CRYPTOSMART-SIM CARD SECURITY TARGET

Publication date: 2019-05-15

Document revision: 911b7d5

Author	Validation	Approval
Julien Kowalski	Eric Laubacher	

Engineering. Réseaux. Communications.
6, rue Dewoitine - 78140 VELIZY - FRANCE
Phone: 01 39 46 50 50 Fax : 01 39 46 25 25
International phone: +33 1 39 46 50 50 International fax: +33 1 39 46 25 25
Web : www.ercom.fr email : info@ercom.fr

TABLE OF CONTENT

TABLE OF FIGURES

TABLES

1. Security Target Introduction

1.1 Security Target and TOE Identification

This document constitutes the Security Target (ST) of the Cryptosmart-SIM card, version 5.1 developed by Ercom.

- **ST name**: Cryptosmart-SIM card 5.1 – Security target
- **ST version**: 5.1
- **ST Date** : 2019-05-15
- **TOE identifier**: Cryptosmart-SIM applet 5.1 on IDEMIA DRAGONFLY V4.0
- **TOE version**: 5.1
- **TOE Developer** : ERCOM & IDEMIA
- **Evaluation sponsor** : ERCOM

This security target addresses a composite TOE evaluation in the sense of [1] where:

- The certified platform IC is an Infineon Technologies Smart Card IC (Security Controller) SLE97CNFX1M50PE
- The Java Card platform is the IDEMIA DRAGONFLY V4.0;
- The application is the Cryptosmart-SIM applet V 5.1.

The platform is identified as follows:

Platform name	DRAGONFLY V4.0
Platform identification	SAAAAR code = “412691”
Reference of the CC certificates of the platform	ANSSI-CC-2016/60-S01 (cf. [2] and [3])
Platform IC reference version	SLE97CNFX1M50PE with Mifare-compatible libraries
Reference of the CC certificates of the underlying IC	BSI-DSZ-CC-0946-V2-2015

The TOE can be uniquely identified by correct platform identification (cf. [4]) and the following fields from card status (cf. [5]):

Applet version	5.1
Source version	914
Build version	`ca c7 fa 53`

The Cryptosmart card developer guide [5] and the Cryptosmart card personalization guide [6] are part of the TOE.

1.2 Conformance Claims

This Security Target claims conformance to **CC version 3.1** with the following documents:

- "Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", April 2017, Version 3.1 Revision 5 (CCMB-2017-04-001)

- "Common Criteria for information Technology Security Evaluation, Part 2: Security Functional requirements", April 2017, Version 3.1 Revision 5 (CCMB-2017-04-002)
- "Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance requirements", April 2017, Version 3.1 Revision 5 (CCMB-2017-04-003)

Conformance is claimed as follows:

- Part 2: conformant.
- Part 3: conformant. The chosen Evaluation Assurance Level is EAL4 augmented with ALC_DVS.2 and AVA_VAN.4

Conformity to a protection profile: This Security Target does not claim conformance with any Protection Profile.

1.3 CONVENTIONS

APDU	Application Protocol Data Unit
CA	Certificate Authority
DH	Diffie-Hellman
ERCOM	ERCOM S.A.
IA	Identification and Authentication
PP	Protection Profile
SFP	Security Function Policy
ST	Security Target
TOE	Target of Evaluation – called CC for system being evaluated
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSP	TOE Security Policy

1.4 TERMINOLOGY

Card admin station	Device for managing a fleet of Cryptosmart-SIM cards.
Administrator	Person authorized to access the administration station and manage the fleet of Cryptosmart-SIM cards. The administrator is the user of the applet before it is delivered to the final user.
Authentication	Service ensuring the identity of a card.
Certificate	Identity data and public key of a user signed by the private key of the certification authority.
Card personalization	This is the process where an administrator injects or generates the cryptographic keys inside the Cryptosmart-SIM card. The name certification comes from the fact a user certificate shall be imported inside the smartcard during this process.
Clone card	Device having the capacity to substitute for a legitimate card, in particular to authenticate itself and generate a valid session key.
CRL	Certificate Revocation List. List of certificates that are no longer authorized.
Cryptosmart-SIM card	Smartcard incorporating the Cryptosmart applet
Host	Represents the module managing communications with the smartcard. By extension it may be viewed as the device embedding the Cryptosmart-SIM card.
Local attacker	Third party person trying to corrupt or recover sensitive data by accessing a Cryptosmart-SIM card directly without knowing the security code. This may be a legitimate user of a card of the same family as the TOE. The legitimate user of the TOE and the administrator are excluded from this definition. Some data shall not be modified or recovered even knowing the TOE security code. In this case even the legitimate user may be considered as an attacker against these data.
MAC	Message Authentication Code, a message sealing and integrity verification mechanism with secret key.
Masquerade	Action aimed at deceiving a correspondent about his real identity.
Online attacker	Third party person trying to corrupt or recover sensitive data by intercepting and/or modifying the flows between equipment using Cryptosmart-SIM cards. The online attacker may possess lost or stolen cards of the same family as the TOE and knowing their Security Code. It may also be a legitimate user of another card of the same family as the TOE. The legitimate TOE user, the legitimate user of the card with which the TOE has to establish a session and the PKI administrator[1] are excluded from this definition.
Reset	Re-initialization of the smartcard volatile memory.
Session key	256-bit key generated by the Cryptosmart-SIM card at each Cryptosmart authentication protocol run. This key is provided to the host under a wrapped form. It is derived by the Cryptosmart-SIM card into keys transmitted to the host. These keys protect the flows exchanged between hosts.
User	Person carrying a user type Cryptosmart-SIM card and knowing his security code
IR_SFR	Platform SFR irrelevant to the Composite TOE
RP_SFR-SERV	Platform SFR relevant to fulfill/support security services, i.e. Composite TOE SFRs
RP_SFR-MECH	Platform SFR relevant to support non tampering and non bypassability

1 As a dishonest PKI administrator may issue any certificates he may impersonate any other user.

1.5 REFERENCES

[1] Common Criteria for information Technology Security Evaluation, «CCDB-2012-04-001 Composite product evaluation for Smart Card and similar devices version 1.1 revision 1,» 2012.

[2] ANSSI, «Rapport de certification ANSSI-CC-2016/60 DragonFly v4.0 sur composant SLE97CNFX1M50PE; Identification Hardware 412691; Card Manager GOP Ref V21.06.01,» 2016.

[3] ANSSI, Rapport de surveillance ANSSI-CC-2016/60-S01 DragonFly v4.0 sur composant SLE97CNFX1M50PE Identification Hardware 412691 Card Manager GOP Ref V21.06.01, 2018.

[4] Oberthur Technology, «Security Target Lite DragonFlye 4.0 issue: 1,» 2016.

[5] Ercom, «Cryptosmart-SIM card Developer's guide - revision: cbb03ac,» 2019.

[6] Ercom, Cryptosmart Card Personalization guide, ed. e9a4cd9, 2018.

[7] Samsung, «Samsung UCM Unified Credential Management Project - Boot Plugin Java API documentation and APDU format definition,» 2017.

[8] Oracle, «Java Card System Protection Profile - Open Configuration, version 3.0,» 2010.

[9] ANSSI, «Référentiel Général de Sécurité version 2.0 - Annexe B2 - Gestion des clés cryptographiques,» 2012.

[10] O. technologies, «Dragonfly 4.0 Application Management guide».

[11] ANSSI, «Référentiel Général de Sécurité version 2.0 - Annexe B1 - Mécanismes cryptographiques,» 2012.

[12] NIST, «FIPS PUB 197 ADVANCED ENCRYPTION STANDARD (AES),» 2001.

[13] NIST, «FIPS PUB 180-4 Secure Hash Standard (SHS),» 2015.

[14] NIST, «FIPS PUB 198-1 The Keyed-Hash Message Authentication Code (HMAC),» 2008.

[15] NIST, «Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication,» 2016.

[16] RSA Laboratories, «PKCS #1 v2.1: RSA Cryptography Standard,» 1999.

[17] RSA Laboratories, «PKCS #3: Diffie-Hellman Key Agreement Standard,» 1993.

[18] Oracle, «Java Card 3.0.4 API, Classic Edition».

[19] Common Criteria for Information Technology Security Evaluation, «CCMB-2012-09-001 Part 1: Introduction and general model" Version 3.1 Revision 4,» 2012.

[20] Common Criteria for information Technology Security Evaluation, «CCMB-2012-09-002 Part 2: Security Functional requirements" Version 3.1 Revision 4,» 2012.

[21] Common Criteria for information Technology Security Evaluation, «CCMB-2012-09-003 Part 3: Security Assurance requirements" Version 3.1 Revision 4,» 2012.

[22] «(U)SIM Java Card Platform Protection Profile - Basic Configuration, version 2.0.2».

[23] ANSSI, «Note d'application 4758/ANSSI/SDE/PSS/CCN - certification de produits ouverts de type carte a puce,» 2014.

[24] Oracle, «Java Card 3 Platform, Runtime Environment Specification, Classic Edition, Version 3.0.4,» 2011.

[25] NIST, «FIPS PUB 186-4 Digital Signature Standard (DSS),» 2013.

2. TOE DESCRIPTION

2.1 TOE OVERVIEW

The TOE consists of a Javacard applet developed by Ercom running on a smartcard running the IDEMIA **DragonFly v4.0** Javacard OS on Infineon chip.

The TOE has three main usages:

- Provide the ability to authenticate a distant Cryptosmart-SIM card and negotiate a shared key with it;
- Act as a cryptographic processor, by providing cryptographic function which are:
  - User authentication by using a security code
  - Secure key storage, either for 256 bits symmetric keys or 2048 bits RSA keys
  - Key export to host interface, either for 256 bits symmetric keys or 2048 bits RSA keys
  - A subset of PKCS#11 functionalities. The smartcard offers in device cryptography. This allows performing cryptographic computation without exposing the secret key. Operations supported by the TOE are:
    - Symmetric encryption and decryption using AES-256 .
    - Symmetric MAC generation and verification using HMAC-SHA256. This feature is out of the scope of the evaluation.
    - RSA-2048 decryption using PKCS#1.5 and PKCS#1.5-OAEP padding schemes. Decryption using the PKCS#1.5 padding scheme is out of the scope of the evaluation.
    - RSA-2048 signature using PKCS#1.5 padding scheme with no hash function (which allows the usage of any hash function: digest shall be performed by the host)
  - 2048 bits RSA key generation
  - 256 bits symmetric key generation
  - Management of the extractable property for RSA and symmetric keys
  - Management of the key usage property for symmetric keys
  - “Local encryption key” obtention by deriving internal keys
  - Wrapped Device Encryption Key generation and unwrapping
  - Random Number generation
- Provide a secure storage area.

The PKCS#11 C interfaces to smartcard functionalities is provided by the card driver which translates PKCS#11 C functions calls into smartcard commands. This driver is out of the scope of the evaluation and is not required to use the TOE.

The smartcard PKCS#11 functionalities are called by extension PKCS#11 functions in this document.

2.2 TOE SAMPLE USAGE: THE CRYPTOSMART SYSTEM

This section presents a Cryptosmart-SIM card usage example.

Ercom has developed a product family called Cryptosmart for mobile and nomadism security. It protects any kind of devices (mobile, laptops, fixed phones, vehicles) on any kind of networks (mobile, wireless, wireline, satellite) for any kind of applications (mail, voice, SMS, video, business applications). In all cases, it always uses a Cryptosmart-SIM card.

The Cryptosmart system consists in:

- A Cryptosmart Gateway which interconnects external devices with the enterprise/customer IT center for data communications via a VPN and for encrypted-clear calls ; it also provides routing and SIP functionalities for secure end-to-end VOIP calls between two devices.
- Cryptosmart PC Suite and Cryptosmart Mobile Suite : it provides protection of the devices, protection of the data flows, and protection of voice calls
- Cryptosmart card (the TOE being the SIM instance of the Cryptosmart Card) providing strong authentication and key management functions for both the Cryptosmart Gateway and the devices ;
- Cryptosmart cardAdminStation for administration and management of Cryptosmart-SIM cards.

The Cryptosmart (-SIM) card acts as a root of trust for the Cryptosmart system by

- protecting essential secrets which allow user authentication,
- managing cryptographic keys which allow the protection of user's sensitive data and communications.

For example, thanks to the use of the Cryptosmart-SIM card (TOE), Cryptosmart Mobile Suite provides the following security features for PDA/Smart phones:

- Strong authentication
- Voice encryption
- SMS protection
- VPN
- Local device encryption
- Network firewall

The secure communications provided by the Cryptosmart solution protect the user's sensitive data in confidentiality, integrity and authenticity.

The following diagram presents an example of architecture with Cryptosmart Mobile Suite, Cryptosmart PC Suite, Cryptosmart Gateway, and Cryptosmart cardAdminStation. Thus, users can:

- Get access to their professional mail securely
- Get access to their intranet securely
- Get access to the internet via a proxy and thus benefit from the already established policy of their organization in terms of internet access
- Make secure end-to-end calls
- Call any anyone in their organization while securing the outside path of the communication thanks to the PBX interconnection
- Send secure SMS

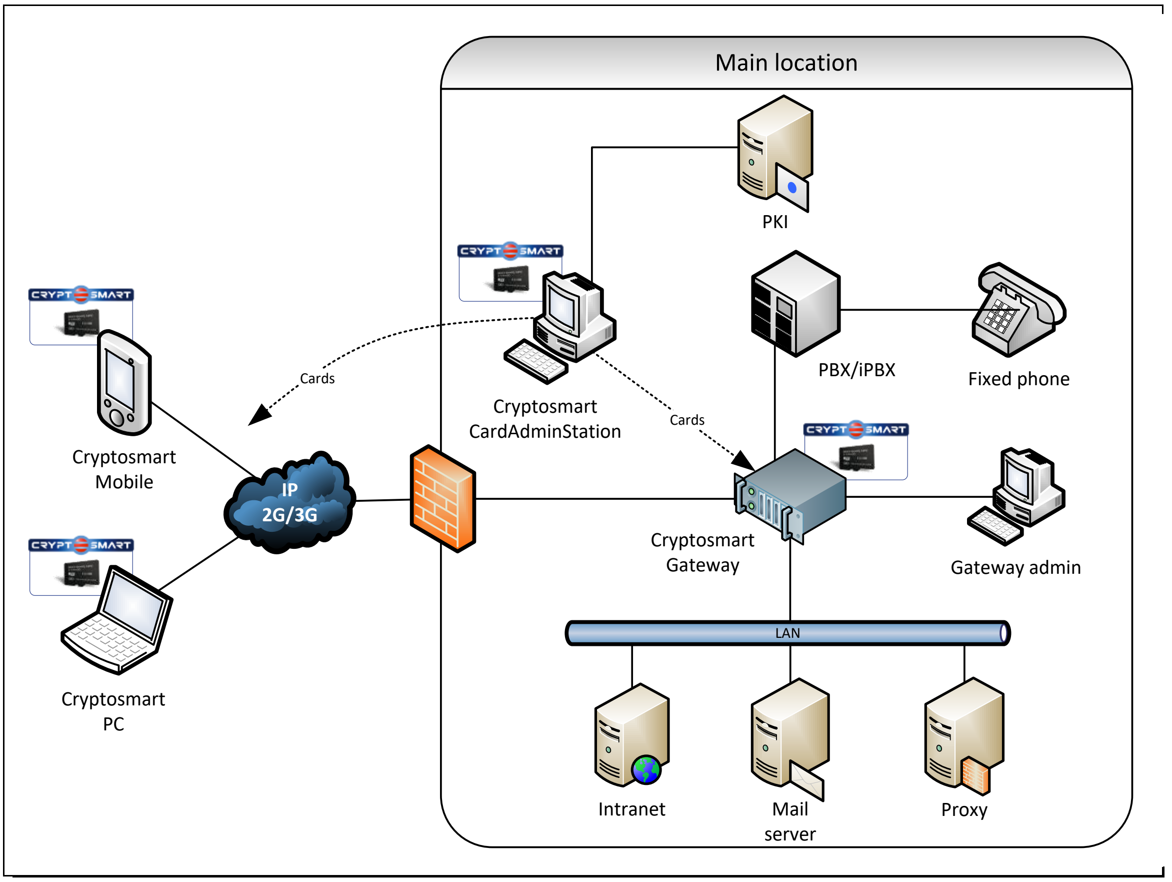


Figure 1: Cryptosmart sample architecture

Mobile devices are by definition easily lost or theft. Usage of the TOE in this system allows a secure storage of user secrets which are never directly stored on the mobile device.

- User data remain protected in case of theft by an attacker as protection keys are stored by the smartcard and using the Cryptosmart card requires a security code;
- The most sensitive cryptographic keys never appear on the host device: cryptographic operation involving them are performed in card;
- The device does not become a threat for the customer system as data required for authentication remain in the smartcard, not accessible to the attacker. The attacker may not enter the customer system by authenticating as the user he compromised.

The TOE also implements the “UCM Bootplugin API” ([7]) in order to support ODE (Full Disk Encryption) on Samsung Devices. This API allows user authentication and the secure management of a device encryption key (DEK), as well as an “SCP11 like” APDU protection.

Usage of the TOE in the Gateway for authentication purposes provides the following advantages in case of Gateway compromise:

- The attacker may not authenticate as a valid Gateway as authentication private keys are stored by the card and are not extractable;
- The Cryptosmart authentication protocol involves symmetric secrets as a second security layer over the asymmetric one. Smartcard usage allows protecting these secrets. Otherwise they would have been stored readable on the Gateway.

2.3 TOE DESCRIPTION

2.3.1 USAGE OF THE TOE

There exists two modes of delivery for the Cryptosmart-SIM card:

1. The Cryptosmart-SIM card is delivered to customers in a state in which no key is installed. An administrator is then in charge of personalizing it for each user. Card personalization is done in this case by using a software tool called card admin station provided by Ercom which exists in two versions:
    - A command line version allowing automation of the personalization process;
    - A graphical version allowing card by card personalization.
2. The Cryptosmart-SIM card is delivered to customers fully personalized. In this latter case personalization is performed by IDEMIA.

The personalization process consists in providing to the smartcard its cryptographic keys and certificates. Cryptographic keys may either be

- Generated inside the smartcard allowing an extremely secure configuration as these keys may never be extracted outside the smartcard;
- Imported inside the smartcard.

This latter case reaches the same security as the first if the customer's information system stores imported keys in a secure way. It allows a more flexible management of the cards: a card containing the same keys can be generated. It allows card replacement without interrupting the services offered by the smartcard.

Card personalization requires usage of an external PKI to generate the user certificate on which distant user authentication relies. As other keys, the corresponding private key may be either generated internally to the smartcard or externally by customer's PKI.

