ORGANIZATIONAL CARDS - S1

Electronic ID Application

Issue Date: 2019-07-01

TABLE OF CONTENTS

TABLES

FIGURES

1 OBJECT

1.1 VERSION HISTORY

Date	Version	Revision
2019-02-15	01	First Official Release
2019-07-01	02	Added §**Error! Reference source not found.**: Version History Correction §3.1.1: Swap Table 2 and Table 3

1.2 INTRODUCTION

This document describes the possible usages of the ORGANIZATIONAL CARDS Application, which is based on ID.me 1.6-i application (hereafter called ID.me). This document is service based: for each category of service the affiliated use cases are listed and described.

1.3 REFERENCES

All the documents referenced in this specification are listed in the following document:

REFERENCE	DOCUMENT
[SAC]	International Civil Aviation Organization, ICAO Doc 9303, Machine Readable Travel Documents – 7th edition, 2015
[GP 2.1.1]	Global Platform Card Specification Version 2.1.1 March 2003
[EN-419212-1]	Application Interface for smart cards used as Secure Signature Creation Devices — Part 1: Basic services, 10 January 2015
[EN-419212-2]	Application Interface for smart cards used as Secure Signature Creation Devices — Part 2: Additional services, 10 January 2015
[CEN]	CEN/ISSS WS/E-Sign Area K Reference: Application Interface for smart cards used as Secure Signature Creation Devices : CEN/TS 15480-2 - Part 2: European Citizen Card
[19794-2]	ISO/IEC 19794-2 (2005) – Finger minutiae data
[BSI TR-03110 part 1]	Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token v2.21 Part 1: eMRTDs with BAC/PACEv2 and EACv1
[BSI TR-03110 part 2]	Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token v2.21 Part 2: Protocols for electronic Identification, Authentication and trust Services (eIDAS)
[BSI TR-03110 part 3]	Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token v2.21 Part 3: Common Specifications
[BSI TR-03110 part 4]	Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token v2.21 Part 4: Applications and Document Profiles
[BSI TR-03110]	[BSI TR-03110 part 1] [BSI TR-03110 part 2] [BSI TR-03110 part 3] [BSI TR-03110 part 4]
[PRMD1123151V]	Avis relatif aux paramètres de courbes elliptiques définis par l'Etat français NOR: PRMD1123151V (Le 18 avril 2012)- ANSSI
[IAS-ECC]	Identification Authentication Signature - European Citizen Card Technical Specifications Revision: 1.0.1
[7816-4]	ISO/IEC 7816-4 (2003) – Part 4: Inter-industry commands for interchange.
[7816-3]	ISO/IEC 7816-3 (2004) - Part 3: Electronic signals and transmission protocols
[X9.62]	ANSI X9.62 (2005) - Public Key Cryptography for the Financial Services Industry – The Elliptic Curve Digital Signature Algorithm (ECDSA)
[PKCS1]	PKCS #1 v2.1 RSA Cryptography Standard

1.4 ABBREVIATIONS

ABBREVIATION	
ADF	Application Dedicated File
AID	Application Identifier
AT	Authentication Template
AMB	Access Mode Byte
APDU	Application Protocol Data Unit
BER	Basic Encoding Rules
CA	Certification Authority
CAR	Certification Authority Reference
CHR	Certificate Holder Reference
CAv2	Chip Authentication Version 2
CCT	Cryptographic Checksum Template
CHAT	Certificate Holder Authorization Template
CRT	Control Reference Template
CVCA	Country Verifying certification Authority
CSE	Current Security Environment
CT	Confidentiality Template
DES	Data Encryption Standard
DF	Dedicated File
DOCP	Data Object Control Parameters
DST	Digital Signature Template
EAC	Extended Access Control
EACv2	Extended Access Control version 2
EF	Elementary File
FCI	File Control Information
FCP	File Control Parameters
HT	Hash Template
IAS	Identification, Authentication and electronic Signature
ICC	Integrated Circuit Card
IFD	Interface Device
LJFS	Light JavaCard File System
MAC	Message Authentication Code
MF	Master File
MSE	Manage Security Environment

PACE	Password Authenticated Connection Establishment
PIN	Personal Identification Number
PK – DH	Public key – Diffie Hellmann (asymmetric key base algorithm)
PSO	Perform Security Operation
RFU	Reserved for Future Use
Root	The applet instance having the default selection privilege
RSA	Rivest Shamir Adleman
SDO	Security Data Object
SCB	Security Condition Byte
SE	Security Environment
SEID	Security Environment Identifier byte
SSE	Static Security Environment
SSESP	Static Security Environment for Security Policy
SK	Secret key – Symmetric key based algorithm
SM	Secure Messaging
TAv2	Terminal Authentication Version 2
TLV	Tag Length Value

1.5 CONVENTIONS

- **Hexadecimal Notation**

The values expressed in hexadecimal are between simple hooks (‘ ’). For example, the decimal value 27509 is noted ‘6B 75’ in hexadecimal.

- **Decimal Notation**

The decimal values are expressed in rough format. For example the hexadecimal-noted value ‘08’ is noted 8 in decimal.

- **Binary Notation**

The binary values are followed by a “b” in lower case. For example, the value 8 is noted 00001000b in binary.

- **Various Notations**

The free or not fixed values are noted ‘XX … XX’ (several bytes) or ‘XX’ (only one byte). The symbol “ || ” is used to represent the concatenation of two elements.

M/O - M for Mandatory and O for Optional.

2 NOTES

The ORGANIZATIONAL CARDS application has been initialized according to the ORGANIZATIONAL CARDS electronic profile, so only some of the ORGANIZATIONAL CARDS features can actually be used.

ORGANIZATIONAL CARDS features (and related commands) that are not comprised in the ORGANIZATIONAL CARDS electronic profile are:

1. Biometric authentication: BIO (fingerprint, iris, facial)
2. Device authentication with privacy protection
3. Extended Access Control protocol Version 2

The use cases descriptions in the present document complies with the following principles:

- A use case is described with the successive necessary steps and completed with the related ADPU commands that are required. Only nominal executions are discussed.
- Some cases allow multiple different scenarios of APDU commands. Only cases with minimum transmissions are developed here
- Commands that do not require a secure channel are detailed with a ‘00’ CLA. Secure Messaging establishment is described in chapter 8 of [ID.ME-SPEC] and is not further described in this document.
- Correct command execution may necessitate access rights conditions fulfilment. The access rights management is described in chapter 4 of [LJFS-SPEC] and is not further described in this document.

3 FILE SYSTEM

3.1 FILE MANIPULATION

3.1.1 SELECT FILE

A successful Select File sets a current file within a logical channel. Subsequent command may implicitly refer to the current file through that logical channel.

Selecting a DF (which may be the MF) sets it as current DF. After such a selection, an implicit current EF may be referred to through that logical channel.

Selecting an EF sets a pair of current files: the EF and its parent file.

After the answer to reset, the MF is implicitly selected through the basic logical channel, unless specified differently in the historical bytes or in the initial date string.

The following conditions shall apply to each open logical channel.

Unless otherwise specified, the correct execution of the command modifies the security status according to the following rules:

- When the current EF is changed, or when there is no current EF the security status if any specific to a former current EF is lost.
- When the current DF is a descendant of or identical to the former current DF, the security status specific to the former current DF is maintained.
- When the current DF is neither a descendant of nor identical to the former current DF the security status specific to the former current DF is lost. The security status common to all common ancestors of the previous and new current DF is maintained.

The following rules shall be applied:

- ➢ After a successful selection of a DF there is no selected EF.
- ➢ After a successful selection of the Root or an ADF, the associated application is selected and becomes the current application; there is no selected EF.

