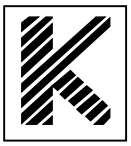

KEYCORP LIMITED

Keycorp MULTOS

Common Criteria

Public Security Target

26 May 2004

Author: Thales Microelectronics & Keycorp MULTOS Team

Checked: _____

Document Number: SIM-SP-0212

Date: _____

Revision Number: 1.0

Table of Contents

1. Introduction

1.1 Security Target Identification

Security Target Title: Keycorp MULTOS Common Criteria Public Security Target.

Security Target Version Number: 1.0

Identity of the Target of Evaluation (TOE): The Target of Evaluation is Keycorp MULTOS including AMD with ID (as assigned by MULTOS CA) 0029v002 and Infineon Technologies SLE66CX322P Smartcard Integrated Circuit.

Version of the TOE: Keycorp MULTOS Version I4C implements version 4.06 of the MULTOS specifications, and also the Change Requests CR0099, CR0103, CR0109, CR0113.

Common Criteria: Version 2.1, August 1999.

1.2 Security Target Overview

The integrated circuit card (ICC), or smartcard, is an ideal tool for the delivery of distributed, secure information processing at low cost. However, an application developed for one smartcard is usually not portable to another. Furthermore, many current smartcard operating systems allow only one application per card, meaning end users must carry a multitude of cards, one for each function or service required. Keycorp Ltd, in its role as a member of the MAOSCO consortium, is developing an open, high-security multi-application operating system to address the current shortcomings of smartcard operating systems. This operating system is called MULTOS.

In order to satisfy the objectives set for it, MULTOS should be able to:

a) Execute an application written for MULTOS - application execution should be independent of the underlying smartcard hardware.

b) Load many applications - applications should be able to co-exist on the smartcard.

c) Ensure that applications are securely loaded and segregated - they should not be able to interfere with each other or with MULTOS.

In summary, MULTOS provides a common development and operating platform for smartcard applications. It allows multiple applications to be loaded onto a single smartcard and execute without interfering with or being interfered with by other applications. It also allows applications written for MULTOS to execute on different types of smartcard independent of the underlying smartcard hardware.

This Security Target is based on CC Smartcard Integrated Circuit with Multi-Application Secure Platform Protection Profile PP/0010, Version 2.0, Issue November 2000, registered at the French Certification Body. Compliance claim with this PP is not possible as the chip was not evaluated in compliance with the PP9806.

Please note that the PP/0010 is upwardly compatible with the PP/9806 and PP/9911. Therefore, this ST is also based on Smartcard IC with Embedded Software Protection Profile PP/9911, Version 2.0, Issue June 1999.

In comparison with PP/0010, this security target includes:

- One additional organisational security policy relative to the application code hash digest in the ALC.
- One additional security objective for the environment: O.CODE_HASH.

1.3 Common Criteria Conformance

This ST has been built with Common Criteria for Information Technology Security Evaluation, Version 2.1 (ISO15408), as the following:

- Part 2 conformant.
- Part 3 conformant with EAL4 level augmented.
- Based on PP/9911 Version 2.0.
- Based on PP/0010 Version 2.0.

The EAL4 level from CC Part 3 is augmented with the assurance components ADV_IMP.2, ALC_DVS.2 and AVA_VLA.4.

Note: Items which are common to PP/9806 and PP/0010 are indicated by a "*" in this Security Target.

2. Target of Evaluation Description

This part of the Security Target describes the Target of Evaluation as an aid to the understanding of its security requirements and addresses the product type, the intended usage and the general IT features of the TOE.

2.1 Product Type

2.1.1 Product Description

MULTOS is an operating system for integrated circuit cards (also known as smartcards). It is designed to allow multiple smartcard applications to be securely loaded and executed on a smartcard.

The user of the smartcard accesses the applications loaded on it via an Interface Device (IFD), which could be a Point-of-Sale terminal, Automatic Teller Machine, or some other device which supports ISO 7816 smartcard protocols.

Communications across the IFD-MULTOS interface comprise a message transmitted by the smartcard when it is reset (the Answer-to-Reset or ATR message), followed by command-response pairs, where a command is a message from the IFD to MULTOS and a response is a message from MULTOS to the IFD.

By means of these command-response pairs, MULTOS allows:

a) Applications to be loaded onto and deleted from the smartcard.

b) An IFD to access data and applications which are loaded on the card.

c) Information specific to the card to be retrieved by an IFD.

MULTOS is a single-threaded operating system. Only one application can be executing at any given time. MULTOS does not provide mechanisms for concurrency or multi-tasking. Following power-on of the smartcard and initialisation, the basic execution sequence for MULTOS is as follows:

a) Wait for input from the IFD.

b) Parse the input.

c) If the input is a MULTOS command, process the command and write a response to the IFD.

d) Otherwise, execute the currently selected application and write to the IFD any output created by the application.

e) Loop back to a).

Applications to be loaded on MULTOS-based smartcards are written in a hardware-independent language called MULTOS Executable Language (MEL). MEL applications are interpreted by MULTOS, rather than being compiled and executed directly on the smartcard processor.

MULTOS also provides for shared code routines, called Codelets, which can be called by an executing application. Codelets can be loaded into MULTOS during IC manufacture or at smartcard

personalisation time. A codelet has its own code address space but executes in the context of the calling application, so has access to the application's data.

MULTOS is targeted to operate on the Infineon Technologies SLE66CX160P and SLE66CX320P Smartcard Integrated Circuits (ICs). The IC provides the microprocessor to execute the instructions comprising the executable code of MULTOS.

The Infineon SLE66CX162P or SLE66CX322P - a single-chip microcontroller embedded in the plastic card, consists of the following elements:

– Five pins which allow the interface device (IFD) to communicate with an MCD, as follows:

- VCC and GND, which supply power to the MCD.
- RESET, which is used by the IFD to reset the MCD.
- I/O, a bi-directional serial channel along which commands and responses are sent using standardised communication protocols.
- CLK, which is used to supply the SLE66CxxxP processor with a clock.

– An eight-bit CPU, which supports the standard 8051 instruction, set together with a set of instructions that are specific to this microcontroller.

– At least 64 Kbytes of mask ROM.

– At least 16 Kbytes of EEPROM (including a 32-byte Security PROM and one-time programmable (OTP) area).

– 256 bytes of "internal" RAM and at least 2 Kbytes of "external" RAM.

– A 1024-bit crypto co-processor used to support public key cryptographic algorithms.

– A hardware random number generator.

– A timer with prescaler.

– An interrupt controller (interrupts unused in Keycorp MULTOS).

– A CRC module (unused by Keycorp MULTOS).

– A memory management protection unit.

– A phased locked loop unit.

– An interrupt module.

– A UART (unused by Keycorp MULTOS).

– A dual key DES and elliptic curve accelerator.

Both the internal and external RAM are held on the one silicon die of the Infineon SLE66CX162P or SLE66CX322P IC. "Internal" and "external" refer to different addressable memory spaces in the 8051 microprocessor architecture.

MULTOS requires the target IC to execute instructions correctly according to its specification.

2.1.2 Intended Method of Use

MULTOS is intended to provide a hardware-independent environment for the execution of multiple applications that provide a variety of functions and services to the holder of the smartcard. Applications may be developed and supplied by different organisations from different industries, and consequently may provide many different services e.g., financial, communication or access control. The security requirements of different applications may also vary (i.e., some applications may require a high level of security while others may only have a low level or no security requirements).

Mondex International (MXI) is the organisation which owns the rights to MULTOS. MXI has in turn issued an exclusive licence to MAOSCO to develop and use the MULTOS specifications. MAOSCO issues licences to organisations to implement MULTOS and to develop MULTOS applications. Therefore, the security provided by MULTOS has two purposes: to protect the applications that are loaded onto MULTOS smartcards, and to protect MXI's control of and interest in MULTOS.

A user of a MULTOS-equipped smartcard will be able to select any of the loaded applications and execute them. The user will access the facilities of the smartcard via an appropriate IFD. MULTOS implements a command interface for handling commands received from the IFD.

MULTOS provides a number of system calls (called primitives) which allow the currently executing application to request particular services from MULTOS.

MULTOS provides the following features:

- MULTOS will ensure all requests to load applications are appropriately authorised. MULTOS will support a capability to ensure the authenticity and integrity of an application when loading the application onto the smartcard. MULTOS will also ensure all requests to delete applications are appropriately authorised. Reasons for wishing to delete applications may be because they are found to contain errors, because an updated application is available, or to make room on the smartcard for a more desirable application.

MULTOS will support a capability to load encrypted applications onto the smartcard, decrypt such applications and make them available to the smartcard user for execution.

- MULTOS will ensure no application loaded on the smartcard can interfere with the operation of any other loaded application or with MULTOS. MULTOS will also ensure that an application's code and data will not be available to other applications after it has been deleted.

- MULTOS will provide the capability to authenticate a card as a valid MULTOS equipped smartcard.

- MULTOS will provide the capability to restrict the use of regulated features of the smartcard (e.g., strong cryptography) to authorised applications.

- MULTOS defines certain functions (installing keys, loading applications and deleting applications) as sensitive functions. For each of these functions, if the number of failed attempts to execute the function reaches a pre-defined limit over the life of the smartcard, MULTOS will permanently disable the function. In the case of installing keys, this means the card is unusable, as no applications can be loaded until keys have been installed. In the cases of application loading and deleting, other functions of the card remain available.

It is assumed that authorised applications which are loaded and executed by MULT OS are responsible for the secure processing of their own information. MULTOS provides an environment for secure loading and execution of smartcard applications.

2.2 Target of Evaluation Environment

2.2.1 Target of Evaluation Location and Usage

MULTOS will initially be developed in software. Following successful implementation and testing, the MULTOS executable will be masked in Read Only Memory (ROM) and embedded on smartcards.

Once the MULTOS chip has been embedded on a target smartcard, interaction with it will be via commands issued to the card from an IFD or service requests (i.e., MULTOS system calls, known as primitives) made by an executing application.

2.2.2 Supporting Firmware

MULTOS requires firmware run-time libraries to support writing data to EEPROM. These libraries are supplied by Infineon Technologies. They provide low-level routines to support writing data to EEPROM, which is used on the target smartcard for the storage of applications. MULTOS requires the run-time libraries to execute correctly according to specification, to ensure data is written to the correct address within EEPROM.

2.2.3 Supporting Security Infrastructure

It is assumed MULTOS-equipped smartcards and MULTOS applications will be manufactured and distributed within a commercial framework providing a procedural security infrastructure.
Figure 2 provides a simplified context diagram of the MULTOS commercial framework and security infrastructure.

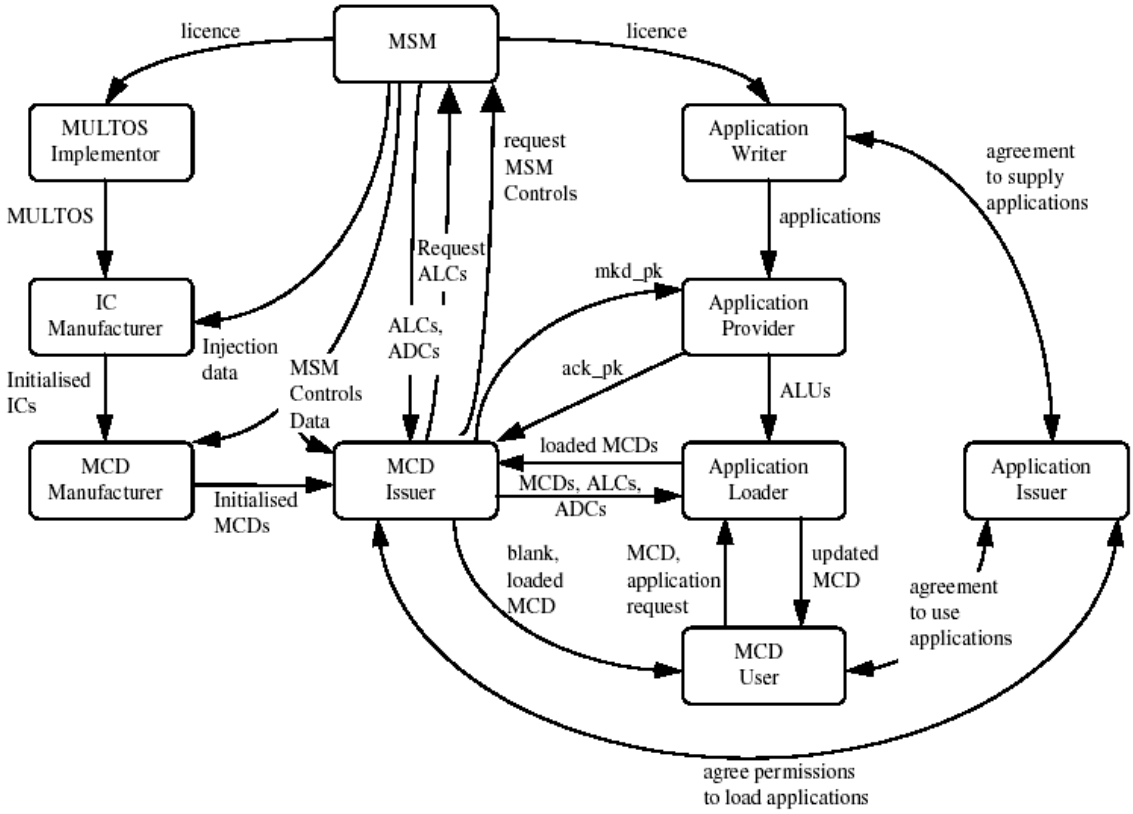

Figure 2: MULTOS Infrastructure Context Diagram

In Figure 2, the box labelled “MSM” includes MXI and MAOSCO in addition to the MSM role.

The following roles and responsibilities are assumed within the infrastructure:

a) **MULTOS Security Manager (MSM):** defines and polices the MULTOS security infrastructure and provides criteria and services necessary for MULTOS participants to operate within the infrastructure. It acts as Certification Authority for the security infrastructure. It is assumed only one MSM exists. The MSM must be trusted by all participants in the infrastructure.

b) **MULTOS Implementor:** the organisation that implements a MULTOS version. The MULTOS Implementor is licensed by MAOSCO and provides its MULTOS version to the IC Manufacturer. The MULTOS Implementor requests the MSM to provide MSM Controls Data, although this may be delivered to the MCD Manufacturer or MCD Issuer.

c) **Integrated Circuit (IC) Manufacturer:** manufacturer of silicon from which chips and smartcards are made. It is assumed the IC Manufacturer is trusted to perform its tasks correctly. It is assumed that all security measures concerned with card manufacture are completed at the IC manufacture stage. This includes:

- The inclusion of security keys in the ROM mask.
- The injection of security data into non-volatile memory.

Security keys and data are provided by the MSM.

The initialised ICs are provided to MCD Manufacturers:

d) **MULTOS Carrier Device (MCD) Manufacturer:** responsible for embedding the IC in its plastic carrier and for background printing on the card. The result is an initialised MCD. This operation is assumed not to be security sensitive. The MCD Manufacturer may also receive MSM Controls Data from the MSM and enable the MCDs. Initialised and enabled MCDs are provided to MCD Issuers.

e) **MCD Issuer:** responsible for issuing to users the MCD itself. The MCD Issuer may also enable initialised MCDs, by loading MSM Controls Data received from the MSM onto the MCDs. MCD Issuers retain the ultimate authority over what applications are loaded on their MCDs. MCD Issuers register applications with the MSM, provide information related to the applications and receive application load and delete certificates from the MSM.

f) **Application Writer:** licensed by MAOSCO to produce applications for MULTOS. Supplies applications under contract to Application Issuers.

g) **Application Issuer:** an organisation that wishes to offer an application to MCD Users. The Application Issuer agrees with an MCD Issuer that the application can be loaded onto MCDs belonging to the MCD Issuer.

h) **Application Provider:** the organisation that takes responsibility for an application, by certifying it with the organisation's public key and encrypting it where necessary. The Application Provider is a role that can be performed by an Application Writer, Application Issuer or MCD Issuer, rather than necessarily, being an organisation in its own right.

i) **Application Loader:** responsible for performing the technical operation of loading applications onto MCDs. The Application Loader enters into an agreement with one or more Application Issuers and MCD Issuers for loading applications supplied by one or more Application Providers.

j) **MCD User:** final user of the MCD.

The MSM authorises potential MULTOS platforms (known as MA-cards). To receive MSM authorisation, a platform must comply with criteria covering attributes of the platform itself and the procedures associated with its manufacture.

MA-cards are assumed to satisfy the following requirements:

a) They are manufactured in a controlled environment conforming to MSM rules.

b) They are subject to type approval by the MSM.

c) They possess a level of tamper resistance.

2.2.4 Application Load Units (ALU)

An Application Load Unit (ALU) is generated by an Application Provider to load applications. An ALU may be uncertified or certified. An uncertified ALU simply contains a clear text copy of the application. A certified ALU contains, in addition to the application, an application signature, which authenticates the application. The Application Provider may also encrypt parts of the application, in which case a Key Transformation Unit is included in the certified ALU.

2.2.5 Key Transformation Unit (KTU)

An Application Provider wishing to utilise application confidentiality will generate a Key Transformation Unit (KTU). The KTU contains descriptors for the areas of the application's code and data that have been encrypted. Each descriptor contains the start address of the protected area, the length of the protected area, an indicator of the algorithm used and the key used to encrypt the contents of the area. The descriptors and some header information (including application identifier and target MCD number) are then encrypted, using the target MCD's public transport key, and included in the KTU.