During card personalization, the administrator also:

- Sets a minimal security code length and a default code. The TOE user can change this code. This minimal security length for security codes must be comprised between 4 and 8, or be 0 to avoid usage of a user security code. This must be set at card creation, and a user won't be able to reduce the initial code length.
- Asks for PUK code creation: the TOE generates 15 PUK codes and exports them. PUK codes may never be exported afterward. PUK codes have a length of 8 digits.
- Generates and injects a recycle code.
- Inject cryptographic keys inside the smartcard which are:
    - The family key which is a key shared by every card of a family. (note: an administrator may administrate several families);
    - The wrapping key. The wrapping key may also be generated internally to the smartcard.
    - The local encryption master keys.
    - The APDU encryption initialization key
    - The user authentication key and corresponding certificate (used in Cryptosmart authentication protocol)
- Sets the card type (user card or gateway card)

PUK codes are used for unlocking the smartcard in case the user blocks his card by entering wrong security codes.

The family key is derived into an identity protection key used during Cryptosmart authentication protocol run.

The wrapping key is a key divided into an encryption key and an integrity key. These key are used to export keys negotiated by the Cryptosmart authentication protocol to the host while protected from disclosure and modification.

The local encryption key is a key which may never be extracted from the TOE but may be derived into several keys. Those derived keys are exported to the host.

The card type defines user cards and gateway card. Gateway cards have the same functionalities as user cards but are optimized to be used on a server. This is useful in the case if the TOE is used in standalone appliances where cryptographic operation shall be performed without user interaction.

A specific command ends the personalization process: from now the card only allows key generation or import for PKCS#11 functions. The keys injected by the administrator (APDU encryption initialization key and user authentication key) can neither be changed by the user nor be reached directly by the user (he may only use them). The card is in a state where it can be delivered to the user.

When a user receives its smartcard he should change the default security code set by the administrator. He may use any functionality of the smartcard described hereafter.

The user may block his smartcard by entering a wrong security code several times (only 2 successive failures are authorized. The third failure blocks the card).

The card can be unblocked by entering a PUK code (which has been generated during the personalization process). The card status contains the current PUK code number to use. To unblock the card the user shall enter this precise PUK code (entering another PUK code even generated by the TOE during personalization will not unblock the card and count as failure). After 10 unsuccessful PUK code entries the whole content of the card but the recycle code is wiped. The card may only be recycled. Once entered successfully a PUK code unblock the card and cannot be used again (the current PUK code number is increased).

If the user blocks the card and no PUK code are available the card content is wiped.

The user unlocks the card by entering his security code. He may explicitly lock the smartcard by logging off.

Once the smartcard unlocked the user may use any functionality described § 2.1.

As the user certificate has a limited validity date, the smartcard has the same validity date. At this date the user shall receive a new smartcard. Continuity of cryptographic services which do not involve this certificate is ensured by personalizing both cards with the same keyset. The old card shall then be recycled by using the recycle code. Card recycling erases any data contained in the card. This card may be re- personalized for another user.

2.3.2 TOE FUNCTIONALITIES

This part describes the services that the TOE offers to its users. These services are:

- User authentication;
- Cryptosmart authentication protocol run;
- Cryptosmart “stateless” authentication protocol run;
- Signature generation (both symmetric and asymmetric);
- Encryption (AES) and decryption (AES and RSA);
- External key storage;
- Key generation;
- Local protection key obtention
- Wrapped Device Encryption Key generation and unwrapping
- User data storage
- Key or user data export.

The TOE offers also an interface for the host to get TOE status which contains the current card state, the serial number, current security code entry failures and other information about the TOE.

Note: the symmetric signature generation services are not included in the TOE.

User authentication

Each TOE user possesses a 4 to 8 digits security code. The TOE authenticates its user by confronting the security code given by the user to the value it stores. A successful authentication unlocks the card functionalities and allows the user to perform cryptographic operations.

After 3 security code entry failures the TOE is locked in a “BLOCKED” state and may only be unblocked by a correct PUK code entry or erased and reset by a correct recycle code entry.

PUK codes also allow authenticating the user. Anyway PUK code entry is forbidden in any state other than BLOCKED.

Security code change

The TOE allows the user to change its security code. This can only be done if the user has previously authenticated himself to the TOE.

Security code choice is free to the user with the only limitation that its length shall be comprised between the lower bound set by the administrator and 8.

Key negotiation (Cryptosmart authentication)

To establish a secure channel between two devices hosting a Cryptosmart-SIM card, the smartcard performs:

- The mutual authentication of both components.
- The negotiation of a session key which will be derived into cryptographic parameters and keys.

The authentication of the distant user is based on standard X.509 certificates.

This key negotiation is based on a DH based security proven authentication protocol.

The Cryptosmart-SIM card provides two implementations of the authentication protocol: One is statefull for being used in a connected mode, the other one is stateless for being used in an asynchronous mode.

Keys negotiated during the authentication protocol are exported under a wrapped form: protected both in confidentiality and integrity. They must be derived to get keys shared between both devices.

The Cryptosmart-SIM card wrapping method since version 5.1 is different from the 5.0 version. A compatibility mode is provided. This mode shall be deactivated for being in the evaluated configuration.

Key protection

The TOE offers keys storage capabilities for external application (Cryptosmart suite, S/MIME application, third party local encryption software, etc.) and for PKCS#11 purposes. A user may generate cryptographic key through the Cryptosmart-SIM card. He can store cryptographic keys inside the smartcard. The TOE protects these keys by conditioning their access to the user authentication.

Keys stored may be either RSA keys with a 2048 bits length or 256 bits symmetric keys. Both types of keys may be extractable or not. A user can never have access to the value of non extractable keys. He can only use them to perform cryptographic operation.

Symmetric keys have a key usage attribute which defines if the key may be used for encryption, signature or both usages.

Key derivation

Key derivation is performed inside the smartcard. This allows the master session key (negotiated during the authentication protocol) to remain unexposed even to the legitimate user. The derivation process also derives a fresh key from the input, returned under a wrapped form.

RSA signature

The TOE allows the user to sign buffers using RSA with a PKCS#1.5 padding scheme. The signature private key is one of the RSA keys stored internally.

Note: the TOE only performs operation with private key. Public key operation (RSA signature verification) is performed by the host.

RSA Decryption

The TOE allows the user to decrypt buffers using RSA with a PKCS#1.5 or OAEP padding scheme. The decryption private key is one of the RSA keys stored internally.

Note 1: the TOE only performs operation with private key. Public key operation (RSA encryption) is performed by the host.

Note 2: Decryption of data using the PKCS#1.5 padding scheme is excluded from the evaluated configuration.

HMAC-SHA256 signature/ verification

The TOE allows the user to sign buffers or verify buffer signature using SHA256 in HMAC mode. The signature key is one of the symmetric keys stored internally.

This feature is excluded from the evaluated configuration.

AES encryption / decryption

The TOE allows buffer encryption or decryption using AES in ECB or CBC mode. The key is one of the symmetric keys stored internally.

By default symmetric keys have a “disable security check” attribute set to false: each symmetric key is limited to perform 150 block operation (encryption and decryption). This check can be disabled at key creation.

Remark: RSA, AES and HMAC operation offer PKCS#11 functionalities. The related key management is provided by the TOE. A user can generate, import cryptographic keys, set their extractable attribute. An administrator can also generate or import cryptographic keys and set their extractable attributes during the certification process. In this case the user will be able to manage these keys just as if they were imported by him.

Local protection keys obtention

The TOE has 2 “local protection master keys”, which are not extractable. The TOE allows deriving these keys for the user to obtain 256 bits keys which may be used for local protection.

Wrapped Device Encryption Key generation and unwrapping

The TOE is able to generate and export a 256bits key, wrapped by internal keys.

The TOE allows unwrapping it depending on its internal state.

Random number generation

The TOE allows random number generation using the card processor true random generator which is post-processed by a cryptographic deterministic random number generator.

Wipe order

The TOE allows the user to send it a wipe command. This command erases every secret contained inside the TOE: the TOE has the same content as when it was delivered to the customer. The only commands allowed in this state are to get the card status and to recycle the card, to be able to reuse it.

2.3.3 TOE additional features

This part describes some features of the TOE independent from user actions.

APDU encryption

Because an attacker has the possibility to listen to the communication channel between the TOE and the host, the TOE implements an encryption mechanism which protects this channel. All sensitive command and associated answer is sent encrypted. Security code entries, PUK unlock commands, and every other command sent when the user has authenticated to the TOE is sent protected in confidentiality.

The encryption mechanism provides the following protection to communication between the TOE and the host:

- Protection in confidentiality through AES encryption;
- Protection in integrity through symmetric signature any command modified by an attacker is detected and rejected;
- Protection against replay: an attacker can't replay an encrypted command he previously intercepted.

The TOE provides two implementations of APDU encryption:

1. The first one is proprietary and used for all Cryptosmart specific commands
2. The second one is based on a variant of the SCP11 standard and is used for supported ISO7816 commands.

Power consumption management

The TOE implements functions to efficiently manage power consumption. The TOE is a smartcard which must be powered on to perform its operations. If unpowered, the user shall re-authenticate to be able to use it.

The TOE implements an ephemeral security code mechanism to manage these re-authentications. Once the user is authenticated, the TOE generates an ephemeral security code at random and sends it to the host. This ephemeral security code is managed by the TOE driver on the host.

The TOE driver uses it to automatically re-authenticate the user after a TOE power off. The user has no action to perform by himself: the loss of power becomes transparent. When the host device is locked by the user, the driver erases the ephemeral security code from its memory forbidding its compromise by an attacker.

If authentication using this ephemeral code fails, the TOE erases the ephemeral code it stores and returns with error.

"Light" unlock

When the host device has been locked, some software shall be able to perform cryptographic operations in background. To handle this case the TOE has a special state called RESUMED_USER. In this state the user has a limited access to the TOE. Some PKCS #11 operations are forbidden: each key has a user defined "useable in resumed" attribute. If this attribute is set to false, any command using the corresponding key is forbidden (as if the TOE was locked).

The TOE may reach the RESUMED_USER after a power loss by sending a dedicated correctly encrypted command. The fact that this command is correctly encrypted implicitly authenticates the user as:

- The symmetric signature of the APDU encryption mechanism proves that the driver possesses the correct encryption key
- The driver may possess the correct encryption key if and only if the user already successfully authenticated to the TOE.

2.3.4 TOE LIFE CYCLE

This part describes the different states of the TOE, with the corresponding possible transitions.

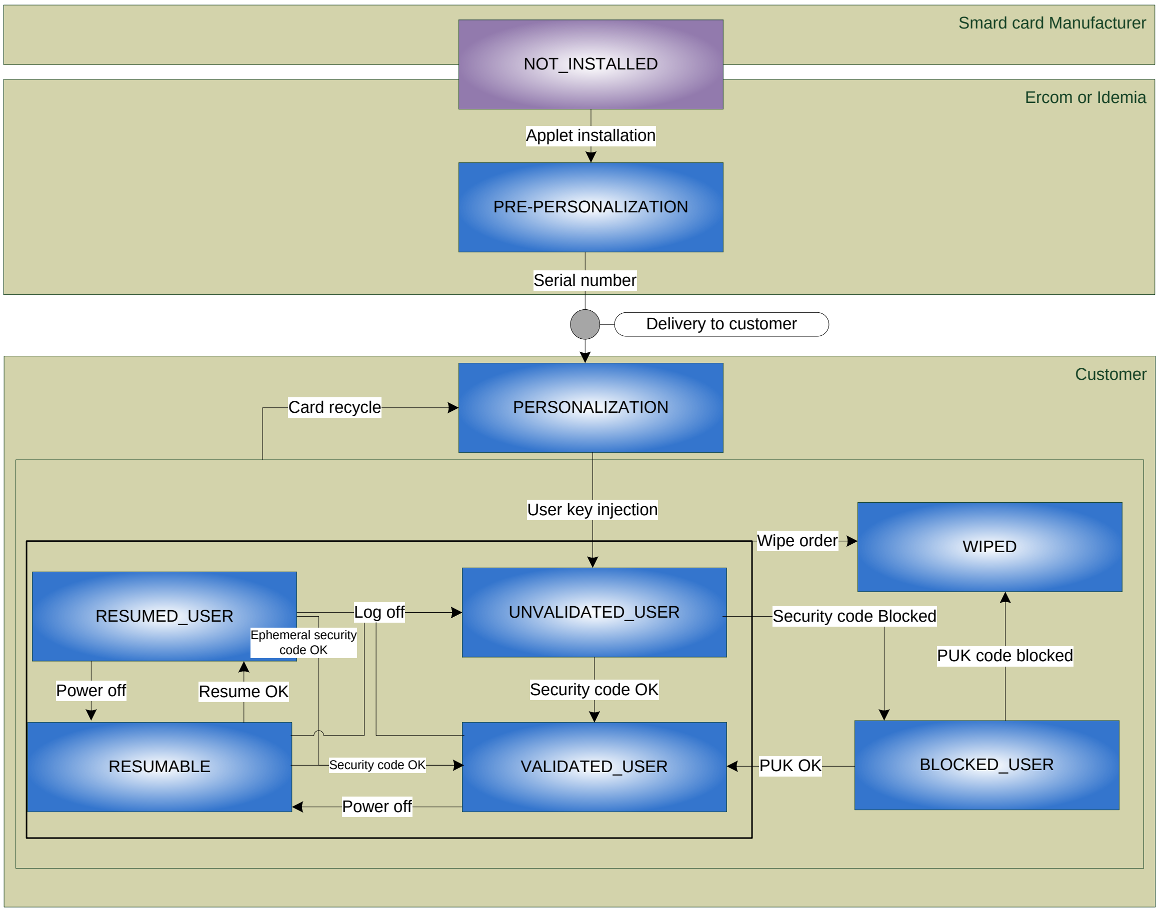


Figure 2: Cryptosmart-SIM card life cycle (delivered in Personalization state)

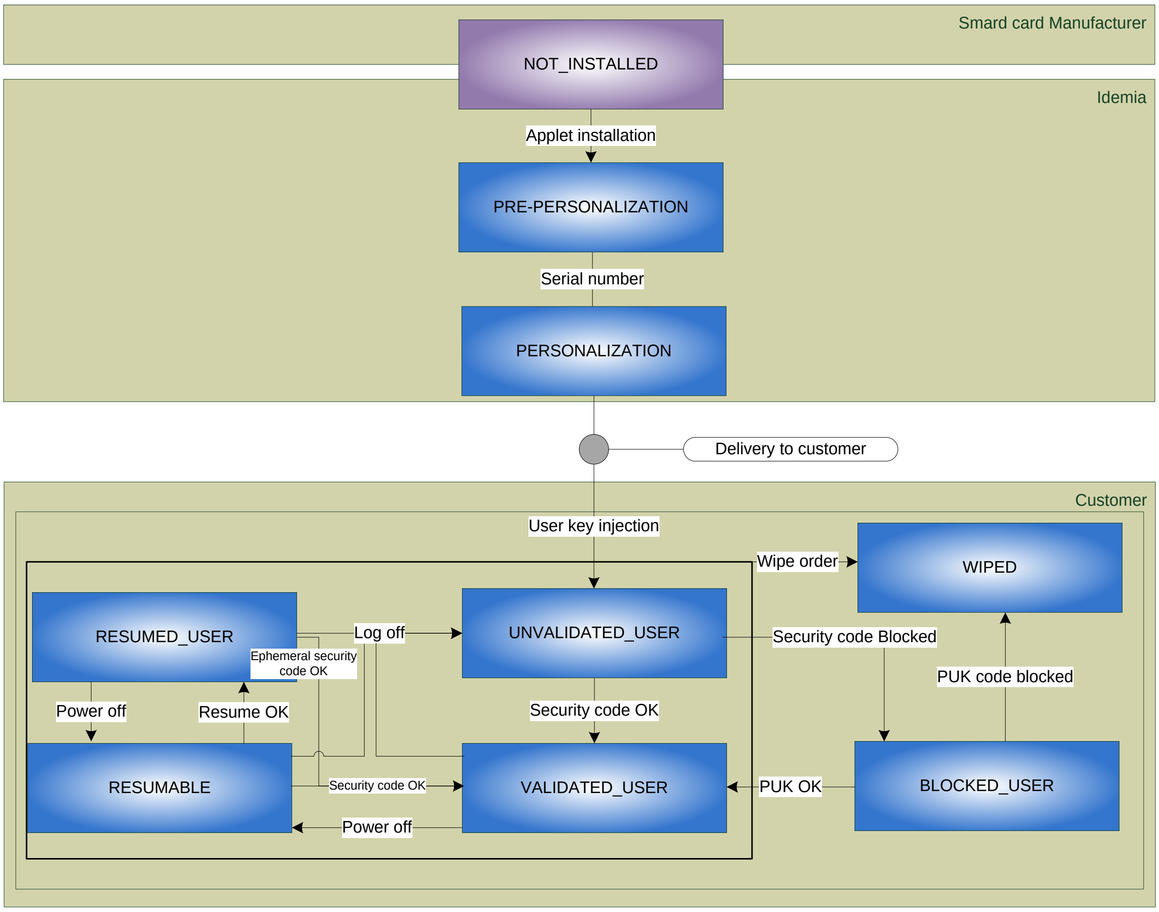


Figure 3: Cryptosmart-SIM card life cycle (delivered in UNVALIDATED_USER state)

Cryptosmart-SIM cards consist of a Cryptosmart Java applet installed on a Javacard platform compatible with Javacard 3.0.4 and Global Platform 2.2.
Cryptosmart-SIM cards can be in one of the states described in Table 1.

Life Cycle State	Comment
NOT_INSTALLED	Card delivered by the card manufacturer to ERCOM. This state is never seen by the end customer.
PRE-PERSONALIZATION	Card with Cryptosmart applet loaded.
PERSONALIZATION	Card with Cryptosmart applet and serial number loaded.
UNVALIDATED_USER	Card state when the card contains its keys. The card is not Security Code unlocked.
VALIDATED_USER	Card unlocked with validated Security Code or ephemeral security code.
RESUMABLE	Card after a card reset from a validated user state
RESUMED_USER	Card after a successful resume command from RESUMABLE state
BLOCKED_USER	Card blocked after repeated Security Code errors.
WIPED	Card with every keys erased

Table 1 : Cryptosmart-SIM card life states

NOT_INSTALLED AND PRE-PERSONALIZATION

During these states IDEMIA:

- Inserts the applet inside the card;
- Inserts the card serial number;
- Configures card functionalities;
- Inserts delivery information;

At the end of this phase the TOE can be delivered to the customer if he requires performing the PERSONALIZATION step himself.

PERSONALIZATION

This state allows card personalization i.e.:

- User key pair generation or import
- User symmetric key generation or import
- User certificate import
- PUK codes generation
- Family key insertion
- Recycle code insertion
- Security code and its properties insertion

Only a subset of the functions of the TOE is accessible in this state.

Personalization operations can be performed using the card-admin station which also allows management of the cards.

Once every user keys have been inserted, the card may get into the UNVALIDATED_USER state. This state transition is explicitly ordered by a call to the function “create card”.

UNVALIDATED_USER

The smartcard is in this state when no security code has been successfully entered inside one session. This state allows access only to the functions:

- Get card information;
- Initialize the APDU encryption;
- User authentication;
- Wipe the card;
- Card recycle.

Once the security code has been successfully entered, the card shall get into the VALIDATED_USER state.

If the maximum security code entry attempt has been reached, the card shall get into the BLOCKED_USER state.

After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

A wipe order turns the card into a WIPED state.