If the selection is aborted due to an error, the current files selection is unchanged.

When selecting and EF, the current DF becomes the parent DF of the selected EF.

Following the Root or an ADF selection, the current DF is the Root or the ADF, and there is no current EF.

Upon IFD request, the command may return file FCP.

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'A4'
P1	See below
P2	See below
Lc field	Absent or length of command data field '02' – to pass a FID 'xx' – to pass DF name or relative path (shall be modulo 2)
Data field	FID or DF name.
Le field	Variable

Table 1: SELECT FILE Command

Note that this command is always sent in clear text. If a secure channel session is in progress, a clear text select ADF does not break the SM session.

The P1 and P2 bytes of the command are coded as follows:

B8	B7	B6	B5	B4	B3	B2	B1	MEANING	COMMAND DATA FIELD
0	0	0	0	0	0	X	x	Selection by file identifier	
				0	0	0	0	Select Root	Root identifier (0x3F00)
				0	0	0	1	Select child DF	DF identifier
				0	0	1	0	Select EF under the current DF	EF identifier
				0	0	1	1	Select parent DF of the current DF. Upper limit = ADF or Root	None
				0	1	0	0	Select by DF name	AID
				1	0	0	1	Selection by path from the current DF	Path without the current DF identifier

Table 2: SELECT FILE Command P1 byte coding

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
0	0	0	0	0	1	0	0	Return FCP template, mandatory use of FCP tag and length
				1	1			No data in response field

Table 3: SELECT FILE Command P2 byte coding

RESPONSE PARAMETER	MEANING
Data Field	Absent or FCP
SW1 - SW2	'6283' - Warning Selected file deactivated '6285' - The Selected file is in Terminate state '6A82' - File or application not found '6A86' - Incorrect parameters P1-P2 '6A87' - Lc inconsistent with parameters P1-P2

Table 4: SELECT FILE command response

3.1.2 UPDATE BINARY

Prerequisite: selection of the EF to write, or selection of the location of the file to write if the SFI is provided (see 3.1.1).

File contents write is performed through the dedicated command:

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'D6'
P1	See below
P2	See below
Lc field	Length of data field
Data field	Bytes to write
Le field	Number of bytes to read

Table 5: UPDATE BINARY command

P1								P2							
B8	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B8**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**
0	Offset in the currently selected file over 15 bits '00' ≤ Offset ≤ '7FFF'														
1	0	0	Short File Identifier 1 ≤ SFI ≤ 30					Offset in the file over 8 bits							

Table 6: P1/P2 value for UPDATE BINARY command

RESPONSE PARAMETER	MEANING
Data Field	None
SW1 - SW2	'6700' - Wrong length; no further indication '6981' - Command incompatible with file structure '6982' - Security status not satisfied '6986' - Command not allowed (no current EF) '6A82' - File not found '6B00' - Wrong parameters P1-P2 (offset + length makes update outside the file) ‘6A84’ – not enough resources to perform operation.

Table 7: UPDATE BINARY command response

Note that it is possible to write either the current file, or another file located in the same EF using its Short File Identifier.

3.1.3 READ BINARY

Prerequisite: selection of the EF to read, or selection of the location of the file to read if the SFI is provided (see 3.1.1).

File contents read is performed through the dedicated command:

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'B0'
P1	See below
P2	See below
Le field	Number of bytes to read

Table 8: READ BINARY command

P1								P2							
B8	B7	B6	B5	B4	B3	B2	B1	B8	B7	B6	B5	B4	B3	B2	B1
0	Offset in the currently selected file over 15 bits '00' ≤ Offset ≤ '7FFF'														
1	0	0	Short File Identifier 1 ≤ SFI ≤ 30					Offset in the file over 8 bits							

Table 9: P1/P2 value for READ BINARY command

RESPONSE PARAMETER	MEANING
Data Field	Data read
SW1 - SW2	‘6700’ - In case of Extended Length, this SW is returned if Le is greater than the APDU buffer size. '6282' - End of file reached before reading ‘Le’ bytes '6981' - Command incompatible with file structure '6982' - Security status not satisfied '6986' - Command not allowed (no current EF) '6A82' - File not found '6B00' - Wrong parameters P1-P2 : Offset + length is beyond the end of file

Table 10: Read Binary command response

Note that it is possible to read either the current file, or another file located in the same EF using its Short File Identifier.

It must be noted that the particular case of requesting 0x100 bytes through an extended length command will result in retuning 0xE7 bytes.

3.2 SDO MANIPULATION

3.2.1 SDO Identification

SDO are coded according to BER TLV rule. The TAG field constitutes the unique identifier of the SDO in its area.

SDO TAG is structured as follows:

```
'BF' || 1b || ObjClass (7 bits) || 0b || 00b || ObjRef (5 bits) ≡ 3 bytes
```

Where:

- 'BF' means context-specific class, constructed data object in the BER TLV encoding rules.
- 1b means that another byte is following the current one.
- ObjClass is one of the authorized classes listed in Table 11.
- 0b means the current byte is the last one of the BER TLV coding.
- ObjRef (1 ≤ ObjRef ≤ 31) codes the unique reference identifying the object.
- Note: The value 00000b for ObjRef shall not be used.

3.2.2 SDO Classes

SDO CLASS	IDENTIFIER VALUE	TAG MIDDLE BYTE CODING
User authentication: PIN / Password	'01'	'81'
Biometric authentication: BIO	'02'	'82'
3DES Symmetric key set of 2 keys	'0A'	'8A'
AES Symmetric key set of 2 keys	'0B'	'8B'
Asymmetric keys RSA – Private portion	'10'	'90'
Asymmetric keys RSA – Public portion	'20'	'A0'
Asymmetric keys EC – Private portion	'12'	'92'
Asymmetric keys EC – Public portion	'22'	'A2'
Security environment	'7B'	'FB'

Table 11: Security data objects (SDO) class identifiers

3.2.3 SDO Data Retrieval

It is possible to retrieve a SDO contents (header and contents) through the GET DATA command.

The GET DATA command is described below:

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'CB'
P1	'3F'
P2	'FF'
Lc field	Length of command data field
Data field	Extended header list: '4D' L_{4D} {References of the data to retrieve} for classic get data, empty for "get security status".
Le field	Variable

Table 12: GET DATA command description

RESPONSE PARAMETER	MEANING
Data Field	Data requested
SW1 - SW2	'6982' - Security status not satisfied '6A86' - Incorrect parameters P1-P2 '6A80' - Incorrect parameters in the command data field '6A88' - Referenced data or reference data not found '6A81' - Function not supported – data are not retrievable

Table 13: SDO GET DATA command response

TAG			LENGTH	VALUE
'4D'			'04'	Extended Header List
	'70'		'02'	Inter industry template for further objects
		'C4'	'80'	Security status

Table 14: Tag 4D for security status retrieval

3.2.4 SDO Update

Prerequisite: DF selection (see 3.1.1)

The SDO update is performed through the PUT DATA command. A SDO creation is conducted through a PUT DATA command. SDO creation is performed in the current directory.

PUT DATA command description:

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'DB'
P1	'3F'
P2	'FF'
Lc field	Length of command data field
Data field	SDO contents, BER TLV formatted.
Le field	None

Table 15: PUT DATA command description

RESPONSE PARAMETER	MEANING
Data Field	None
SW1 - SW2	'6700' - Wrong length; no further indication '6982' - Security status in incoming data not satisfied '6A86' - Incorrect parameters P1-P2 '6A80' - Incorrect parameters in the command data field '6A88' - Reference data not found '6985' - Conditions of use not satisfied '6A81' - Function not supported – data cannot be put

Table 16: SDO PUT DATA command response

3.3 SECURITY MANAGEMENT

3.3.1 Security environments

The Security Environment (SE) data object is used for referencing SDO, cryptographic algorithms, modes of operation and any additional data needed for secure messaging or security operations. All those different concepts are listed into a SE with Control Reference Template (CRT).