2.2.6 Application Load and Delete Certificates (ALCs & ADCs)

Application Load Certificates (ALCs) and Application Delete Certificates (ADCs) are generated by the MSM to respectively load and delete an application on to and from an MCD. Each ALC contains the unique Application ID of the application for which it is created. Each ALC refers to a particular domain, which defines the set of MCDs that the application may be loaded on to and deleted from. The domain is defined by a set of load permissions and may be:

a) A specific MCD.

b) A subset of the cards issued by an MCD Issuer.

c) All cards issued by an MCD Issuer.

d) Limited to a subset of cards enabled on specific dates.

e) A combination of the above.

An ALC contains load controls that define exactly what load operations are allowed.
The load controls specify:

a) If application certification has been used.

b) If application confidentiality has been used.

c) If reloading a deleted application is permitted.

The ALC also contains feature permissions, which define what regulated features the application may use. For the initial version of MULTOS, the only regulated features are strong cryptography functions.

The ADC for an application is created at the same time as the ALC. It contains the same unique Application ID and the same set of load permissions as the corresponding ALC.

3. Target of Evaluation Security Environment

This section describes the security aspects of the environment in which the TOE is intended to be used and addresses the description of the assets to be protected, the threats, the organisational security policies and the assumptions.

3.1 Assets

Assets are security relevant elements of the TOE that include:

Assets linked to the IC with Multi-Application Secure Platform itself:

- The IC specifications, design, development tools.
- The IC Dedicated software.
- Multi-Application Platform Software.
- Multi-Application Platform specifications, implementation, test programs and related documentation.
- The TSF data (such as IC and Multi-Application Platform specific data, Initialisation data, IC pre-personalisation requirements and personalisation data,).

Assets linked to the eventual Integrated Applications:

- Native-Application software.
- Native-Application TSF data such as keys and identification data.

Assets are also linked to Loaded-Applications on the platform:

- Application provider User Data:
  - Loaded-Application software loaded on the platform.
  - Loaded-Application SF data. (SF data for the eventual Loaded Application Security Functions)
- The TOE resources:
  - Card resources: memory space and computation power made available to a Loaded-Application and its security functions.

Assets are also linked to end user, card holder and application provider:

- End User Data for users of Native Applications
- End User Data for users of Loaded Applications.

NOTE: even if the PP scope does not include the applications, the TOE must provide security mechanisms such that Native or Loaded Applications can protect the Ed User data when required.

Assets have to be protected in terms of confidentiality, authenticity and control of their origin.

3.2 Assumptions

Security always concerns the whole system. The weakest element of the chain determines the total system security. Assumptions described hereafter must be considered for a secure system using Smartcard products.

3.2.1 Assumptions on the Target of Evaluation Delivery Process (Phases 4 to 7)

Procedures shall guarantee the control of the TOE delivery and storage process and conformance to its objectives as described in the following assumptions:

A.DLV_PROTECT*

Procedures shall ensure protection of TOE material/information under delivery and storage.

A.DLV_AUDIT*

Procedures shall ensure that corrective actions are taken in case of improper operation in the delivery process and storage.

A.DLV_RESP*

Procedures shall ensure that people dealing with the procedure for delivery have got the required skill.

3.2.2 Assumptions on Phases 4 to 6

A.USE_TEST*

It is assumed that appropriate functionality testing of the TOE is used in phases 4, 5 and 6.

A.USE_PROD*

It is assumed that security procedures are used during all manufacturing and test operations through phases 4, 5, 6 to maintain confidentiality and integrity of the TOE and of its manufacturing and test data (to prevent any possible copy, modification, retention, theft or unauthorised use).

3.2.3 Assumption on Phase 7

A.USE_DIAG*

It is assumed that secure communication protocols and procedures are used between Smartcard and terminal.

3.2.4 Assumption on Loaded-Application Development (Phase A1)

A.APPLI_CONT

Whenever a Loaded-Application is to be loaded on the platform, it is assumed that its development and production follow the Administrator Guidance.

3.3 Threats

The TOE as defined in chapter 2 is required to counter the threats described hereafter. A threat agent wishes to abuse the assets either by functional attacks or by environmental manipulation, by specific hardware manipulation, by a combination of hardware and software manipulations or by any other type of attacks.

Threats have to be split in:

- Threats against which specific protection within the TOE is required (class I).
- Threats against which specific protection within the environment is required (class II).

3.3.1 Unauthorised Full or Partial Cloning of the Target of Evaluation

T.CLON*

Functional cloning of the TOE (full or partial) appears to be relevant to all phases of the TOE life-cycle, from phase 1 to phase 7, but only phases 1 and 4 to 7 are considered here, since functional cloning in phases 2 and 3 are purely in the scope of Smartcard IC PP. Generally, this threat is derived from specific threats combining unauthorised disclosure, modification or theft of assets at different phases. This threat addresses User Data and potentially TSF data.

3.3.2 Threats on Phase 1

During phase 1, three types of threats have to be considered:

a) Threats on the Smartcard Embedded Software and its development environment, such as unauthorised disclosure, modification or theft of the Smartcard Embedded Software and/or initialisation data.

b) Threats on the assets transmitted from the IC designer to the Smartcard software developer during the Smartcard ES development.

c) Threats on the Smartcard Embedded Software and initialisation data transmitted during the delivery process from the Smartcard software developer to the IC designer.

Unauthorised disclosure of assets

This type of threat covers unauthorised disclosure of assets by attackers who may possess a wide range of technical skills, resources and motivation. Such attackers may also have technical awareness of the product.

T.DIS_INFO* (type b)

Unauthorised disclosure of the assets delivered by the IC designer to the Smartcard Embedded Software developer, such as sensitive information on IC specification, design and technology, software and tools if applicable.

T.DIS_DEL* (type c)

Unauthorised disclosure of the Asset Smartcard Embedded Software and any additional *application data* (such as IC pre-personalisation requirements) during the delivery to the IC designer.

NOTE application data means TSF data

T.DIS_ES1 (type a)

Unauthorised disclosure of ES (technical or detailed specifications, implementation code) and/or TSF data (such as secrets, or control parameters for protection system, specification and implementation for security mechanisms).

T.DIS_TEST_ES (type a and c)

Unauthorised disclosure of the Smartcard ES test programs or any related information.

Theft or unauthorised use of assets

Potential attackers may gain access to the TOE and perform operations for which they are not authorised. For example, such an attacker may personalise, modify or influence the product in order to gain access to the Smartcard application system.

T.T_DEL* (type c)

Theft of the Smartcard Embedded Software and any additional *application data* (such as pre-personalisation requirements) during the delivery process to the IC designer.

NOTE application data means TSF data

T.T_TOOLS (type a and b)

Theft or unauthorised use of the Smartcard ES development tools (such as PC, development software, databases).

T.T_SAMPLE2 (type a)

Theft or unauthorised use of TOE samples (e.g. bond-out chips with the Embedded Software).

Unauthorised modification of assets

The TOE may be subjected to different types of logical or physical attacks, which may compromise security. Due to the intended usage of the TOE (the TOE environment may be hostile), the TOE security may be bypassed or compromised reducing the integrity of the TOE security mechanisms and disabling their ability to manage the TOE security. This type of threats includes the implementation of malicious Trojan horses.

T_MOD_DEL* (type c)

Unauthorised modification of the Smartcard Embedded Software and any additional *application data* (such as IC pre-personalisation requirements) during the delivery process to the IC designer.

Note: Application data means TSF data.

T.MOD (type a)

Unauthorised modification of ES and/or TSF data or any related information (technical specifications).

3.3.3 Threats on Delivery for/from Phase 1 to Phases 4 to 6

Threats on data transmitted during the delivery process from the Smartcard developer to the IC packaging manufacturer, the Finishing process manufacturer or the Personaliser.

These threats are described hereafter:

T.DIS_DEL1

Unauthorised disclosure of Native-Application and ES personalisation Data during delivery to the IC Packaging manufacturer, the Finishing process manufacturer or the Personaliser.

T.DIS_DEL2

Unauthorised disclosure of Native-Application and ES personalisation Data delivered to the IC Packaging manufacturer, the Finishing process manufacturer or the Personaliser.

T.MOD_DEL1

Unauthorised modification of Native-Application and ES personalisation Data during delivery to the IC Packaging manufacturer, the Finishing process manufacturer or the Personaliser.

T.MOD_DEL2

Unauthorised modification of Native-Application and ES personalisation Data delivered to the IC Packaging manufacturer, the Finishing process manufacturer or the Personaliser.

3.3.4 Threats on Phases 4 to 7

During these phases, the assumed threats could be described in four types:

- Unauthorised disclosure of assets.
- Theft or unauthorised use of assets.
- Unauthorised modification of assets.
- Threats on Native-Applications and on Loaded-Applications.

Unauthorised disclosure of assets

This type of threat covers unauthorised disclosure of assets by attackers who may possess a wide range of technical skills, resources and motivation. Such attackers may also have technical awareness of the product.

T.DIS_ES2

Unauthorised disclosure of ES, Native-Application, and Loaded-Application TSF Data (such as data protection system, memory partitioning, cryptographic programs and keys).

Theft or unauthorised use of assets

Potential attackers may gain access to the TOE and perform operation for which they are not allowed. For example, such attackers may personalize the product in an unauthorised manner, or try to gain fraudulently access to the Smartcard system

T.T_ES

Unauthorised use of TOE. (e.g. bond out chips with embedded software).

T.T_CMD

Unauthorised use of instructions or commands or sequence of commands sent to the TOE.

Unauthorised modification of assets

The TOE may be subjected to different types of logical or physical attacks, which may compromise security. Due to the intended usage of the TOE (the TOE environment may be hostile), the TOE security parts may be bypassed or compromised reducing the integrity of the TOE security mechanisms and disabling their ability to manage the TOE security. This type of threat includes the implementation of malicious Trojan horses, Trapdoors, downloading of viruses or unauthorised programs.

T.MOD_TSF

Unauthorised modification or destruction of TOE Security Function Data. (By any mean including probing, electronic perturbation etc.)

T.MOD_LOAD

Unauthorised loading of Native Applications. This includes also illegal modification of eventual Native Applications. As the TOE described in a Security Target claiming this PP must include eventual Native Applications, their loading or modification must be blocked during the usage phase. The threat includes bypassing this blocking.

T.MOD_EXE

Unauthorised execution of Platform or application software.

T.MOD_SHARE

Unauthorised modification of Platform or application behavior by interaction of different programs.

T.MOD_SOFT*

Unauthorised modification of Smartcard Embedded Software and data.

3.3.5 Threats on Phases 6 to 7

Threats on assets linked to Loaded-Applications

These threats are specific to the Multi-Application Platform, and thus do not appear in PP/9911. They are centered on threats to loading/unloading of Loaded-Applications and to threats using a Loaded-Application to attack another.

T.LOAD_MAN

Loading an application on the platform bypassing the Administrator. This threat could lead to undue usage of card resources, and for unverified application to attack on other Loaded-Application or Native-Application TSF or User data.

T.LOAD_APP

Loading an application that purports to be another Loaded-Application. This attacks card resources and end user data.

T.LOAD_OTHER

Loading the software representation of a Loaded-Application intended for a specific platform domain onto other platform domains, thus taking from the Loaded-Application representation the security feature of being confined to a specific domain. This is an attack on Loaded-Application User Data.

T.LOAD_MOD

Intercepting application load units and altering code or data without the permission of the Loaded-Application Provider. This attacks application provider user data.

T.APP_DISC

Intercepting application load units and gaining access to confidential code or data. This is an attack on application provider user data's confidentiality and knowledge.

T.APP_CORR

Loading an application that partially or completely overwrites other Loaded-Applications, either corrupting or gaining access to code or data. This is an attack on Application Provider user data.

T.APP_REMOVE

Removing a Loaded -Application without the involvement of the Administrator. This is an attack on Application Provider user data.

T.ERR_REMOVE

Removing a Loaded-Application leaving confidential data and/or code in memory which can be examined This is an attack on Application Provider user data.

T.DEL_REMOVE

Removing a Loaded-Application at the same time deleting part or all of another Loaded-Application. This is an attack on Application Provider user data.

T.APP_READ

Using a loaded application to read confidential data or code belonging to another Loaded-Application. This attacks the confidentiality of User Data.

T.APP_MOD

Using a Loaded-Application to modify data or code belonging to another Loaded-Application without its authorisation. This is an attack on Application Provider user data (and also end user data).

T.RESOURCES

Total or partial destruction of card resources delivered by the platform.

NOTE: T.APP_DISC is also present during phase A1.

3.3.6 Threats on Phase 7

Unauthorised disclosure of assets

T.DIS_DATA

Unauthorised disclosure of User (application provider and end user) data and TSF data.

Unauthorised modification of assets

T.MOD_DATA

Unauthorised modification or destruction of User (application provider and end user) Data and TSF data.

Table 2 given below indicates the relationship between the phases of the Smartcard life cycle, the threats and the type of the threats:

Threats	Phase 1	Phase A1	Phase 4	Phase 5	Phase 6	Phase 7
T.CLON*	Class II		Class I	Class I	Class I	Class I
T.DIS_INFO*	Class II					
T.DIS_DEL*	Class II					
T.DIS_DEL1	Class II		Class II	Class II	Class II	
T.DIS_DEL2			Class II	Class II	Class II	
T.DIS_ES1	Class II					
T.DIS_TEST_ES	Class II					
T.DIS_ES2			Class I	Class I	Class I	Class I
T.T_DEL*	Class II					
T.T_TOOLS	Class II					
T.T_SAMPLE2	Class II					

T.T_ES			Class I	Class I	Class I	Class I
T.T_CMD			Class I	Class I	Class I	Class I
T.MOD_DEL*	Class II					
T.MOD_DEL 1	Class II		Class II	Class II	Class II	
T.MOD_DEL 2			Class II	Class II	Class II	
T.MOD	Class II					
T.MOD_TSF			Class I	Class I	Class I	Class I
T.MOD_SOFT*			Class I	Class I	Class I	Class I
T.MOD_LOAD			Class I	Class I	Class I	Class I
T.MOD_EXE			Class I	Class I	Class I	Class I
T.MOD_SHARE			Class I	Class I	Class I	Class I
T.DIS_DATA						Class I
T.MOD_DATA						Class I
T.LOAD_MAN					Class I	Class I
T.LOAD_APP					Class I	Class I
T.LOAD_OTHER					Class I	Class I
T.LOAD_MOD					Class I/II	Class I/II
T.APP_DISC		Class II			Class I/II	Class I/II
T.APP_CORR					Class I	Class I
T.APP_REMOVE					Class I	Class I
T.ERR_REMOVE					Class I	Class I
T.DEL_REMOVE					Class I	Class I
T.APP_READ					Class I	Class I
T.APP_MOD					Class I	Class I
T.RESOURCES					Class I	Class I

Table 2: Relationship between phases and threats

Note: Phases 2 and 3 are covered in the scope of Smartcard IC PP.

3.4 Organisational Security Policies

If an application is approved to uses strong cryptography, it is mandatory for the MCD Issuer to provide a hash of the application in order for the MSM to include it in the Key Header of the Application Load Certificate and so to certify the application.

If an application is not approved to use strong cryptography, it is strongly recommended to MCD issuers to provide also a hash of the application to be included in the Key Header of the Application Load Certificate to certify the application.

4. Security Objectives

The security objectives of the TOE cover principally the following aspects:

- Integrity and confidentiality of assets.
- Protection of the TOE and associated documentation and environment during development and production phases.

4.1 Security Objectives for the Target of Evaluation

The TOE shall achieve the following IT security objectives, and for that purpose, when IC physical security features are used, the specification of those IC physical security features shall be respected. When IC physical security features are not used, the Security Objectives shall be achieved in other ways:

O.TAMPER_ES

The TOE must prevent tampering with its security critical parts. Security mechanisms have especially to prevent the unauthorised change of functional parameters, security attributes and secrets such as the life cycle sequence flags and cryptographic keys.

O.SIDE

The ES must be designed to avoid interpretations of electrical signals from the hardware part of the TOE.

O.CLON*

The TOE functionality must be protected from cloning.

O.OPERATE*

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

O.FLAW*

The TOE must not contain flaws in design, implementation or operation.

O.DIS_MECHANISM2

The TOE shall ensure that the ES security mechanisms are protected against unauthorised disclosure.

O.DIS_MEMORY*

The TOE shall ensure that sensitive information stored in memories is protected against unauthorised disclosure.

NOTE sensitive information means *User Data and TSF data.*

O.MOD_MEMORY*

The TOE shall ensure that *sensitive information* stored in memories is protected against any corruption or unauthorised modification.

NOTE sensitive information means *User Data and TSF data.*

The following security objectives are necessary to meet the new threats specific to Multi-Application Platforms. This is why these objectives are new and not present in PP/9911.

O.ROLLBACK

The TOE must be in a well-defined valid state before a loading of an application, even in case of failure of the previous loading or removal. A failure must not hinder the resources that the TOE can deliver. A rollback operation can be achieved either through specific commands or automatically.

O.RESOURCE

The TOE must provide the means of controlling the use of resources by its users and subjects so as to prevent permanent unauthorised denial of service. (For example it must prevent a Loaded-Application from taking control of the whole permanent memory (EEPROM) thus prohibiting other Loaded-Applications from using it).

O.LOAD

Loaded-Applications are only to be loaded onto a platform with the permission of the administrator.

O.SECURITY

The application load process must be able to guarantee, when required, the integrity, confidentiality, and to verify the claimed origin of the Loaded-Application code and data;

O.EFFECT_L

Loading an application must have no effect on the code and data of existing Loaded-Applications;

O.REMOVE

Removal of a Loaded-Application and consequent reuse of the Loaded-Application space is only to be performed with the authorisation of the administrator. The space must not hold any information relative to data or code linked to the removed Loaded-Application.

O.EFFECT_R

Removal of a Loaded-Application must have no effect on the code and data of the remaining independent Loaded-Applications;

O.SEGREGATE

Loaded-Applications are to be segregated from other Loaded-Applications. A Loaded-Application may not read from or write to another Loaded-Application’s code or data without its authorisation.