VALIDATED_USER

In this state, every user function is accessible.

The VALIDATED_USER state corresponds to the operational state: a fully personalized card, with the user security code successfully entered.

After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

After a log off, the card gets into an UNVALIDATED_USER state.

After a card reset (power off), the card gets into a RESUMABLE state if allowed by configuration.

After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

A wipe order turns the card into a WIPED state.

RESUMABLE

This state allows every one of the following actions and no other:

- Get card information
- Enter the security code
- Wipe the card;
- Resume command
- Log off

Once the security code has been successfully entered, the card shall get into the VALIDATED_USER state.

If the resume command is successful and contains the ephemeral security code, the card shall get into the VALIDATED_USER state.

After too many unsuccessful security code entries, the card shall get into a BLOCKED_USER state.

If the resume command is successful without the ephemeral security code, the card shall get into the RESUMED_USER state.

If the resume command is unsuccessful, the card shall get into the UNVALIDATED_USER state.

After a log off command, the card must get into the UNVALIDATED_USER state.

After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

A wipe order turns the card into a WIPED state.

RESUMED_USER

The smartcard is in this state when a successful resume without ephemeral security command has been sent to the card from the RESUMABLE state.

This state allows only a restricted usage of the TOE:

- The authentication protocol and the derivation function are accessible
- Usage of the Cryptosmart-SIM card as a cryptographic token is allowed, but the usage of some keys in this state may be forbidden by the user

Once the security code has been successfully entered, the card shall get into the VALIDATED_USER state.

After too many unsuccessful security code entries, the card shall get into a BLOCKED_USER state.

After a log off command the card must get into the UNVALIDATED_USER state.

After a card reset, the card shall get into a RESUMABLE state.

After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

A wipe order turns the card into a WIPED state.

BLOCKED_USER

The card is in this state once the maximum security code entry has been reached. The card shall refuse any command except a PUK code entry or commands to get the card status.

This state allows every one of the following actions, and no other:

- Get card information
- Enter PUK code

After a successful PUK code entry, the card shall get into a VALIDATED_USER state.

After 10 successive wrong PUK code entries, the card shall get into a WIPED state.

If no PUK code is available the card shall immediately get into a WIPED state.

After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

A wipe order turns the card into a WIPED state.

WIPED

The card is in this state once the PUK code entry maximum attempts number has been reached; or the card has received a “wipe” order.

The card shall refuse any command except a recycle or a get card status command.

In this state every customer secret has been erased except the recycle key.
After a successful recycle code entry, the card shall get into a PERSONALIZATION state.

2.3.5 TOE USER

The TOE possesses two users. They are identified implicitly depending on TOE state

The card administrator is the TOE user while it is in PERSONALIZATION or WIPED states. He performs card personalization actions. In WIPED state the card administrator may recycle the card (using the recycle key). As in these states the commands require no authentication the administrator shall operate the card in a secure environment.

The second user is the “local user”. He is the TOE user while it is in UNVALIDATED_USER, VALIDATED_USER, RESUMABLE, RESUMED_USER, or BLOCKED states.

2.3.6 USER CARDS AND GATEWAY CARDS

The Cryptosmart-SIM card can be personalized into two card types: user cards and gateway cards.

Gateway cards are Cryptosmart-SIM card intended to be used in a server and may be used heavily. For these reasons, gateway cards differ from user cards by the possibility to reuse the private Diffie-Hellman parameter for Cryptosmart mutual authentication protocol run.

Otherwise the card would only offer a limited number of authentications as only a limited write number in EEPROM are allowed. The host has to reset this parameter for the card to use a new one.

In the scope of this evaluation we assert that the host sends a command to renew the private DH parameter after each authentication protocol run.

The administrator has also the possibility to set a 0 length security code for Gateway cards. In the scope of this evaluation we assert this is not the case.

2.4 INTEGRATION IN JCS LIFE CYCLE

[8] describes the life cycle of a javacard system (JCS) as follows:

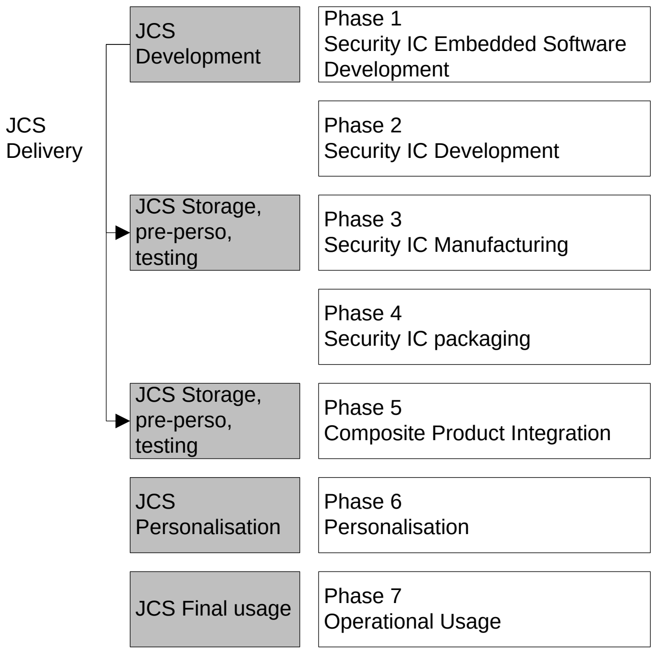


Figure 4: JCS life cycle

Phase 1 to phase 5 are performed as described in [4].

Personalization in phase 6 is decomposed into two parts:

- One is concerning the personalization of the javacard platform.
- One concerns installing the Cryptosmart applet and performing pre-personalization operations. This corresponds to PRE-PERSONALIZATION state of the TOE

This later part may be performed either by Ercom on Ercom premises (Velizy) or by IDEMIA on any premise covered by ANSSI-CC-2016/60 certificate.

The delivery point is at the end of phase 6.

Phase 7 corresponds to other subsequent states (PERSONALIZATION, UNVALIDATED_USER, VALIDATED_USER, RESUMABLE, RESUMED_USER, BLOCKED_USER and WIPED) of the TOE.

TOE Component	format	Sender	Delivery method
Cryptosmart-SIM card	smartcard	Ercom	Private transporter / postal
		IDEMIA	Secure transport

TOE Component	format	Sender	Delivery method
Guide [5] and [6]	Pdf document	Ercom	E-Mail

Table 2: TOE delivery method

2.5 COMPOSITE TOE SCOPE

The TOE to be evaluated is composed of TOE part 1 and TOE part 2.

TOE part 1 identifies the DragonFly 4.0 platform on the certified IC. This TOE part includes software (the firmware, the Card Manager and the operating system) embedded on a microcontroller

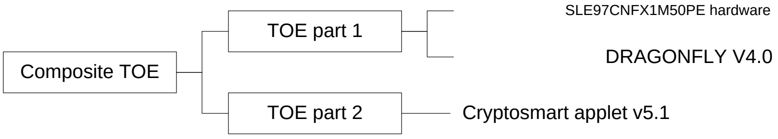
TOE part 1 is a javacard platform compliant with Java Card 3.0.4 and Visa GlobalPlatform 2.2.

TOE part 2 identifies the Cryptosmart-SIM applet v5.1. This applet is a Java Card application which enables the previously described features.

The TOE communicates with a terminal device (for example a PC with a card reader) by APDU messages compliant with the ISO/IEC 7816-4 standard. This terminal device is identified as “the host” in this document.

APDU messages may only be transmitted using the contact interface in the evaluated configuration.

This means that the contactless interface on the platform must be deactivated.



The TOE scope is represented Figure 5. [4] gives full details on TOE part 1 scope.

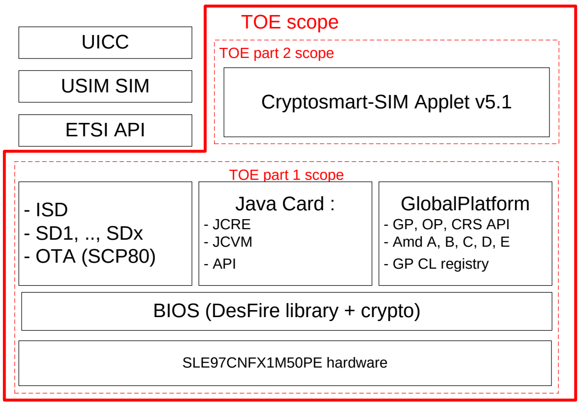


Figure 5: TOE scope

SD and CM are not part of the TOE as CM is deactivated before delivery to customers.

The TOE part 1 provides the following services used by TOE part 2:

- Security code management and user authentication through the PIN object;
- Cryptography services including symmetric encryption and decryption, RSA encryption and decryption, RSA signature and verification and generation of random data;
- Physical protection of stored data ensuring their confidentiality and integrity.

The TOE part 2 provides the services which will be used by the Cryptosmart user. These services were previously described § 2.2.

2.5.1 TOE PART 2 SCOPE OF EVALUATION

The scope of evaluation encompasses all the security features offered by TOE part 2 with the exception of the symmetric and RSA signature service.

The evaluated TOE also fixes TOE configuration choices and some constraints on TOE usage which are described in the TOE user guide [5]. The javacard configuration is described in the platform's security target [4]: "*The [platform] is configured so that an applet can be downloaded and installed on it, even after the smart card has been issued to the Cardholder. This allows MNOs as Card issuers to dynamically respond to customers' needs. For instance, if the Card issuer decides to upgrade some of the applications offered to the customer, this could be made without issuing a new card. Moreover, applications from different vendors can coexist in a single card, and they can even share information between each other*."

Moreover the case of a minimal security code length of 0 is out of the scope of this evaluation.

TOE configuration is performed either by IDEMIA or Ercom during the PRE-PERSONALIZATION state, where they initialize the internal random generator; insert the product serial number and create files which may be used by the customer. The evaluation scope includes any such configuration.

In terms of life cycle the evaluated configuration includes every state starting from PERSONALIZATION.

3. Security Problem Definition

3.1 Assets to protect

This section lists the assets which are security relevant for the TOE. Assets for the TOE are divided into three categories:

- Primary assets which are user data from IT environment which shall be protected using the security functions offered by the TOE. This intended protection is addressed by organizational security policies;
- Secondary assets which are needed by the TOE to offer its security services. We divide secondary assets into:
  - User assets: these are data created by or for the user.
  - TSF assets: these are data created by the TOE or for the TOE internal usage.

3.1.1 Primary assets

D.PLAINTEXT_DATA

These are user data which need protection in confidentiality. TOE user uses the TOE to protect them in confidentiality.

D.USER_STORED_KEYS

These are the RSA or symmetric keys stored by the user inside the TOE. Each of the key is stored with their attributes. The asset includes both the key and attributes.

D.USER_STORED_DATA

The TOE allows the user to store raw data. This asset represents these data, such as the user certificate.

3.1.2 User assets

D.SESSION_KEYS

These are the keys negotiated during the Cryptosmart authentication protocol. These keys are derived into keys which will be used by external application (the Cryptosmart suite for instance). These keys are present in plaintext inside the TOE (when managed by the Cryptosmart applet) or in wrapped form when outside the TOE.

D.USER_SIGNATURE_KEY

This is the RSA signature key used during the authentication protocol. This key is the private RSA key which public key is contained in the user certificate. It is not extractable.

D.USER_AUTH_CODE

These are the codes allowing getting the TOE in a VALIDATED_USER state.
Authentication codes are:

- the security code of the user: it allows to login successfully to the TOE (a TOE state change from UNVALIDATED_USER to VALIDATED_USER);
- the PUK codes: it allows to unblock the TOE (a TOE state change from BLOCKED_USER to VALIDATED_USER)
- the ephemeral security code: it allows a TOE state change from RESUMABLE to VALIDATED_USER
- the recycle code: it allows a TOE state change from any state to PERSONALIZATION

3.1.3 TSF data

D.TOE_INTERNAL_DATA

Internal data are the keys which impact only internal behavior of the TOE.
Some of them are inserted in the PERSONALIZATION state. These keys are:

- The keys derived from the family key
- The wrapping key (includes both encryption and MAC keys)
- The APDU initialization key which is an RSA encryption key used to initialize the APDU encryption mechanism. It allows the TOE to exchange APDU encryption keys (D.APDU_KEYS) with the host

Some others are generated and stored inside the TOE. These are:

- The RNG internal state.

D.TOE_INTERNAL_INFORMATION

Internal information is data used by the TOE with no need of confidentiality but which shall not be modified. This information is needed for a proper behavior of the TOE and are:

- The card state.
- The card flags giving information on already performed security initialization function.
- The commercial version number of the applet.
- The applet source code release identifier.
- The build version of the applet.
- The applet compilation date.
- The card serial number.
- The remaining number of times an incorrect Security Code that can be presented before the card is blocked.
- The number identifying the current PUK code.
- The number of times an incorrect PUK code can be presented before the card is blocked.

D.APDU_KEYS

These are the APDU encryption and APDU integrity keys.

D.APP_CODE

The code of the TOE, (TOE part 1 and TOE part 2)

3.1.4 Security needs

Asset	Confidentiality	Integrity	Availability
D.PLAINTEXT_DATA	Yes	No	No
D.USER_STORED_KEYS	Yes	Yes	No
D.USER_STORED_DATA	Yes	Yes	No
D.SESSION_KEYS	Yes	Yes	No
D.USER_SIGNATURE_KEY	Yes	Yes	No
D.USER_AUTH_CODE	Yes	Yes	No
D.TOE_INTERNAL_DATA	Yes	Yes	No
D.TOE_INTERNAL_INFORMATION	No	Yes	No
D.APDU_KEYS	Yes	Yes	No
D.APP_CODE	No	Yes	No

Table 3: Assets security needs

3.2 Assumptions

A.TRUSTED_ADMIN

The smartcard administrator is not careless, willfully negligent or hostile and does not voluntarily disclose data he inserted in the TOE during configuration.

A.TRAINED_ADMIN

The smartcard administrator is correctly trained to the TOE usage.

A.CONFIGURATION

The TOE is correctly configured while in PERSONALIZATION state. Configuration is considered correct if:

- The signature private key matches the public key contained in the user certificate.
- The administrator has correctly set the required security code length.

A.KEY_QUALITY

Keys generated outside the TOE are of appropriate quality. Appropriate quality may be reached by following the ANSSI rules for key generation from [9].

A.SECURE_KEY_MANAGEMENT

Some keys are generated outside the TOE and may be stored by the administrator (to recreate a card for the same user). An attacker may not have knowledge of these keys during their generation, transmission to the administrator or by accessing their backup.

PLATFORM RELEVANT ASSUMPTIONS

A.VERIFICATION

All the bytecodes are verified at least once, before the loading, in order to ensure that each bytecode is valid at execution time.
This verification applies for applet loading in post-issuance phase, in pre-issuance, this verification is ensured by ALC_DVS.2.
For application code loaded post-issuance and verified off-card, the verification authority provides digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification. Such evidence is generated according to [10].

A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" outside the API.

3.3 Threats

The threats listed below concern directly the TOE part 2. All the platform threats from [4] are also relevant to the composite security target.

The following threats are adapted from the javacard protection profile ([8]) and the platform security target [4]:

T.TOE_DATA_CORRUPTION

A local or distant attacker can modify the user keys, user data or TSF data stored or in use inside the TOE. This may for instance allow the attacker to replace user key by keys he knows or weaken targeted keys. A consequence of this threat is leading to attacks defeating OSP.SYMMETRIC_ENCRYPTION
Impacted assets: every identified asset except D.PLAINTEXT_DATA

T.TOE_DATA_COMPROMISE

A local or distant attacker can obtain the user keys, user data or TSF data stored or in use inside the TOE. A consequence of this threat is leading to attacks defeating OSP.SYMMETRIC_ENCRYPTION
Impacted assets: every identified asset except D.PLAINTEXT_DATA

T.ABUSE_FUNC

An attacker with high attack potential may use TOE functions intended for installation or configuration of the TOE which shall not be used for operational cryptographic keys or user data in order

- to disclose or manipulate user data or keys, or
- to enable attacks against the integrity or confidentiality of user data or keys by:
  - manipulating (explore, bypass, deactivate or change) security features or functions of the TOE or
  - disclosing or manipulating TSF Data.
- to enable attacks against the integrity of TOE part 2 code (D.APP_CODE)

Impacted assets: every identified asset

T.MASQUERADE

An attacker with high attack potential may masquerade as an authorized data source or receiver to perform operations that will be attributed to the authorized user or may gain undetected access to cryptographic module causing potential violations of integrity or confidentiality of the user data, the user keys or the TSF data.
For instance the attacker can have access or modify the user security code or PUK codes, in order to get illegitimate access to the TOE.
Impacted asset: D.USER_AUTH_CODE D.USER_STORED_KEYS D.USER_STORED_DATA D.PLAINTEXT_DATA D.ORIGINAL_DATA D.TOE_INTERNAL_DATA

This security target also considers the following threats:

T.KEY_DERIVE

A local or distant attacker with high attack potential is able to compute private key from publicly available data (for instance compute an RSA private key from the corresponding public key, or perform a brute force attack on a symmetric key from known plaintext).
Impacted assets: D.USER_SIGNATURE_KEY D.USER_STORED_KEYS

T.INTERFACE_EAVESDROP

A local attacker with high attack potential can act on the TOE interfaces in order to

- get knowledge of user data or user keys, or
- modify user data or user keys when transmitted to the TOE.

Impacted assets: D.USER_AUTH_CODE D.USER_STORED_KEYS D.PLAINTEXT_DATA D.ORIGINAL_DATA D.USER_STORED_DATA

3.3.1 Relevant Threats from the platform

T.LIFE_CYCLE

An attacker accesses to an application outside of its expected availability range thus violating irreversible life cycle phases of the application (for instance, an attacker repersonalizes the application)

3.4 OSP

This part describes the organizational security policies the TOE shall conform to. We divide these OSP into 3 categories the directives and rules the TOE shall follow; the security policies that TOE usage allows to cover; and OSP which allow reaching the aimed security level.

3.4.1 Directives and rules

OSP.RGS_CRYPTO

Cryptographic mechanisms conform to rules and recommendation from [11].
Keys are generated with mechanisms conformant to rules and recommendation from [9].

3.4.2 IT security policy

OSP.MUTUAL_AUTHENTICATION

The TOE shall allow to authenticate remote users and to negotiate a shared secret key. The TOE shall forbid an attacker with high attack potential to cause the TOE to run successfully the Cryptosmart authentication protocol by forging or replaying authentication packets. (The attacker's goal is to be falsely authenticated by the TOE which could lead to the possibility of man in the middle attacks)

OSP.KEY_STORAGE

The TOE shall allow storage of cryptographic keys in such a way they are protected from modification or disclosure.

OSP.SYMMETRIC_ENCRYPTION

The TOE shall allow protecting the confidentiality of information represented by user data which may get known by an attacker using a symmetric encryption algorithm.