3.3.1.1 STATIC SECURITY ENVIRONMENTS FOR SECURITY POLICY (SSESP)

The Static Security Environments for Security policies are involved into the security architecture. The SE is referenced into the SCB of an access rule (see 3.3.3). For that reason, and for performance issues, SE sets could not be created or updated after card issuance. Therefore, SE sets are created during card personalization and cannot be changed afterwards.

The card may contain several SSESP sets, each of them containing at most 14 SSESP numbered from 1 to 14. A SSESP set can exist under the Root or under an ADF or under a DF.

As the SE#1 is the "default SE", that is always restored in the current SE.

If the SSESP references multiple keys to protect access to an object, the algorithm present in the associated CRT shall be unique for all Key references. This means tag '83' or '84' can be present many time in a CRT template while tag '80' is present only once.

3.3.1.2 CURRENT SECURITY ENVIRONMENT (CSE)

The Current Security Environment helps clarifying any required additional data needed for the execution of some commands. For updating the content of the CSE, the MSE SET functionality is used, see Table 11 for details.

In order to avoid collision, and to ensure card interoperability, the following rules shall apply when addressing the CSE:

- After an application selection either at card reset or using the SELECT command, the CSE is cleared.
- At each moment, at most, there should be only one Control reference template of each type (AT, KAT, HT, CCT, DST or CT).
- In case of failure updating the CSE content, the previous CSE content is kept and is still valid.

3.3.2 Manage Security Environment (MSE Set)

This command sets the security environment with requested algorithms and SDOs. Put more simply, it selects the keys and algorithms used for a subsequent cryptographic operation.

The current security environment contains all the received information related to key selection.

This operation is performed by supplying to the card control references templates (CRTs) that contain all the necessary information to update the current security environment.

The MSE Set Command is described below:

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'22'
P1	See below
P2	See below
Lc field	Length of command data field
Data field	See below
Le field	None

Table 17: MSE SET command description

TYPE	USAGE	HEADER				DATA FIELD
		P1	**P2**	**Tag**	**Length**	**Value**
Authentication Template (CRT AT)	Mutual auth. 3DES	'C1'	'A4'	'80'	'01'	'0C' : symmetric mutual authentication based on 3DES with SHA-1 '8C' : symmetric mutual authentication based on 3DES with SHA-256
				'83'	'01'	Symmetric key set reference
	Mutual auth. AES	'C1'	'A4'	'80'	'04'	'FF40104C': symmetric mutual authentication based on AES with SHA-256
				'83'	'01'	Symmetric key set reference
	Role auth. 3DES	'81'	'A4'	'80'	'01'	'1C' : Role authentication based on 3DES
				'83'	'01'	Symmetric key set reference
	Role auth. AES	'81'	'A4'	'80'	'04'	'FFA01200' : Role authentication based on AES
				'83'	'01'	Symmetric key set reference
	Role auth. RSA	'81'	'A4'	'80'	'01'/'04'	'1E': Role authentication based on RSA SHA-1 '9E': Role authentication based on RSA SHA-256 'FFA14000' : Role authentication based on RSA SHA-1[2] 'FFA34000' : Role authentication based on RSA SHA-224[2] 'FFA44000' : Role authentication based on RSA SHA-256[2] 'FFA54000' : Role authentication based on RSA SHA-384[2] 'FFA64000' : Role authentication based on RSA SHA-512[2]
				'83'	'0C'	Private key reference
	Mutual auth. SAC[1]	'C1'	'A4'	'80'	Var.	Object Identifiers of the SAC protocol to use (see [[SAC]])
				'83'	'01'	Reference of the password used for SAC authentication : 01: contents of EF 4001 02: contents of EF 4002 1X: global pin (id 1X)
				'84'	'01'	Reference of the domain parameters to be used for authentication
	Client/Server authentication	'41'	'A4'	'80'	'01'	'02': IFC/ICC authentication RSA PKCS#1 V1.5 '03': IFC/ICC authentication RSA PKCS#1 PSS -SHA-1 '04': IFC/ICC authentication ECC -SHA-1 with data formatting
				'84'	'01'	Asymmetric private key set reference
	Device auth. w/ privacy prot. (phase 1)[2]	'41'	'A4'	'80'	Var.	Algorithm identifier for Device Authentication with privacy protection as specified in 0. [#]
				'83'	'01'	Reference of the domain parameters to be used for key agreement.
	Device auth. w/ privacy prot. (phase 3)[2]	'81'	'A4'	'83'	'0C'	Name of the authentication public key to be used for external authentication.

	Device auth. w/ privacy prot. (phase 5)[2]	'41'	'A4'	'80'	Var.	Algorithm identifier for Device Authentication with privacy protection as specified in 0. [2]
				'84'	'01'	Reference of the authentication private key to be used for internal authentication.
	Terminal Authentication version 2[3] (cert. ver.)	'81'	'B6'	'83'	Var.	Name of the authentication public key to be used for certificate verification.
	Terminal Authentication version 2[3] (ext. auth.)	'81'	'A4'	'80'	Var.	Algorithm identifier for Terminal Authentication version 2 as specified in 0
				'83'	Var.	Name of the authentication public key to be used for external authentication.
				'91'	Var.	Compressed ephemeral public of the terminal.
	Chip Authentication version 2[3]	'41'	'A4'	'80'	Var.	Algorithm identifier for Chip Authentication version 2 as specified in 0
				'84'	'01'	Reference of the private key to be used for authentication.
Confidentiality template (CRT CT)	Message decryption	'41'	'B8'	'80'	'01'or'04'	'1A': Data decryption algorithm reference – RSA PKCS#1 V1.5 '2A': Data decryption algorithm reference – RSA PKCS#1 V2.1 : OAEP 'FF300400': Key Agreement Algorithm reference - ECDH
				'84'	'01'	Mutual authentication asymmetric key (private portion)reference
Digital Signature Template (CRT DST)	Data Signature	'41'	'B6'	'80'	Var.	'12': algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-1 '32': algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-224 '42': algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-256 '52': algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-384 '62': algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-512 '15': algorithm identifier for signature using RSA PKCS#1 v2.1 - SHA-1 '35': algorithm identifier for signature using RSA PKCS#1 v2.1 - SHA-224 '45': algorithm identifier for signature using RSA PKCS#1 v2.1 - SHA-256 '55': algorithm identifier for signature using RSA PKCS#1 v2.1 - SHA-384 '65': algorithm identifier for signature using RSA PKCS#1 v2.1 - SHA-512 'FF110800': algorithm identifier for signature using ECDSA - SHA-1 'FF130800': algorithm identifier for signature using ECDSA - SHA-224 'FF140800': algorithm identifier for signature using ECDSA - SHA-256 'FF150800': algorithm identifier for signature using ECDSA - SHA-384 'FF160800': algorithm identifier for signature using ECDSA - SHA-512
				'84'	'01'	Asymmetric key (private portion) reference

	Certificate verification	'81'	'B6'	'80'	'01'	'11': algorithm for signature using ISO 9796-2 scheme 1 - SHA-1 '31': algorithm for signature using ISO 9796-2 scheme 1 - SHA-224 '41': algorithm for signature using ISO 9796-2 scheme 1 - SHA-256 '51': algorithm for signature using ISO 9796-2 scheme 1 - SHA-384 '61': algorithm for signature using ISO 9796-2 scheme 1 - SHA-512
				'83'	Var.	Asymmetric key (public portion) reference (length = '01') CHR reference (length = '0C')
Hash template (CRT HT)		'41'	'AA'	'80'	'01'	'10': algorithm identifier for SHA-1 '30': algorithm identifier for SHA-224 '40': algorithm identifier for SHA-256 '50': algorithm identifier for SHA-384 '60': algorithm identifier for SHA-512

[1]: Reference for the MSE Set command used for SAC can be found in [[SAC]].
[2]: Only applies to the IAS PKI configuration.
[3]: Only applies to the EAC PKI configuration.