4.2 Security Objectives for the Environment

4.2.1 Objectives on Phase 1

O.DEV_TOOLS*

The Smartcard ES shall be designed in a secure manner, by using exclusively software development tools (compilers assemblers, linkers, simulators, etc.) and software-hardware integration testing tools (emulators) that will result in the integrity of program and data.

O.DEV_DIS_ES

The Embedded Software developer shall use established procedures to control storage and usage of the classified development tools and documentation, suitable to maintain the integrity and the confidentiality of the assets of the TOE.

It must be ensured that tools are only delivered and accessible to the parties authorised personnel.

It must be ensured that confidential information on defined assets is only delivered to the parties' authorised personnel on a need-to-know basis.

O.SOFT_DLV*

The Embedded Software must be delivered from the Smartcard software developer (Phase I) to the IC designer through a trusted delivery and verification procedure that shall be able to maintain the integrity of the software and its confidentiality, *if applicable*

NOTE: In this PP it will be always considered applicable.

O.INIT_ACS

Initialisation Data shall be accessible only by authorised personnel (physical, personnel, organisational, technical procedures).

O.SAMPLE_ACS

Samples used to run tests shall be accessible only by authorised personnel.

4.2.2 Objectives on the Target of Evaluation Delivery Process (Phases 4 to 7)

O.DLV_PROTECT*

Procedures shall ensure protection of TOE material/information under delivery including the following objectives:

- Non-disclosure of any security relevant information.
- Identification of the element under delivery.
- Meet confidentiality rules (confidentiality level, transmittal form, reception acknowledgment).
- Physical protection to prevent external damage.

- Secure storage and handling procedures (including rejected TOEs).
- Traceability of TOE during delivery including the following parameters:
- Origin and shipment details.
- Reception, reception acknowledgement.
- Location material/information.

O.DLV_AUDIT*

Procedures shall ensure that corrective actions are taken in case of improper operation in the delivery process (including if applicable any non-conformance to the confidentiality convention) and highlight all non-conformance to this process.

O.DLV_RESP*

Procedures shall ensure that people (shipping department, carrier, reception department) dealing with the procedure for delivery have got the required skill, training and knowledge to meet the procedure requirements and be able to act fully in accordance with the above expectations.

4.2.3 Objectives on Delivery from Phase 1 to Phases 4, 5 and 6

O.DLV_DATA

Native-Application and ES data must be delivered from the Smartcard embedded software developer (phase 1) either to the IC Packaging manufacturer, the Finishing Process manufacturer or the Personaliser through a trusted delivery and verification procedure that shall be able to maintain the integrity and confidentiality of the Native-Application Data.

(Note: some application data are not required for embedding and are then delivered directly to phases 4 to 6).

4.2.4 Objectives on Phases 4 to 6

O.TEST_OPERATE*

Appropriate functionality testing of the TOE shall be used in phases 4 to 6.

During all manufacturing and test operations, security procedures shall be used through phases 4, 5 and 6 to maintain confidentiality and integrity of the TOE and its manufacturing and test data.

4.2.5 Objectives on Phase 7

O.USE_DIAG*

Secure communication protocols and procedures shall be used between the Smartcard and the terminal.

O.CODE_HASH

The documentation relative to the generation of Application Load Certificate MCD issuers shall indicate that:

- If an application is approved to use strong cryptography, it is mandatory to provide a hash of the application in order for the MSM to include it in the Key Header of the Application Load Certificate.

- If an application is not approved to use strong cryptography, it is strongly recommended to provide also a hash of the application to be included in the Key Header of the Application Load Certificate.

4.2.6 Objectives on Loaded-Application Development and Loading (Phases A1 and A2)

This Objective is specific to Loaded-Application development in the Smartcard IC with Multi-Application Platform environment.

O.APPLI_DEV

The Loaded-Application provider must:

- Follow the Administrator Guidance.
- Provide trusted delivery channel so that the integrity and origin of the Loaded-Application can be verified and that its confidentiality can be maintained.

5. IT Security Requirements

This part of the ST defines the detailed IT security requirements that shall be satisfied by the TOE or its environment.

5.1 Target of Evaluation Security Requirements

This chapter defines the functional requirements for the TOE using only functional requirement components drawn from the CC part 2.

The assurance level for this PP is EAL4 augmented. The minimum strength level for the TOE security functions is " SOF-high "(Strength of Functions High).

The assignment and selection operations are written in **bold style** for a better readability.

5.1.1 Security Audit Automatic Response (FAU_ARP)

5.1.1.1 FAU_ARP.1 Security alarms

FAU_ARP.1.1 Abend Iteration. The TSF shall cause **the MCD to abend and become mute** upon detection of a potential security violation.

FAU_ARP.1.1 Shutdown Iteration. The TSF shall cause **the MCD to enter Shutdown mode** upon detection of a potential security violation.

5.1.2 Security audit analysis (FAU_SAA)

5.1.2.1 Potential violation analysis

FAU_SAA.1.1 Abend Iteration. The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.

FAU_SAA.1.2 Abend Iteration. The TSF shall enforce the following rules for monitoring audited events:

a) Accumulation or combination of:

- **An application attempts to execute MEL code outside of its code space or the code space of the codelet that it calls.**
- **An application attempts to access data outside of its data space or the Public data segment.**
- **An apparent corruption of the MSM Controls Data or security data held within the EEPROM of MULTOS.**
- **An unexpected hardware event occurred.**

- **MULTOS determines that it has executed an invalid sequence of instructions (possibly due to electromagnetic or mechanical interference).**
- **An apparent corruption of an application's code space held within the Application Pool Block in the EEPROM of MULTOS.**

known to indicate a potential security violation.

b) **none**.

FAU_SAA.1.1 Shutdown Iteration. The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.

FAU_SAA.1.2 Shutdown Iteration. The TSF shall enforce the following rules for monitoring audited events:

a) Accumulation or combination of:

- **An EEPROM write fails.**
- **A critical process is interrupted.**
- **There have been too many failed attempts to load MSM Controls Data.**

known to indicate a potential security violation.

b) **none**.

5.1.3 Cryptographic key management (FCS_CKM)

5.1.3.1 FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1. The TSF shall perform **a read of cryptographic key** in accordance with a specified cryptographic key access method, **a temporary copy key in RAM** that meets the following: **none**.

5.1.3.2 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, **erasure of a temporary copy key present in RAM**, that meets the following: **none**.

5.1.4 FCS_COP Cryptographic operations

5.1.4.1 FCS_COP.1 Cryptographic operations

FCS_COP.1.1 Iteration 1. The TSF shall perform **digital signature verification** in accordance with a specified cryptographic algorithm **RSA decryption algorithm** and cryptographic key sizes **of 768 bits for modulus and 3 for public exponent** that meet the following **ANSI X9.31, PKCS # 1, PKCS#2 and IEEE-P13-63**.

FCS_COP.1.1 Iteration 2. The TSF shall perform **application code integrity checking and MCD authentication** in accordance with a specified cryptographic algorithm **asymmetric hash algorithm** and cryptographic key sizes **of 576 bits for hash modulus and 3 for public exponent** that meet the following **none**.

FCS_COP.1.1 Iteration 3. The TSF shall perform **application code integrity checking** in accordance with a specified cryptographic algorithm **secure hash algorithm (SHA-1)** and cryptographic key sizes **no key** that meet the following **FIPS PUB 180-2**.

FCS_COP.1.1 Iteration 4. The TSF shall perform **recovering of protected code or data segments of an application and recovering of MSM Controls Data** in accordance with a specified cryptographic algorithm **DES decryption** and cryptographic key sizes **of 8 byte (single key) or 16 byte (double key)** that meet the following **FIPS-PUB 46-3**.

5.1.5 Access control policy FDP_ACC

5.1.5.1 FDP_ACC.2 Complete access control

FDP_ACC.2.1 Load Application SFP Iteration. The TSF shall enforce the **Load Application SFP** on **MULTOS ES and Application Load Certificate**, and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.1 Delete Application SFP Iteration. The TSF shall enforce the **Delete Application SFP** on **MULTOS** ES and Application Delete Certificate, and all operations among subjects and objects covered by the SFP.

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

5.1.6 Access control functions FDP_ACF

5.1.6.1 FDP_ACF.1 Security attribute based access control

FDP_ACF.1.1 Load Application SFP Iteration. The TSF shall enforce the **Load Application SFP** to objects based on **Unique Application Identifier present in the ALC, Unique Application Identifier of loaded-applications, MCD Enabled Flag, Application Load Permissions, MCD Load Permissions, History List.**

FDP_ACF.1.2 Load Application SFP Iteration. The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed. **See the table below.**

FDP_ACF.1.3 Load Application SFP Iteration. The TSF shall explicitly authorize access of subjects to objects based on the following additional rules. **See the table below.**

FDP_ACF.1.4 Load Application SFP Iteration. The TSF shall explicitly deny access of subjects to objects based on the **table below.**

Security attributes	Governing access rules	Authorizing access rules	Denying access rules
Unique Application Identifier present in the ALC and Unique Application Identifier of loaded-applications	Verify there is other application currently loaded on this MCD with the same Application Identifier.	Establish that there is no other application currently loaded on this MCD with the same Application Identifier. Application load process continues.	Establish that there is another application currently loaded on this MCD with the same Application Identifier. Application load process is aborted.
MCD enabled flag	Verify the MCD is enabled (ie that the MSM Controls Data for this MCD has been installed)	Establish that the MCD is enabled (ie that the MSM Controls Data for this MCD has been installed). Application load process continues.	Establish that the MCD is not enabled (ie that the MSM Controls Data for this MCD has not been installed). Application load process is aborted.
Application Load Permissions and MCD Load Permissions	Verify the application load permissions are compatible with the MCD permissions which were installed when the card was enabled	Establish that the application load permissions are compatible with the MCD permissions which were installed when the card was enabled. Application load process continues.	Establish that the application load permissions are not compatible with the MCD permissions which were installed when the card was enabled. Application load process is aborted.
History list	Determine if the application is being re-load a second time on to this MCD, and whether that is permitted	If the application is being re-load a second time on to this MCD, and that is permitted. Application load process continues.	If the application is being re-load a second time on to this MCD, and that is not permitted. Application load process is aborted.

FDP_ACF.1.1 Delete Application SFP Iteration. The TSF shall **enforce the Delete application SFP** to objects based on **Unique Application Identifier present in the ADC, Unique Application Identifier of loaded-applications, Application Load Permissions, MCD Load Permissions.**

FDP_ACF.1.2 Delete Application SFP Iteration. The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed. **See the table below.**

FDP_ACF.1.3 Delete Application SFP Iteration. The TSF shall explicitly authorize access of subjects to objects based on the following additional rules. **See the table below.**

FDP_ACF.1.4 Delete Application SFP Iteration. The TSF shall explicitly deny access of subjects to objects based on the **table below.**

Security attributes	Governing access rules	Authorizing access rules	Denying access rules
Unique Application Identifier present in the ADC and Unique Application Identifier of loaded-applications	Verify an application with the Application Identifier specified in the ADC is loaded on this MCD	Establish that an application with the Application Identifier specified in the ADC is loaded on this MCD. Application deletion process continues.	Establish that no application with the Application Identifier specified in the ADC is loaded on this MCD. Application deletion process is aborted.
Application Load Permissions and MCD Load Permissions	Verify the application permissions are compatible with the MCD permissions which were installed when the card was enabled	Establish that the application permissions are compatible with the MCD permissions, which were installed when the card was enabled. Application deletion process continues.	Establish that the application permissions are not compatible with the MCD permissions, which were installed when the card was enabled. Application deletion process is aborted.

5.1.7 Data authentication FDP_DAU

5.1.7.1 FDP_DAU.1 Basic data authentication

FDP_DAU.1.1. The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of **application's code spaces.**

FDP_DAU.1.2. The TSF shall provide **MULTOS** with the ability to verify evidence of the validity of the indicated information.

5.1.8 Export to outside TSF control FDP_ETC

5.1.8.1 FDP_ETC.1 Export of user data without security attributes

FDP_ETC.1.1. The TSF shall enforce the **Load Application SFP and Delete Application SFP** when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.1.2. The TSF shall export the user data without the user data's associated security attributes.

5.1.9 Import from outside TSF control FDP_ITC

5.1.9.1 FDP_ITC.1 Import of user data without security attributes

FDP_ITC.1.1. The TSF shall enforce the **Load Application SFP and Delete Application SFP** when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.1.2. The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

FDP_ITC.1.3. The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: **none**.

5.1.10 Residual information protection FDP_RIP

5.1.10.1 FDP_RIP.1 Subset residual information protection

FDP_RIP.1.1. The TSF shall ensure that any previous information content of a resource is made unavailable upon the **de-allocation of the resource from** the following objects: **application's code and data spaces.**

5.1.11 Rollback (FDP_ROL)

5.1.11.1 FDP_ROL.1 Basic rollback

FDP_ROL.1.1. The TSF shall enforce **Load Application SFP** to permit the rollback of the **load of an application** on the **application's code and data.**

FDP_ROL.1.2. The TSF shall permit operations to be rolled back within **a failure occurs during loading of an application.**

5.1.12 Stored data integrity (FDP_SDI)

5.1.12.1 FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1. The TSF shall monitor user data stored within the TSC for **memory corruption** on all objects, based on the following attributes: **four-byte check sum**.

FDP_SDI.2.2. Upon detection of a data integrity error, the TSF shall **abend the current session**.

5.1.13 Authentication failures (FIA_AFL)

5.1.13.1 FIA_AFL.1 Authentication failure handling

FIA_AFL.1.1. The TSF shall detect when **20** unsuccessful authentication attempts occur related to **execution of SetMSMControls command, DeleteMELApplication command and CreateMELApplication command.**

FIA_AFL.1.2. When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall **permanently disable the incriminated command.**

5.1.14 User attribute definition (FIA_ATD)

5.1.14.1 FIA_ATD.1 User attribute definition

FIA_ATD.1.1. The TSF shall maintain the following list of security attributes belonging to individual users: **See the table below**.

User	Security attribute
MULTOS Security Manager	Global Key Certification Key (kck)
	MCD-specific Asymmetric Transport Key (mkd)
	MCD-specific Transport Key variable (tkv)
Application Provider	Application Provider's Asymmetric Key (ack)
MCD Issuer	MCD Issuer Identifier

5.1.15 User Authentication (FIA_UAU)

5.1.15.1 FIA_UAU.1 Timing of authentication

FIA_UAU.1.1. The TSF shall allow **processing of Check Data command** on behalf of the user to be performed before the user is authenticated.

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

5.1.15.2 FIA_UAU.4 Single-use Authentication Mechanisms

FIA_UAU.4.1. The TSF shall prevent reuse of authentication data related to **application's load and delete authentication mechanisms.**

5.1.16 User identification (FIA_UID)

5.1.16.1 FIA_UID.1 Timing of identification

FIA_UID.1.1. The TSF shall allow **processing of Check Data command** on behalf of the user to be performed before the user is identified.

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

5.1.17 User-subject Binding (FIA_USB)

5.1.17.1 FIA_USB.1 User-subject binding

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

5.1.18 Management of function in the TSF (FMT_MOF)

5.1.18.1 FMT_MOF.1 Management of security functions behaviour

FMT_MOF.1.1. The TSF shall restrict the ability to **determine the behaviour of** the functions **Application Load Certificate Control SF and Application Deletion Certificate Control SF** to **MSM**.

FMT_MOF.1.1. The TSF shall restrict the ability to **enable** the functions **Application Load Certificate Control SF and Application Deletion Certificate Control SF** to **MSM**.

5.1.19 Management of security attributes (FMT_MSA)

5.1.19.1 FMT_MSA.1 Management of security attributes

FMT_MSA.1.1. The TSF shall enforce the **Load Application SFP and Delete Application SFP** to restrict the ability to **load** the following security attributes to **the MSM**:

- **MCD Issuer Product Identifier.**
- **MCD Issuer Identifier.**
- **MCD Batch Number.**
- **RFU 2 (Reserved for Future Use).**
- **RFU 4.**
- **RFU 5.**
- **RFU 6.**
- **MCD-unique Identifier.**
- **asymmetric transport key set (mkd).**

5.1.19.2 FMT_MSA.2 Secure security attributes

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

5.1.19.3 FMT_MSA.3 Static attribute initialisation

FMT_MSA.3.1. The TSF shall enforce the **Load Application SFP and Delete Application SFP** to provide **MSM Controls Data** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2. The TSF shall allow the **MSM** to specify alternative initial values to override the default values when an object or information is created.

5.1.20 Management of TSF data (FMT_MTD)

5.1.20.1 FMT_MTD.1 Management of TSF data

FMT_MTD.1.1. The TSF shall restrict the ability to **load** the **MSM Controls Data** to **MSM**.

5.1.20.2 FMT_MTD.2 Management of limits on TSF data

FMT_MTD.2.1. The TSF shall restrict the specification of the limits for **MSM Controls Data** to **MSM**.

FMT_MTD.2.2. The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: **MCD becomes mute**.

5.1.21 Security management roles (FMT_SMR)

5.1.21.1 FMT_SMR.1 Security roles

FMT_SMR.1.1. The TSF shall maintain the roles:

- **MULTOS Security Manager (MSM)**
- **MCD Issuer**
- **Application Provider**

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

5.1.22 Unobservability (FPR_UNO)

5.1.22.1 FPR_UNO.1 Unobservability

FPR_UNO.1.1. The TSF shall ensure that **any users** are unable to observe the **cryptographic** operations on **Application Load Certificate, Application Delete Certificate, Application Load Unit, MSM Controls Data and hash digest of the contents of a selected area of MCD's memory** by **MULTOS**.

