serve different purposes, they share the same namespace.
APDU	Application Protocol Data Unit, an ISO 7816-4 defined communication format between the card and the off-card applications. Cards receive requests for service from the CAD in the form of APDUs. These are encapsulated in Java Card System by the javacard.framework.APDU class ([JCAPI22]). APDUs manage both the selection-cycle of the applets (through Java Card RE mediation) and the communication with the Currently selected applet.
APDU buffer	The APDU buffer is the buffer where the messages sent (received) by the card depart from (arrive to). The Java Card RE owns an APDU object (which is a Java Card RE Entry Point and an instance of the javacard.framework.APDU class) that encapsulates APDU messages in an internal byte array, called the APDU buffer. This object is made accessible to the currently selected applet when needed, but any permanent access (out of selection-scope) is strictly prohibited for security reasons.
Applet	The name given to any Java Card technology-based application. An applet is the basic piece of code that can be selected for execution from outside the card. Each applet on the card is uniquely identified by its AID.
Applet deletion manager	The on-card component that embodies the mechanisms necessary to delete an applet or library and its associated data on smart cards using Java Card technology.
BCV	The bytecode verifier is the software component performing a static analysis of the code to be loaded on the card. It checks several kinds of properties, like the correct format of CAP files and the enforcement of the typing rules associated to bytecodes. If the component is placed outside the card, in a secure environment, then it is called an off-card verifier. If the component is part of the embedded software of the card it is called an on-card verifier.
CAD	Card Acceptance Device or card reader. The device where the card is

	inserted, and which is used to communicate with the card. Unless explicitly said otherwise, in this document, CAD covers PCD.
CAP file	A file in the Converted applet format. A CAP file contains a binary representation of one or several packages of classes that can be installed on a device and used to execute the packages' classes on a Java Card virtual machine. A CAP file can contain a user library, or the code of one or more applets.
Class	In object-oriented programming languages, a class is a prototype for an object. A class may also be considered as a set of objects that share a common structure and behavior. Each class declares a collection of fields and methods associated to its instances. The contents of the fields determine the internal state of a class instance, and the methods the operations that can be applied to it. Classes are ordered within a class hierarchy. A class declared as a specialization (a subclass) of another class (its super class) inherits all the fields and methods of the latter. Java platform classes should not be confused with the classes of the functional requirements (FIA) defined in the CC.
Context	A context is an object-space partition associated to a CAP file. Applets in Java technology-based packages contained within the same CAP file belong to the same context. The firewall is the boundary between contexts (see "Current context").
Current context	The Java Card RE keeps track of the current Java Card System context (also called "the active context"). When a virtual method is invoked on an object, and a context switch is required and permitted, the current context is changed to correspond to the context of the applet that owns the object. When that method returns, the previous context is restored. Invocations of static methods have no effect on the current context. The current context and sharing status of an object together determine if access to an object is permissible.
Currently selected applet	The applet has been selected for execution in the current session. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command from the CAD or PCD with this applet's AID, the Java Card RE makes this applet the currently selected applet over the I/O interface that received the command. The Java Card RE sends all further APDU commands received over each interface to the currently selected applet on this interface ([JCRE22], Glossary).
Default applet	The applet that is selected after a card reset or upon completion of the PICC activation sequence on the contactless interface ([JCRE22], §4.1).
DPA	Differential Power Analysis is a form of side channel attack in which an attacker studies the power consumption of a cryptographic hardware device such as a smart card.
Embedded Software	Pre-issuance loaded software.
Firewall	The mechanism in the Java Card technology for ensuring applet isolation and

	object sharing. The firewall prevents an applet in one context from unauthorized access to objects owned by the Java Card RE or by an applet in another context.
Installer	The installer is the on-card application responsible for the installation of applets on the card. It may perform (or delegate) mandatory security checks according to the card issuer policy (for bytecode-verification, for instance), loads and link CAP files on the card to a suitable form for the Java Card VM to execute the code they contain. It is a subsystem of what is usually called “card manager”; as such, it can be seen as the portion of the card manager that belongs to the TOE. The installer has an AID that uniquely identifies him, and may be implemented as a Java Card applet. However, it is granted specific privileges on an implementation-specific manner ([JCRE3], §10).
Interface	A special kind of Java programming language class, which declares methods, but provides no implementation for them. A class may be declared as being the implementation of an interface, and in this case must contain an implementation for each of the methods declared by the interface (See also shareable interface).
Java Card RE	The runtime environment under which Java programs in a smart card are executed. It is in charge of all the management features such as applet lifetime, applet isolation, object sharing, applet loading, applet initializing, transient objects, the transaction mechanism and so on.
Java Card RE Entry Point	An object owned by the Java Card RE context but accessible by any application. These methods are the gateways through which applets request privileged Java Card RE services: the instance methods associated to those objects may be invoked from any context, and when that occurs, a context switch to the Java Card RE context is performed. There are two categories of Java Card RE Entry Point Objects: Temporary ones and Permanent ones. As part of the firewall functionality, the Java Card RE detects and restricts attempts to store references to these objects.
Java Card RMI	Java Card Remote Method Invocation is the Java Card System version 2.2 and 3 Classic Edition mechanism enabling a client application running on the CAD platform to invoke a method on a remote object on the card. Notice that in Java Card System, version 2.1.1, the only method that may be invoked from the CAD is the process method of the applet class and that in Java Card System, version 3 Classic Edition, this functionality is optional.
Java Card System	Java Card System includes the Java Card RE, the Java Card VM, the Java Card API and the installer.
Java Card VM	The embedded interpreter of bytecodes. The Java Card VM is the component that enforces separation between applications (firewall) and enables secure data sharing.
Logical channel	A logical link to an application on the card. A new feature of the Java Card System, version 2.2 and 3 Classic Edition, that enables the opening of