Table 18: CRT description for MSE Set Command

3.3.3 Security Attributes

Security attribute referencing the compact format for contact or contactless communication link is coded as follows:

```
'8C' or '9C' || L8C or 9C|| AMB || SCB1 || SCB2 || … || SCBn
```

There are as many SCB as there are bits set to 1 in an AMB (from bits 7 to 1). In case of a bit set to 0, as there is no SCB the access rule linked to the command is never. For the same reason, a RFU bit is always set to 0.

Examples:

- AMB = '81' = 10000001b there is one SCB following the AMB.
- AMB = '47' = 01000111b there are four SCB following the AMB.

There might be more than one AMB + SCB combination in the '8C' or '9C' (nested with the 'A1' Data object encoding the access conditions depending on the access medium contact/contactless) data object.

3.3.3.1 AMB CODING

The following tables show the AMB respectively for DF and EF.

In case of a DF all the commands apply to the DF itself.

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
0	-	-	-	-	-	-	-	Bits 7 to 1 according to this table
	1	-	-	-	-	-	-	DELETE FILE (DF ITSELF)
	-	1	-	-	-	-	-	TERMINATE FILE
	-	-	1	-	-	-	-	ACTIVATE FILE
	-	-	-	1	-	-	-	DEACTIVATE FILE
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	CREATE FILE EF (EF CREATION)
	-	-	-	-	-	-	0	RFU

Table 19: Access mode byte for DFs

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
0	-	-	-	-	-	-	-	Bits 7 to 1 according to this table
	1	-	-	-	-	-	-	DELETE FILE
	-	1	-	-	-	-	-	TERMINATE EF
	-	-	1	-	-	-	-	ACTIVATE FILE
	-	-	-	1	-	-	-	DEACTIVATE FILE
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	UPDATE BINARY
	-	-	-	-	-	-	1	READ BINARY

Table 20: Access mode byte for EFs

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	1	-	-	-	-	-	-	COMPARE BINARY
	-	0	-	-	-	-	-	RFU
	-	-	0	-	-	-	-	RFU
	-	-	-	0	-	-	-	RFU
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	UPDATE BINARY, ERASE BINARY
	-	-	-	-	-	-	1	READ BINARY

Table 21: Complementary access mode bytes for EFs

The following tables show the AMB coding according to the SDO type. The bit B8 is always set to 1.

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	1	-	-	-	-	-	-	CHANGE REFERENCE DATA
	-	1	-	-	-	-	-	VERIFY
	-	-	1	-	-	-	-	RESET RETRY COUNTER
	-	-	-	0	-	-	-	RFU
	-	-	-	-	1	-	-	SAC AUTHENTICATION
	-	-	-	-	-	1	-	PUT DATA
	-	-	-	-	-	-	1	GET DATA

Table 22: User authentication (Pin/Password)

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	0	-	-	-	-	-	-	RFU
	-	1	-	-	-	-	-	Verify
	-	-	1	-	-	-	-	Reset Retry Counter
	-	-	-	0	-	-	-	RFU
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	Put Data
	-	-	-	-	-	-	1	Get Data

Table 23: User authentication (Biometry object)

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	1	-	-	-	-	-	-	PSO COMPUTE DIGITAL SIGNATURE
	-	1	-	-	-	-	-	INTERNAL AUTHENTICATION
	-	-	1	-	-	-	-	PSO DECIPHER
	-	-	-	1	-	-	-	GENERATE ASYMMETRIC KEY PAIR
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	PUT DATA
	-	-	-	-	-	-	1	GET DATA

Table 24: RSA Key, private key part

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	1	-	-	-	-	-	-	PSO VERIFY CERTIFICATE
	-	1	-	-	-	-	-	EXTERNAL AUTHENTICATE
	-	-	0	-	-	-	-	RFU
	-	-	-	1	-	-	-	GENERATE ASYMMETRIC KEY PAIR
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	PUT DATA
	-	-	-	-	-	-	1	GET DATA

Table 25: RSA Key, public key part

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	1	-	-	-	-	-	-	PSO VERIFY CERTIFICATE
	-	1	-	-	-	-	-	EXTERNAL AUTHENTICATE
	-	-	0	-	-	-	-	RFU
	-	-	-	1	-	-	-	GENERATE ASYMMETRIC KEY PAIR
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	PUT DATA
	-	-	-	-	-	-	1	GET DATA

Table 26: EC Key, public key part

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	1	-	-	-	-	-	-	PSO COMPUTE DIGITAL SIGNATURE
	-	1	-	-	-	-	-	INTERNAL AUTHENTICATION
	-	-	1	-	-	-	-	PSO DECIPHER
	-	-	-	1	-	-	-	GENERATE ASYMMETRIC KEY PAIR
	-	-	-	-	1	-	-	PERFORM SAC AUTHENTICATION
	-	-	-	-	-	1	-	PUT DATA
	-	-	-	-	-	-	1	GET DATA

Table 27: EC Key, private key part

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	0	-	-	-	-	-	-	RFU
	-	1	-	-	-	-	-	EXTERNAL AUTHENTICATE FOR ROLE
	-	-	0	-	-	-	-	RFU
	-	-	-	1	-	-	-	MUTUAL AUTHENTICATE
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1	-	PUT DATA
	-	-	-	-	-	-	1	GET DATA

Table 28: Symmetric key set

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	0	-	-	-	-	-	-	RFU
	-	0	-	-	-	-	-	RFU
	-	-	0	-	-	-	-	RFU
	-	-	-	0	-	-	-	RFU
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	0	-	PUT DATA (not supported)
	-	-	-	-	-	-	1	GET DATA

Table 29: Security Environment

3.3.3.2 SCB CODING

The Security Condition Byte coding is described as follow:

SECURITY CONDITIONS				SECURITY ENVIRONMENT				MEANING
B8	B7	B6	B5	B4	B3	B2	B1	
0	0	0	0	0	0	0	0	No condition - i.e. use of data object is free
1	1	1	1	1	1	1	1	Never
-	-	-	-	0	0	0	0	No reference to a security environment - Forbidden
-	-	-	-	Not all equal				Static Security Environment for Security Policies (SSESP) identifier (SEID nibble) from one to fourteen
-	-	-	-	1	1	1	1	RFU - Reserved for future use
0	-	-	-	-	-	-	-	At least one condition (i.e., b7 OR b6 OR b5)
1	-	-	-	-	-	-	-	All conditions (i.e., b7 AND b6 AND b5)
-	1	-	-	-	-	-	-	Secure messaging (and device authentication)
-	-	1	-	-	-	-	-	External authentication
-	-	-	1	-	-	-	-	User authentication (e.g., PIN/password)

Table 30: Security condition byte coding

Bits B8 to B5 indicate the security conditions applying on the SDO/File. Bits B4 to B1 identify a security environment referencing other SDO to protect the current SDO. The mechanisms defined in the security environment shall be used according to the indications in bits B7 to B5 for command protection and / or external/mutual authentication and / or user authentication.