OSP.RSA_PRIVATE_KEY_OPERATION

The TOE shall allow performing the following operation using an RSA private key:

- RSA signature with PKCS#1.5 padding
- RSA decryption with PKCS#1.5-OAEP padding

3.4.3 Security ensuring OSP

OSP.LOCAL_AUTHENTICATION

Users must be authenticated prior to accessing any controlled TOE resources with the exception of read access to public objects. After a pre-defined authentication failures number, access to controlled resources shall be blocked.
Authentication may be performed by:

- User entered security code verification;
- PUK code verification (only if the TOE is in BLOCKED state)
- Automatic authentication by the driver using an ephemeral security code which must have been generated by the TOE;
- Implicit authentication by proof of knowledge of secrets for the secure communication channel with the TOE. This last case allows only a limited usage of the TOE.

OSP.PUK_UNBLOCK

User can unblock access to TOE controlled resources by using a PUK code. Any PUK code may only be used once. After a pre-defined PUK entry failures sensitive data shall be wiped.

OSP.DATA_WIPE

The TOE provides a way to wipe controlled sensitive assets.

OSP.ACCESS_CONTROL

The TOE must limit the extent of each user's abilities to use the TOE functions in accordance with the current TOE authentication state.

OSP.TOE_AUDIT

The TOE provides an interface to the hosted application, which allows peeking in its internal state. Moreover the TOE informs the host of occurred security events in order for the host to record them in a security journal. Reported security events shall include:

- User authentication failure
- Mutual authentication and session key establishment faults
- Cryptographic errors

4. SECURITY OBJECTIVES

4.1 SECURITY OBJECTIVES FOR THE TOE

This section defines the security objectives for the composite TOE. They are satisfied either by technical countermeasure implemented by the Cryptosmart applet, by the platform or by a combination of the two.

O.USER_AUTHENTICATION

The TOE shall authenticate the user before providing access to any controlled resources with the exception of read access to card state or APDU encryption initialization.
Authentication shall be performed by:

- User entered security code verification;
- PUK code verification (only if the TOE is in BLOCKED state)
- Automatic authentication by the driver using an ephemeral security code which must have been generated by the TOE;
- Implicit authentication by proof of knowledge of secrets for the secure communication channel with the TOE. This last case allows only a limited usage of the TOE.

This objective does not apply to TOE administrator which may only use the TOE in PERSONALIZATION state.

O.PUK_UNBLOCK

When in BLOCKED state the TOE shall allow the user to unblock it by usage of a PUK code. PUK codes must have been generated by the TOE.
Each PUK code may only be successfully used once.

O.STRONG_SECCODE

The TOE shall restrict the choice of the user security code according to administrator defined minimum length.
The TOE shall restrict PUK codes and ephemeral security codes to TOE generated secrets.

O.LIMITED_AUTH_NUMBER

The TOE shall go in BLOCKED state after 3 successive failed security code entries.
The TOE shall go in WIPED state after 10 failed PUK code entries.
The TOE shall manage only 15 PUK codes. When the card is blocked and no PUK codes are available the TOE shall go in WIPED state.

O.FUNCTION_ACCESS_CONTROL

The TOE shall restrict the access to its services, depending on the TOE state and the services explicitly assigned to this state. Assignment of services to state shall be done by default and may not be changed.

O.CRYPTOGRAPHIC_OPERATION

The TOE shall ensure the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have sufficient entropy.
The TOE shall perform valid AES encryption and decryption, MAC signature, RSA signature and decryption.

O.STRONG_MUTUAL_AUTHENTICATION

The TOE shall provide cryptographic functions to authenticate remote users and to negotiate a shared secret key.
The TOE shall make impossible for an online attacker to get knowledge of the session key negotiated with the remote card whose identifier is furnished by the TOE, even if he later acquires knowledge of the private RSA keys of both cards and if he has intercepted all the flows between cards of the same family before and after the session.
An online attacker must be unable to cause the TOE to run successfully the authentication protocol without a valid certificate (even without getting knowledge of the negotiated key).

O.KEY_MANAGEMENT

The TOE shall provide a means to securely manage and store user keys (D.USER_STORED_KEYS and D.USER_SIGNATURE_KEY). This concerns the correct generation, access and destruction of cryptographic keys. This includes:

- Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes;
- The TOE shall be able to import and store user keys;
- User keys shall be associated with security attributes. Assignment of security attribute to a key shall be done by explicit action of the TOE administrator or TOE user or by default.
- User may only change security attributes to more restrictive values
- The TOE shall restrict the access to the user keys according to their security attributes and the TOE state.
- Keys shall be destroyed in accordance with specified cryptographic key destruction methods.

O.PROTECT_SESSION_KEY

The TOE shall make impossible for an attacker to recover session keys (D.SESSION_KEYS) when stored by the host.
The TOE shall make impossible for an attacker to modify session keys when stored by the host without detection.

O.APDU_ENCRYPTION

The TOE shall provide an encryption mechanism to protect the confidentiality and integrity of commands to and responses from the TOE.
The TOE shall encrypt every response when in VALIDATED_USER, RESUMABLE or RESUMED_USER state if the command was sent encrypted with the exception of mutual authentication packets.
In VALIDATED_USER, RESUMABLE or RESUMED_USER state every command sent in plaintext shall be refused; with the exception of read access to card state or APDU encryption initialization.

O.SENSITIVE_MEMORY_ERASING

The TOE shall erase by zeroization sensitive assets that are not currently in use:

- session key after usage;
- security code when card is blocked;
- ephemeral security code on ephemeral security code entry error;
- PUK code after usage

O.WIPE

The TOE shall be able to wipe every personalization data (except the recycle key) and user keys. Wipe shall be initiated:

- On user request, or
- When the PUK codes tries limits is reached, or
- When the TOE has no available PUK code and the security code tries limit has been reached.

4.2 SECURITY OBJECTIVES FOR THE ENVIRONMENT

OE.ADMIN

The card administrator shall not be careless, willfully negligent, or hostile, and shall follow the instructions provided by the administrator documentation.

OE.CARD_ADMIN_STATION

The card admin station must check the consistency of data injected inside the TOE in PERSONALIZATION state.

OE.KEY_GENERATOR

The keys injected inside the TOE must have been generated by a key generator which mechanisms conform to rules and recommendation from [9].

OE.SECURE_PERSONALIZATION

The TOE personalization (injection of user keys, user certificate, recycle key and family key) must be performed in an environment not accessible physically or remotely by an attacker.

OE.SECURE_KEY_MANAGEMENT

Externally generated TOE sensitive data (such as user keys, user certificate, family key, recycle key, etc.) must be transmitted to the card administrator in a secure way, and stored in an environment not accessible physically or remotely by an attacker.

OE.SECURE_SECCODE_ENTRY

The user enters or changes his security code only in a safe environment, with no one able to see the security code. In case of compromising suspicion, the user changes its code immediately.

OE.NON_TRIVIAL_SECCODE

The user security code shall be non-guessable (i.e. different from “0000”, “1234” or analog values). The TOE administrator shall have set a strictly positive minimal length for the user security code.

OE.HOST_CORRECT_BEHAVIOR

The host behaves as expected. In particular:

- while running the Cryptosmart authentication protocols it correctly verifies that the distant certificate is valid and bound to the expected distant user’s identity;
- while running the stateless Cryptosmart authentication protocol it correctly manages the exported TOE state
- it does not leak the APDU encryption key. Moreover the host shall detect and inform the user when the APDU encryption initialization public key has changed.

OE.HOST_AUDIT

The host records in a security journal occurred security events such as:

- User authentication failure
- Mutual authentication and session key establishment faults
- Cryptographic errors

This journal is generated from error codes returned by the TOE.

PLATFORM RELEVANT OE

OE.VERIFICATION

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

Moreover for application code loaded post-issuance and verified off-card, the verification authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification. Such evidence shall be generated according to [10].

OE.APPLET

No applet loaded post-issuance shall contain native methods

5. SECURITY REQUIREMENTS

This section defines the security requirements satisfied by the TOE.

The SFR are grouped under the security objective they cover. First are presented the SFR directly enforcing the security objective, then the dependencies of these SFR. SFR which are dependencies for several other SFRs are grouped together § 5.1.13.

5.1 SECURITY FUNCTIONAL REQUIREMENTS

5.1.1 O.USER_AUTHENTICATION ENFORCING SFR

USER ATTRIBUTE DEFINITION (FIA_ATD)

FIA_ATD.1 User attribute definition	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FIA_ATD.1.1	The TSF shall maintain the following list of security attributes belonging to individual users:[• *security code;* • *security code failed entries counter;* • *user security code block flag;* • *PUK code number;* • *PUK code failed entries counter;* • *authenticated state (not authenticated, authenticated or light authenticated) ;* • *ephemeral security code;*].

USER AUTHENTICATION (FIA_UAU)

FIA_UAU.1 Timing of authentication	
Hierarchical to:	No other components.
Dependencies:	• FIA_UID.1 Timing of identification
FIA_UAU.1.1	The TSF shall allow [• *Get card information;* • *Enter the security code;* • *Card recycling;* • *Card wiping;* • *Initializing the APDU encryption key;* • *Renew the APDU encryption IV;* • *Usage of the TOE if it is in the PRE-PERSONALIZATION state* • *Usage of the TOE if it is in the PERSONALIZATION state*] 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.

FIA_UAU.5 Multiple authentication mechanisms	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FIA_UAU.5.1	The TSF shall provide [• *Authentication through security code presentation* • *Authentication through ephemeral security code presentation* • *Authentication through PUK code presentation* • *Light authentication by successful usage of the trusted channel with the resume command*] to support user authentication
FIA_UAU.5.2	The TSF shall authenticate any user's claimed identity according to the [*current TOE state and expected TOE state after authentication :* • *From UNVALIDATED_USER, RESUMABLE or RESUMED_USER to VALIDATED_USER state by using "Authentication through security code presentation"* • *From RESUMABLE to VALIDATED_USER state by using "Authentication through ephemeral security code presentation"* • *From BLOCKED to VALIDATED_USER state by using "Authentication through PUK code presentation"* • *From RESUMABLE to RESUMED_USER state by using "Light authentication by successful usage of the trusted channel with the resume command"*].

5.1.2 O.PUK_UNBLOCK ENFORCING SFR

This objective is handled by the "FIA_UAU.5 Multiple authentication mechanisms": the TOE is unblocked by authentication by PUK codes.

This last SFR ensures PUK codes may only be used once:

USER AUTHENTICATION (FIA_UAU)

FIA_UAU.4 Single-use authentication mechanisms	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FIA_UAU.4.1	The TSF shall prevent reuse of authentication data related to [*authentication through PUK code presentation*].

5.1.3 O.STRONG_SECCODE ENFORCING SFR

SPECIFICATION OF SECRETS (FIA_SOS)

FIA_SOS.1 Verification of secrets	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FIA_SOS.1.1	The TSF shall provide a mechanism to verify that secrets meet [*minimum security code length is greater than 4 or the value set by the administrator*].

FIA_SOS.2 TSF Generation of secrets	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FIA_SOS.2.1	The TSF shall provide a mechanism to generate secrets that meet [*indistinguishability from an uniform distribution*].
FIA_SOS.2.2	The TSF shall be able to enforce the use of TSF generated secrets for [• *Authentication through ephemeral security code presentation* • *Authentication through PUK code presentation*].

Application note:

- ephemeral security code length is fixed to 8 bytes.
- PUK code length is fixed to 8 digits

5.1.4 O.LIMITED_AUTH_NUMBER ENFORCING SFR

AUTHENTICATION FAILURES (FIA_AFL)

FIA_AFL.1 Authentication failure handling	
Hierarchical to:	No other components.
Dependencies:	• FIA_UAU.1 Timing of authentication
FIA_AFL.1.1/userSC	The TSF shall detect when [*3*] unsuccessful authentication attempts occur related to [*user authentication by security code*].
FIA_AFL.1.2/userSC	When the defined number of unsuccessful authentication attempts has been [*met*], the TSF shall [• *turn the TOE to the BLOCKED state if a PUK code is available* • *turn the TOE to the WIPED state if no PUK code are available*].
FIA_AFL.1.1/ephSC	The TSF shall detect when [*1*] unsuccessful authentication attempts occur related to [*user authentication by ephemeral security code*].
FIA_AFL.1.2/ephSC	When the defined number of unsuccessful authentication attempts has been [*met*], the TSF shall [*wipe the stored ephemeral security code*].
FIA_AFL.1.1/PUK	The TSF shall detect when [*10*] unsuccessful authentication attempts occur related to [*user authentication by PUK code*].
FIA_AFL.1.2/PUK	When the defined number of unsuccessful authentication attempts has been [*met*], the TSF shall [*turn the TOE to the WIPED state*].

5.1.5 O.FUNCTION_ACCESS_CONTROL ENFORCING SFR

The TOE checks access rights for every command based on its current state. This part aims at defining the SFP which has this effect.

ACCESS CONTROL POLICY (FDP_ACC)

FDP_ACC.2/access Complete access control	
Hierarchical to:	• FDP_ACC.1 Subset access control
Dependencies:	• FDP_ACF.1 Security attribute based access control
FDP_ACC.2.1/access	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFP*] on [***subject****: local user, administrator;* ***objects****: TOE part 2]* and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/access	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

The subject (the local user or administrator) operates (sends a command) an object (the Cryptosmart applet, i.e. TOE part 2).

There is only one possible operation between TOE part 2 and the local user or administrator which is sending a command as an APDU.

Each Cryptosmart command possesses an attribute per TOE state which is a Boolean representing its authorization for this state.

This SFP aims at verifying whether the operation is authorized or not based on the command function identifier and the TOE state.

ACCESS CONTROL FUNCTIONS (FDP_ACF)

FDP_ACF.1/access Security attribute based access control	
Hierarchical to:	No other components.
Dependencies:	• FDP_ACC.1 Subset access control • FMT_MSA.3 Static attribute initialization
FDP_ACF.1.1/access	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFP*] to objects based on the following: [the *TOE state and related operation authorization*].
FDP_ACF.1.2/access	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [*the authorization attribute for the command is set to true for the current TOE state, as stated in* ***Table 4*** and Table 5].
FDP_ACF.1.3/access	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [*none*].
FDP_ACF.1.4/access	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [*none*].

Function identifier	Function	PP	P	UU	VU	R	RU	B	W
0x00	Security code verification			X	X	X	X		
0x01	Authentication start				X		X		
0x02	Incoming authentication packet processing				X		X		
0x03	Recall	X	X	X	X	X	X	X	X
0x04	Certificate decoding result transmission				X		X		
0x05	Authentication reset				X		X		
0x06	Key derivation				X		X		
0x07	Continue	X	X	X	X	X	X	X	X
0x08	Generate Kapdu			X				X	X
0x09	Renew APDU encryption IV			X		X		X	
0x0A	Resume					X			
0x0B	Log off			X	X	X	X	X	X
0x0C	Wipe card			X	X	X	X	X	X
0x0D	Enumerate RSA keys		X		X		X		
0x0E	Import external RSA key		X		X				
0x0F	Generate RSA key		X		X				
0x10	Private RSA key export		X		X				
0x11	Delete RSA key		X		X				
0x12	RSA key attributes change		X		X				
0x13	RSA public key export		X		X		X		
0x14	Buffer RSA decryption		X		X		X		
0x15	Buffer RSA signing without hash		X		X		X		
0x16	Renew DH value				X				
0x17	Get status	X	X	X	X	X	X	X	X
0x18	Enumerate symmetric keys		X		X		X		
0x19	Generate symmetric key		X		X				
0x1A	Import symmetric key		X		X				
0x1B	Delete symmetric key		X		X				
0x1C	Symmetric encryption (AES)		X		X		X		
0x1D	Symmetric signature (HMAC-SHA256)		X		X		X		

Function identifier	Function	PP	P	UU	VU	R	RU	B	W
0x1E	Get local encryption key				X		X		
0x1F	Symmetric key export		X		X				
0x20	Generate Kapdu			X				X	
0x21	Symmetric key attribute change		X		X				
0x22	Renew ephemeral security code				X				
0x24	Stateless authentication start				X		X		
0x25	Stateless authentication get certificate keys				X		X		
0x26	Stateless authentication sign				X		X		
0x27	Stateless authentication verify				X		X		
0x28	Fuse CC flag			X	X	X	X		
0xE0	Create file	X							
0xE2	Write file	X	X	X	X	X	X	X	X
0xE3	Read file	X	X	X	X	X	X	X	X
0xEA	Security code change				X				
0xFA	Card recycle		X	X	X	X	X	X	X
0xFB	Create card		X						
Function identifier	Function	PP	P	UU	VU	R	RU	B	W

Table 4: Function access rights

This table uses the abbreviation:

- PP as PRE-PERSONALIZATION
- P as PERSONALIZATION
- UU as UNVALIDATED_USER
- VU as VALIDATED_USER
- R as RESUMABLE
- RU as RESUMED_USER
- B as BLOCKED
- W as WIPED

Command	Encryption required	PP	P	UV and R	VU and RU	B	W
GET RESPONSE		X	X	X	X	X	X
Get scp key id and version		X	X	X	X	X	X
Get card certificate		X	X	X	X	X	X

Command	Encryption required	PP	P	UV and R	VU and RU	B	W
INTERNAL AUTHENTICATE				X	X	X	X
Generate card's private static key		X	X				
Get card's public static key		X	X	X	X	X	X
Set card's certificate		X	X				
Get LockApplet Status		X	X	X	X	X	X
Get LockApplet Info		X	X	X	X	X	X
Present PIN	X			X	X		
Change PIN	X			X	X		
Present PUK	X			X	X		
Lock		X	X	X	X	X	X
Generate Wrapped DEK	X			X	X		
Unwrap DEK	X				X		
Generate DEK	X				X		
Get DEK	X				X		
Generate Password	X				X		
Get Password	X				X		

Table 5: Lock applet functions access control

MANAGEMENT OF SECURITY ATTRIBUTES (FMT_MSA)

The security attributes for this policy are fixed and may not be changed:

FMT_MSA.3/access Static attribute initialization	
Hierarchical to:	No other components.
Dependencies:	• FMT_MSA.1 Management of security attributes • FMT_SMR.1 Security roles
FMT_MSA.3.1/access	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFP*] to provide [*fixed*] default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2/access	The TSF shall allow the [*local user, administrator*] to specify alternative initial values to override the default values when an object or information is created.

The “fixed” property for file access authorization means these default values were fixed at compilation time by the applet developer, and may not be changed.

FMT_MSA.1/access Management of security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] • FMT_SMR.1 Security roles • FMT_SMF.1 Specification of Management Functions
FMT_MSA.1.1/access	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFP*] to restrict the ability to [*modify*] the security attributes [*state authorization*] to [*none*].