The functional requirement must be understood in the sense of protection against observation of the mechanisms and TSF data used and of User data manipulated during the operation. The intent is to protect against side channel attacks.

5.1.23 Fail secure (FPT_FLS)

5.1.23.1 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:

a) An application attempts to execute MEL code outside of its code space or the code space of the codelet that it calls

b) An application attempts to access data outside of its data space or the Public data segment

c) An apparent corruption of the MSM Controls Data or security data held within the EEPROM of MULTOS

d) An unexpected hardware event occurred

e) MULTOS determines that it has executed an invalid sequence of instructions (possibly due to electromagnetic or mechanical interference)

f) An EEPROM write fails

g) A critical process is interrupted

h) There have been too many failed attempts to load MSM Controls Data.

5.1.24 TSF Physical protection (FPT_PHP)

5.1.24.1 FPT_PHP.3 Resistance to physical attack

FPT_PHP.3.1. The TSF shall resist the **following physical tampering scenarios** to the **following list of TSF devices/elements** by responding automatically such that the TSP is not violated.

Physical tampering scenarios	**TSP devices/elements**
Abnormal use of reset signal	All TSF devices/elements
Abnormal use of power signal	All TSF devices/elements
Clock rate variations	The processor
Dynamic power analysis	Cryptographic operations

5.1.25 Trusted recovery (FPT_RCV)

5.1.25.1 FPT_RCV.4 Function recovery

FPT_RCV.4.1. The TSF shall ensure that the **following list of SFs and failure scenarios** have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

Security functions	Failure scenarios
Application Load Certificate Control SF	Reser/power down during command processing
Application Delete Certificate Control SF	Reser/power down during command processing Too many failed Delete Command
Unprotected/Protected Application Load Unit SF	Reser/power down during command processing Too many failed Create Command
Confidential Application Load Unit SF	Reser/power down during command processing Too many failed Create Command
MSM Controls Data Load Management SF	Reser/power down during command processing Too many failed Set MSM Controls Command
Application Execution Management SF	Application abend Reser/power down during command processing or application execution
Critical Data Overwrite SF	Reser/power down during command processing or application execution
Reset Protection SF	Reser/power down during command processing or application execution
Integrity Checks SF	Reser/power down during command processing or application execution
Start-up Validity Checks and Initialisation SF	Reser/power down during command processing or application execution
All Security Functions	EEPROM write failure Power loss Integrity failure

5.1.26 Reference mediation (FPT_RVM)

5.1.26.1 FPT_RVM.1 Non-bypassability of the TSP

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

5.1.27 Domain separation (FPT_SEP)

5.1.27.1 FPT_SEP.1 TSF Domain separation

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

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

5.1.28 Inter-TSF TSF data consistency (FPT_TDC)

5.1.28.1 FPT_TDC.1 Inter-TSF basic TSF data consistency

FPT_TDC.1.1. The TSF shall provide the capability to consistently interpret **Application Load Certificate, Application Delete Certificate, Key Transformation Unit, Application Provider Signature and MSM Controls Data** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2. The TSF shall use **signatures format on the certificates, the Application Load Unit, the Key Transformation Unit and MSM Controls Data** when interpreting the TSF data from another trusted IT product.

5.1.29 TSF self test (FPT_TST)

5.1.29.1 FPT_TST.1 TSF Testing

FPT_TST.1.1. The TSF shall run a suite of self tests **at the conditions when MULTOS is powered-down/powered-up or reset** to demonstrate the correct operation of the TSF.
FPT_TST.1.2. The TSF shall provide authorised users with the capability to verify the integrity of TSF data.
FPT_TST.1.3. The TSF shall provide authorised users with the capability to verify the integrity of the stored TSF executable code.

5.1.30 Resource allocation (FRU_RSA)

5.1.30.1 FRU_RSA.1 Maximum quotas

FRU_RSA.1.1. The TSF shall enforce maximum quotas of the following resources: **EEPROM and X-RAM** that **applications, functions, codelets and primitives** can use **simultaneously**.

5.2 TOE Security Assurance Requirements

The Assurance requirement is EAL4 augmented with additional assurance components listed in the following section. These components are hierarchical ones to the components specified in EAL4.

5.2.1 ADV_IMP.2: Implementation of the TSF

Developer action elements:

ADV_IMP.2.1D. The developer shall provide the implementation representation for the entire TSF.

Content and presentation of evidence elements:

ADV_IMP.2.1C. The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.2.2C. The implementation representation shall be internally consistent.

ADV_IMP.2.3C. The implementation representation shall describe the relationships between all portions of the implementation.

Evaluator action elements:

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

ADV_IMP.2.2E. The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

Dependencies:

ADV_LLD.1. Descriptive low-level design.

ADV_RCR.1. Informal correspondence demonstration.

ALC_TAT.1. Well defined development tools.

5.2.2 ALC_DVS.2: Sufficiency of Security Measures

Developer action elements:

ALC_DVS.2.1D. The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.2.1C. The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.2.2C. The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C. The evidence shall justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

Evaluator action elements:

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

ALC_DVS.2.2E. The evaluator shall confirm that the security measures are being applied.

Dependencies:

No dependencies.

5.2.3 AVA_VLA.4: Highly Resistant

Developer action elements:

AVA_VLA.4.1D. The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.4.2D. The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.4.1C. The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.4.2C. The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.4.3C. The evidence shall show that the search for vulnerabilities is systematic.

AVA_VLA.4.4C. The analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.

Evaluator action elements:

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

AVA_VLA.4.2E. The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.4.3E. The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.4.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.4.5E. The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential.

Dependencies:

ADV_FSP.1. Informal functional specification.

ADV_HLD.2. Security enforcing high-level design.

ADV_IMP.1. Subset of the implementation of the TSF.

ADV_LLD.1. Descriptive low-level design.

AGD_ADM.1. Administrator guidance.

AGD_USR.1. User guidance.

5.3 Security Requirements for the IT Environment

The TOE has no asserted dependencies on the IT environment.

6. Target of Evaluation Summary Specification

6.1 Target of Evaluation Security Functions

This section defines the TOE security functions. The *italic paragraph parts* correspond to *actions provided by the security functions* whereas the normal paragraph parts correspond to the context in which the security functions take place.

The minimum strength level of these security functions is "SOF high". Note that cryptographic primitives are out of scope.

Table 3 shows how these security functions satisfy the TOE security functional requirements.

6.1.1 Application Load Certificate Control SF (SF1)

SF1 ensures that the MSM Controls Data has been loaded before loading any application.
SF5 maintains a flag to indicate whether or not MSM Controls Data has been loaded successfully onto the MCD. This flag is contained in MULTOS security data.

SF1 authenticates an application load certificate as having been authorised by the MSM, using the MSM's GKCK (kck_pk), prior to validating the loaded-application. SF1 calculates an asymmetric hash of the key header and compares it with the deciphered Key Certificate, using a RSA algorithm and kck_pk.

SF1 ensures an authorised application has appropriate permissions (MCD Issuer product identifier, MCD Issuer identifier, MCD enables dates, MCD number, four permissions field reserved for future use) before load is validated. In this way, SF1 checks the eight application's permissions against the eight MCD's permissions.

SF1 checks if an application, which has been previously loaded and then, deleted, is authorised by the MSM to be reloaded. The ALC contains a value that indicates if reloads of the application onto the same MCD are authorised. The value can be zero or a random number generated by the MSM.
A value of zero means that the MSM has authorised multiple reloads of the application.

SF1 ensures the application is not already loaded on the MCD. When an attempt is made to load the application, the AID (unique Application Identifier) contained in the ALC is checked against the AID associated with each application already loaded on the MCD. If a match is found, this indicates the application has already been loaded onto the MCD and the load attempt will fail.

SF1 supports a means to ensure an application's code loaded onto the MCD is the same as the code originally approved for access to strong cryptography primitives.

If an application is approved to use strong cryptography, a one-way cryptographic hash (asymmetric hash or SHA-1 algorithm) of its code segment is created and submitted to the MSM as part of the application details when the MCD Issuer requests ALCs. MSM authorises the hash value by including it in the signed ALC.
When the application is loaded on to an MCD, SF1 calculates the hash value over the application's code segment and compares it with the hash value contained in the ALC.
If the two values differ, the load request is rejected.

Although the Application Writer is responsible for obtaining approval to use strong cryptography in an application, the MCD Issuer determines if application authentication will be applied. Therefore, if the MCD Issuer wished to change the application's code, it could do so, specify that application authentication was not required and then load arrange for an application not approved for strong cryptography to be loaded onto an MCD. The hash value mechanism is designed to counter this unauthorised use of strong cryptographic primitives. This mechanism depends on appropriate supporting procedures, to ensure MSM is provided with a trustworthy hash value for inclusion in the ALC.

When loading the ALU components in the Application Pool Block in EEPROM, SF1 checks if there is enough space available. If it is not the case, SF1 returns an error.

If load application fails, SF1 ensures that the temporary loaded-application is erased.

Permutational/probabilistic/cryptographic mechanisms used in this security function: RSA algorithm, asymmetric hash algorithm, and SHA-1 algorithm.

6.1.2 Application Delete Certificate Control SF (SF2)

SF2 authenticates the Application Delete Certificate with the key kck_pk that is stored in the MCD's ROM. The authentication is done through an asymmetric hash of the ADC and a comparison with the signature provided.

SF2 delete an application only after receiving and authenticating a valid application delete certificate. SF2 checks that the Application ID extracted from the certificate matches to a loaded application and that the permissions are correct (the delete process uses the same interpretation of permissions as the load process). Only after all these checks have passed will SF2 delete the application.

Permutational/probabilistic/cryptographic mechanisms used in this security function: RSA algorithm, asymmetric hash algorithm.

6.1.3 Unprotected/Protected Application Load Unit SF (SF3)

SF3 manages the Unprotected/Protected Application Load Unit which is composed of the Application Code (clear text copy of the application) and the Application Signature (for the protected ALU).
The protected ALU is used for application authentication.

If the ALC indicates that application authentication is required (it is optional), the application is authenticated by its application signature. *When application authentication is invoked, SF3 verifies the authenticity of the application. Once the application has been loaded onto the MCD, SF3 creates a digest of the application using the one-way hash function. SF3 then decrypts the application signature using the Application Provider's public key (ack_pk), which is contained within the ALC.* ack_pk is certified by the MSM. The MSM signs ack_pk using the secret GKCK (kck_sk). SF1 can verify the authenticity of ack_pk by decrypting it with the public GKCK (kck_pk). *The hash digest from within decrypted signature is compared with the application digest generated by SF3.* If they are equal, then the authenticity of the application is confirmed, since only the Application Provider could create the application signature and the Application Provider's public key is certified by the MSM. *If the decrypted signature does not match the application digest generated by SF3, application authentication fails and SF3 deletes the loaded application from the MCD.*

Permutational/probabilistic/cryptographic mechanisms used in this security function: RSA algorithm, asymmetric hash algorithm.

6.1.4 Confidential Application Load Unit SF (SF4)

SF4 manages the Confidential Application Load Unit which is composed of the Application Code (clear text copy of the application), the Application Signature (for application authentication) and the Key Transformation Unit (for application confidentiality).

If application confidentiality is required (it is optional) SF4 allows the loading of applications which have protected areas of code or data. In order to protect the confidentiality of an application, the Application Provider is able to encrypt the relevant areas of the application using DES CBC or Triple DES CBC. The DES or Triple DES encryption key and descriptors for each of the encrypted areas are then encrypted using the public transport key (mkd_pk) of the MCD onto which the application is to be loaded. This information is placed into a KTU. The KTU is appended to the ALU. This ensures the confidentiality of sensitive parts of an application before it is loaded onto an MCD.

Once the application is loaded, SF4 allows the decryption of the protected areas of the application so that the application can be securely executed on the MCD.

Once SF1 authenticates the Application Load Certificate, SF4 decrypts the KTU using the MCD-specific secret transport key (mkd_sk). SF4 uses the DES or Triple DES key recovered from the decrypted KTU to decrypt the protected application areas and complete the process of loading the application. This ensures that the protected application can be executed once it is safely loaded onto the MCD (where its confidentiality is protected by the Application Execution Management SF).

SF4 ensures only an authentic MCD is able to load and execute a protected application.

Since the MCD-specific secret transport key is required in order to decrypt the protected application areas, only the target MCD can gain access to those areas.

By successfully decrypting the KTU and recovering the DES or Triple DES key to decrypt the protected application areas, SF4 also authenticates the MCD as a valid MCD. This ensures the confidentiality of the protected application in the event it is loaded onto a smartcard that is not an authentic MCD.

Permutational/probabilistic/cryptographic mechanisms used in this security function: RSA algorithm, DES algorithm.

6.1.5 MSM Controls Data Load Management SF (SF5)

During implementation of MULTOS in silicon for the target processor, MULTOS security data is injected into non-volatile memory. The MULTOS security data includes an MCD-unique identifier (the MCD id), an MCD-unique symmetric transport key (tkv) and a security flag.

MSM Controls Data for a specific MCD includes the MCD-unique identifier, the MCD's permissions and the MCD-unique transport key (mkd). MSM Controls Data is encrypted by the MSM using the MCD-unique symmetric transport key (tkv). MSM Controls Data is provided to the MCD Issuer for loading on the target MCD.

SF5 ensures the MCD only allow MSM Controls Data to be loaded once. The MCD Issuer presents the MSM Controls Data to the target MCD. This is done by submitting the Set MSM Controls command

to MULTOS via an IFD. *Before loading, SF5 checks the security data flag to verify if the MSM Controls Data has not already been installed. If the MSM Controls Data have already been loaded, the attempt to load is denied.*

SF5 ensures encrypted MSM Controls Data is able to be loaded on target MCD. Since MSM Controls Data is encrypted using a symmetric key specific to the target MCD, only the target MCD is able to decrypt the data and load it successfully. Furthermore, the MCD is able to load only its own MSM Controls Data, since it will not be able to decrypt any other MCD's MSM Controls Data.

This ensures an MCD cannot load MSM Controls Data intended for another MCD and therefore cannot masquerade as another MCD (e.g., in order to load applications not intended for it).

SF5 verifies the integrity of the MSM Controls Data. SF5 generates a hash digest of the MSM Controls Data (less the last 16 bytes) and compares it with the attached hash digest (last 16 bytes of the decrypted data). If the two digests match, the MSM Controls Data has been received without corruption or tampering.

SF5 ensures the MCD only loads its own unique MSM Controls Data. If the decrypted MCD-unique identifier does not match the MCD-unique identifier stored in non-volatile memory of the targeted MCD, the MSM Controls Data is rejected.

If the MSM Controls Data is loaded successfully, SF5 sets the security flag to '0x5A' in MULTOS security data to indicate this has occurred. This ensures that once the MCD Issuer has enabled and issued the MCD, bogus MSM Controls Data cannot be created and loaded onto the MCD.

Permutational/ probabilistic/cryptographic mechanisms used in this security function: DES algorithm, asymmetric hash algorithm.

6.1.6 Application Execution Management SF (SF6)

SF6 ensures each application is restricted to accessing its own code and data. The only exceptions to the restriction on an application's code and data access are as follows:

a) Accessing data in the Public Data Area.

b) Application delegation.

c) Accessing Codelets.

d) Accessing data via MULTOS primitives.

SF6 also allows strong cryptography provided by the MCD to be regulated so that only authorised applications can access them.

SF6 maintains separate storage and execution space for applications loaded onto an MCD. SF6 manages a pool of loaded applications. SF6 ensures each application, including its code and data areas, is kept separate from every other application loaded on the MCD. This ensures an application that is restricted to its own code and data space cannot gain access to the code or data of another loaded application. Each application is allocated to its own Application Pool Block within the Application Pool. Each Application Pool Block contains a unique identifier of the application loaded into the block. The intent of this mechanism is to allocate a portion of EEPROM memory to an application where that portion does not overlap regions allocated to any other applications and to tag these regions of memory with the application ID of that application.

An application is able to read code for execution only from its own code space or from a pool of common routines controlled by SF6. SF6 executes only applications written in the MULTOS Execution Language (MEL). MEL is an interpreted language. MEL applications are executed on an Application Abstract Machine, which enables memory accesses by applications to be checked at the time of interpretation**.** *SF6 ensures any attempt by an application to access code for execution is restricted to its own code space or to Codelets, which are controlled by SF6.* This ensures an application is unable to compromise the integrity or confidentiality of the code of other applications loaded on the MCD.

SF6 ensures no application is able to write to the code space of any application, including itself. SF6 ensures any attempt by an application to write data is restricted to the application's own data space or to the Public data area. Any attempt by the application to write data outside these areas, including to its own or another applications code space, is blocked by SF6 and the application is terminated.

SF6 ensures no application is able to read from or write to the data space of another application except via a mechanism provided and controlled by SF6. SF6 ensures any attempt by an application to read or write data is restricted to the application's data space or to the Public data area. The Public data area is available for reading and writing by all applications and provides the mechanism for applications to communicate information with each other. This ensures an application is unable to compromise the integrity or confidentiality of the data of other applications loaded on the MCD.

No application is able to cause the execution of another application except via a mechanism provided and controlled by SF6. SF6 also provides a mechanism for an application to delegate execution to another application. On delegation, a full context switch occurs, so the only information from the delegating application which is available to the delegated application is whatever might be held in the Public data area. ("Full context switch" means that SF6 writes all information related to the execution of the delegating application to an area under its control, then commences execution of the delegated application. When the delegated application ends its execution, the execution context of the delegating application is restored and it is able to continue execution from the point of delegation.) occurs, so the only information from the delegating application which is available to the delegated application is whatever might be held in the Public data area.
This ensures an application cannot make use of another application to compromise the integrity or confidentiality of other applications. Applications execute only within their own environment and cannot be made to execute in another application's environment.