For instance:

- If bit B8 is set to 1, then all the conditions set in bits B7 to B5 shall be satisfied.
- If bit B8 is set to 0, then at least one of the conditions set in bits B7 to B5 shall be satisfied.
- If bit B7 is set to 1, then the control reference template of the security environment identified in bits B4 to B1, indicates that the secure messaging shall apply to the command and to the response data field.

When a security condition cannot be resolved, the card returns SW12 = '6982' (*e.g.*, the referenced security condition is missing in the SE).

4 AUTHENTICATION

4.1 USER AUTHENTICATION

4.1.1 User authentication with passphrase or biometry

This functionality allows the user to authenticate through a password or biometric data. The command used for this is the VERIFY command.

COMMAND	
CLA	ISO
INS	'20' (for PIN) or '21' (for biometry)
P1	'00'
P2	PIN or Biometry Reference
Lc	Variable
Data In	Verification data
Le	None

Table 31: Verify Command

RESPONSE		
Data Out	None	
SW1 SW2	'6A86'	P1 ≠ '00' and P1 ≠ 'FF'
	'6700'	PIN length is out of valid boundaries
	'6A88'	Referenced PIN not found
	'6982'	Security Status not satisfied
	'6983'	Referenced PIN not successfully verified AND no subsequent tries are allowed (remaining tries counter reached 0)
	'6984'	Referenced PIN usage counter reached 0
	'6300'	No retry limit: user authentication failed
	'63Cx'	With retry limit, having x = remaining tries: user authentication failed
	'9000'	user authentication successful

Table 32: VERIFY command response encoding for PIN verification

RESPONSE		
Data Out	None	
SW1 SW2	‘6A86’	P1 ≠ ‘00’ and P1 ≠ ‘FF’
	‘6700’	Wrong data length
	‘6A80’	Invalid biometry data format
	‘6A88’	Referenced biometry not found
	‘6982’	Security Status not satisfied
	‘6983’	Referenced Biometry not successfully verified, and no subsequent tries are allowed (remaining tries counter reached 0)
	‘6984’	Referenced Biometry usage counter reached 0
	‘6300’	No retry limit: user authentication failed (Only for Biometry)
	‘6363’	More biometric data needed
	‘63Cx’	With retry limit, having x = remaining tries: user authentication failed
	‘9000’	user authentication successful

Table 33: VERIFY response encoding for biometry verification

4.1.2 Conditional usage and security of PIN in contact mode

The PIN/Biometry to verify must not be blocked.

If the referenced PIN/Biometry remaining tries counter has reached 0, or the usage counter has reached 0, no verification is done.

If the referenced PIN/Biometry is successfully verified, the remaining tries counter returns to its maximum (if the tries counter is not infinite) and the validation flag is set to "true".

A successful verification decrement the PIN/Biometry usage counter (if applicable).

When calling VERIFY to check the PIN/Biometry status (i.e. When P1=”00” and Lc=”00”):

- The card always answers, despite the value of the usage counter (i.e. the command is accepted and treated including when the usage counter reached zero).
- The access conditions set for the VERIFY command do apply.
- If the PIN/Biometry is already verified, the card returns ‘9000’, otherwise ‘63Cx’ or ‘6300’ is returned.

When calling VERIFY to set the PIN/Biometry status ‘unverified’ (Devalidate) (i.e. When P1=”FF” and Lc=”00”):

- The access conditions set for the VERIFY command do apply.

- The usage counter is not modified and is not decremented after successful verify command (with P1="FF" and Lc="00").
- The Retry counter is not reset after successful Verify (Devalidate) command (i.e. When P1="FF" and Lc="00").
- If the PIN/Biometry is already 'unverified', the card returns '9000'

Only 2 basic states are available for the PIN in contact mode as Figure 1 depicts.

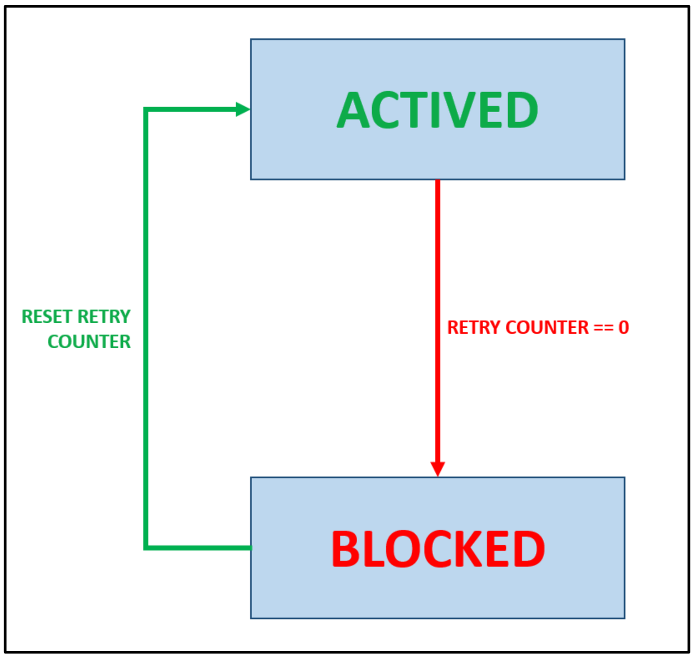


Figure 1: PIN life-cycle diagram in contact mode

4.1.3 Conditional usage and security of BIOMETRY

For Biometry the same rules of PIN in contact mode are valid, so refer to 4.1.2 .

4.1.4 PIN/Biometry object status reset

This use case allows resetting the verification flag of the object. The command used to perform this operation is described below:

COMMAND	
CLA	ISO
INS	‘20’ (for PIN) or ‘21’ (for biometry)
P1	‘FF’
P2	PIN or Biometry SDO Identifier
Lc	00
Data In	None
Le	None

Table 34: Status reset command

RESPONSE		
Data Out	None	
SW1 SW2	‘6A86’	P1 ≠ ‘00’ and P1 ≠ ‘FF’
	‘6700’	PIN length is out of valid boundaries
	‘6A88’	Referenced PIN not found
	‘9000’	PIN status is set to ‘unverified’

Table 35: VERIFY response encoding for PIN devalidation

RESPONSE		
Data Out	None	
SW1 SW2	‘6A86’	P1 ≠ ‘00’ and P1 ≠ ‘FF’
	‘6700’	Data length is not null
	‘6A88’	Referenced biometry not found
	‘9000’	Biometry status is set to ‘unverified’

Table 36: VERIFY response encoding for biometry devalidation

4.1.5 PIN/Biometry object status retrieval

The status of a PIN/Biometry object is the state of the verification flag, and number of remaining tries. These elements can be retrieved through this command:

COMMAND	
CLA	ISO
INS	'20' (for PIN) or '21' (for biometry)
P1	'00'
P2	PIN or Biometry SDO Identifier
Lc	00
Data In	None
Le	None

Table 37: Status retrieval command

RESPONSE		
Data Out	None	
SW1 SW2	'6A86'	P1 ≠ '00' and P1 ≠ 'FF'
	'6A88'	Referenced PIN not found
	'6300'	No retry limit, PIN is not verified
	'63Cx'	With retry counter, having x = remaining tries, PIN is not verified
	'9000'	The PIN has already been 'verified'

Table 38: VERIFY response encoding for PIN status retrieval

RESPONSE		
Data Out	None	
SW1 SW2	'6A86'	P1 ≠ '00' and P1 ≠ 'FF'
	'6A88'	Referenced biometry not found
	'6300'	No retry limit, biometry is not verified
	'6Cxx'	With retry counter, having x = remaining tries, biometry is not verified
	'9000'	The biometry has already been 'verified'

Table 39: VERIFY response encoding for biometry status retrieval

4.1.6 Reset retry counter

The user can reset the retry counter of a PIN/Biometry object through the reset retry counter command, described below. The user has the possibility to input a new reference data in this operation. If the PIN/Biometry is blocked, the RESET RETRY COUNTER command unblocks the PIN.