	simultaneous sessions with the card, one per logical channel. Commands issued to a specific logical channel are forwarded to the active applet on that logical channel. Java Card platform, version 2.2.2 and 3 Classic Edition, enables opening up to twenty logical channels over each I/O interface (contacted or contactless).
NVRAM	Non-Volatile Random-Access Memory, a type of memory that retains its contents when power is turned off.
Object deletion	The Java Card System version 2.2 and 3 Classic Edition mechanism ensures that any unreferenced persistent (transient) object owned by the current context is deleted. The associated memory space is recovered for reuse prior to the next card reset.
Package	A package is a namespace within the Java programming language that may contain classes and interfaces. A package defines either a user library, or one or more applet definitions. A package is divided in two sets of files: export files (which exclusively contain the public interface information for an entire package of classes, for external linking purposes; export files are not used directly in a Java Card virtual machine) and CAP files where CAP file may contain one or more packages.
PCD	Proximity Coupling Device. The PCD is a contactless card reader device.
PICC	Proximity Card. The PICC is a card with contactless capabilities.
RAM	Random Access Memory, is a type of computer memory that can be accessed randomly
SCP	Smart Card Platform. It is comprised of the integrated circuit, the operating system and the dedicated software of the smart card.
Shareable interface	An interface declaring a collection of methods that an applet accepts to share with other applets. These interface methods can be invoked from an applet in a context different from the context of the object implementing the methods, thus “traversing” the firewall.
SIO	An object of a class implementing a shareable interface.
Subject	An active entity within the TOE that causes information to flow among objects or change the system’s status. It usually acts on the behalf of a user. Objects can be active and thus are also subjects of the TOE.
SWP	The Single Wire Protocol is a specification for a single-wire connection between the SIM card and a Near Field Communication (NFC) chip in a mobile handset
Transient object	An object whose contents are not preserved across CAD sessions. The contents of these objects are cleared at the end of the current CAD session or when a card reset is performed. Writes to the fields of a transient object are not affected by transactions.
User	Any application interpretable by the Java Card RE. That also covers the packages. The associated subject(s), if applicable, is (are) an object(s) belonging to the javacard.framework.applet class.

9.2 References

[CC1]	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1. Revision 5. April 2017. CCMB-2017-04-001.
[CC2]	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1. Revision 5. April 2017. CCMB-2017-04-002.
[CC3]	Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 3.1. Revision 5. April 2017. CCMB-2017-04-003.
[CEM]	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1. Revision 5. April 2017. CCMB-2017-04-004.
[GP]	GlobalPlatform Card Specification Version 2.3.0 including: ● Card Confidential Card Content Management Card Specification v2.3 - Amendment A V1.1; ● Card Technology Contactless Services Card Specification v2.3 - Amendment C V1.3; ● Card Technology Secure Channel Protocol '03' Card Specification v2.3 – Amendment D V1.1.2; ● Card Technology Security Upgrade for Card Content Management Card Specification v2.3 – Amendment E V1.1; ● Card Technology Secure Channel Protocol '11' Card Specification v2.3 – Amendment F V1.2.1; ● Card Technology Executable Load File Upgrade Card Specification v2.3 – Amendment H V1.1; ● Card Technology Secure Element Configuration V2.0.
[JCVM22]	Java Card Platform, version 2.2 Virtual Machine (Java Card VM) Specification. June 2002. Published by Sun Microsystems, Inc.
[JCAPI22]	Java Card Platform, version 2.2 Application Programming Interface. June 2002. Published by Sun Microsystems, Inc.
[JCRE22]	Java Card Platform, version 2.2 Runtime Environment (Java Card RE) Specification. June 2002. Published by Sun Microsystems, Inc.
[JCVM3]	Java Card Platform, versions 3.0 up to 3.1, Classic Edition, Virtual Machine (Java Card VM) Specification. Published by Oracle.
[JCAPI3]	Java Card Platform, versions 3.0 up to 3.1, Classic Edition, Application Programming Interface. Published by Oracle.
[JCRE3]	Java Card Platform, versions 3.0 up to 3.1, Classic Edition, Runtime Environment (Java Card RE) Specification. Published by Oracle.
[JCBV]	Java Card 3 Platform Off-card Verification Tool Specification, Classic Edition, Version 1.0. Published by Oracle.
[JAVASPEC]	The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[PP0084b]	BSI-CC-PP-0084-2014, Security IC Platform Protection Profile with Augmentation

	Packages, Version 1.0, 13 January 2014
[PP JCS]	Java Card System Protection Profile, Version 3.1, April 2020, registered and certified by the German certification body (BSI) under the following references: [BSI-CC-PP-0099-V2-2020].
[THD89]	THD89 1.0.3 Secure Element Version 1.0 Security Target Lite, version 0.1 Jun. 2023.
[AIS20]	BSI. Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren. Version 3.0 (15.05.2013).
[AIS31]	BSI. Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren. Version 3 (15.05.2020).
[AGD_PRE]	TMCOS 4.0 0.3 On THD89 1.0.3 Preparative procedures v1.4
[AGD_OPE]	TMCOS 4.0 0.3 On THD89 1.0.3 Operational User Guidance v1.6