SF6 ensures no application is able to write to the code space of MULTOS and no application is able to read from or write to the data space of MULTOS except via a mechanism provided and controlled by SF6. SF6 provides system primitives that can be invoked by applications, which return to the application specific system data values and allow specific system data values to be updated. Any other attempt by an application to access MULTOS code or data is blocked by SF6.
This ensures no application is able to compromise the integrity of MULTOS or the confidentiality of its sensitive information.

SF6 ensures only applications specifically authorised by the MSM can access strong cryptography primitives. The ALC contains a flag indicating whether or not the application is authorised to use MULTOS's strong cryptography primitives. This information is stored with the application when it is loaded onto the MCD. *Every time an application attempts to call a strong cryptography primitive, SF6 checks the control flag to determine if the application is allowed to make the call. If it is, SF6 will process the call. If the flag indicates access is not authorised, SF6 will return an error condition to the application.* The MSM wishes to control which applications can access strong cryptography.
This is necessary to comply with government restrictions on the use by Application Writers of strong cryptography. An Application Writer must obtain appropriate documentation (e.g., an export license) from the appropriate government body before the MSM will authorise the application's use of strong

cryptography. The MSM authorises an application to use strong cryptography by digitally signing its ALC with the cryptography access flag set to allowed.

SF6 ensures a series of functions that allow MULTOS to address all required X-RAM and EEPROM it needs as follows: MULTOS needs to be able to access data held in up to 64K of EEPROM/X-RAM.

6.1.7 Critical Data Overwrite SF (SF7)

SF7 ensures that no part of an application's code or data, excluding data the application has placed into the Public data area, can be accessed after the application has been deleted.
When SF7 deletes an application from the MCD, it overwrites the application's code and data spaces with a fixed pattern of bytes. In this way, any other application subsequently loaded into the same space will be unable to determine any information relating to the deleted application.
Data that the application has written to the Public data area is not overwritten, since this provides the means for the application to communicate with other applications. By placing data in the Public data area, an application is effectively deciding the data can be accessed by any application.

6.1.8 Reset Protection SF (SF8)

When allocating memory to an application, a number of pointers must be manipulated.
These pointers are held in EEPROM memory and are susceptible to corruption if the MCD should loose power while being updated. *To protect against this, SF8 establishes a critical region around the operations that update these pointers. If the MCD is powered down or reset while in this critical region, SF8 will permanently shutdown the MCD. In this way, SF8 ensures that critical memory allocation operations occur as an atomic operation (i.e., they are either not initiated or are guaranteed to complete).*

6.1.9 Integrity Checks SF (SF9)

SF9 protects MULTOS critical data by applying an integrity check to the following information:

a) MISA injected security data.

b) MSM Controls Data.

c) Application code spaces.

SF9 calculates a four-byte check sum over the MISA (MULTOS Injection Security Application) injected security data and the MSM controls data when the MSM controls have been successfully set. This check sum is re-calculated and verified by SF9 before sending out a response to any command. A failure of the check sum causes SF9 to abend the session. This integrity check hence ensures that the smartcard will not attempt to send any response that may be based on corrupted data.

In addition, when an application is loaded onto the MCD (by successful execution of the Create MEL Application command), SF9 calculates a four-byte check sum over the application's code space. SF9 stores this check sum in the application pool block for the application. When an application is selected as the current file, SF9 calculates the check sum over its code space and compares it with the stored check sum to confirm the continued integrity of the application. If the calculated and stored check sums do not match, SF9 abends the session.

A full integrity check verifies the checksum of the full MSM Controls data whereas a partial integrity check excludes the verification of the codelets within the MSM Controls data. A full integrity check is performed at startup and a partial integrity check is performed just prior to sending a response to every command in order to make sure that security data and MSM Controls data remain unchanged.

Therefore, MULTOS is not vulnerable to attempts to corrupt its memory.

Permutational/probabilistic/cryptographic mechanisms used in this security function: 4-byte checksum.

6.1.10 Start-up Validity Checks and Initialisation SF (SF10)

If the MCD is reset or loses power while MULTOS is processing a command or executing an application, SF10 will perform the usual validity checks and initialisation when MULTOS is restarted:

a) The MCD validity check allows SF10 to determine that MULTOS is still in a valid state (if it is not, SF10 will shutdown permanently).

b) SF10 erases the Public data area (to protect any sensitive information placed there by an application executing at the time of reset/power loss).

c) SF10 erases from the Application Pool any application in the Application Pool that is in the "opened" state (since the application load process has been interrupted).

d) SF10 initialises the Active Application Block to the shell application if any is present, or otherwise to a null value to indicate that no application is currently selected.

e) SF10 rolls back any uncommitted writes in the Data Item Buffer.

The validity check can fail for the following reasons:

a) An attempt to write to EEPROM has failed; when EEPROM (which has a limited life span) starts to fail, the MCD can no longer function correctly and SF10 needs to shutdown the MCD.

b) *A check of the integrity of the security data (comprising Initialisation Security Data and MSM Controls Data) fails, the MCD can no longer function correctly and SF10 abends.*
A change to the MCD's security data could indicate an attempt to attack the MCD or a failure of the MCD memory.

c) *The Application Memory Manager module detects it was in the middle of a critical operation when the system was reset. SF10 permanently shuts down the MCD* in this circumstance.
The Memory Manager Software Module's data will be inconsistent, with no means to recover it to a consistent state. Critical operations involve the manipulation of memory addresses associated with an application and cannot be recovered.

d) The maximum number of failed attempts to execute the Set MSM Controls command has been reached; since MSM Controls Data cannot be successfully loaded, it is not possible to load applications, so SF10 permanently shuts down the MCD.

The Data Item Buffer or Data Item Stack holds a "stack" of data item copies. Each data item copy held in this stack contains a copy of a particular Static data item which MULTOS has, or is in the process of, updated as the result of executing an application MEL instruction or primitive.
This data item stack also contains information that allows SF10 to determine, for each data item copy in the stack, whether the source data item has been successfully and completely updated.

The data item copy contains the following items. These items are located within the data item copy in the order given, with the first item at the lowest address:

- Flags and byte counts that allow navigation through the data item stack to find the most recent data item copy, to create a new data item copy, or to determine whether the most recent data item copy is a copy of an item which is in the process of being updated.
- A pointer to the start of the data item which the data item copy refers to.
- A copy of the data item.

When a data item copy is created SF10 marks it as ACTIVE and when the source data item is successfully and completely updated SF10 marks the data item copy as USED. If the card is reset and SF10 founds an ACTIVE block on the stack, SF10 will copy it back to its original location and mark it as USED.

At the end of initialisation, MULTOS is in the Ready state, waiting to process commands from the IFD. It therefore returns to a known secure state following a reset or power-down/power-up.

Permutational/probabilistic/cryptographic mechanisms used in this security function: 4-byte checksum.

6.1.11 Tamper Resistant Software Behaviours SF (SF11)

MisExecution Detection

When required, SF11 detects possible mis-executions of the operating system due to unexpected external electro-magnetic or mechanical interference. SF11 calculates a parameter in two independent ways. SF11 then compares the two results. If the two results do not match, SF11 will make the MCD become mute. By doing this, SF11 will always trap a single mis-execution of the code which causes one of the parameters to contain an incorrect value.

Failed Command Counter

This counter is a software counter-measure against power analysis.

To limit the number of allowed failed attacks; SF11 maintains a count of the number of unsuccessful attempts to perform critical operations that rely on cryptographic mechanism. It makes infeasible attacks on these operations that rely on brute force attacks on the underlying cryptographic mechanism that supports it. *SF11 uses this to protect the RSA decryption mechanism used when loading and deleting applications as well as to protect the DES decryption mechanism used during the loading of the MSM Controls data. If a sufficient number (20 attempts) of unsuccessful commands is presented for any of these operations, SF11 will take appropriate action for the operation in question:*

- *If the number of failed attempts is exceeded for Create MEL Application command, SF11 disables the Create MEL Application command for this MCD to deter further attack.* The MCD is not disabled, as it is still valid to permit access to applications that have already been successfully loaded.
- *If the number of failed attempts is exceeded for Delete MEL Application command, SF11 disables the Delete MEL Application command for this MCD to deter further attack.* The MCD is not disabled, as it is still valid to permit access to applications that have already been successfully loaded.

- *If the number of failed attempts is exceeded for the Set MSM Controls command, SF11 disables the Set MSM Controls command for the remainder of the current session. Additionally SF11 sets a shutdown flag so that the MCD will permanently shutdown upon next reset.* Since the MSM Controls cannot be set and the MCD cannot progress to the next stage in the lifecycle, the MCD is shutdown to prevent further attack.

SF11 uses a down count that is initialised to its full value during manufacturing. SF11 decrements the counter before beginning the cryptographic mechanism that is being protected and re-increments it if the operation is successful.

6.1.12 Tamper Resistant Hardware Behaviours SF (SF12)

SF12 ensures normal behaviour if power voltages outside normal operating ranges are applied to the smartcard to induce non-documented behaviour. SF12 is able to detect voltages outside its normal operating range and will halt the processor (the Infineon Technologies SLE66CxxxP) until the reset signal is asserted. Therefore, this is handled in the same manner as an unexpected reset or power loss and does not represent a vulnerability.

SF12 ensures normal behaviour despite clock rate variations. The Infineon Technologies SLE66CxxxP is clocked by an external source. In normal usage, the Interface Device (IFD) the smartcard is communicating with will supply this. However, an attacker could apply a slow clock rate in order to reveal operating characteristics of the processor, or a fast clock rate in order to induce non-documented behaviour. *SF12 implements a clock rate detection mechanism that is able to detect clock rates that are slower or faster than its normal operating range. If such clock rates are detected, SF12 will halt until the reset signal is asserted.* Therefore, clock signals are handled in the same manner as a reset and do not represent a vulnerability.

Power supply analysis attack

Apart from the commonly known standard attacks on cryptographic algorithms, for example, brute force attack, known plaintext attack, chosen ciphertext attack, two methods of attacking the security of algorithms have recently emerged, known as Timing attacks and Differential Power Analysis attacks. These two forms of attack work by analysing the performance of the system implementing cryptographic algorithms to gain secret information relating to the algorithms, such as the secret keys. Both forms of attack require that the attacker has access to the system on which the algorithms are implemented and requires fairly inexpensive and readily available analysis equipment.

The timing attack is aimed at measuring the amount of time required to perform private key operations to deduce information about the keys. Crypto systems often take slightly different amounts of time to process different inputs for various reasons, for example, performance optimisations to bypass unnecessary operations, branching and conditional statements, processor instructions for multiplication and division, and other causes. This form of attack exploits timing measurements from vulnerable systems to find an entire secret key.

Prevention of timing attacks requires that timing of operations is not key or data dependent. While a timer delay may be used to make the computation time uniform, this method still allows system responsiveness or power consumption to reveal the CPU usage. Power attacks are based on the measurement of power consumption of the device performing the cryptographic computations. Different instructions may have specific power consumption profiles with respect to time and amplitude. This information along with detailed knowledge of the device and the implementation of the crypto-algorithm can be used to draw conclusions regarding the data values used in the

computations and by that may reveal information about the internal keys. The power attacks are classified as SPA (simple power analysis) and DPA (differential power analysis). SPA reveals information using power consumption profiles, electromagnetic radiation, etc., while DPA applies highly sophisticated mathematical and statistical methods to the profiles to extract secret keys.

The basic principle of these attacks is that power supply current (and/or other measurement of processor activity such as execution timing) gives visibility into processing of a cryptographic operation. It is therefore possible to correlate a part of the cryptographic operation against a set of test vectors, potentially reducing the effort required to break the algorithm. This requires a trigger event such as I/O or a distinctive supply current event and a gathering of repeated cycles of the cryptographic operation with different test data.

SF12 is implemented by an active shield metal layer. This feature allows detection of attempts to probe or force electrical signals through the metal layer.

SF12 is implemented by a CURSE (Current Scrambling Engine) unit on-chip, which scrambles the chip's current consumption.

SF12 employs the random output of the RNG to insert a random pattern of waitstates during operation, hence hiding exact time of execution of the chip's instructions and processes.

SF12 carries out of the random number generator a quality check at regular intervals and the card is abended if the quality check fails.

The CURSE, in combination with the use of random waitstates and the other hardware security features provide powerful countermeasures against timing and power analysis attacks.

Permutational/probabilistic/cryptographic mechanisms used in this security function: Random number generator.

6.1.13 Smartcard Authentication SF (SF13)

SF13 provides a means for MCD Issuers to determine that an MCD is an authentic initialised MCD prior to loading it with MSM Controls Data.

On request, SF13 provides a digest of the contents of a selected area of memory within an initialised MCD. For that, SF13 uses the Check Data Command. This digest can be used for comparison with the results of the same request applied to a known authentic initialised MCD, in order to verify the authenticity of the target MCD.

The digest had to be representative of the contents of the memory which is subject to authentication (i.e., the selected area of memory together with a fixed portion of MULTOS data). *So SF13 incorporates a portion of fixed MULTOS data in the digest. SF13 requires as input:*

a) The start address of the memory area to be checked

b) The length of the memory area to be checked

c) A random challenge value.

SF13 performs a bit-wise exclusive OR function on the random challenge value and the first part of the fixed transport key (tkf). The result of this operation is concatenated, by SF13, with the second part of

tkf and a one way hash algorithm applied to it. Using this hash as an initial value, SF13 computes a hash digest over the contents of the indicated memory area.

The inclusion in the digest of fixed MULTOS data (in the form of tkf) enables the authenticity of the MCD to be checked by comparing the digest with the result produced from the same request applied to the same area of memory on a known authentic initialised MCD. The random challenge value ensures the returned digest cannot be spoofed.

SF13 ensures it is not possible to infer from the digest any information regarding the contents of the memory area checked.

The digest is formed using a one-way hash function over the specified memory area and random challenge value. This acts to prevent any useful information being returned in the digest and therefore prevents any potential compromise of sensitive MULTOS information.

SF13 is only available on initialised MCDs (i.e., which have not yet been enabled).

This function is only useful for authenticating MCDs before they are enabled. Allowing its use after the MCD is enabled could provide a means for probing for information related to applications loaded on the MCD. As it serves no useful purpose once the MCD is enabled and, despite the way in which the digest is constructed, could be used to attack the MCD, it is prudent to disable this function after loading MSM Controls Data. If this function is called on an enabled MCD, an error condition is returned. No digest is calculated.

Permutational/probabilistic/cryptographic mechanisms used in this security function: RSA algorithm.

6.1.14 Mapping between Security Functions and Security Functional Requirements

Table 3 below shows which functions satisfy which requirements and that all requirements are met. Note that each security function contributes to the satisfaction of at least one TOE security functional requirement.

Functional requirements/ Security functions	**SF1**	**SF2**	**SF3**	**SF4**	**SF5**	**SF6**	**SF7**	**SF8**	**SF9**	**SF10**	**SF11**	**SF12**	**SF13**
FAU_ARP.1						X		X	X	X	X	X	
FAU_SAA.1						X		X	X	X	X	X	
FCS_CKM.3	X	X	X	X	X								X
FCS_CKM.4	X	X	X	X	X								X
FCS_COP.1	X	X	X	X	X								X
FDP_ACC.2	X	X											
FDP_ACF.1	X	X											
FDP_DAU.1									X				
FDP_ETC.1	X	X											
FDP_ITC.1	X	X											
FDP_RIP.1							X						

FDP_ROL.1	X												
FDP_SDI.2									X				
FIA_AFL.1											X		
FIA_ATD.1	X	X	X	X	X								
FIA_UAU.1													X
FIA_UAU.4	X	X											
FIA_UID.1													X
FIA_USB.1	X	X	X	X	X								
FMT_MOF.1					X								
FMT_MSA.1					X								
FMT_MSA.2					X								
FMT_MSA.3					X								
FMT_MTD.1					X								
FMT_MTD.2					X								
FMT_SMR.1	X	X	X	X	X								X
FPR_UNO.1											X	X	
FPT_FLS.1						X		X	X		X	X	
FPT_PHP.3											X	X	
FPT_RCV.4										X			
FPT_RVM.1	X	X	X	X	X	X	X	X	X	X	X	X	X
FPT_SEP.1					X	X							
FPT_TDC.1	X	X	X	X	X								
FPT_TST.1									X	X		X	
FRU_RSA.1	X					X							

Table 3: Mapping between security functions and security functional requirements

7. Protection Profile Claims

7.1 Protection Profile Reference

None.

7.2 Protection Profile Tailoring

None.

7.3 Protection Profile Additions

•

None.

8. Rationale

8.1 Security Objectives Rationale

This section demonstrates that the stated specific security objectives address all security environment aspects identified. Each specific security objective being correlated to at least one threat or one assumption.

8.1.1 Threats and Security Objectives

The following tables show which security objectives counter which threats phase by phase.
During phase 1, the Smartcard ES is being developed and the pre-personalisation and personalisation requirements are specified for all other phases.

The Target of Evaluation (TOE) is a functional product designed during phase 1, considering that the only purpose of the Embedded Software is to control and protect the operation of the Smartcard during phase 4 to 7 (operational phases). The global security requirements of the TOE mandate to consider, during the development phase, the security threats of the other phases. This is why the PP addresses the functions used in phases 4 to 7 but developed during phase 1. Then, the limit of the TOE corresponds to phase 1 including the TOE delivery to the IC manufacturer.

T.CLON

The TOE being constructed can be cloned, but also the construction tools and document can help clone it. During phase 1, since the product does not exist, it cannot contribute to countering the threat. For the remaining phases 4 to 7, TOE participates to countering the threats.

T.DIS_INFO

This threat addresses disclosure of specification, design and development tools concerning the IC and delivered to the software developer (during phase 1) in order to meet with the overall security objectives of the TOE. This threat is countered by development environment.