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'2C' for PIN, '2D' for BIO
P1	'02' (New reference data) or '03' (No data field)
P2	SDO identifier
Lc field	Absent if P1 = '03', Length of command data field if P1 = '02'
Data field	Absent if P1 = '03', new reference data if P1 = '02'
Le field	None

Table 40: RESET RETRY COUNTER command

RESPONSE PARAMETER	MEANING
Data Field	None
SW1 - SW2	'6A86' - P1 ≠ '02' or P1 ≠ '03' '6700' - The length of the new reference data doesn't match with the length of the PIN reference container length (P1 = '02') or Lc ≠ '00' (P1 = '03'). '6A88' - Referenced PIN not found '6984' - Reference data not usable – Usage counter of referenced PIN raised 0.

Table 41: RESET RETRY COUNTER command response for PIN RESET

RESPONSE PARAMETER	MEANING
Data Field	None
SW1 - SW2	'6A86' - P1 ≠ '03' '6700' - Lc ≠ '00' '6A88' - Referenced Bio not found '6984' - Reference data not usable – Usage counter of referenced Bio raised 0.

Table 42: RESET RETRY COUNTER command response for biometry RESET

4.1.7 Change reference data

It is possible to change the reference data of a PIN object through the CHANGE REFERENCE DATA command. Note that to change the reference data, the original password must be presented.

The command is described below.

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'24'
P1	'00'
P2	SDO identifier
Lc field	Length of command data field
Data field	Current Password || New reference data (i.e. new PIN or password)
Le field	None

Table 43: CHANGE REFERENCE DATA command

RESPONSE PARAMETER	MEANING
Data Field	None
SW1 - SW2	'6A86' - P1 ≠ '00' '6700' - PIN length is out of valid boundaries [2*Lmin - 2*Lmax]" '6A88' - Referenced PIN not found '63Cx' - Referenced PIN not successfully verified AND subsequent tries are allowed (error counter not null), x = remaining tries allowed '6300' - No retry limit: Change reference data command failed '6983' - Referenced PIN not successfully verified AND no subsequent tries are allowed (remaining tries counter reached 0) '6984' - Referenced PIN usage counter reached 0

Table 44: CHANGE REFERENCE DATA command response

4.2 SYMMETRIC AUTHENTICATION

4.2.1 Mutual symmetric authentication

Mutual symmetric authentication allows both terminal and card to authenticate, and to engage a secure messaging session using computed ephemeral keys. The whole process is described hereby.

Prerequisite: select the correct Key Set using Manage Security Environment (MSE Set). The CRT to be used is:

USAGE	HEADER		DATA FIELD		
	P1	P2	Tag	Length	value
Mutual auth.	'C1'	'A4'	'80'	Var.	Symmetric mutual authentication algorithm
			'83'	'01'	Symmetric key set reference

Table 45: CRT for symmetric mutual authentication

Smart card ⇔ IFD mutual authentication and K.ICC + K.IFD calculation		
CARD ACCEPTING DEVICE (IFD)		**Smart card**
.../...		.../...
READ BINARY SN.ICC Reading of the card serial number (for key diversification and for cryptogram calculation).	→ ←	Send SN.ICC Card Serial Number in the context of the current application
Retrieval of the Card Key Set for device authentication		
GET CHALLENGE Get RND.ICC	→ ←	Generate and return RND.ICC (8 bytes-long)
Generate RND.IFD (8 bytes) and K.IFD (32 bytes) S = RND.IFD || SN.IFD || RND.ICC || SN.ICC || K.IFD Data = Authentication token computed over S MUTUAL AUTHENTICATE (Data || MAC) Verify the MAC Decipher Data Verify RND.IFD & SN.IFD, restore K.ICC	→ ←	Verify the MAC Decipher Data Verify RND.ICC & SN.ICC, restore K.IFD, Generate K.ICC S' = RND.ICC || SN.ICC || RND.IFD || SN. IFD || K.ICC Data = Authentication token computed over S' Returns (Data)
.../...		.../...

Figure 2: Mutual authentication process

STEP 1: READ SERIAL NUMBER

The READ BINARY command is described in (READ BINARY). The EF SN.ICC must be read.

STEP 2: GET CHALLENGE

The GET CHALLENGE command requires a random challenge from the card. The challenge is kept until next command is received.

COMMAND	
CLA	ISO
INS	'84'
P1	'00'
P2	'00'
Lc	None
Data In	None
Le	'08'
Data Out	Challenge

Table 46: Get challenge command description

RESPONSE		
Data Out	Challenge	
SW1 SW2	'6A86'	P1 ≠ '00' or P2 ≠ '00'
	'6700'	Lc ≠ '00'

Table 47: GET Challenge response description

STEP 3: MUTUAL AUTHENTICATE

The MUTUAL AUTHENTICATE command is then sent to authenticate the terminal and retrieve the card cryptogram.

COMMAND	
CLA	‘00’
INS	‘82’
P1	‘00’ Algorithm reference -> no further information (information available in current SE)
P2	‘00’ Secret reference -> no further information (information available in the current SE)

Lc	Lc = ‘48’ (3DES cipher block)
	Lc = ‘50’ (AES cipher block)

Data In	IncomingEncryptedText || MAC Having: o IncomingEncryptedText = ENC[K_{ENC}](RND.IFD || SN.IFD || RND.ICC || SN.ICC || K.IFD) o MAC = MAC[K_{MAC}](IncomingEncryptedText) Where ENC = 3DES-CBC or AES-CBC MAC=ISO/IEC 9797-1 algorithm 3 padding 2 (3DES) or CMAC (AES) RND.IFD: 8 byte-long random number generated by the IFD SN.IFD: 8 byte-long serial number of the IFD (8 least significant bytes) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the IAS card (8 least significant bytes) K.IFD: 32 byte-long random number generated by the IFD

Le	Le = ‘48’ (3DES cipher block)
	Le = ‘50’ (AES cipher block)

Table 48: Mutual authenticate command description

RESPONSE		
Data Out	OutgoingEncryptedText || MAC Having: OutgoingEncryptedText = ENC[K_{ENC}](RND.ICC || SN.ICC || RND.IFD || SN.IFD || K.ICC) and MAC = MAC[K_{MAC}](OutgoingEncryptedText) Where ENC = 3DES-CBC or AES-CBC MAC= ISO/IEC 9797-1 algorithm 3 padding 2 (3DES) or CMAC (AES) RND.IFD: 8 byte-long random number generated by the IFD SN.IFD: 8 byte-long serial number of the IFD (8 least significant bytes) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the ID.me application(8 least significant bytes) K.ICC: 32 byte-long random number generated by the ICC	
SW1 SW2	'6A86'	P1P2 ≠ '0000'
	'6700'	Lc ≠ '48' or Lc ≠ '50' (3DES or AES respectively)
	'6300'	No retry limit: device authentication failed
	'63Cx'	With retry limit, having x = remaining tries. device authentication failed
	'6983'	Referenced key set not successfully used AND no subsequent tries are allowed (remaining tries counter reached 0)
	'6984'	Reference data not usable – Usage counter of the key raised 0
	'6985'	Authentication process not respected (No Get Challenge just before), or SE content doesn't allow processing the command
	'6A88'	Data needed for mutual authentication not found.