5.1.6 O.CRYPTOGRAPHIC_OPERATION ENFORCING SFR

CRYPTOGRAPHIC OPERATION (FCS_COP)

FCS_COP.1 Cryptographic operation	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] • FCS_CKM.4 Cryptographic key destruction
FCS_COP.1/AES **FCS_COP.1.1**	The TSF shall perform [*Encryption and decryption*] in accordance with a specified cryptographic algorithm [*AES*] and cryptographic key sizes [*256 and 128 bits*] that meet the following: [[12]]
FCS_COP.1/SHA256 **FCS_COP.1.1**	The TSF shall perform [*Message Digest computation*] in accordance with a specified cryptographic algorithm [*SHA256*] and cryptographic key sizes [*none*] that meet the following: [[13]]
FCS_COP.1/HMAC **FCS_COP.1.1**	The TSF shall perform [*MAC generation and verification*] in accordance with a specified cryptographic algorithm [*HMAC-SHA256*] and cryptographic key sizes [*256 and 128 bits*] that meet the following: [[14]]
FCS_COP.1/EMAC **FCS_COP.1.1**	The TSF shall perform [*MAC generation and verification*] in accordance with a specified cryptographic algorithm [*Retailed AES CBC-MAC*] and cryptographic key sizes [*256 bits*] that meet the following: [*none*]
FCS_COP.1/CMAC **FCS_COP.1.1**	The TSF shall perform [*MAC generation and verification*] in accordance with a specified cryptographic algorithm [*AES CMAC*] and cryptographic key sizes [*256 bits*] that meet the following: [[15]]
FCS_COP.1/RSA_sig **FCS_COP.1.1**	The TSF shall perform [*signature*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*2048 bits*] that meet the following: [[16]]
FCS_COP.1/RSA_verif **FCS_COP.1.1**	The TSF shall perform [*signature verification*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*2048 bits*] that meet the following: [[16]]
FCS_COP.1/RSA_enc **FCS_COP.1.1**	The TSF shall perform [*decryption*] in accordance with a specified cryptographic algorithm [*RSA*] and cryptographic key sizes [*2048 bits*] that meet the following: [[16]]
FCS_COP.1/random **FCS_COP.1.1**	The TSF shall perform [*random number generation*] in accordance with a specified cryptographic algorithm [*True random generator post-processed with AES*] and cryptographic key sizes [*128 bits*] that meet the following: [*none*]
FCS_COP.1/DH **FCS_COP.1.1**	The TSF shall perform [*key agreement*] in accordance with a specified cryptographic algorithm [*Diffie-Hellman*] and cryptographic key sizes [*2048 bits*] that meet the following: [[17]]

The following SFR applies to the public key transmitted by the host during stateless authentication protocol signature verification:

FDP_ITC.1/pubkey Import of user data without security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] • FMT_MSA.3 Static attribute initialization

FDP_ITC.1.1/pubkey	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFPs*] when importing user data, controlled under the SFP, from outside of the TOE.
FDP_ITC.1.2/pubkey	The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.
FDP_ITC.1.3/pubkey	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [*none*].

The dependency FMT_MSA.3 will not be fulfilled as the TOE does not define security attribute for signature verification keys: its usage is fully controlled by SFP_OPERATION_ACCESS_CONTROL which provides access control on the functions involving it.

5.1.7 O.STRONG_MUTUAL_AUTHENTICATION ENFORCING SFR

The mutual authentication is performed using the Cryptosmart authentication protocol. This protocol is based on the following SFR:

- FCS_COP.1/random
- FCS_COP.1/DH
- FCS_COP.1/RSA_sig
- FCS_COP.1/HMAC
- FCS_COP.1/SHA256
- FCS_COP.1/AES
- FCS_COP.1/EMAC

5.1.8 O.KEY_MANAGEMENT ENFORCING SFR

5.1.8.1 KEY GENERATION AND IMPORT

CRYPTOGRAPHIC KEY MANAGEMENT (FCS_CKM)

FCS_CKM.1 Cryptographic key generation	
Hierarchical to:	No other components.
Dependencies:	• [FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation] • FCS_CKM.4 Cryptographic key destruction
FCS_CKM.1/RSA FCS_CKM.1.1	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*Javacard API RSA key pair generation*] and specified cryptographic key sizes [*2048 bits*] that meet the following: [*none*].
FCS_CKM.1/random FCS_CKM.1.1	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*random generation*] and specified cryptographic key sizes [*128 bits and 256 bits*] that meet the following: [*none*].
FCS_CKM.1/local_prot FCS_CKM.1.1	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*hash with constant*] and specified cryptographic key sizes [*256 bits*] that meet the following: [*none*].

IMPORT FROM OUTSIDE OF THE TOE (FDP_ITC)

FDP_ITC.2/keys Import of user data with security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] • [FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path] • FPT_TDC.1 Inter-TSF basic TSF data consistency
FDP_ITC.2.1/keys	The TSF shall enforce the [*SFP_KEY_ACCESS_CONTROL access control SFP*] when importing user data, controlled under the SFP, from outside of the TOE.
FDP_ITC.2.2/keys	The TSF shall use the security attributes associated with the imported user data.
FDP_ITC.2.3/keys	The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.
FDP_ITC.2.4/keys	The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.
FDP_ITC.2.5/keys	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [*none*].

Application note: When the administrator imports user keys in the TOE, there is no usage of any trusted path as the security of the communication link is ensured by the environment.

5.1.8.2 KEY DESTRUCTION

CRYPTOGRAPHIC KEY MANAGEMENT (FCS_CKM)

FCS_CKM.4 Cryptographic key destruction	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]
FCS_CKM.4/zero FCS_CKM.4.1	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [*zeroization*] that meets the following: [*none*].
FCS_CKM.4/java FCS_CKM.4.1	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [*The keys are reset in accordance with* [18] *in class Key with the method clearKey()*] that meets the following: [18].

Application note: The SFR FCS_CKM.4/java only applies to keys stored in javacard Key objects. The SFR FCS_CKM.4/zero applies to keys stored as arrays.

5.1.8.3 SECURITY ATTRIBUTES AND ACCESS CONTROL

This section defines a SFP on Ob.USER_STORED_KEYS, Ob.USER_SIGNATURE_KEY and Ob.APDU_INITIALIZATION_KEY objects.

These objects are defined as:

- Ob.USER_STORED_KEYS corresponds to the D.USER_STORED_KEYS asset
- Ob.USER_SIGNATURE_KEY corresponds to the D.USER_SIGNATURE_KEY asset
- Ob.APDU_INITIALIZATION_KEY corresponds to the APDU initialization key which is part of the D.TOE_INTERNAL_DATA asset

These objects have the following attributes considered in this SFP:

- Key identifier
- Key type (symmetric or RSA)
- Useable in resume flag
- The “encryption” key usage flag
- The “sign” key usage flag
- Extractable flag
- The “disable security checks” flag
- A usage counter

Note: for asymmetric keys the key usage, “disable security checks” flags and usage counter are ignored (cf. FDP_ACF.1/keys) and are therefore not implemented.

The possible operations on these objects are:

- Generation;
- Deletion;
- Import inside the TOE
- Export from the TOE
- Encrypt or decrypt
- Sign

ACCESS CONTROL POLICY (FDP_ACC)

FDP_ACC.2/keys Complete access control	
Hierarchical to:	• FDP_ACC.1 Subset access control
Dependencies:	• FDP_ACF.1 Security attribute based access control
FDP_ACC.2.1/keys	The TSF shall enforce the [*SFP_KEY_ACCESS_CONTROL access control SFP*] on [***subject****: local user, administrator;* ***objects****: Ob.USER_STORED_KEYS, Ob.USER_SIGNATURE_KEY and Ob.APDU_INITIALIZATION_KEY*] and all operations among subjects and objects covered by the SFP.
FDP_ACC.2.2/keys	The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

ACCESS CONTROL FUNCTIONS (FDP_ACF)

FDP_ACF.1/keys Security attribute based access control	
Hierarchical to:	No other components.
Dependencies:	• FDP_ACC.1 Subset access control • FMT_MSA.3 Static attribute initialization
FDP_ACF.1.1/keys	The TSF shall enforce the [*SFP_KEY_ACCESS_CONTROL access control SFP*] to objects based on the following: [*TOE state and object attribute*].
FDP_ACF.1.2/keys	The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [• *Generation or import operation are allowed only if no object with same identifier and key type already exists;* • *Export operation is always denied if the extractable attribute is set to false;* • *Any operation is always denied if the TOE is in RESUMED_USER state and the "usable in resumed" attribute is set to false;* • *Local protection key obtention is authorized only if the key type is symmetric and the key identifier attribute is 0 or 1* • *The encrypt or decrypt operation is authorized only if* ○ *The key type is asymmetric and the key identifier attribute is different from 0 or 1, or* ○ *The key type is symmetric and the "encryption" key usage attribute is set to true* • *The sign operation is authorized only if* ○ *The key type is asymmetric and the key identifier attribute is different from 0 or 1, or* ○ *The key type is symmetric and the "sign" key usage attribute is set to true* *]*
FDP_ACF.1.3/keys	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [*none*].
FDP_ACF.1.4/keys	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [• *The encrypt or decrypt operation is refused if the "disable security checks" key attribute is set to false and the usage counter as reached 10.000*

	].

Export from the TOE (FDP_ETC)

FDP_ETC.1/keys Export of user data without security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]
FDP_ETC.1.1/keys	The TSF shall enforce the [*SFP_KEY_ACCESS_CONTROL access control SFP*] when exporting user data, controlled under the SFP(s), outside of the TOE.
FDP_ETC.1.2/keys	The TSF shall export the user data without the user data's associated security attributes

Management of security attributes (FMT_MSA)

FMT_MSA.1/keys Management of security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] • FMT_SMR.1 Security roles • FMT_SMF.1 Specification of Management Functions
FMT_MSA.1.1/keys	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFP*] to restrict the ability to [*modify*] the security attributes [*Useable in resume, Extractable, "encryption" and "sign" key usage*] to [*local user, administrator*].

FMT_MSA.2/keys Secure security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] • FMT_MSA.1 Management of security attributes • FMT_SMR.1 Security roles
FMT_MSA.2/keys	The TSF shall ensure that only secure values are accepted for [*Extractable, usable in resumed, "encryption" and "sign" key usage attributes*].

Application note: This SFR applies:

- On key creation:
    - For keys with identifier 0 or 1 only fixed values are accepted, i.e. the "extractable", "encryption" and "sign" key usage attributes must be set to false, and the "usable in resumed" must be set to false for key with identifier 0 and to true for key with identifier 1;
- On attribute change: a user shall not be able to change security attributes to less restrictive ones. He cannot change any flags from false to true. It shall be impossible to allow an object to be exported when export was previously forbidden, or to allow performing previously forbidden operation.

Inter-TSF TSF data consistency (FPT_TDC)

FPT_TDC.1 Inter-TSF basic TSF data consistency	
Hierarchical to:	No other components.
Dependencies:	No dependencies
FPT_TDC.1.1	The TSF shall provide the capability to consistently interpret [• *User key attributes*] when shared between the TSF and another trusted IT product.
FPT_TDC.1.2	The TSF shall use [• *The user key attribute is defined as two bytes: the first represent the key identifier, the second is formed as follows:* ○ *Bit number starts at 0* ○ *Bit 1 is 1 if the extractable attribute is true, 0 otherwise* ○ *Bit 3 is 1 if the useable in resumed attribute is true, 0 otherwise* ○ *Bit 4 is 1 if the encryption attribute is true, 0 otherwise* ○ *Bit 5 is 1 if the signature attribute is true, 0 otherwise* ○ *Bit 6 is 1 if security checks are disabled, 0 otherwise*] when interpreting the TSF data from another trusted IT product.

5.1.9 O.PROTECT_SESSION_KEY ENFORCING SFR

Session key are outputted symmetrically wrapped. This is based on already defined SFR FCS_COP.1.1/AES and FCS_COP.1.1/EMAC.

Wrapping keys may be either generated internally (FCS_CKM.1.1/random) or imported inside the TOE.

The FDP_ITC.2/keys dependency of FCS_COP.1 can't apply as SFP_KEY_ACCESS_CONTROL do not apply to wrapping keys. The following SFR allows fulfilling the dependency for wrapping key import:

IMPORT FROM OUTSIDE OF THE TOE (FDP_ITC)

FDP_ITC.1/systemkey Import of user data without security attributes	
Hierarchical to:	No other components.
Dependencies:	• [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] • FMT_MSA.3 Static attribute initialization
FDP_ITC.1.1/systemkey	The TSF shall enforce the [*SFP_OPERATION_ACCESS_CONTROL access control SFPs*] when importing user data, controlled under the SFP, from outside of the TOE.
FDP_ITC.1.2/systemkey	The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.
FDP_ITC.1.3/systemkey	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: [*none*].

The dependency FMT_MSA.3 will not be fulfilled as the TOE does not define security attribute for wrapping keys: their usage is fully controlled by SFP_OPERATION_ACCESS_CONTROL which provides access control on the functions involving the wrapping key.

5.1.10 O.APDU_ENCRYPTION ENFORCING SFR

TRUSTED PATH (FTP_TRP)

FTP_TRP.1/Cryptosmart Trusted path	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FTP_TRP.1.1/Cryptosmart	The TSF shall provide a communication path between itself and [*local*] users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [*modification, disclosure and replay*].
FTP_TRP.1.2/Cryptosmart	The TSF shall permit [*local users*] to initiate communication via the trusted path.
FTP_TRP.1.3/Cryptosmart	The TSF shall require the use of the trusted path for [• *user authentication by security code entry;* • *card resume command;* • *Renew APDU encryption key* • *PUK code unlock;* • *any command and TOE response in VALIDATED_USER, RESUMABLE or RESUMED_USER state, except* ○ *get status command* ○ *card recycle* ○ *Authentication packets transmission (but their responses for the host will be encrypted)* ○ *Log off* ○ *Wipe order* ○ *Continue or recall commands depending on initial command encryption* ○ *APDU encryption IV renewal*].

FTP_TRP.1/Scp11 Trusted path	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FTP_TRP.1.1/Scp11	The TSF shall provide a communication path between itself and [*local*] users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [*modification, disclosure and replay*].
FTP_TRP.1.2/Scp11	The TSF shall permit [*local users*] to initiate communication via the trusted path.
FTP_TRP.1.3/Scp11	The TSF shall require the use of the trusted path for [• *ISO7816 VERIFY command;* • *ISO7816 CHANGE REFERENCE DATA command;* • *ISO7816 RESET RETRY COUNTER command* • *ISO7816 PUT DATA on tag "unwrap DEK" (0x7F)* • *ISO7816 GET DATA on tags:* • *TAG_GENERATE_DEK (0xCE)* • *TAG_GENERATE_WRAPPED_DEK (0xCF)*

	• *TAG_GET_DEK (0xCC)* • *TAG_GET_PASSWORD (0xCD)*].

5.1.11 O.SENSITIVE_MEMORY_ERASING ENFORCING SFR

RESIDUAL INFORMATION PROTECTION (FDP_RIP)

FDP_RIP.1 Subset residual information protection	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FDP_RIP.1.1/applet	The TSF shall ensure that any previous information content of a resource is made unavailable upon the [*deallocation of the resource from*] the following objects: [• *session key* • *security code* • *ephemeral code* • *PUK code*].
FDP_RIP.1.1/APDU	The TSF shall ensure that any previous information content of a resource is made unavailable upon the [*allocation of the resource to*] the following objects: [*the APDU Buffer*].
FDP_RIP.1.1/TRANSIENT	The TSF shall ensure that any previous information content of a resource is made unavailable upon the [*deallocation of the resource from*] the following objects: [*any transient object*][2].

5.1.12 O.WIPE ENFORCING SFR

The wipe action is managed by already defined FCS_CKM.4.1/zero, FCS_CKM.4.1/java and FDP_RIP.1:

- User keys are erased using FCS_CKM.4.1/java;
- Personalization data are respectively erased by:
  - Security codes are erased using FDP_RIP.1;
  - Keys derived from the family key by FCS_CKM.4.1/java
  - Wrapping keys by FCS_CKM.4.1/java;
  - Local protection master keys by FCS_CKM.4.1/java

Wipe can be initiated:

- On user request.
- When the PUK codes tries limits is reached, which is handled by FIA_AFL.1/PUK
- When the TOE has no available PUK code and the security code tries limit has been reached which is handled by FIA_AFL.1/userSC

[2] Transient objects are object which content is stored in RAM as opposed to object stored in persistent memory (EEPROM)

5.1.13 Generic dependencies SFR

This section groups SFR which are dependencies for different other SFRs.

User identification (FIA_UID)

Application note: as the TOE is single user, user authentication provides identification. Therefore the FIA_UID dependency does not need to be fulfilled.

Security management roles (FMT_SMR)

FMT_SMR.1 Security roles	
Hierarchical to:	No other components.
Dependencies:	• FIA_UID.1 Timing of identification
FMT_SMR.1.1	The TSF shall maintain the roles [*local user, administrator*].
FMT_SMR.1.2	The TSF shall be able to associate users with roles.

Specification of Management Functions (FMT_SMF)

FMT_SMF.1 Specification of Management Functions	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FMT_SMF.1.1	The TSF shall be capable of performing the following management functions: [• *key attribute change;* • *security code minimum length setting;* • *TOE personalization (import of cryptographic keys)*].

5.2 TOE SECURITY ASSURANCE REQUIREMENTS

The evaluation target must comply with parts 2 and 3 of the Common Criteria version 3.1 for the EAL4 level, augmented with ALC_DVS.2 and AVA_VAN.4. Table 6 summarizes the aimed assurance components.

Assurance class	Assurance component	Level	Description
Development	ADV_ARC	1	Security architecture.
	ADV_FSP	4	Functional specification
	ADV_IMP	1	Implementation representation
	ADV_INT	N.A.	TSF internals
	ADV_SPM	N.A.	Security policy modelling
	ADV_TDS	3	TOE design
Guidance documents	AGD_OPE	1	Operational user guidance
	AGD_PRE	1	Preparative procedure
Life-cycle support	ALC_CMC	4	Configuration Management (CM) Capabilities
	ALC_CMS	4	CM scope
	ALC_DEL	1	Delivery
	ALC_DVS	**2**	**Development security**
	ALC_FLR	N.A.	Flaw remediation
	ALC_LCD	1	Life cycle definition
	ALC_TAT	1	Tools and techniques
Security target evaluation	ASE_CCL	1	Conformance claims
	ASE_ECD	1	Extended component definition
	ASE_INT	1	ST introduction
	ASE_OBJ	2	Security objectives
	ASE_REQ	2	Security requirements
	ASE_SPD	1	Security problem definition
	ASE_TSS	1	TOE summary specification
Tests	ATE_COV	2	Coverage
	ATE_DPT	1	Depth
	ATE_FUN	1	Functional tests
	ATE_IND	2	Independent testing
Vulnerability assessment	**AVA_VAN**	**4**	**Vulnerability analysis**

Table 6: Cryptosmart-SIM card aimed assurance levels

6. GLOBAL TOE SPECIFICATION

This chapter provides the TOE summary specification, a high-level definition of the security functions claimed to meet the functional and assurance requirements.

6.1 TOE SECURITY FUNCTIONS

The TOE security functions include the following:

- Identification and authentication (IA): this family includes all functions related to identification and authentication of users and administrators;
- Cryptography (CR): this family includes all cryptography related functions;
- Protection and filtering (PR): this family includes all functions related to protecting user data;
- Security management (GS): this family includes all functions related to managing security policies.
- Platform provided security functions (PTF): this family includes all functions entirely managed by the underlying platform.