T.DIS_DEL

This threat addresses disclosure of specifications, test programs, related documents, ES and data which is delivered from phase 1 to phase 2 for software embedding. As the TOE does not yet exist, the threat can only be countered by development environmental procedures.

T.DIS_DEL1

This threat addresses disclosure of software or Native-Application and ES Data during delivery from phase 1 to phases 4 to 6. As the data is not yet implemented in the TOE, the threat can only be countered by environmental procedures

T.DIS_DEL2

This threat addresses disclosure of software or data which has been delivered, from phase 1, to phases 4 to 6. As the data is not yet implemented in the TOE, the threat can only be countered by environmental procedures.

T.DIS_ES1

The ES and accompanying documents are created and used during phase 1. As during this phase the product does not yet exist, it cannot contribute to countering the threat which must be countered by development environment.

T.DIS_ES2

Disclosure of ES and TSF data can compromise security. During phases 4 to 7, the TOE must counter the unauthorised disclosure of the ES and the Loaded-Application Data.

T.DIS_TEST_ES

Tests concerning the embedded software or software to be embedded is carried out in phase 1.
This threat is countered by environmental development procedures, of which the tests themselves are part.

TT_DEL

The threat addresses the theft of software or ES and Native-Application Data which is delivered for software embedding, from phase 1 to phase 2. As the data is not yet implemented in the TOE, the threat can only be countered by developmental environmental procedures.

T.T_TOOLS

TOE development tools are used only during phase 1, so this threat can only exist during phase 1.
As the TOE is not yet manufactured, this threat is countered by environmental procedures.

T.T_SAMPLE2

TOE samples are used only during phase 1, so this threat can only exist during phase 1.
The theft or unauthorised use of samples are countered by environmental procedures.

T.MOD_DEL

This threat addresses modification of software or TSF data which is delivered for software embedding, in phase 1. As the TOE does not exist during this phase, the threat must be countered by development procedures.

T.MOD_DEL1

This threat addresses modification of Native-Application Data during delivery from embedded software developer, phase 1, to the IC packaging manufacturer, phase 4, the finishing process manufacturer, phase 5, and for the Personaliser, phase 6. As the data is not yet loaded on the TOE, the threat can only be countered by environmental procedures.

T.MOD_DEL2

This threat addresses modification of Native-Application Data which is delivered to the IC packaging manufacturer, phase 4, the finishing process manufacturer, phase 5, and for the Personaliser, phase 6. As the data is not yet loaded on the TOE, the threat can only be countered by environmental procedures.

T.MOD

Modification of ES and TSF Data can be done during ES design in phase 1. Since the product does not exist, the threat can only be countered by environment procedures.

T.MOD_SOFT

Once present on the TOE, the software and Application data can be modified in an unauthorised way during any phases from 4 to 7. This threat is countered by the TOE.

T.T_ES

This threat covers the unauthorised use of cards during the different phases of the card life cycle as well as the misappropriation of rights of Smartcards. This threat covers phases 4 to 7 and is countered by the TOE.

T.T_CMD

This threat includes the diversion of the hardware or the software, or both, in order to execute non authorised operations. This threat covers phases 4 to 7 and is countered by the TOE.

T.MOD_LOAD, T.MOD_EXE, T.MOD_SHARE

The loading of Native Applications, execution and modification of software can endanger the security of the TOE, and especially create interference between applications. This threat covers phases 4 to 7 and is countered by the TOE.

New threats not specific to Multi-Application platforms:

T.MOD_TSF

Modification of TOE Security function can only appear when the TOE exists, thus only during phases 4 to 7. This threat is countered by the TOE.

T.DIS_DATA

Threats on end user data can only appear after the data has been created, thus in usage phase 7.
This threat is countered by the TOE.

T_MOD_DATA

Threats on end user data can only appear after the data has been created, thus in usage phase 7.
This threat is countered by the TOE.

New Threats specific to Multi-Application platforms:

These threats are present during phases 6 to 7 depending when the Loaded-Applications are loaded. It can be supposed that Loaded-Applications are mostly used during phase 7.

T.LOAD_MAN

This threat comes from illegal loading of Loaded-Applications which can for example clone legal Loaded-Applications on other cards. Loading can be done in phases 6 and 7. This threat is countered by the TOE.

T.LOAD_APP

This threat is a complement to the precedent one. In this case, an illegal Loaded-Application is loaded in place of a legal one. The attacking party can be the same as above. This threat appears during phases 6 and 7 and is countered by the TOE.

T.LOAD_OTHER

This threat addresses loading a Loaded-Application to a domain to which it should not have access. This means that the other Loaded-Application can be attacked. This threat appears during phases 6 and 7 and is countered by the TOE.

T.LOAD_MOD

This threat alters code or data without the permission of the Loaded-Application Provider.
This threat appears during phases 6 and 7 and is countered by the TOE and the environment.

T.APP_DISC

This is an attack on the Loaded-Application provider know how, and possibly on confidential data loaded along with the Loaded-Application. This threat appears during phases 6 and 7 and is countered by the TOE and the environment.

T.APP_CORR

This attack destroys partly or completely the other Loaded-Application, or more subtly can divert the Loaded-Application to create a dangerous state. This threat appears during phases 6 and 7 and is countered by the TOE.

T.APP_REMOVE

This threat addresses illegal removal of a legal Loaded-Application. It attacks reliability of services. This threat appears during phases 6 and 7 and is countered by the TOE.

T.ERR_REMOVE

This opportunistic threat takes advantage of a removal operation to attack the confidentiality of Loaded-Application provider know how, or confidential data. This threat appears during phases 6 and 7 and is countered by the TOE.

T.DEL_REMOVE

This threat is on remaining Loaded-Applications which can be damaged during the removal operation. This threat appears during phases 6 and 7 and is countered by the TOE.

T.APP_READ

This threat loads a Trojan horse to illegally access to confidential data belonging to other Loaded-Applications. This threat appears during phases 6 and 7 and is countered by the TOE.

T.APP_MOD

This threat loads a Trojan horse to illegally access to modify data or code belonging to another Loaded-Application. This threat appears during phases 6 and 7 and is countered by the TOE.

T.RESOURCES

This threat is aimed at the reliability of service of the platform or Loaded-Application.
This threat appears during phases 6 and 7 and is countered by the TOE.

Threat T.APP_DISC is also present during Loaded-Application development, phase A1 when it is countered by the environment.

8.1.2 Threats Addressed by Security Objectives

8.1.2.1 Security objectives for the TOE

During phase 1, as the TOE does not yet exist, there is no threat on the TOE itself. For the phases 4 to 7, the following table indicates that each threat is mapped to at least one specific security objective during the life of the TOE:

Threats/Obj	TAMPER_ ES	SIDE	OPERATE*	FLAW*	DIS_ MECHAN2	DIS MEMORY*	MOD_ MEMORY*	CLON*
T.CLON*					X	X		X
T.DIS_ES2		X	X	X	X	X		
T.T_ES	X		X	X			X	
T.T_CMD	X		X	X			X	
T.MOD_SOFT*	X		X	X			X	
T.MOD_LOAD	X		X	X			X	
T.MOD_EXE	X		X	X		X	X	
T.MOD_SHARE	X		X	X		X	X	
T.MOD_TSF	X		X	X			X	
T.DIS_DATA		X	X	X		X		
T.MOD_DATA	X		X	X			X	

Table 6: Mapping of security objectives to threats relative to phases 4 to 7

The TOE shall use state of the art technology to achieve the following IT security objectives; for that purpose, when IC physical security features are used, the specification of these physical security features shall be respected:

T.CLON*

To realize the threat, it is necessary to have knowledge of the security mechanism, which is prevented by O.DIS_MECHANISM2, and of the TSF data, which is prevented by O.DIS_MEMORY*.
The general threat is countered by the dedicated objective O.CLON*.

T.DIS_ES2

Illegal disclosure of ES is countered by O.DIS_MECHANISM2 and disclosure of Application by O.DIS_MEMORY*. More specifically this can be achieved by incorrect operation of the TOE, which

is countered by O.OPERATE* and O.FLAW*, or by a direct observation during operation which is countered by O.SIDE

T.T_ES

Unauthorised use of TOE can be achieved by a degradation of the security mechanisms which is countered by O.OPERATE* and O.FLAW*, or by modification of the security mechanisms countered by O.TAMPER_ES or by modification of TSF data, countered by O.MOD_MEMORY*.

T.T_CMD

To be able to have an unauthorised use of sequences sent to the TOE, it is necessary to achieve a degradation of the security mechanisms which is countered by O.OPERATE* and O.FLAW*, or to modify the security mechanisms countered by O.TAMPER_ES or by modification of TSF data, countered by O.MOD_MEMORY*.

T.MOD_SOFT*

The modification of embedded software of the TOE is countered by a correct operation of security mechanisms O.OPERATE* and O.FLAW*. The threat includes modification of the security mechanisms themselves which is countered by O.TAMPER_ES and modification of TSF data, countered by O.MOD_MEMORY*.

T.MOD_LOAD

To be able to load illegally programs on TOE, it is necessary to achieve a degradation of the security mechanisms which is countered by O.OPERATE* and O.FLAW*, or to modify the security mechanisms countered by O.TAMPER_ES or by modification of TSF data, countered by O.MOD_MEMORY*.

T.MOD_EXE

To be able to illegally execute programs on TOE, it is necessary to bypass or degrade the access security mechanisms. This is countered by O.OPERATE* and O.FLAW*. Modification of the security mechanisms is countered by O.TAMPER_ES. It is also possible to gain access through modification of TSF data, which is countered by O.MOD_MEMORY*, or through disclosure of TSF data which is countered by O.DIS_MEMORY*.

T.MOD_SHARE

To be able to modify programs on TOE, it is necessary to bypass or degrade security mechanisms. This is countered by O.OPERATE* and O.FLAW*. Modification of the security mechanisms is countered by O.TAMPER_ES. Illegal Modification of Application data is countered by O.MOD_MEMORY*. This is also countered by protection of the confidentiality of TSF data: O.DIS_MEMORY*.

T.MOD_TSF

The illegal modification of TSF data of the TOE is countered by O.MOD_MEMORY* which addresses also TSF data. It is also possible to degrade or bypass access mechanisms, which is countered by O.TAMPER_ES and O.OPERATE*. Absence of design flaws, O.FLAW*, is necessary to counter the threat.

T.DIS_DATA

The disclosure of application user and TSF data on the TOE is countered by O.DIS_MEMORY*. To fulfill the threat, it can be necessary to degrade or bypass access mechanisms, which is countered by O.SIDE and O.OPERATE*. Absence of design flaws, O.FLAW*, is necessary to counter the threat.

T.MOD_DATA

The modification of application user data on the TOE is countered by O.MOD_MEMORY*. To fulfill the threat, it can be necessary to degrade or bypass access mechanisms, which is countered by O.TAMPER_ES, O.OPERATE*. Absence of design flaws, O.FLAW*, is necessary to counter the threat.

Threats/Obj	ROLLBACK	RESOURCE	LOAD	SECURITY	EFFECT_L	REMOVE	EFFECT_R	SEGREGATE
T.LOAD_MAN			X					
T.LOAD_APP			X					
T.LOAD_OTHER					X			
T.LOAD_MOD				X				
T.APP_DISC				X				
T.APP_CORR					X			
T.APP_REMOVE						X		
T.ERR_REMOVE						X		
T.DEL_REMOVE							X	
T.APP_READ								X
T.APP_MOD								X
T.RESOURCES	X	X						

Table 7: Mapping of security objectives to threats relative to phase 6 and 7

The TOE shall use state of the art technology to achieve the following IT security objectives.

T.LOAD_MAN

O.LOAD imposes that application be loaded only with the permission of the administrator, which counters the threat.

T.LOAD_APP

O.LOAD controls the origin of the Loaded Application before loading, thus if necessary control is made by the administrator, it counters T.LOAD_APP.

T.LOAD_OTHER

Loading an application into an another illegal domain is countered by O.EFFECT_L, which prevents applications from having non-authorised effects on applications loaded in other domains.

T.LOAD_MOD

Alteration of Loaded Application during loading is prevented by O.SECURITY, which guarantees its integrity.

T.APP_DISC

Divulgence of Loaded Application during loading is prevented by O.SECURITY, which guarantees its confidentiality.

T.APP_CORR

Loading an application so it corrupts another application is countered by O.EFFECT_L, which prevents applications from having non-authorised effects on applications loaded in other domains.

T.APP_REMOVE

Removal of application without the consent of the administrator is countered by O.REMOVE, which imposes the authorisation of the administrator.

T.ERR_REMOVE

Removal of application leaving confidential data is countered by O.REMOVE which imposes that the space left does not hold any information linked to removed application.

T.DEL_REMOVE

Deletion of part of a Loaded Application by removal of another is countered by O.EFFECT_R, which ensures that removal has no effect on other Loaded Applications.

T.APP_READ

Use of a Loaded Application to illegally read data contained in another application is countered by O.SEGREGATE which ensure that illegal reading of data of another application is not possible.

T.APP_MOD

Use of a Loaded Application to illegally modify data or code contained in another application is countered by O.SEGREGATE, which ensure that illegal modification of data or code of another application is not possible.

T.RESOURCES

Destruction or hoarding of card resources is prevented by O.ROLLBACK which guarantees that a failure does not compromise card resources and by O.RESOURCES which controls the use of card resources by Loaded Applications.

8.1.2.2 Security objectives for the environment

The following tables map the security objectives for the environment relative to the various threats in addition to the Smartcard PP.

Threats/Obj	DEV_TOOLS	DEV_DIS_ES	SOFT_DLV*	INIT_ACS	SAMPLE_ACS
T.CLON*		X	X	X	X
T.DIS_INFO*		X			
T.DIS_DEL*			X		
T.DIS_ES1		X		X	

T.DIS_TEST_ES		X			
T.T_DEL*			X		
T.T_TOOLS	X				
T.T_SAMPLE2					X
T.MOD_DEL*			X		
T.MOD		X		X	

Table 8: Mapping of security objectives for the environment to threats relative to phase 1

T.CLON*	Cloning requires knowledge of: Development data and access to tools, which is countered by O.DEV_DIS_ES The software which is countered by O.SOFT_DLV* Initialisation data which is countered by O.INIT_ACS Cloning can also be done by using samples; this is countered by O.SAMPLES_ACS.
T.DIS_INFO*	Disclosure of IC assets is countered by O.DEV_DIS_ES, which guarantees the storage of classified information.
T.DIS_DEL*	Disclosure of embedded software and corresponding data during delivery is countered by O.SOFT_DLV*.
T.DIS_ES1	Disclosure of ES is countered by O.DEV_DIS_ES, which guarantees the storage of classified information, and by O.INIT_ACS which guarantees a controlled access to initialisation data.
T.DIS_TEST_ES	Disclosure of ES test program is countered by O.DEV_DIS_ES, which guarantees the storage of classified information.
T.T_EL*	Theft of software delivered to IC manufacturer is countered by O>SOFT_DLV* which ensures trusted delivery.

T.T_TOOLS	Theft or unauthorised access to development tools is countered by O.DEV_TOOLS* which controls the accesses.
T.T_SAMPLE2	Theft of samples is countered by O.SAMPLE_ACS controlled access.
T.MOD_DEL*	Modification of software and related information is countered O.SOFT_DLV*.
T.MOD	Unauthorised modifications of software are countered by access control specified by O.DEV_DIS_ES and that of TSF data by O.INIT_ACS.

Threats	**DLV_DATA**	**TEST_OPERATE***
T.DIS_DEL1	X	
T.DIS_DEL2		X
T.MOD_DEL1	X	
T.MOD_DEL2		X

Table 9: Mapping of security objectives for the environment to threats relative on delivery from phase 1 to phases 4 to 6

T.DIS_DEL1

Unauthorised disclosure of Native-Application and ES data during delivery is countered by O.DLV_DATA, which specifies a trusted delivery maintaining the confidentiality.

T.DIS_DEL2

Unauthorised disclosure of Native-Application and ES data after delivery is countered by O.TEST_OPERATE* which specifies maintenance of the confidentiality.

T.MOD_DEL1

Unauthorised modification of Native-Application and ES data during delivery is countered by O.DLV_DATA, which specifies a trusted delivery maintaining the integrity.

T.MOD_DEL2

Unauthorised modification of Native-Application and ES data after delivery is countered by O.TEST_OPERATE* which specifies maintenance of the integrity and its test.

Threats	**O.APPLI_DEV**
T.LOAD_MOD	X
T.APP_DISC	X

Table 10: Mapping of security objectives for the environment to threat on phases A1 and A2 (development and delivery for phase A1 to phases 6 and 7)

T.LOAD_MOD

Modification of Code and data of a Loaded-Application during its transfer and loading is countered by O.APPLI_DEV, which ensures the mechanisms to verify their integrity.

T.APP_DISC

Gaining access to confidential code and data of a Loaded-Application during its transfer and loading is countered by O.APPLI_DEV, which ensures the confidentiality.

8.1.3 Assumptions and Security Objectives for the Environment

This section demonstrates that the combination of the security objectives is suitable to satisfy the identified assumptions for the environment.

Each of the assumptions for the environment is addressed by objectives.

Table 11 demonstrates which objectives contribute to the satisfaction of each assumption.
For clarity, the table does not identify indirect dependencies.

This section describes why the security objectives are suitable to provide each assumption.