Table 49: MUTUAL AUTHENTICATE response encoding for SK scheme

It is the responsibility of the user to check the elements returned by the card and compute the session keys that will be used for the subsequent secure messaging session.

4.2.2 External symmetric authentication

Prerequisite: select the correct symmetric Key Set using Manage Security Environment (MSE Set). The CRT to be used is:

USAGE	HEADER				DATA FIELD
	P1	P2	Tag	Length	Value
Role auth.	'81'	'A4'	'80'	'01'	Symmetric external authentication algorithm
			'83'	'01'	Symmetric key set reference

Table 50: CRT for symmetric external authentication

The external symmetric authentication allows a user to authenticate to the card. The process is described hereby:

IFD		ICC
READ BINARY SN.ICC Reading of the card serial number (for key diversification, the S/N is part of the cryptogram calculation).	-> <-	Send SN.ICC Card Serial Number in the context of the current application
GET CHALLENGE	-> <-	Generate Random number RND.ICC

Compute the authentication token with S.
S=RND.ICC || SN.ICC
EXTERNAL AUTHENTICATE
Data = authentication token
	->
	<-
		MAC checking
		Decipher
		Verify RND.ICC & SN.ICC **(IFD is authenticated)**

Table 51: External authentication procedure

STEP 1: READ SERIAL NUMBER

The READ BINARY command is described in (**READ** BINARY). The EF SN.ICC must be read.

STEP 2: GET CHALLENGE

The GET CHALLENGE command requires a random challenge from the card. The challenge is kept until next command is received.

COMMAND	
CLA	ISO
INS	‘84’
P1	‘00’
P2	‘00’
Lc	None
Data In	None
Le	‘08’
Data Out	Challenge

Table 52: Get challenge command description

RESPONSE		
Data Out	Challenge	
SW1 SW2	‘6A86’	P1 ≠ ‘00’ or P2 ≠ ‘00’
	‘6700’	Lc ≠ ‘00’

Table 53: Get Challenge response description

STEP 3: EXTERNAL AUTHENTICATION

The external authenticate for role is then sent to allow the user authentication to the terminal.

COMMAND	
CLA	ISO
INS	'82'
P1	'00' Algorithm reference -> no further information (information available in current SE)
P2	'00' Secret reference -> no further information (information available in the current SE)
Lc	Variable
Data In	Symmetric Role authentication: ENCRYPT || MAC Having: o ENCRYPT = ENC[K_{ROLE_ENC}](RND.ICC||SN.ICC) o MAC = MAC [K_{ROLE_MAC}](ENCRYPT) Where o ENC = 3DES-CBC or AES-CBC o MAC= ISO/IEC 9797-1 algorithm 3 padding 2 (3DES) or CMAC (AES) SN.IFD: 8 byte-long serial number of the IFD (8 least significant bytes) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the IAS card (8 least significant bytes)
Le	None

Table 54: External authentication command description

RESPONSE		
Data Out	None	
SW1 SW2	'6A86'	P1P2 ≠ '0000'
	'6700'	Wrong length
	'6985'	Authentication process not respected (wrong order of operations)
	'6A88'	Data needed for external authentication not found
	'6300'	External Authentication failed in symmetric scheme only when SDO does not have a retry counter attribute
	'63Cx'	x = remaining tries if SDO has a retry counter attribute, in symmetric scheme only
	'6983'	Authentication method blocked in symmetric scheme only
	'6984'	Reference data not usable in symmetric scheme only
	'9000'	External authentication successful

Table 55: EXTERNAL AUTHENTICATE response description

4.3 ASYMETRIC AUTHENTICATION

4.3.1 Client/Server authentication

Prerequisites: select the correct asymmetric Key using Manage Security Environment (MSE Set).

Client / server authentication consists of using the Card as a crypto box. It computes a signature for the computer used by the card holder to access remote services. The computer gets authenticated with the signature computed by the card using an asymmetric cryptographic scheme.

4.3.1.1 OVERVIEW

4.3.1.1.1 CRYPTOGRAPHY INVOLVED

For the RSA scheme:

Two RSA algorithms are supported for Client/Server Authentication. Both are described extensively in [PKCS1].

PKCS#1 v1.5:

The ID.me applet pads and encrypts the incoming data M received from the IFD as described in RSA PKCS#1 v1.5

```
M' = '00' || '01' || PS || '00' || M
```

Where PS is one or several 'FF' bytes in order to have M' length equals to the key length.

The Message M' is then encrypted using the private key. It **assumes** the incoming data M is formatted as described by PKCS#1 v1.5.

For security reasons, the digest info to sign should not be larger than 40% of the key size. If not, the C/S authentication is rejected.

Supported key sizes for RSA signature are 1024, 1536, 2048, 2560, 3072 bits.

The following table gives the detail of digest info sizes and compatibility with RSA key size

		SHA1		SHA224		SHA256		SHA384		SHA512	
Key Size	max data length	SSL	non-qualified	SSL	non-qualified	SSL	non-qualified	SSL	non-qualified	SSL	non-qualified
1024	51	46	39	**54**	47	**58**	51	**74**	**67**	**90**	**83**
1536	76	46	39	54	47	58	51	74	67	**90**	**83**
2048	102	46	39	54	47	58	51	74	67	90	83
2560	128	46	39	54	47	58	51	74	67	90	83
3072	153	46	39	54	47	58	51	74	67	90	83

Table 56 - Key Sizes and data length for CS authentication

Red values indicate that the security requirement stating that digest info should not be larger than 40% of the key size is not respected. Meaning the C/S authentication is rejected by the card. See SW returned in Table 61.

PKCS#1 v2.1 PSS:

The signature is performed according to [PKCS1], described as RSA PSS. Signature is based on SHA1, SHA224, SHA256, SHA384 or SHA512 algorithm.

For the ECDSA scheme:
The signature is performed according to [X9.62]. Supported domain parameters are profile dependent. See 7.3 for supported domain parameters. Signature is based on SHA1, SHA224, SHA256, SHA384 or SHA512 algorithm.

4.3.1.1.2 PROTOCOL STEPS

The user must also read all the necessary certificates stored in the card to ensure that the key is certified by the appropriate authority, using**READ** BINARY .

The external symmetric authentication allows a user to authenticate to the card. The process is described hereby:

Notation:

- M: digest info field generated by the IFD and sent to the card
- C/S(M): Signature calculated by the entity

IFD		ICC
Send command INTERNAL AUTHENTICATE ○ Data=digestinfo	-> <-	The card calculates the authentication token according to the algorithm described here below C/S(M)

Table 57: Process for client/server authentication

4.3.1.1.3 SETTING THE CRYPTOGRAPHIC CONTEXT

The service is realized by setting the relevant template in the Current security Environment (CSE).

The template is a CRT AT encoded as follows:

TAG	LENGTH	MEANING			M/O
'A4'	LA4				M
▨	▨	**TAG**	**LENGTH**	**MEANING**	
▨	▨	**'84'**	**'01'**	Private Key Reference	M
▨	▨	**'80'**	**'01'**	Algorithm identifier for C/S authentication (see Table 59)	M

Table 58 - CRT AT encoding for Client/Server Authentication

C/S authentication method	Value
RSA PKCS#1 V1.5	'02'
RSA PKCS#1 PSS -SHA-1	'03'
ECC -SHA-1	'04'

Table 59: Client/Server Authentication - algorithm identifier

4.3.1.2 INTERNAL AUTHENTICATE

The INTERNAL AUTHENTICATE command computes a digital signature that can be checked by using the digital certificates retrieved from the card.