6.1.1 IDENTIFICATION AND AUTHENTICATION (IA)

SF.IA_ROLES

The TOE maintains two roles:

- The administrator which is the TOE user while in PERSONALIZATION and WIPED states
- The local user which is the TOE user while in UNVALIDATED_USER, VALIDATED_USER, RESUMABLE, RESUMED_USER and BLOCKED states.

SF.IA_AUTH_PARAMETER

The TOE shall maintain the following elements related to its user:

- The user security code;
- The corresponding failed entries counter;
- The user ephemeral security code;
- The TOE state which includes the security code validation flag and the light unlock validation flag.
- 15 PUK codes
- The current authorized PUK code

SF.IA_SECURITY_CODE

The TOE is able to authenticate the local user by a security code. This security code is composed of numeric characters and has a maximum size of 8 characters. After 3 successive failed attempts, the TOE turns into a BLOCKED state.
The administrator may fix a minimum length requirement, which is set to 4 by default.
The security code must be transmitted to the TOE using the APDU encryption mechanism.

SF.IA_PUK_CODE

The TOE is able to authenticate the local user by a PUK code. This method of authentication is only available if the TOE is in the BLOCKED state. After 10 failed attempts authentication by PUK code is disabled. The TOE must be recycled.
PUK codes are composed of 8 (random) numeric characters generated by the TOE.
Already used PUK codes will be refused.
The PUK code must be entered using the APDU encryption mechanism.

SF.IA_EPHEMERAL_SECURITY_CODE

The TOE is able to authenticate the local user by an ephemeral security code. This ephemeral security code is composed of numeric characters and has a size of 8 characters. It is generated by the TOE after a successful authentication by security code or PUK code. This security code is re-generated after successive successful re-authentication with the security code or at user request.
Using a wrong ephemeral security code shall cause the TOE to erase the stored (right) ephemeral security code, which disables this authentication mechanism.
Ephemeral security code is composed of 8 (random) numeric characters generated by the TOE when the user has authenticated using his security code or a PUK code..
The ephemeral security code must be entered using the APDU encryption mechanism.

SF.IA_LIGHT_AUTH

The TOE is able to implicitly authenticate the user by successfully sending a correctly formatted resume command using the APDU encryption mechanism, with a null buffer as ephemeral security code argument. Usage of the APDU encryption mechanism proves knowledge of the 128 bits APDU encryption key. After such light authentication the TOE turns into a RESUMED_USER state which allows limited usage of the TOE.
This kind of authentication is only authorized after a successful security code authentication in case of card loss of power.

6.1.2 CRYPTOGRAPHY (CR)

SF.CR_STATEFULL_AUTHENTICATION

The TOE is able to conduct a session key negotiation phase with another card by using a SIGMA-R like protocol. This protocol allows the negotiation of a common secret with a perfect forward secrecy property and authenticates the distant card user. Unpowering the card does cause the protocol to abort.

Once the distant card user has been authenticated and the session key has been generated it is transmitted to the local user under a wrapped form and erased from memory.

SF.CR_STATELESS_AUTHENTICATION

The TOE is able to conduct a session key negotiation phase with another card by using a SIGMA like protocol. This protocol allows the negotiation of a common secret with a perfect forward secrecy property and authenticates the distant card user. Unpowering the card does not cause the protocol to abort.

Once the distant card user has been authenticated and the session key has been generated it is transmitted to the local user under a wrapped form and erased from memory

SF.CR_KEY_DERIVATION

The TOE is able to derive a cryptographic key into several other keys using a cryptographic key derivation mechanism. Once the derivation succeeded the initial key is updated, transmitted to the local user under a wrapped form with the derived keys (in clear) and every key is erased from memory.

SF.CR_LOCAL_PROTECTION_KEY

The TOE is able to derive a cryptographic key into several other keys using a cryptographic key derivation mechanism. Once the derivation succeeded the TOE outputs the obtained key to the TOE user.

SF.CR_RSA_ENCRYPT

The TOE is able to decrypt data using the RSA algorithm with PKCS#1.5-OAEP padding and 2048 bits keys.

SF.CR_RSA_SIGN

The TOE is able to sign data using the RSA algorithm with 2048 bits keys and PKCS#1.5 padding without use of any digest algorithm.

SF.CR_AES

The TOE is able to encrypt and decrypt data using the AES algorithm using 128 or 256 bits keys.

SF.CR_MAC

The TOE is able to perform MAC using HMAC-SHA256 and 128 or 256 bits keys.

The TOE is able to perform MAC using AES in retail CBC MAC mode (EMAC) and 256 bits keys (2*128).

The TOE is able to perform MAC using AES in CMAC mode and 256 bits keys.

SF.CR_RNG

The TOE is able to generate random numbers using the platform secure random generator post processed using AES with 128 bits keys.

SF.CR_KEY_GENERATION

The TOE is able to generate cryptographic keys using the following methods:

- RSA keys are generated using the built in javacard generator
- Symmetric keys are generated using the TOE random number generator

SF.CR_KEY_DESTRUCTION

The TOE destroys cryptographic keys by clearing them in memory.

6.1.3 PROTECTION AND FILTERING (PR)

SF.PR_ACCESS_RIGHTS

The TOE is able to perform access control for each command sent by the user. This is done by confronting the TOE state to:

- The command access rights.

Access control ensures the local user is authenticated before performing any action except:

- Get card information;
- Enter the security code;
- Card recycling;
- Card wiping;
- Initializing the APDU encryption key;
- Renew the APDU encryption IV;

The administrator is not required to be authenticated (the administrator may only use the TOE while in PERSONALIZATION state, and security is ensured by the environment in this state).
The command access rights are fixed in TOE code and may not be changed.
This applies in particular for accessing the TOE internal file system thus providing a secure storage area.

SF.PR_KEY_ACCESS_CONTROL

The TOE is able to perform access control for user keys. This is done by confronting the TOE state to:

- The “extractable” and “useable in resumed” key attributes

Key identifier, key type, extractable and useable in resumed flags attributes must be set by the user at asset creation.

SF.PR_RESIDUAL_WIPE

The TOE erases sensitive data when not needed any more (changed security code, after PUK code usage, etc.).

SF.PR_APDU

The TOE offers a trusted channel where every Cryptosmart command sent to the TOE in VALIDATED_USER and RESUMED_USER state can be AES encrypted and AES based MAC sealed using TOE generated keys.
Usage of this trusted channel is required for:

- initial user authentication;
- any command and TOE response in VALIDATED_USER, or RESUMED_USER state, except
  - response with a “distant card” order or transmission of an authentication packet;
  - get status command
  - APDU encryption keys renewal
- the “resume” command in RESUMABLE state

SF.PR_APDU_SCP11

The TOE offers a trusted channel where every supported ISO7816 command sent to the TOE in VALIDATED_USER and RESUMED_USER state can be protected using the SCP11 protocol.
Usage of this trusted channel is required for:

- ISO7816 user authentication;
- DEK obtention
- Password obtention
- Wrapped DEK unwrapping

6.1.4 SECURITY MANAGEMENT (GS)

SF.GS_ADMIN

An administrator can perform the following administration operation:

- Set the minimum security code length requirements
- Set the default user security code
- Create the PUK code within the TOE and get them
- Import a recycle key
- Import the SMS wrapping key
- Import, export or generate RSA keys (within the limits of the access control policy)
- Import, export or generate symmetric keys (within the limits of the access control policy)
- Delete RSA or symmetric keys (within the limits of the access control policy)
- Change user key attributes (within the limits of the access control policy)

An administrator may not change access rights to TOE functions.
Data are imported with security attributes: in case of symmetric or RSA keys attributes are specified by the administrator during their import; other data attributes are statically set when imported.

SF.GS_USER

A user can perform the following administration operation:

- Change its security code
- Import, export or generate RSA keys (within the limits of the access control policy)
- Import, export or generate symmetric keys (within the limits of the access control policy)
- Delete RSA or symmetric keys (within the limits of the access control policy)
- Change user key attributes (within the limits of the access control policy)

A user may not change access rights to TOE functions.

Data are imported with security attributes: symmetric or RSA keys attributes are specified by the user during their import.

7. RATIONALES

7.1 SECURITY OBJECTIVES RATIONALE

	T.TOE_DATA_CORRUPTION	T.TOE_DATA_COMPROMISE	T.ABUSE_FUNC	T.MASQUERADE	T.KEY_DERIVE	T.INTERFACE_EAVESDROP	T.LIFE_CYCLE	OSP.RGS_CRYPTO	OSP.MUTUAL_AUTHENTICATION	OSP.KEY_STORAGE	OSP.SYMMETRIC_ENCRYPTION	OSP.RSA_PRIVATE_KEY_OPERATION	OSP.LOCAL_AUTHENTICATION	OSP.PUK_UNBLOCK	OSP.DATA_WIPE	OSP.ACCESS_CONTROL	OSP.TOE_AUDIT	A.TRUSTED_ADMIN	A.TRAINED_ADMIN	A.CONFIGURATION	A.KEY_QUALITY	A.SECURE_KEY_MANAGEMENT	A.VERIFICATION	A.APPLET
O.USER_AUTHENTICATION				X									X											
O.PUK_UNBLOCK														X										
O.STRONG_SECCODE													X											
O.LIMITED_AUTH_NUMBER													X	X										
O.KEY_MANAGEMENT				X						X						X								
O.STRONG_MUTUAL_AUTHENTICATION									X															
O.PROTECT_SESSION_KEY	X	X																						
O.APDU_ENCRYPTION	X	X		X		X																		
O.CRYPTOGRAPHIC_OPERATION					X			X			X	X		X										
O.FUNCTION_ACCESS_CONTROL	X	X	X	X			X			X						X								
O.WIPE														X	X									
O.SENSITIVE_MEMORY_ERASING		X																						
OE.ADMIN	X	X																X	X					
OE.CARD_ADMIN_STATION	X								X											X				
OE.KEY_GENERATOR					X																X			

	T.TOE_DATA_CORRUPTION	T.TOE_DATA_COMPROMISE	T.ABUSE_FUNC	T.MASQUERADE	T.KEY_DERIVE	T.INTERFACE_EAVESDROP	T.LIFE_CYCLE	OSP.RGS_CRYPTO	OSP.MUTUAL_AUTHENTICATION	OSP.KEY_STORAGE	OSP.SYMMETRIC_ENCRYPTION	OSP.RSA_PRIVATE_KEY_OPERATION	OSP.LOCAL_AUTHENTICATION	OSP.PUK_UNBLOCK	OSP.DATA_WIPE	OSP.ACCESS_CONTROL	OSP.TOE_AUDIT	A.TRUSTED_ADMIN	A.TRAINED_ADMIN	A.CONFIGURATION	A.KEY_QUALITY	A.SECURE_KEY_MANAGEMENT	A.VERIFICATION	A.APPLET
OE.SECURE_PERSONALIZATION	X	X	X																					
OE.SECURE_KEY_MANAGEMENT	X	X																				X		
OE.SECURE_SECCODE_ENTRY		X																						
OE.NON_TRIVIAL_SECCODE		X																						
OE.HOST_CORRECT_BEHAVIOR		X				X			X															
OE.HOST_AUDIT																	X							
OE.VERIFICATION																							X	
OE.APPLET																								X

Table 7: Tracing between security objectives and security problem definition

T.TOE_DATA_CORRUPTION

This threat is covered by the following security objectives:

- O.FUNCTION_ACCESS_CONTROL: ensures write access to assets is controlled by the TOE. An attacker may not directly modify them as he must be authenticated to get access to TOE controlled assets;
- O.PROTECT_SESSION_KEY: ensures that modifications on outputted session keys will be detected;
- O.APDU_ENCRYPTION: ensures that an attacker may not alter asset while imported in the TOE by the user;
- OE.SECURE_PERSONALIZATION: ensures the personalization environment is secure. An attacker may not alter asset while imported in the TOE by the admin;
- OE.SECURE_KEY_MANAGEMENT: ensures the key are managed in a secure way while out of TOE scope. An attacker may not alter asset while transmitted from key generator to admin; or alter their stored value;
- OE.ADMIN: ensure that the attacker may not be the admin, nor can he corrupt the admin to alter assets;
- OE.CARD_ADMIN_STATION: ensures that the assets injected in TOE are consistent (the private key and certificate are correctly related).

T.TOE_DATA_COMPROMISE

This threat is covered by the following security objectives:

- O.FUNCTION_ACCESS_CONTROL: ensures read access to assets is controlled by the TOE. An attacker may not directly get them as he must be authenticated to get access to TOE controlled assets;
- O.PROTECT_SESSION_KEY: ensures outputted session keys are protected from disclosure;
- O.APDU_ENCRYPTION: ensures that an attacker may not eavesdrop assets while imported in the TOE by the user;
- O.SENSITIVE_MEMORY_ERASING: ensures that an attacker may not recover previously erased sensitive data
- OE.SECURE_PERSONALIZATION: ensures the personalization environment is secure. An attacker may not have knowledge of asset while imported in the TOE by the admin;
- OE.SECURE_KEY_MANAGEMENT: ensures the key are managed in a secure way while out of TOE scope. An attacker may not have knowledge of assets while transmitted from key generator to admin; or while stored for backup;
- OE.ADMIN: ensure that the attacker may not be the admin, nor can he corrupt the admin to get assets;
- OE.SECURE_SECCODE_ENTRY: ensures that the user security code is not compromised by an attacker while entered;
- OE.HOST_CORRECT_BEHAVIOR: ensures that the APDU encryption initialization key is trusted. When changed the user is informed and is in measure to refuse to enter his security code if he did not change his Cryptosmart-SIM card.
- OE.NON_TRIVIAL_SECCODE: ensures that the user security code is not guessable by an attacker

T.ABUSE_FUNC

This threat is covered by the following security objectives:

- O.FUNCTION_ACCESS_CONTROL: ensures that access to TOE function is controlled by the TOE. Functions for personalization may not be abused as they are only accessible in PERSONALIZATION state, and OE.SECURE_PERSONALIZATION ensures the attacker may not use the TOE.

T.MASQUERADE

This threat is covered by the following security objectives:

- O.USER_AUTHENTICATION which states that authorized data sources must be authenticated preventing than attacker to impersonate a legitimate user;

- O.APDU_ENCRYPTION which links every command to a previous successful authentication. An attacker may not send commands to the TOE by getting control of the communication link after the user has authenticated to the TOE;
- O.KEY_MANAGEMENT and O.FUNCTION_ACCESS_CONTROL which define security attributes based access control. This restricts access to keys according to TOE state and key security attributes (the TOE state is related to the fact that the user is authenticated, and how he had authenticated).

T.KEY_DERIVE

This threat is covered by the following security objectives:

- O.CRYPTOGRAPHIC_OPERATION and OE.KEY_GENERATOR ensure that generated keys have sufficient entropy generated by the TOE or not. The attacker may not bruteforce them.
- O.CRYPTOGRAPHIC_OPERATION ensures that RSA keys are generated using a cryptographically correct algorithm. This prevents the attacker to mount a mathematical attack on these keys.

T.INTERFACE_EAVESDROP

O.APDU_ENCRYPTION directly covers this threat by protecting the confidentiality of data transmitted by the user to the TOE and reciprocally.

OE.HOST_CORRECT_BEHAVIOR covers this threat by ensuring that the APDU encryption initialization key is trusted. When changed the user is informed and is in measure to refuse to enter his security code if he did not change his Cryptosmart-SIM card. This avoids the possibility of man in the middle when APDU encryption is initialized.

T.LIFE_CYCLE

This threat is covered by the following security objectives:

- O.FUNCTION_ACCESS_CONTROL: ensures that personalization function are not accessible to an attacker once the TOE is delivered to its end user.

OSP.RGS_CRYPTO

The objective O.CRYPTOGRAPHIC_OPERATION ensures that cryptographic algorithms and key generation algorithms conform to the RGS.

OSP.MUTUAL_AUTHENTICATION

This OSP is covered by the following security objectives:

- O.STRONG_MUTUAL_AUTHENTICATION which directly cover the OSP by providing the possibility for the user to run an authentication protocol;
- OE.HOST_CORRECT_BEHAVIOR which ensures that the certificate verification is performed correctly.
- OE.CARD_ADMIN_STATION which ensures that assets injected in TOE are consistent (private key and certificate): users are authenticated using the certificate related to the private key used in protocol. (this only allows the protocol to successfully run. Otherwise it would always fail at certificate or signature verification)

OSP.KEY_STORAGE

This OSP is covered by the following security objectives:

- O.KEY_MANAGEMENT and O.FUNCTION_ACCESS_CONTROL which enable a strong security attributes based access control

OSP.SYMMETRIC_ENCRYPTION

The objective O.CRYPTOGRAPHIC_OPERATION directly covers this OSP.

OSP.RSA_PRIVATE_KEY_OPERATION

The objective O.CRYPTOGRAPHIC_OPERATION directly covers this OSP.

OSP.LOCAL_AUTHENTICATION

The objective O.USER_AUTHENTICATION directly answers to the first part of the OSP (the user must be authenticated before performing sensitive action)

The objective O.STRONG_SECCODE ensures an attacker may not guess the security code, allowing to falsely authenticating to the TOE

The objective O.LIMITED_AUTH_NUMBER directly answers to the second part of the OSP concerning authentication error handling.

OSP.PUK_UNBLOCK

The objective O.PUK_UNBLOCK provides the ability to unblock the TOE

The objective O.CRYPTOGRAPHIC_OPERATION ensures an attacker may not guess PUK code as they are TOE generated

The objectives O.LIMITED_AUTH_NUMBER and O.WIPE directly answer to the second part of the OSP concerning PUK authentication error handling.

OSP.DATA_WIPE

The objective O. WIPE directly covers this OSP.

OSP.ACCESS_CONTROL

The objective O.FUNCTION_ACCESS_CONTROL covers this OSP by providing function access control.

The objective O.KEY_MANAGEMENT strengthens the coverage of this OSP for user keys by providing a second level of access control for user keys.

OSP.AUDIT

This OSP is directly covered by the objective on the environment OE.HOST_AUDIT

A.TRUSTED_ADMIN

This assumption is directly covered by the objective on the environment OE.ADMIN

A.TRAINED_ADMIN

This assumption is directly covered by the objective on the environment OE.ADMIN

A.CONFIGURATION

This assumption is covered by OE.CARD_ADMIN_STATION which ensures that assets injected in TOE are consistent.

A.KEY_QUALITY

This assumption is directly covered by the objective on the environment OE.KEY_GENERATOR

A.SECURE_KEY_MANAGEMENT

This assumption is directly covered by the objective on the environment OE.SECURE_KEY_MANAGEMENT

A.VERIFICATION

This assumption is directly covered by the objective on the environment OE.VERIFICATION

A.APPLET

This assumption is directly covered by the objective on the environment OE.APPLET

7.2 SECURITY REQUIREMENTS RATIONALE

7.2.1 STUDY OF DEPENDENCIES

The following table summarizes the dependencies of the security requirement components and justifies their satisfaction or non-satisfaction.