Phases		**Delivery process for phases 4 to 7**			**Phases 4 to 6**	**Phase 7**	**Phase A1**
	Assumptions/Objectives	DLV_ PROTECT*	DLV_ AUDIT*	DLV_RESP*	TEST_OPERATE*	USE_DIAG*	APPLI_DEV
4 to 7	DLV_PROTECT*	X					

4 to 7	DLV_AUDIT		X				
4 to 7	DLV_RESP*			X			
4 to 7	USE_TEST*				X		
4 to 7	USE_PROD*				X		
7	USE_DIAG*					X	
A1	APPLI_CONT						X

Table 11: demonstrates mapping of security objectives for the environment to assumptions

8.2 Security Requirements Rationale

8.2.1 Security Functional Requirements Rationale

This section demonstrates that the combination of the security requirements is suitable to satisfy the identified security objectives.

Table 12 demonstrates which security functional requirement contributes to the satisfaction of each TOE security objective. For clarity, the table does not identify indirect dependencies.

Security Functional Requirements	**O.TAMPER_ES**	**O.SIDE**	**O.OPERATE***	**O.DIS_ MECHAN.2**	**O.DIS MEMORY**	**O.MOD_ MEMORY**	**O.FLAW***	**O.CLON***
EAL4 Requirements							X	
FAU_ARP.1	X		P	P	X	X		
FAU_SAA.1	X		P	P	X	X		
FCS_CKM.3	X		P		P	P		P
FCS_CKM.4	X		P		P	P		X
FCS_COP.1	X				X	X		P
FDP_ACC.2	X		P	X	X	X		P
FDP_ACF.1	X		P	X	X	X		P
FDP_DAU.1	X		P		X	X		P
FDP_ETC.1					X			
FDP_ITC.1					X	X		
FDP_RIP.1	X				P			
FDP_SDI.2			P			X		
FIA_AFL.1	X		P					P
FIA_ATD.1	X		P					
FIA_UAU.1	X				X	X		P
FIA_UAU.4	X				X	X		P
FIA_UID.1	X				X	X		P
FIA_USB.1	X				X	X		P
FMT_MOF.1	X		X	X	P	P		P
FMT_MSA.1	X		P	X	P	P		P

FMT_ MSA.2	X		P	X	P	P		P
FMT_ MSA.3	X		P	X	P	P		P
FMT_MTD.1					X	X		P
FMT_SMR.1	X		X					
FPR_UNO.1		X	P		X			X
FPT_FLS.1	X							
FPT_PHP.3	X		X	X	X	X		X
FPT_SEP.1	X			X	X	X		
FPT_TDC.1	X					X		
FPT_TST.1			X		X	X		

Security Requirements	**ROLLBACK**	**RESOURCE**	**LOAD**	**SECURITY**	**EFFECT_L**	**REMOVE**	**EFFECT_R**	**SEGREGATE**
FAU_ARP.1		X						
FAU_SAA.1		X						
FCS_CKM.3				X		X		
FCS_CKM.4				X		X		
FCS_COP.1			X	X		X		
FDP_ACC.2			X			X		X
FDP_ACF.1					X		X	X
FDP_ITC.1			X	X				
FDP_RIP.1						X		
FDP.ROL.1	X							
FIA_UID.1			X			X		
FIA_UAU.1			X			X		
FMT_MSA.1			X			X		
FMT_ MSA.2			X			X		
FMT_ MSA.3			X			X		
FMT_MTD.1								X
FMT_MTD.2		X	X					
FMT_SMR.1			X			X		
FPT_FLS.1	X				X		X	X
FPT.RCV.4	X				X		X	
FPT.RVM.1			X			X		X
FPT_SEP.1					X		X	X
FRU_RSA.1		X						

Table 12: Mapping of security functional requirements and objectives

This section describes how the security objectives for the TOE are met by the security requirements. The assurance requirements contribute to the satisfaction of the O.FLAW* security objective. They are suitable because they provide the assurance that the TOE is designed, implemented and operates so that the IT security requirements are correctly provided.

O.TAMPER_ES

This objective is met through:

Protection of critical parts from tampering:

If a failure occurs, a secure state is preserved (FPT_FLS.1), so critical parts are preserved. Resistance to physical attack (FPT_PHP.3) also allows protection of critical parts. The security domain separation (FPT_SEP.1) between the TSF and untrusted subjects, and between subjects in the TSC (loaded-applications for example) allow for better security policy and for better protection of critical parts. Inter-TSF basic TSF data consistency (FPT_TDC.1) also allows better protection from tampering.

Prevention of unauthorised changes:

Identification and authentication: Thanks to FIA_ATD.1 and FIA_USB.1 only authorised users are able to load certificates, ALU or MSM Controls Data. The only command available before identification and authentication permits verification only of the validity of an MCD, it does not allow any unauthorised changes. Moreover, brute force attacks are prevented with FIA_AFL.1. Cryptographic aspects do not permit any unauthorised changes.

Security management: Only two security roles are allowed (FMT_SMR.1) which decreases the risk of unauthorised changes of the TOE. MULTOS Security Manager is the only authorised role able to manage security functions behaviour via the security attributes contained in the MSM Controls Data (FMT_MOF.1, FMT_MSA.1, FMT_MSA.3). Application Provider is the only authorised role able to sign application certificate.

Protection of parameters and keys:

Cryptographic keys are temporarily copied in RAM (FCS_CKM.3) to be used for cryptographic operations (FCS_CKM.4) and then they are destroyed (FCS_COP.1) to prevent an unauthorised user gaining access to them.

Thanks to FDP.ACC.2 and FDP_ACF.1, only authorised operations are allowed between subjects and objects defined in the Access Control Policy. It prevents fake Load and Delete Certificate being loaded onto an MCD. Moreover, FDP_DAU.1 and FDP_RIP.1 do not allow the application's code and data to be modified or disclosed.

If a potential violation is detected, the MCD will abend current operation and become mute or it will shutdown to prevent critical parts being disclosed. FAU_SAA.1 and FAU_ARP.1 define potential violations and actions to be taken in each case to ensure protection of critical parts.

O.SIDE

Interpretation of side channel information leakage is countered by FPR_UNO, which ensures that observation of signals cannot reveal information that could allow illegal access and operations.

O.OPERATE

Correct operation of security functions is assured by:

Security management: Only security roles defined in FMT_SMR.1 are able to provide security management. MSM ensures management of security functions behaviour. In this way, operations of security functions are secured to be determined by authorised users.

Protection of TSF: Resistance to physical attacks (FPT_PHP.3) and TSF testing (FPT_TST) permit to protect security data which security functions are reliant on.
In this way, correct operations of security functions are ensured.

On a second level, other SFR are active: FAU_ARP.1, FAU_SAA.1, FCS_CKM.3, FCS_CKM.4, FDP_ACC.2, FDP_ACF.1, FDP_DAU.1, FDP_SDI.2, FIA_AFL.1, FIA_ATD.1, FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, and FPR_UNO.1.

O.DIS_MECHAN2

Protection of security mechanisms against unauthorised disclosure is assured by the following:

Protection of TSF: Resistance to physical attack (FPT_PHP.3) prevent unauthorised users to directly or indirectly observe the security mechanisms by using physical attack whereas TSF Domain separation (FPT_SEP.1) prevent unauthorised users to interfere with the TSF and to access to security mechanisms.

Security management: Only MULTOS Security Manager is able to determine the behaviour and enable (FMT_MOF.1, FMT_MSA.3) security mechanisms by loading security attributes contained in MSM Controls Data (FMT_MSA.1). MSM Controls Data is encrypted by MSM to ensure only secure values are accepted for security attributes (FMT_MSA.2).

User data protection: FDP_ACC.2 and FDP_ACF.1 ensure user data is not corrupted (in order to permit security mechanisms unauthorised disclosure). In the same way, FAU_SAA.1 and FAU_ARP.1 allow tracking of possible attacks against user data.

O.DIS_MEMORY

User data protection: FDP_ACC.2 and FDP_ACF.1 are used to ensure that applications are not loaded on a fake MCD. FDP_DAU.1 guarantees the validity of an application's code space and to ensure MULTOS will not send responses that may be based on corrupted data. FDP_ETC.1, FDP_ITC.1 and partially FDP_RIP.1 fulfil this objective.

Authentication and identification: FIA_UAU.1, FIA_UID.1 and FIA_USB.1 allow only authorised users to access memory. FIA_UAU.4 guarantees the authenticity of the application thanks to single-use authentication mechanisms.

Cryptographic support: FCS_COP.1, FCS_CKM.3 and FCS_CKM.4 are used for authentication.

Security management: Only MSM is authorised to load encrypted MSM Controls Data (FMT_MTD.1). Because of the encryption, an unauthorised user cannot disclose this data. FMT_MOF.1, FMT_MSA.1, FMT_MSA.2 and FMT_MSA.3 are used as support.

Unobservability: FPR_UNO.1 is used so that data is not revealed during operations.

Protection of the TSF: FPT_PHP.3 prevents unauthorised users disclosing sensitive information in memories using physical attack. FPT_SEP.1 ensures separations of applications so as an application can only access his own space data. FPT_TST.1 provides initialisations of memories, when the MCD is powered-up or reset, to prevent unauthorised disclosure of data.

Security audit: FAU_ARP.1 and FAU_SAA.1 monitor problems related to unauthorised disclosure of sensitive information.

O.MOD_MEMORY

User data protection: FDP_DAU.1 and FDP_SDI.2 ensure that the application's code has not been corrupted or modified by an unauthorised user. FDP_ITC.1 and partially FDP_ACC.2 and FDP_ACF.1 also fulfil this objective.

Authentication and identification: FIA_UAU.1, FIA_UID.1 and FIA_USB.1 allow only authorised users to access memory. FIA_UAU.4 guarantees the authenticity of the application thanks to single-use authentication mechanisms. This objective is supported partially by FIA_AFL.1 and FIA_ATD.1.

Cryptographic support: FCS_COP.1, FCS_CKM.3 and FCS_CKM.4 are used for authentication.

Security management: Only the MSM is authorised to load encrypted MSM Controls Data (FMT_MTD.1). Because of the encryption, an unauthorised user cannot modify this data. FMT_MOF.1, FMT_MSA.1, FMT_MSA.2 and FMT_MSA.3 are used as support.

Protection of the TSF: FPT_PHP.3 prevents unauthorised users modifying or corrupting sensitive information in memories using physical attack. FPT_TST.1 provides validity checks, when the MCD is powered-up or reset, to prevent unauthorised modification or corruption of data.

Security audit: FAU_ARP.1 and FAU_SAA.1 monitor problems related to corruption or unauthorised modification of sensitive information.

O.FLAW

The objective is met by good design and testing as specified by EAL4 augmented conformity requirements.

O.CLON

The protection against cloning objective is assured by the following:

Good key housekeeping: FCS_CKM.4 is a protection against unauthorised disclosure of cryptographic keys. Cryptographic keys had to be held securely to prevent cloning.

Unobservability of TSF data: FPR_UNO.1 prevents observation of TSF data, which is necessary for cloning the TOE.

Resistance to physical attacks: Physical attacks are able to determine some important data of the TOE (i.e. cryptographic keys) which is necessary for cloning. FPT_PHP.3 prevents that.

Other SFR also participate in cloning prevention:

Cryptographic support: FCS_CKM.3, FCS_COP.1

Data protection: FDP_ACC.2 and FDP.ACF.1 manage access to users' data, which must not be modified to prevent cloning. Indeed, a fake application can be used to access to sensitive data. FDP_DAU.1 is used to verify the validity of applications code spaces.

Identification and authentication: FIA_AFL.1, FIA_UAU.1, FIA_UID.1, FIA_UAU.4, FIA_USB.1 are used to prevent unauthorised users gaining access to sensitive data using identification and authentication mechanisms. It is a way to protect the TOE from cloning.

Security management: Management of security functions is assured by the one and only MSM that loads encrypted MSM Controls Data on the MCD.

MSM Controls Data contains security attributes that determine the behaviour of some security functions. Encryption with tkv (a key generated by MSM) ensures security attributes are secure and unauthorised users are not able to use MSM Controls Data contents for cloning.

O.ROLLBACK

This objective is assured by the following:

The TOE is in a valid state before a loading of an application thanks to FPT_FLS.1 and FPT_RCV.4. In the case of a failure in the loading, FDP_ROL.1 allows the erasure of the application's code and data automatically.

O.RESOURCE

Resource preservation objective is assured by the following:

Management of limits on TSF data: FMT_MTD.2.

Maximum quotas: FRU_RSA.1.

Backed by FAU_ARP.1 Security Alarms and FAU_SAA.1 Potential violation analysis.

O.LOAD

The control of the administrator is assured by the following:

Identification and authentication: No load operation can be done before identification/authentication operation succeeds (FIA_UAU.1 and FIA_UID.1).

Non-bypassability of the TSP: FPT_RVM.1 ensures TSP is not bypassed.

Security management: MSM is the only user authorised to load MSM Controls Data. MSM Controls Data contains data used to determine the behaviour of the security functions used to manage the load and delete process of the applications (FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, and FMT_SMR.1). FMT_MTD.2 also implements this objective.

User data protection: The complete access control (FDP_ACC.2 and FDP_ITC.1) of the load operations allows verification of the ALC or ADC that is produced by the MSM by comparing security attributes of the application and security attributes stored on the MCD.

Cryptographic support: FCS_COP.1.

O.SECURITY

Loading of applications requirements is assured by the following:

Import of User Data without Security Attributes: FDP_ITC.1

Cryptographic support: FCS_COP.1, FCS_CKM.3 and FCS_CKM.4.

O.EFFECT_L

Separation of the Loaded-Applications is assured by the following:

Domain separation: Security domains of the Loaded-Applications are held separate (FPT_SEP.1) and a Loaded-Application cannot interfere with the code or data of another Loaded-Application.

Security attributes: The Unique Application Identifier identifies each application stored in the memory of the MCD (FDP_ACF.1). Indeed, each application is stored in an Application Pool Block (which contains its code and data) which is identified by the Unique Application Identifier.

And the following SFR which assure correct operation:

Failure with preservation of secure state: FPT_FLS.1.

Function recovery: Security functions involved in application load have the capacity to either complete or to recover to a consistent and secure state.

O.REMOVE

Safety of the removal is assured by the following:

Identification and authentication: No delete operation can be done before identification/authentication operation succeeds (FIA_UAU.1 and FIA_UID.1).

Non-bypassability of the TSP: FPT_RVM.1 ensures only authorised users are able to remove a Loaded-Application.

Security management: the MSM is the only user authorised to load MSM Controls Data. The MSM Controls Data contains data used to determine the behaviour of the security functions used to manage the load and delete process of the applications (FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, and FMT_SMR.1).

User data protection: The complete access control (FDP_ACC.2) of the load and delete operations permits the verification of the ALC or ADC that is produced by the MSM (the administrator) by comparing security attributes of the application and security attributes stored on the MCD. FDP_ITC.1 also implements this objective.

Cryptographic support: FCS_CKM.3, FCS_CKM.4 and FCS_COP.1.

Information protection: FDP_RIP.1 guarantees that the code and the application's data are erased and are no longer accessible after the removal.

O.EFFECT_R

Separation of the unloaded applications from the other Loaded-Applications is assured by the following:

Domain separation: Security domains of the Loaded-Applications are held separate (FPT_SEP.1) and an unloaded application cannot interfere with the code or data of another Loaded-Application.

Security attributes: The Unique Application Identifier permits the identification of each application stored in the memory of the MCD (FDP_ACF.1). Indeed, each application is stored in an Application Pool Block, (which contains its code and data) which is identified by the Unique Application Identifier.

And the following SFR which assures correct operation:

Failure with preservation of secure state: If an application attempts to modify code or data of remaining Loaded-Applications, a secure state is preserved (FPT_FLS.1).

Function recovery: Security functions involved in application deletion have the property to either complete or to recover to a consistent and secure state.

O.SEGREGATE

Segregation of Loaded-Applications is assured by the following:

User data protection and security management: The Unique Application Identifier, contained in MSM Controls Data (FMT_MTD_1), is used to identify the Application Pool Block in which the application's code and data are stored. It is directly coupled to the application load process but is also a basic requirement of segregation of Loaded-Applications (FDP_ACC.2).

Domain separation: FPT_SEP.1 ensures separation between the security domains of Loaded-Applications.

The TSF fulfilling the SFR are protected by:

Failure with preservation of secure state: If a failure occurs (Loaded-Application trying to read from or write to another's Loaded-Application's code or data without authorisation), a secure state is preserved (FPT_FLS.1).

Non-bypassability of the TSP: FPT_RVM.1 ensures TSP is not bypassed.

8.2.2 Strength of Function Level Rationale

Due to the definition of the TOE, it is very important that the claimed SOF should be high since the product critical security mechanisms only have to be defeated by attackers possessing a high level of expertise, opportunity and resources, and successful attack is judged beyond normal practicality.

8.2.3 Security Assurance Requirements Rationale

The assurance requirements of this Protection Profile are summarized in the following table:

Requirements	Name	Type
EAL4	Methodically designed, tested and reviewed	Assurance level
ADV_IMP.2	Implementation of the TSF	Higher hierarchical component
ALC_DVS.2	Sufficiency of security measures	Higher hierarchical component
AVA_VLA.4	Highly resistant	Higher hierarchical component

Evaluation Assurance level rationale

An assurance requirement of EAL4 is required because the platform is designed to support Loaded-Applications intended to defend against sophisticated attacks. This evaluation assurance level was selected since it is designed to permit a developer to gain maximum assurance from positive security engineering based on good commercial practices. EAL4 represents a high practical level of assurance expected for a future commercial grade product. In order to provide a meaningful level of assurance that the TOE provides an adequate level of defense against such attacks, the evaluators should have access to the low level design and source code.

Assurance augmentations rationale

Additional assurance requirements are also required due to the definition of the TOE and to the conformance to the ITSEC evaluation level E3 with a strength of mechanism high.