Component	Dependencies	Satisfaction
FIA_ATD.1	No dependencies.	
FIA_UAU.1	FIA_UID.1	Not fulfilled (justified)
FIA_UAU.5	No dependencies.	
FIA_UAU.4	No dependencies.	
FIA_SOS.1	No dependencies.	
FIA_SOS.2	No dependencies.	
FIA_AFL.1	FIA_UAU.1[3]	FIA_UAU.1
FDP_ACC.2/access	FDP_ACF.1	FDP_ACF.1/access
FDP_ACF.1/access	FDP_ACC.1	FDP_ACC.2/access
	FMT_MSA.3	FMT_MSA.3/access
FMT_MSA.3/access	FMT_MSA.1	FMT_MSA.1/access
	FMT_SMR.1	FMT_SMR.1
FMT_MSA.1/access	FDP_ACC.1 or FDP_IFC.1	FDP_ACC.2/access
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FCS_COP.1	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	FCS_CKM.1 FDP_ITC.1/systemkey FDP_ITC.1/pubkey
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.1	FCS_CKM.2 or FCS_COP.1	FCS_COP.1
	FCS_CKM.4	FCS_CKM.4
FCS_CKM.4	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	FDP_ITC.1/systemkey FCS_CKM.1
FDP_ACC.2/keys	FDP_ACF.1	FDP_ACF.1/keys
FDP_ACF.1/keys	FDP_ACC.1	FDP_ACC.2/keys
	FMT_MSA.3	FMT_MSA.3/access
FDP_ETC.1/keys	FDP_ACC.1 or FDP_IFC.1	FDP_ACC.2/keys
FDP_ITC.2/keys	FDP_ACC.1 or FDP_IFC.1	FDP_ACC.2/keys

[3] This SFR satisfies the dependency requirement for each FIA_AFL.1 instantiation (FIA_AFL.1/userSC, FIA_AFL.1/ephSC and FIA_AFL.1/PUK)

	FTP_ITC.1 or FTP_TRP.1	FTP_TRP.1
	FPT_TDC.1	FPT_TDC.1
FMT_MSA.1/keys	FDP_ACC.1 or FDP_IFC.1	FDP_ACC.2/keys
	FMT_SMR.1	FMT_SMR.1
	FMT_SMF.1	FMT_SMF.1
FMT_MSA.2/keys	FDP_ACC.1 or FDP_IFC.1	FDP_ACC.2/keys
	FMT_MSA.1	FMT_MSA.1/keys
	FMT_SMR.1	FMT_SMR.1
FPT_TDC.1	No dependencies	
FDP_ITC.1/systemkey	FDP_ACC.1 or FDP_IFC.1	FDP_ACC.2/access
FDP_ITC.1/pubkey	FMT_MSA.3	Not fulfilled (justified)
FTP_TRP.1/Cryptosmart	No dependencies.	
FTP_TRP.1/Scp11	No dependencies.	
FDP_RIP.1	No dependencies.	
FMT_SMR.1	FIA_UID.1.	FIA_UAU.1
FMT_SMF.1	No dependencies.	

Table 8: SFR dependencies

UNRESOLVED DEPENDENCIES JUSTIFICATION

FIA_UAU.1 ← FIA_UID.1: the TOE is single user for a given state, combination of TOE state and user authentication provides identification. Therefore the FIA_UID dependency does not need to be fulfilled

FMT_SMR.1 ← FIA_UID.1: the TOE is single user for a given state, combination of TOE state and user authentication provides identification. Therefore the FIA_UID dependency is replaced by **FIA_UAU.1**

FDP_ITC.1/systemkey ← FMT_MSA.3: the TOE does not define security attribute for wrapping keys. Their usage is fully controlled by SFP_OPERATION_ACCESS_CONTROL which provides access control on the functions involving the wrapping key

FDP_ITC.1/pubkey ← FMT_MSA.3: TOE does not define security attribute for signature verification keys: its usage is fully controlled by SFP_OPERATION_ACCESS_CONTROL which provides access control on the functions involving it

7.2.2 SECURITY REQUIREMENTS / OBJECTIVES CONSISTENCY MATRIX

	O.USER_AUTHENTICATION	O.PUK_UNBLOCK	O.STRONG_SECCODE	O.LIMITED_AUTH_NUMBER	O.FUNCTION_ACCESS_CONTROL	O.CRYPTOGRAPHIC_OPERATION	O.STRONG_MUTUAL_AUTHENTICATION	O.KEY_MANAGEMENT	O.PROTECT_SESSION_KEY	O.APDU_ENCRYPTION	O.SENSITIVE_MEMORY_ERASING	O.WIPE
FIA_ATD.1	X	X										
FIA_UAU.1	X											
FIA_UAU.5	X	X										
FIA_UAU.4		X										
FIA_SOS.1			X									
FIA_SOS.2			X									
FIA_AFL.1				X								
FDP_ACC.2/access					X							
FDP_ACF.1/access					X							
FMT_MSA.3/access					X							
FMT_MSA.1/access					X							
FCS_COP.1/*						X	X		X			
FCS_CKM.1/*								X				
FCS_CKM.4/*								X			X	X
FDP_ACC.2/keys								X				
FDP_ACF.1/keys								X				
FMT_MSA.1/keys								X				
FMT_MSA.2/keys								X				
FPT_TDC.1								X				
FDP_ETC.1/keys								X				
FDP_ITC.2/keys								X				
FDP_ITC.1/systemkey									X			
FDP_ITC.1/pubkey							X					
FTP_TRP.1/Cryptosmart	X									X		
FTP_TRP.1/Scp11										X		
FDP_RIP.1											X	X
FMT_SMR.1			X		X			X				
FMT_SMF.1			X		X			X				

Table 9: security requirements / objectives consistency matrix

7.2.3 RATIONALE

O.USER_AUTHENTICATION is covered by:

- FIA_ATD.1 which defines the authentication attributes for the local user (security code, ephemeral security code and validation flag and counter);
- FIA_UAU.1 which defines the commands an unauthenticated user may perform;
- FIA_UAU.5 which defines the different ways to authenticate an user :
  - Through its security code;
  - Through an ephemeral security code if the TOE is in a RESUMABLE state;
  - Through a PUK code if the TOE is in a BLOCKED state;
  - Implicitly by ability to use the host-card trusted path if the TOE is in a RESUMABLE state.
- FTP_TRP.1/Cryptosmart which defines a trusted path between the TOE and the local user. Successful usage of this trusted channel in RESUMABLE state implicitly authenticates the user by proof of knowledge of the protection keys.

O.PUK_UNBLOCK is covered by:

- FIA_ATD.1 which defines the authentication attributes for the local user, in particular the TOE state;
- FIA_UAU.5 which defines how to unblock the TOE: the user is authenticated by using PUK code, which turns the TOE from BLOCKED to VALIDATED_USER state;
- FIA_UAU.4 which ensures each PUK code may only be used once.

O.STRONG_SECCODE is covered by:

- FIA_SOS.1 which ensures the user security code has the required length property;
- FIA_SOS.2 which ensures that PUK codes and ephemeral code are correctly generated by the TOE
- FMT_SMF.1 which allows the administrator to set a minimum security code length
- Correct management of security features (FMT_SMR.1 and FMT_SMF.1)

O.LIMITED_AUTH_NUMBER is covered by:

- FIA_AFL.1 which defines the maximum possible failures for each authentication method and the TOE behavior when this maximum tries has been reached.

O.FUNCTION_ACCESS_CONTROL is covered by:

- The SFP_OPERATION_ACCESS_CONTROL access control SFP (FDP_ACC.2/access and FDP_ACF.1/access) which ensure every command passed to the TOE follows the access control policy;
- FMT_MSA.3/access which sets the default values for each command access right;
- FMT_MSA.1/access which ensures no one can change access rights;
- Correct management of security features (FMT_SMR.1 and FMT_SMF.1)

O.CRYPTOGRAPHIC_OPERATION is covered by:

- All FCS_COP.1/* which defines the cryptographic algorithms the TOE shall support;

O.STRONG_MUTUAL_AUTHENTICATION is covered by:

- FCS_COP.1/random
- FCS_COP.1/DH
- FCS_COP.1/RSA_sig
- FCS_COP.1/RSA_verif
- FCS_COP.1/SHA256
- FCS_COP.1/HMAC
- FCS_COP.1/AES
- FCS_COP.1/EMAC
- FDP_ITC.1/pubkey

The mutual authentication is performed using the Cryptosmart authentication protocol. This protocol is based on the listed SFR.

O.KEY_MANAGEMENT is covered by:

- Respective FCS_CKM.1/* which ensures user keys can be correctly generated by the TOE;
- FDP_ITC.2/keys which ensures user keys may be imported to the TOE;
- FCS_CKM.4/java and FCS_CKM.4/zero which ensure secure keys destruction of user keys;
- The SFP_KEY_ACCESS_CONTROL ensures the security of key management, in particular:
  - FDP_ACC.2/keys and FDP_ACF.1/keys defines the access control to user keys;
  - FMT_MSA.1/keys and FMT_MSA.2/keys focus on security attribute management by users
  - FDP_ETC.1/keys defining conditions under which keys may be exported from the TOE;
  - FDP_ITC.2/keys defining conditions under which keys may be imported inside the TOE;
- FPT_TDC.1 which ensures imported security attribute are correctly interpreted
- Correct management of security features (FMT_SMR.1 and FMT_SMF.1)

O.PROTECT_SESSION_KEY is covered by:

- FCS_COP.1/AES and FCS_COP.1/EMAC which ensure correct protection of session keys when stored outside the TOE;
- FDP_ITC.1/systemkey supports FCS_COP.1.1 by defining how wrapping which protect the session keys may be imported inside the TOE.

O.APDU_ENCRYPTION is covered by:

- FTP_TRP.1/Cryptosmart which ensure the usage of a trusted path, protected both in confidentiality and integrity, for sending command to the TOE and getting its responses.
- FTP_TRP.1/Scp11 which ensure the usage of a trusted path, protected both in confidentiality and integrity, for sending command to the TOE and getting its responses.

O.SENSITIVE_MEMORY_ERASING and O.WIPE are both covered by:

- FDP_RIP.1 which ensures non keys sensitive data are erased when no more needed;
- All FCS_CKM.4/* which ensure the correct destruction of keys.

7.2.4 SAR RATIONALE

The aimed assurance level for this security target is EAL4 augmented with ALC_DVS.2 and AVA_VAN.4.

This level has been chosen to be as close as possible the French “qualification renforcée” package. (which also includes ADV_IMP.2 and ALC_FLR.3 which can’t be claimed as these component are not met by the platform).

The AVA_VAN.4 level is the maximum reachable due to the platform certificate.

The ALC_DVS.2 component has no dependencies.

The AVA_VAN.4 component has for dependencies:

- ADV_ARC.1 Security architecture description
- ADV_FSP.4 Complete functional specification
- ADV_TDS.3 Basic modular design
- ADV_IMP.1 Implementation representation of the TSF
- AGD_OPE.1 Operational user guidance
- AGD_PRE.1 Preparative procedures
- ATE_DPT.1 Testing: basic design

Which are all included in the EAL4 package. Therefore every requirement for the chosen SAR is fulfilled.

7.3 GLOBAL TOE SPECIFICATION RATIONALE

7.3.1 MAPPING TOE SFRS TO TOE SECURITY FUNCTIONS

	SF.IA_roles	SF.IA_auth_parameter	SF.IA_security_code	SF.IA_PUK_code	SF.IA_ephemeral_security_code	SF.IA_light_auth	SF.CR_statefull_authentication	SF.CR_stateless_authentication	SF.CR_key_derivation	SF.CR_local_protection_key	SF.CR_RSA_encrypt	SF.CR_RSA_sign	SF.CR_AES	SF.CR_MAC	SF_CR_key_generation	SF.CR_rng	SF.CR_key_destruction	SF.PR_access_rights	SF.PR_Key_access_control	SF.PR_Residual_wipe	SF.PR_apdu	SF.PR_apdu_SCP11	SF.GS_admin	SF.GS_user
FIA_ATD.1		X																						
FIA_UAU.1																		X						
FIA_UAU.5			X	X	X	X																		
FIA_UAU.4				X																				
FIA_SOS.1			X																					
FIA_SOS.2				X	X											X								
FIA_AFL.1			X	X	X																			
FDP_ACC.2/access																		X						
FDP_ACF.1/access																		X						
FMT_MSA.3/access																		X						
FMT_MSA.1/access																		X						
FCS_COP.1							X	X	X		X	X	X	X		X								
FCS_CKM.1										X					X									
FCS_CKM.4																	X							
FDP_ACC.2/keys																			X					
FDP_ACF.1/keys																			X					
FDP_ETC.1/keys																							X	X

	SF.IA_roles	SF.IA_auth_parameter	SF.IA_security_code	SF.IA_PUK_code	SF.IA_ephemeral_security_code	SF.IA_light_auth	SF.CR_statefull_authentication	SF.CR_stateless_authentication	SF.CR_key_derivation	SF.CR_local_protection_key	SF.CR_RSA_encrypt	SF.CR_RSA_sign	SF.CR_AES	SF.CR_MAC	SF_CR_key_generation	SF.CR_rng	SF.CR_key_destruction	SF.PR_access_rights	SF.PR_Key_access_control	SF.PR_Residual_wipe	SF.PR_apdu	SF.PR_apdu_SCP11	SF.GS_admin	SF.GS_user
FDP_ITC.2/keys																							X	X
FPT_TDC.1																							X	X
FMT_MSA.1/keys																							X	X
FMT_MSA.2/keys																							X	X
FDP_ITC.1/systemkey																							X	
FDP_ITC.1/pubkey								X																
FTP_TRP.1/Cryptosmart																					X			
FTP_TRP.1/Scp11																						X		
FDP_RIP.1																				X				
FMT_SMR.1	X																							
FMT_SMF.1																							X	X

Table 10: Mapping toe sfrs to toe security functions

7.3.2 RATIONALE

This section describes how the TOE security functions described in TOE summary specification meet each SFR.

FIA_ATD.1 (User attribute definition) describes the list of attributes belonging to users. This component is met by SF.IA_AUTH_PARAMETER which defines the user related elements maintained by the TOE.

FIA_UAU.1 (Timing of authentication) describes the functions the TOE may perform without having the user authenticated. This component is met by SF.PR_ACCESS_RIGHTS which defines these actions.

FIA_UAU.5 (Multiple authentication mechanisms) describes the different available authentication types. This component is met by SF.IA_SECURITY_CODE SF.IA_PUK_CODE SF.IA_EPHEMERAL_SECURITY_CODE SF.IA_LIGHT_AUTH which each defines an individual authentication type supported by the TOE.

FIA_SOS.1 (Verification of secrets) requires a mechanism to verify secrets length. This component is met by SF.IA_SECURITY_CODE which defines the required secret length.

FIA_SOS.2 (Generation of secrets) describes the methods the TOE shall use to generate secrets and the authentication method which require generated secrets usage. This component is met by SF.IA_PUK_CODE and SF.IA_EPHEMERAL_SECURITY_CODE which specify that they must use TOE generated secrets and SF.CR_RNG which defines the generation method of secrets.

FIA_AFL.1 (Authentication failure handling) describes how the TOE shall react on authentication failure. The authentication mechanism descriptions SF.IA_SECURITY_CODE SF.IA_PUK_CODE SF.IA_EPHEMERAL_SECURITY_CODE each describe how to handle authentication failure.

The **FDP_ACC.2/access (Complete access control)** and **FDP_ACF.2/access (Security attributes based access control)** components are met by SF.PR_ACCESS RIGHTS which defines the access control method to sensitive assets.

FMT_MSA.3/access (Static attributes initialization) defines how attributes used for FDP_ACF.2/access are initialized. SF.PR_ACCESS RIGHTS defines this initialization.

FMT_MSA.1/access (Management of security attributes) requires that attributes used for FDP_ACF.2/access may not be changed. SF.PR_ACCESS RIGHTS meets this requirement by explicitly stating this.

FCS_COP.1 (Cryptographic operation) defines how the TOE shall perform cryptographic operation. This component is instantiated for:

- Diffie-Hellman operation
- AES operation
- SHA256 message digest operation
- HMAC MAC operation
- EMAC MAC operation
- RSA signature generation
- RSA signature verification
- RSA encryption
- Random generation

These requirements are met respectively by:

- SF.CR_STATEFULL_AUTHENTICATION and SF.CR_STATELESS_AUTHENTICATION
- SF.CR_AES and SF.CR_KEY_DERIVATION
- SF.CR_STATEFULL_AUTHENTICATION and SF.CR_STATELESS_AUTHENTICATION and SF.CR_LOCAL_PROTECTION_KEY
- SF.CR_MAC, SF.CR_STATEFULL_AUTHENTICATION and SF.CR_STATELESS_AUTHENTICATION
- SF.CR_MAC and SF.CR_KEY_DERIVATION
- SF.CR_RSA_SIGN, , SF.CR_STATEFULL_AUTHENTICATION and SF.CR_STATELESS_AUTHENTICATION
- SF.CR_STATELESS_AUTHENTICATION
- SF.CR_RSA_ENCRYPT
- SF.CR_RNG

Which each define the cryptographic functions of the TOE.

FCS_CKM.1 (Cryptographic key generation) defines how the TOE shall generate RSA and symmetric keys. This is directly met by SF.CR_KEY_GENERATION.

FCS_CKM.4 (Cryptographic key destruction): This component is met by SF.CR_KEY_DESTRUCTION which describes key destruction.

The **FDP_ACC.2/keys (Complete access control)** component is met by SF.PR_KEY_ACCESS_CONTROL which defines the access control method to user keys.

The **FDP_ACF.2/keys (Security attributes based access control)** component is met by SF.PR_KEY_ACCESS_CONTROL which defines the access control method to user keys.

The **FDP_ETC.1/keys (Export of user data without security attributes)** component is met by SF.GS_admin and SF.GS_user which defines the management operation on user keys.

The **FDP_ITC.2/keys (Import of user data with security attributes)** component is met by SF.GS_admin and SF.GS_user which defines the management operation on user keys.

The **FPT_TDC.1 (Inter-TSF basic TSF data consistency)** component is met by SF.GS_admin and SF.GS_user which defines the management operation on user keys.

FMT_MSA.1/keys (Management of security attributes) defines how and by whom attributes used for FDP_ACF.2/keys may be changed. SF.GS_ADMIN and SF.GS_USER meet this requirement by defining management possibilities respectively for the administrator and the user.

FMT_MSA.2/keys (Secure security attributes) defines the attribute management limitations. This requirement is met by SF.GS_ADMIN and SF.GS_USER which state these limitations.

FDP_ITC.1/systemkey (Import of user data without security attributes) is met by SF.GS_ADMIN which specifies how administrator keys are imported.

FDP_ITC.1/pubkey (Import of user data without security attributes) is met by SF.PR_access_rights which enforce access control on interface for importing this public key.

FTP_TRP.1/Cryptosmart (Trusted path) requires usage of a protected channel (in confidentiality, integrity and against replay) for passing commands to the TOE. It also defines the commands for which this channel is not necessary. This is met by SF.PR_APDU which defines such a trusted channel.

FTP_TRP.1/Scp11 (Trusted path) requires usage of a protected channel (in confidentiality, integrity and against replay) for passing commands to the TOE. It also defines the commands for which this channel is not necessary. This is met by SF.PR_APDU_SCP11 which defines such a trusted channel.