ADV_IMP.2 Implementation of the TSF

The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. ES source code is an example of implementation representation. This assurance component is a higher hierarchical component to EAL4 (only ADV_IMP.1 is found in EAL4.) It is important for a Smartcard that the evaluator evaluates the implementation representation of the entire TSF to determine if the functional requirements in the Security Target are addressed by the representation of the TSF.

ADV_IMP.2 has dependencies with:

- ADV_LLD.1 “Descriptive Low-Level design”.
- ADV_RCR.1 “Informal correspondence demonstration”.
- ALC_TAT.1 “Well defined development tools”.

These components are included in EAL4, and so these dependencies are satisfied.

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.

This assurance component is a higher hierarchical component to EAL4 (only ALC_DVS.1 is found in EAL4). Due to the nature of the TOE, there is a need to justify the sufficiency of these procedures to protect the confidentiality and the integrity of the TOE.

ALC_DVS.2 has no dependencies.

AVA_VLA .4 Highly resistant

Due to the definition of the TOE, it must be shown to be highly resistant to penetration attacks. This is due to the fact that a Smartcard can be placed in a hostile environment, such as electronic laboratories.

This assurance requirement is achieved by the AVA_VLA.4 component. Independent vulnerability analysis is based on highly detailed technical information. The attacker is assumed to be thoroughly familiar with the specific implementation of the TOE. The attacker is presumed to have a high level of technical sophistication.

AVA_VLA.4 has dependencies with:

- ADV_FSP.1 Informal functional specification.
- ADV_HLD.2: Security enforcing high-level design.
- ADV_LLD.1: Descriptive low level design.
- ADV_IMP.1: Subset of the implementation of the TSF.

- AGD_ADM.1: Administrator Guidance.
- AGD_USR.1 “User Guidance”.

All these dependencies are satisfied by EAL4.

8.2.4 Security Requirements are Mutually Supportive and Internally Consistent

The purpose of this part of the ST rationale is to show that the security requirements are mutually supportive and internally consistent.

No detailed analysis is given in respect to the assurance requirement because:

- EAL4 is an established set of mutually supportive and internally consistent assurance requirements.
- The dependencies analysis for the additional assurance components in the previous section has shown that the assurance requirements are mutually supportive and internally consistent (all the dependencies have been satisfied).
- The dependencies analysis for the functional requirements described above demonstrate mutual support and internal consistency between the functional requirements.
- Inconsistency between functional and assurance requirements can only arise if there are functional assurance dependencies which are not met, a possibility which has been shown not to arise in the above section "Security functional requirements dependencies".

Therefore, the dependencies analysis described above demonstrates mutual support and internal consistency between the functional requirements.

8.3 TOE Summary Specification Rationale

8.3.1 Target of Evaluation Security Functions Rationale

The following explains how the security functions work together so as to satisfy the TOE security functional requirements.

FAU_ARP.1 Abend and Shutdown Iterations

This requirement is implemented by the same security functions than FAU_SAA.1 because these actions are induced by the events defined in FAU_SAA.1.

FAU_SAA.1 Abend Iteration

The TSF detects a potential violation as soon as one of the events listed in FAU_SAA.1 appears. The security functions that satisfy the requirement are presented in the following table:

Security Function	Event
SF6	An application attempts to execute MEL code outside of its code space or the code space of the codelet that it calls.
	An application attempts to access data outside of its data space or the Public data segment.
SF9	An apparent corruption of the MSM Controls Data or security data held within the EEPROM of MULTOS.
	An apparent corruption of an application's code space held within the Application Pool Block in the EEPROM of MULTOS.
SF11	MULTOS determines that it has executed an invalid sequence of instructions (possibly due to electromagnetic or mechanical interference).
SF12	An unexpected hardware event occurred.

FAU_SAA.1 Shutdown Iteration

The TSF detects a potential violation as soon as one of the events listed in FAU_SAA.1 appears. The security functions that satisfy the requirement are presented in the table below:

Security Function	Event
SF8	A critical process is interrupted.
SF10	An EEPROM write fails.
SF11	There have been too many failed attempts to load MSM Controls Data.

FCS_CKM.3

This requirement is implemented each time a cryptographic key is used by a cryptographic operation: SF1, SF2, SF3, SF4, SF5 and SF13.

FCS_CKM.4

This requirement is implemented each time a cryptographic key is used by a cryptographic operation: SF1, SF2, SF3, SF4, SF5 and SF13.

FCS_COP.1 Iteration 1, 2 and 3

The following cryptographic operations are implemented in the following security functions:

Cryptographic operations	Security functions
Digital signature verification (RSA)	SF1, SF2, SF3, SF4
Application code integrity checking and MCD authentication (Asymmetric hash)	SF1, SF2, SF3, SF5, SF13
Application code integrity checking (SHA-1)	SF1
Recovering of protected code or data segments of an application and recovering of MSM Controls Data (DES)	SF4, SF5

FDP_ACC.2 Load Application SFP Iteration and Delete Application SFP Iteration

SF1 enforces the Load Application SFP. It covers therefore FDP_ACC.2 Load Application SFP Iteration.

SF2 enforces the Delete Application SFP. It covers therefore FDP_ACC.2 Delete Application SFP Iteration.

FDP_ACF.1 Load Application SFP Iteration and Delete Application SFP Iteration

SF1 covers FDP_ACF.1 Load Application SFP Iteration because it enforces the whole set of rules defined in this requirement.

SF2 covers FDP_ACF.1 Delete Application SFP Iteration because it enforces the whole set of rules defined in this requirement.

FDP_DAU.1

Integrity checks made in SF9 implements this requirement. Indeed, when an application is successfully loaded, MULTOS calculates a 4-bytes checksums over this application's code space. This checksum is stored and compared with a calculated checksum each time this application is selected to verify the validity of the application's code space.

FDP_ETC.1

This requirement is fully implemented by SF1 and SF2.

FDP_ITC.1

This requirement is fully implemented by SF1 and SF2.

FDP_RIP.1

Critical Data Overwrite SF (SF7) applies this requirement by ensuring that no part of an application's code or data can be accessed after the application has been deleted.

FDP_ROL.1

SF1 permits the rollback of loading process by erasing the application's code and data if the application's load fails.

FDP_SDI.2

Monitoring stored user data is ensured by SF9 by using 4-byte checksums to guarantee application's code integrity. If an integrity error occurs, SF9 abends the current session.

FIA_AFL.1

Failed Command Counter Mechanism implemented in SF11 ensures the achievement of this requirement.

FIA_ATD.1

Each security attribute identifies a specific user. The correspondence between security attribute and security function is shown in the following table:

Security attribute	Security function implementing the use of the security attributes belonging to individual users
Global Key Certification Key (kck)	SF1; SF2
Application Provider's Asymmetric Key (ack)	SF3, SF4
MCD-specific Asymmetric Transport Key (mkd)	SF4
MCD-specific Transport Key variable (tkv)	SF5
MCD Issuer Identifier	SF1; SF2

FIA_UAU.1

Users are not able to perform any command before authentication, except the Check Data Command which is implemented in the Smartcard Authentication SF (SF13).

FIA_UAU.4

History List mechanism (SF1): The history list mechanism directly supports the ability of MULTOS to prevent unauthorised reload of applications if such reload has been disallowed. When a request for an Application Load Certificate is made to the MSM, the requestor may specify that the ALC is to contain a random number entry that will be loaded in to the history list of each MCD when the certificate is processed by the MCD. The history list entry will remain on this MCD, even if the application is deleted. If an attempt is made to re-use the certificate a second time on such an MCD, the request will be refused. A random number entry of zero has the special meaning of indicating that the application is not protected against load replay, which allows the Issuer and MSM a means to make this feature optional for those applications not requiring it. So, each Application Identifier must remain unique to prevent unauthorised reload of a deleted application.

Unique Application Identification mechanism (SF1, SF2): All applications that are loaded on to an MCD must have an Applications Identifier that is unique amongst all other applications that are currently loaded on that MCD. When required by the appropriate SF, a comparison is made between the Application Identifier contained in an Application Load or Delete Certificate and that of the other applications loaded on to the MCD. A decision whether to proceed with this load or delete operation is made on the basis of this comparison. So, each Application ID must remain unique to prevent reload of an already-loaded application or to check if an application to be deleted is on the MCD.

Load Permissions mechanism (SF1): A load permissions structure shall be associated with each application. The structure shall define the MCD domain in which the application can be loaded and deleted. A check shall be made of each application's permissions against the controls specified for the target MCD. Among these permissions, MCD-unique Identifier, MCD Issuer Identifier and MCD Issuer Product Identifier must remain unique to prevent load of an application on an unauthorised MCD.

SF1 and SF2 implement these authentication mechanisms and so this requirement.

FIA_UID.1

Users are not able to perform any command before identification, except the Check Data Command which is implemented in the Smartcard Authentication SF (SF13).

FIA_USB.1

User security attributes (keys) are associated with subjects acting on behalf of the users thanks to certification of ALC (SF1), ADC (SF2) and application signature in ALU (SF3), encryption of MSM Controls Data (SF5) and KTU (SF4), authentication of a target MCD (SF13).

This requirement is implemented by SF1, SF2, SF3, SF4, SF5 and SF13.

FMT_MOF.1

MSM Controls Data determines the behaviour of the functions Application Load SF and Application Deletion SF because it contains permissions of the MCD. These permissions are used in the functions Application Load SF and Application Deletion SF to check permissions of the application versus permissions of the MCD. MSM, who encrypts the MSM Controls Data with the MCD-specific transport key tkv, enables the Application Load Certificate Control SF and Application Deletion Certificate Control SF. So SF5 implements this requirement.

FMT_MSA.1

The security attributes contained in the MSM Controls Data (MCD Issuer Product Identifier, MCD Issuer Identifier, MCD Batch Number, RFU 2 (Reserved for Future Use), RFU 4, RFU 5, RFU 6, MCD-unique Identifier, asymmetric transport key set (mkd)) can only be loaded if the MSM Controls Data is encrypted with the MCD-specific transport key tkv (generated by MSM). It is implemented by SF5.

FMT_MSA.2

The controls data are generated by the MSM and are loaded onto the MCD in enciphered form in a secure controlled environment. Once the data has been collected, the data is deciphered and then verified. The multi key DES CBC (cipher block chaining) decrypt function is used by MSM to encipher the controls data and the corresponding encrypt function is used by MULTOS to decipher the data using the MCD specific symmetric transport keys which are stored in the MCD EEPROM at IC manufacturing time.

Verification is done in two steps. The signature (asymmetric hash digest) of the controls data is attached at the tail end of the controls data by MSM before encipherment. MULTOS first deciphers the MSM controls data and then re-calculates the hash digest of the deciphered data, using the asymmetric hash algorithm and the hash modulus and exponent stored in MULTOS ROM, and compares it to the deciphered hash digest. Next, the MCD identifier of the deciphered controls data is compared with the reference MCD identifier stored in MULTOS EEPROM at IC manufacturing time as MISA injected security data.

In this manner, only secure values are accepted for security attributes present in MSM Controls Data and SF5 implements this requirement.

FMT_MSA.3

The default (and unique) values for security attributes used to enforce the Application Load SFP and Application Deletion SFP are provided by the MSM Controls Data which is certified by MSM.
This requirement is implemented by SF5.

FMT_MTD.1

The MSM is the only user able to load the MSM Controls Data because MSM Controls Data is encrypted by the MCD-specific transport key tkv that is generated by MSM. MSM Controls Data is decrypted on the MCD using the transport key stored in non-volatile MCD memory. If MSM Controls Data is not encrypted with the specific transport key, the load will fail. SF5 implements this requirement.

FMT_MTD.2

This requirement is implemented by SF5.

FMT_SMR.1

Users are associated to roles thanks to cryptographic keys. This requirement is implemented by SF1, SF2, SF3, SF4 and SF5.

FPR_UNO.1

The unobservability of cryptographic operations is implemented by SF11 and SF12.

FPT_FLS.1

a), b) (Application Memory Access and Unique Code and Static Data Space mechanisms) are implemented by the Application Execution Management SF (SF6).

c) is implemented by the Integrity Checks SF (SF9).

d) is implemented by the Tamper Resistant Hardware Behavior SF (SF12).

e) (MisExecution Detection Mechanism) and h) (Failed Command Counter mechanism) are implemented by the Tamper Resistant Software Behavior SF (SF11).

f) and g) are implemented by the Reset Protection SF (SF8).

FPT_PHP.3

This requirement is implemented by Tamper Resistant Software Behaviour SF (SF11) and Tamper Resistant Software Hardware Behaviour SF (SF12).

FPT_RCV.4

In the event of a reset, a power down, a power loss or an application abend, MULTOS will perform its validity checks and initialisation before entering its Ready State. In this way, the TSF finds back in a consistent and secure state. SF10 implements this part of the requirement.

If the number of failed attempts to execute the Create or Delete Controls Commands exceeds the MULTOS-defined limit the command is no longer available. Although this operational error reduces the functionality of the MCD, the security of MULTOS is not affected.

If the number of failed attempts to execute the Set MSM Controls Commands exceeds the MULTOS-defined limit, an EEPROM failure or an integrity failure occurs, the MCD is shutdown permanently and the TSF is in a secure state. SF10 implements this other part of this requirement.

FPT_RVM.1

MCD starts up in secure state thanks to Start-up validity checks and initialisation SF (SF10). Then the MCD is preserved in a secure state by Application Execution Management SF (SF6), Critical Data Overwrite SF (SF7), Reset Protection SF (SF8), Integrity Checks SF (SF9), Tamper Resistant Software Behaviour SF (SF11), Tamper Resistant Software Hardware Behaviour SF (SF12) and Smartcard Authentication (SF13). Moreover, interfaces between the TOE and users are protected by cryptographic operations and cryptographic keys associated to authorised users (SF1, SF2, SF3, SF4, SF5). In that way it is not possible to bypass the TSP.

FPT_SEP.1

The TSF maintains a security domain for its own execution by partially loading the Application Load Unit and MSM Controls Data (SF5) (untrusted subjects). If the authentication fails they are erased and the TSF is protected from interference and tampering. They are completely loaded only after they were completely authenticated and then became trusted subjects.

The TSF enforces separation between the security domains of loaded-applications as it is described by the Application Execution Management SF (SF6).

FPT_TDC.1

This requirement is implemented by Application Load Certificate Control SF (SF1), Application Delete Certificate Control SF (SF2), Unprotected/Protected ALU (SF3), Confidential ALU (SF4), MSM Controls Data Load Management (SF5).

FPT_TST.1

This requirement is implemented by the validity check and initialisation, which occur after a power-up/power down or a reset (SF10). It is also implemented by integrity checks (SF9) and hardware self-test (SF12).

FRU_RSA.1

As described in the Application Execution Management SF (SF6), some functions are required to allow MULTOS to address all the required X-RAM and EEPROM it needs. SF1 also checks if there is sufficient amount of EEPROM before loading the ALU components.

8.3.2 Assurance Measures Rationale

Table 5 shows that each assurance requirement is satisfied by at least one assurance measure.
There is no need of further explanation to demonstrate that the assurance measures are suitable to meet the TOE security assurance requirements, as this is obvious.

8.3.3 Protection Profile Claims Rationale

None.

Glossary

Abbreviations and acronyms

Term	Description
ABEND	Abnormal End (of MEL application execution).
ADC	Application Delete Certificate.
ALC	Application Load Certificate.
ALU	Application Load Unit.
APB	Application Pool Block.
ATR	Answer To Reset.
CC	Common Criteria (for Information Technology Security Evaluation, Version 2.1).
CM	Configuration Management.
DES	Data Encryption Standard (algorithm).
EAL	Evaluation Assurance Level.
EEPROM	Electronically Erasable Programmable Read Only Memory.
IC	Integrated Circuit.
IFD	Interface Device (to smartcard).
IT	Information Technology.
KTU	Key Transform Unit.
MCD	MULTOS Carrier Device.
MEL	MULTOS Executable Language (application language).
MSM	MULTOS Security Manager.
OSP	Organisational Security Policies.
PP	Protection Profile.
RAM	Random Access Memory.
ROM	Read Only Memory.
RSA	Rivest-Shamir-Aldeman (algorithm)
SF	Security Function.
SFP	Security Function Policy.
ST	Security Target.
TOE	Target Of Evaluation.
TSC	TSF Scope of Control.
TSF	TOE Security Functions.

TSFI	TSF Interface.
TSP	TOE Security Policy.

Vocabulary

Term	Description
Embedded software	Software embedded in a smartcard IC. Embedded software may be in any part of the non-volatile memory of the IC.
Integrated circuit (IC)	Electronic component(s) designed to perform processing and/or memory functions.
Phases	Refers to the seven phases of the smartcard product lifecycle, as outlined in the “Smartcard Integrated Circuit Protection Profile.”
I/O peripherals	Material components of the TOE that manage its inputs/outputs.
Smartcard	A card according to ISO 7816 requirements, which has a non-volatile, memory and a processing unit embedded within it.
Smartcard embedded software	Composed of embedded software in charge of generic functions of the smartcard IC such as operating system, general routines and interpreters (smartcard basic software) and embedded software dedicated to the applications (smartcard application software).

References

- Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, Version 2.1, August 1999.
- Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, Version 2.1, August 1999.
- Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements, Version 2.1, August 1999.
- Common Criteria for Information Technology Security Evaluation, Protection Profile, Smartcard Integrated Circuit with Multi-Application Secure Platform, Version 2.0, November 2000, registered by French Certification Board under number PP/0010.
- Common Criteria for Information Technology Security Evaluation, Protection Profile, Smartcard Integrated Circuit, Version 2.0, September 1998, registered by French Certification Board under number PP/9806.
- Common Criteria for Information Technology Security Evaluation, Protection Profile, Smartcard Integrated Circuit with Embedded Software, Version 2.0, June 1999, registered by French Certification Board under number PP/9911.