FDP_RIP.1 (Residual Information Protection): This component is met by SF.PR_RESIDUAL_WIPE which describes destruction of sensitive elements (which are not cryptographic keys).

FMT_SMR.1 (Security roles) describes the roles of TOE users. This component is met by SF.IA_ROLES which defines these roles.

FMT_SMF.1 (Specification of Management Functions): This component is met by SF.GS_ADMIN and SF.GS_USER which describe the management functions the administrator or the user may perform.

8. CONSISTENCY OF COMPOSITE PRODUCT SECURITY TARGET

8.1 SEPARATION OF TSF

The composite product is an applet loaded on the platform. The platform offers protection to the applet and its sensitive data against all relevant threat from [4].

8.2 COMPATIBILITY OF SAR

The following table summarizes the SAR for both the composite product and the platform, showing that every platform SAR is greater than or equal to composite SAR.

Assurance class	Assurance component	Platform Level	Composite Level
Development	ADV_ARC	1	1
	ADV_FSP	5	4
	ADV_IMP	1	1
	ADV_INT	2	N.A.
	ADV_TDS	4	3
Guidance documents	AGD_OPE	1	1
	AGD_PRE	1	1
Life-cycle support	ALC_CMC	4	4
	ALC_CMS	5	4
	ALC_DEL	1	1
	ALC_DVS	2	2
	ALC_LCD	1	1
	ALC_TAT	2	1
Security target evaluation	ASE_CCL	1	1
	ASE_ECD	1	1
	ASE_INT	1	1
	ASE_OBJ	2	2
	ASE_REQ	2	2
	ASE_SPD	1	1
	ASE_TSS	1	1
Tests	ATE_COV	2	2
	ATE_DPT	3	1
	ATE_FUN	1	1
	ATE_IND	2	2
Vulnerability assessment	AVA_VAN	4	4

Table 11: SAR compatibility

8.3 Compatibility of SFR

Some TOE SFRs are based on platform SFR. This section aims at identifying these dependencies and verifying that operations performed by composite SFR on platform SFR are appropriate.

Composite SFR	**Based on (platform SFR)** **F**ully (F) or **P**artially (P)	**Statement of compatibility**
FIA_ATD.1	None	User attributes are only related to the Cryptosmart applet
FIA_UAU.1	None	Timing of authentication is only related to the Cryptosmart applet
FIA_UAU.5	Platform's SFRs covering platform's O.PIN-MNGT[4] (P)	Authentication codes (security code, ephemeral code and PUK codes) are stored as OwnerPIN objects.
FIA_UAU.4	Platform's SFRs covering platform's O.PIN-MNGT (P)	The Cryptosmart applet stores PUK code as OwnerPin objects with maximum authentication attempts defined to 1.
FIA_SOS.1	None	
FIA_SOS.2	None.	These secrets are generated by the TOE using the applet's internal random generator.
FIA_AFL.1	Platform's SFRs covering platform's O.PIN-MNGT (F)	The applet bases security code and PUK code management on OwnerPin objects. The maximum unsuccessfull authentication failure is considered reached if the OwnerPin object is blocked.
FDP_ACC.2/access	None.	The SFP_OPERATION_ACCESS_CONTROL policy is entirely managed by the Cryptosmart applet.
FDP_ACF.1/access	None.	
FMT_MSA.3/access	None.	
FMT_MSA.1/access	None.	
FCS_COP.1.1/AES	FCS_COP.1 (P)	The applet only uses 128 and 256 bits AES keys which are consistent with the platform SFR.
FCS_COP.1/SHA256	FCS_COP.1 (F)	The applet directly uses the applet's sha256 implementation using javacard API
FCS_COP.1.1/HMAC	FCS_COP.1 (P)	The applet performs HMAC-SHA256 based on the SHA256 defined by this SFR. HMAC mode is implemented by the applet.
FCS_COP.1.1/EMAC	FCS_COP.1 (P)	The applet performs EMAC (retail cbc-mac) based on AES with 128 bits keys which is consistent with the platform SFR.

[4] Platform's security objective O.PIN-MNGT is covered by the following platform's SFRs (cf. [4]) : FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_ROL.1/FIREWALL, FDP_SDI.2 and the firewall security functions FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL.

Composite SFR	**Based on (platform SFR)** **F**ully (F) or **P**artially (P)	**Statement of compatibility**
FCS_COP.1.1/CMAC	FCS_COP.1 (P)	The applet performs CMAC based on AES with 256 bits keys which is consistent with the platform SFR.
FCS_COP.1.1/RSA_sig	FCS_COP.1 (P)	The applet only uses 2048 bits RSA keys (in CRT form) for signature operation, which is consistent with the platform SFR.
FCS_COP.1.1/RSA_enc	FCS_COP.1 (P)	The applet only uses 2048 bits RSA keys (in CRT form) for decryption operation, which is consistent with the platform SFR.
FCS_COP.1.1/random	FCS_RNG.1/SCP (P) FCS_COP.1 (P)	The applet generates random number by: - Generating a random buffer using the SecureRandom javacard object - Postprocessing it using an AES based DRBG
FCS_COP.1.1/DH	FCS_COP.1 (P)	The applet only uses 2048 bits modulus which is consistent with the platform SFR. Diffie-Hellman computations rely on FCS_COP.1/RSA in SFM mode.
FCS_CKM.1.1/RSA	FCS CKM.1.1 (F)	
FCS_CKM.1.1/random	None.	These keys (symmetric keys and DH private keys) are generated at random using the applet generator.
FCS_CKM.1.1/local_prot	FCS_COP.1 (P)	The applet performs the key generation based on the SHA256 defined by this SFR.
FCS_CKM.4	FCS_CKM.4 (F) for keys values stored in a key object in NVM FDP_RIP.1.1/KEYS (F) for keys values stored in the cryptographic buffer. FDP_RIP.1.1/APDU (F) and FDP_RIP.1.1/TRANSIENT (F) for key values stored in volatile memory	Keys stored as javacard objects are cleared using the clearKey() method. Cryptographic keys which are stored as arrays are erased by zeroization
FDP_ACC.2/keys	None.	
FDP_ACF.1/keys	None.	
FDP_ETC.1/keys	None.	
FDP_ITC.2/keys	FCS_CKM.2 (P)	User keys are imported then stored in javacard objects using the setKey() method.
FPT_TDC.1	None.	
FMT_MSA.1/keys	None.	

Composite SFR	**Based on (platform SFR)** **F**ully (F) or **P**artially (P)	**Statement of compatibility**
FMT_MSA.2/keys	None.	
FDP_ITC.1.1/systemkey	None.	
FDP_ITC.1.1/pubkey	None.	
FTP_TRP.1/Cryptosmart	None.	Trusted path protection is based on composite SFRs FCS_COP.1.1/AES, FCS_COP.1.1/EMAC for confidentiality and integrity protection and FCS_COP.1.1/RSA_enc for key establishment.
FTP_TRP.1/Scp11	None.	Trusted path protection is based on composite SFRs FCS_COP.1.1/AES, FCS_COP.1.1/CMAC for confidentiality and integrity protection and FCS_COP.1.1/DH for key establishment.
FDP_RIP.1	FCS_CKM.4 (F) and FDP_RIP.1.1/KEYS (F) for keys values stored in the cryptographic buffer. FDP_RIP.1.1/APDU (F) and FDP_RIP.1.1/TRANSIENT (F) for key values stored in volatile memory	Session keys are stored as javacard key objects and benefit from the FCS_CKM.4 and FDP_RIP.1.1/KEYS SFR. Keys may transit through temporary location and are then erased either through FDP_RIP.1.1/APDU or FDP_RIP.1.1/TRANSIENT.
FMT_SMR.1	None	
FMT_SMF.1	None	

Table 12: composite SFR dependencies

Platform SFR	Classification	Comment
FDP_UIT.1/CCM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ROL.1/CCM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ITC.2/CCM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_FLS.1/CCM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCS_COP.1/DAP	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCS_COP.1/TOKEN-OT(TDES, AES and RSA)	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCS_COP.1/RECEIPTS-	RP_SFR-MECH	Supports platform's security objectives relevant to support non

OT(TDES, AES)		tampering and non bypassability
FCS_COP.1/CIPHERLOADFILE-OT(TDES and AES)	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_TDC.1/CCM-OT	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACF.1/SD	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMR.1/SD	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.3/SD	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.1/SD	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMF.1/SD	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACC.1/SD	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FTP_ITC.1/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCO_NRO.2/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_IFC.2/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_IFF.1/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.3/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMF.1/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_UID.1/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_UAU.1/SC	RP_SFR-MECH	Supports platform's security

		objectives relevant to support non tampering and non bypassability
FIA_UAU.4/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_AFL.1/SC-OT	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_UAU.7/SC-OT	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPR_UNO.1/SC-OT	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.1/SC	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACC.2/FIREWALL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACF.1/FIREWALL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_IFC.1/JCVM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_IFF.1/JCVM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_RIP.1/OBJECTS	RP_SFR-SERV	Cf. Table 12
FMT_MSA.1/JCRE	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.1/JCVM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.2/FIREWALL_JCVM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.3/FIREWALL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.3/JCVM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMF.1	RP_SFR-MECH	Supports platform's security objectives relevant to support non

		tampering and non bypassability
FMT_SMR.1	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCS_CKM.1	RP_SFR-SERV	Cf. Table 12
FCS_CKM.2	RP_SFR-SERV	Cf. Table 12
FCS_CKM.3	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCS_CKM.4	RP_SFR-SERV	Cf. Table 12
FCS_COP.1	RP_SFR-SERV	Cf. Table 12
FDP_RIP.1/ABORT	RP_SFR-SERV	Cf. Table 12
FDP_RIP.1/APDU	RP_SFR-SERV	Cf. Table 12
FDP_RIP.1/bArray	RP_SFR-SERV	Cf. Table 12
FDP_RIP.1/KEYS	RP_SFR-SERV	Cf. Table 12
FDP_RIP.1/TRANSIENT	RP_SFR-SERV	Cf. Table 12
FDP_ROL.1/FIREWALL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FAU_ARP.1	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_SDI.2	RP_SFR-SERV	Cf. Table 12
FPR_UNO.1	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_FLS.1	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_TDC.1	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_ATD.1/AID	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_UID.2/AID	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_USB.1/AID	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MTD.1/JCRE	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability

FMT_MTD.3/JCRE	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ITC.2/Installer	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMR.1/Installer	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_FLS.1/Installer	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_RCV.3/Installer	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_RIP.1/ODEL	RP_SFR-SERV	Cf. Table 12
FPT_FLS.1/ODEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACC.2/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACF.1/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_RIP.1/ADEL	RP_SFR-SERV	Cf. Table 12
FMT_MSA.1/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.3/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMF.1/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMR.1/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_FLS.1/ADEL	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCO_NRO.2/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_IFC.2/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability

FDP_UIT.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_IFF.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FIA_UID.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_MSA.3/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMF.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FMT_SMR.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FTP_ITC.1/CM	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FDP_ACC.2/JCRMI	IR_SFR	Relates to irrelevant remote method invocation
FDP_ACF.1/JCRMI		
FDP_IFC.1/JCRMI		
FDP_IFF.1/JCRMI		
FMT_MSA.1/EXPORT		
FMT_MSA.1/REM_REFS		
FMT_MSA.3/JCRMI		
FMT_REV.1/JCRMI		
FMT_SMF.1/JCRMI		
FMT_SMR.1/JCRMI		
FPT_FLS.1/SCP	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FRU_FLT.1/SCP	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_PHP.3/SCP	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FPT_RCV.3/SCP	RP_SFR-MECH	Supports platform's security

		objectives relevant to support non tampering and non bypassability
FPT_RCV.4/SCP	RP_SFR-MECH	Supports platform's security objectives relevant to support non tampering and non bypassability
FCS_RNG.1/SCP	RP_SFR-SERV	Cf. Table 12
FMT_SMR.1/DF	IR_SFR	Relates to irrelevant DESFire
FMT_MSA.1/DF		
FDP_ACC.1/DF		
FDP_ACF.1/DF		
FDP_ROL.1/DF		
FMT_MSA.3/DF		
FMT_MTD.1/DF		
FMT_SMF.1/DF		
FDP_ITC.2/DF		
FIA_UID.2/DF		
FIA_UAU.2/DF		
FIA_UAU.5/DF		
FTP_TRP.1/DF		
FPT_RPL.1/DF		
FPT_TDC.1/DF		

8.4 COMPATIBILITY OF SECURITY OBJECTIVES

Only a subset of the platform security objectives (as described in [4]) is relevant to the composite security target.

The table below lists the security objectives provided by the composite TOE and for each of them gives the security objectives from the platform it relies on.

Security objectives from the platform which are not mentioned in this table are not relevant to the composite TOE.

Security objectives of the composite TOE	Rely on the following objectives from the underlying platform, either Fully (F) or Partly (P)
O.USER_AUTHENTICATION	O.PIN-MNGT (user security code and PUK are managed as platform Pin objects) (F)
O.PUK_UNBLOCK	O.PIN-MNGT (PUK codes are managed as platform Pin objects) (F)
O.STRONG_SECCODE	None.

Security objectives of the composite TOE	Rely on the following objectives from the underlying platform, either Fully (F) or Partly (P)
O.LIMITED_AUTH_NUMBER	O.PIN-MNGT (user security code and PUK are managed as platform Pin objects which manage the failure limit) (P)
O.FUNCTION_ACCESS_CONTROL	None.
O.CRYPTOGRAPHIC_OPERATION	O.CIPHER,: cryptographic operation done by the applet use platform provided cryptographic functions. (P)
O.STRONG_MUTUAL_AUTHENTICATION	O.CIPHER: cryptographic operation done by the applet use platform provided cryptographic functions. (P)
O.KEY_MANAGEMENT	O.KEY-MNGT: user keys are managed as platform key object (P)
O.PROTECT_SESSION_KEY	O.CIPHER: key wrapping is performed using platform provided cryptographic functions O.KEY-MNGT (P)
O.APDU_ENCRYPTION	O.CIPHER: APDU encryption is performed using platform provided cryptographic functions (P)
O.SENSITIVE_MEMORY_ERASING	O.KEY-MNGT: for user keys as the platform provides a method to clear keys. O.REALLOCATION (P)
O.WIPE	O.KEY-MNGT: for user keys as the platform provides a method to clear keys. (P)

Table 13: Compatibility of security objectives

The following security objectives for the platform are relevant to support non tampering and non bypassability (ADV_ARC):

- O.CARD-MANAGEMENT
- O.DOMAIN-RIGHTS
- O.APPLI-AUTH
- O.COMM_AUTH
- O.COMM_INTEGRITY
- O.SID
- O.FIREWALL
- O.GLOBAL_ARRAYS_CONFID
- O.GLOBAL_ARRAYS_INTEG
- O.NATIVE
- O.OPERATE
- O.REALLOCATION
- O.RESOURCES
- O.ALARM
- O.TRANSACTION
- O.OBJ-DELETION
- O.LOAD

- O.INSTALL
- O.SCP.SUPPORT
- O.SCP.IC
- O.SCP.RECOVERY

8.5 Compatibility of security objectives for the environment

Platform security objectives for the environment concern:

- USIM security objectives for the environment. Those objective concern applet management by the mobile operator and are not relevant for this TOE
- Policies security objectives for the environment including:
  - Validation and Certification:
    - OE.BASIC-APPS-VALIDATION and OE.SECURE-APPS-CERTIFICATION: these OE is fulfilled for this TOE (The Cryptosmart-SIM applet is being certified)
    - OE.AID-MANAGEMENT: this OE is only relevant for Cryptosmart-SIM card delivered by IDEMIA. In this case Cryptosmart-SIM applet has been signed by the VA. Therefore this OE shall be fulfilled
  - Loading: these OE concern OTA applet management and are not relevant for this TOE
  - Keys: these OE concern mobile operator keys management and are not relevant for this TOE
- Platform security objectives for the environment concern application provider SD key management and is not relevant for this TOE
- GlobalPlatform security objectives for the environment concern security domain management and are not relevant for this TOE
- Applications security objectives for the environment:
  - OE.SHARE-CONTROL: the Cryptosmart-SIM applet does not use any Shareable interface.
- Java Card System Protection Profile - Open Configuration security objectives for the environment:
  - Those OE included in this ST. This allows post issuance applet installation as long as the security domain owner verifies these objectives[5]
- DESFire security objectives are not relevant for this TOE

The objectives for the environment of the composite TOE objectives concerning:

- The environment for pre-personalization;
- The environment for personalization;
- The host behavior;
- Usage constraints.

Therefore these objectives do not contradict themselves.

8.6 Compatibility of threats

The threats of the composite security target can be divided into threats corresponding to:

- The platform specific threats;
- Applet specific threats.

Platform specific threats include:

- (U)SIM threats:
  - T.UNAUTHORIZED_CARD_MNGT , T.PHYSICAL : these threats are independent to TOE threats

[5] following “Joint Interpretation Library -- Certification of "open" smart card products” : *“It is up to the risk manager to rely on the assurance of verification of OE1 and OE2 provided by the actor in charge of the deployment of these applications or to rely on the schema”*

  - T.INTEG-USER-DATA: this threat is analogous to T.TOE_DATA_CORRUPTION and does not contradict it
  - T.COM_EXPLOIT: this threat is analogous to T.INTERFACE_EAVESDROP and does not contradict it
  - T.UNAUTHORIZED_ACCESS is not relevant to the TOE as the Cryptosmart-SIM does not use any shareable interface
  - T.LIFE_CYCLE is relevant to the TOE and included in this ST
- Java Card System Protection Profile Open Configuration threats are independent to TOE threats
- DESFire threats are independent to TOE threats

8.7 COMPATIBILITY OF OSP

The platform has the following OSP:

- Basic and Secure Applications Policies OSP
  - OSP.BASIC-APPS-VALIDATION and OSP.SECURE-APPS-CERTIFICATION: these OSP are fulfilled for this TOE (The Cryptosmart-SIM applet is being certified)
  - OSP.AID-MANAGEMENT and OSP.SHARE-CONTROL: the Cryptosmart-SIM applet does not use any Shareable interface
- Loading Policies OSP concern OTA applet management and do not apply to the TOE
- Key Policies, Platform and GlobalPlatform OSP concern platform SD card management and do not apply to the TOE
- Java Card System Protection Profile - Open Configuration OSP:
  - OSP.VERIFICATION: this OSP is relevant to the TOE and included in this TOE
- DESFIRE OSPs concern DESFIRE and do not apply to the TOE

The OSP of the composite security target are independent and not contradictory to the relevant OSPs of the platform security target.

8.8 COMPATIBILITY OF ASSUMPTIONS

The platform has the following assumptions:

- Actors assumptions concern actors with no relation with current TOE. These assumptions are independent from TOE assumptions
- Java Card System Protection Profile - Open Configuration assumptions:
  - A.APPLET concerns post-issuance applet installation and is independent from TOE assumptions
  - A.VERIFICATION is relevant for current TOE and is included in this ST
- DESFire Assumptions are independent from TOE assumptions