COMMAND	
CLA	ISO
INS	'88'
P1	'00' Algorithm reference -> no further information (information available in current SE)
P2	'00' Secret reference -> no further information (information available in the current SE)
Lc	Lc≤ 40% of the key length in bytes (i.e. '33' for 1024 bits, '4C' for 1536 bits and '66' for 2048)
Data In	Incoming message
Le	Variable
Data Out	SIG = DS [SK.ICC.CS_AUT] (Incoming message)

Table 60: Internal authentication command description

RESPONSE		
Data Field	SIG = DS [SK.ICC.CS_AUT] (Incoming message)	
SW1 SW2	'6A86'	P1P2 ≠ '0000'
	'6700'	Wrong length; no further indication
	'6A88'	Reference data needed for internal authenticate not found

Table 61: INTERNAL AUTHENTICATE response encoding for client/server authenticate

4.3.2 Asymmetric Role Authentication

This section is applicable only if PKI service is activated. See PKI configuration **Error! Reference source not found.**.

Prerequisites: select the correct asymmetric Key using Manage Security Environment (MSE Set).

The CRT to be used is:

USAGE	HEADER		DATA FIELD		
	P1	P2	Tag	Length	Value
Client/Server authentication	'81'	'A4'	'80'	'01'/'04'	Asymmetric external authentication algorithm
			'84'	'01'	Asymmetric private key set reference

Table 62: CRT AT encoding for Asymmetric Role Authentication

The external symmetric authentication allows a user to authenticate to the card. The process is described hereby:

ENTITY TO AUTHENTICATE		ICC
Present the certificate chain	-> <-	Verify step by step the certificate chain. After this steps, PuK.IFD.RA is available in the ICC
GET CHALLENGE	-> <-	Generate Random number RND.ICC
Compute an authentication token M = DS(RND.ICC || SN.ICC)		
EXTERNAL AUTHENTICATE Data = M	-> <-	Verify the signature Verify RND.ICC & SN.ICC

Table 63: Asymmetric Role Authentication sequence

STEP 1: PRESENT THE CERTIFICATE CHAIN.

STEP 2:

GET CHALLENGE

The GET CHALLENGE command requires a random challenge from the card. The challenge is kept until next command is received.

COMMAND	
CLA	ISO
INS	‘84’
P1	‘00’
P2	‘00’
Lc	None
Data In	None
Le	‘08’
Data Out	Challenge

Table 64: Get challenge command description

RESPONSE		
Data Out	Challenge	
SW1 SW2	‘6A86’	P1 ≠ ‘00’ or P2 ≠ ‘00’
	‘6700’	Lc ≠ ‘00’

Table 65: Get challenge response

STEP 3: EXTERNAL AUTHENTICATION

The external authenticate for role is then sent to allow the user authentication to the terminal.

COMMAND	
CLA	ISO
INS	'82'
P1	'00' Algorithm reference → no further information (information available in current SE)
P2	'00' Secret reference → no further information (information available in the current SE)
Lc	Variable
Data field	Asymmetric Role authentication: SIG = DS $_{[SK.IFD.ROLE\_AUT]}$ (RND.ICC||SN.ICC) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the IAS card (8 least significant bytes) RSA [PrK.IFD.AUT](…/…): PrK.IFD.ROLE_AUT.length = length of the RSA key (i.e. 1024, 1536 or 2048 bits). The key is issued from a PSO VERIFY CERTIFICATE.

Table 66: External authentication command description

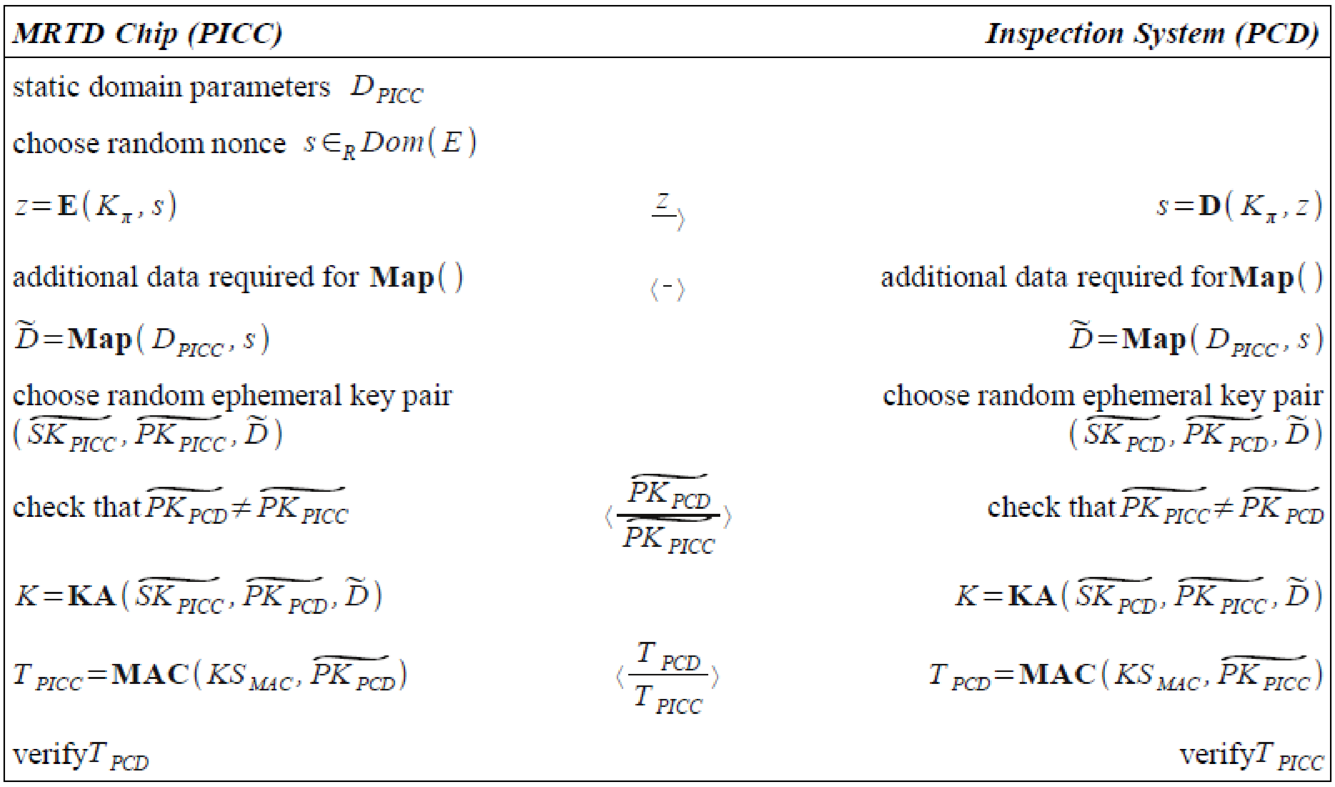
RESPONSE		
Data Out	None	
SW1 SW2	'6A86'	P1P2 ≠ '0000'
	'6700'	Wrong length
	'6985'	Authentication process not respected (wrong order of operations)
	'6A88'	Data needed for external authentication not found
	'6300'	External Authentication failed in symmetric scheme only when SDO does not have a retry counter attribute
	'63Cx'	x = remaining tries if SDO has a retry counter attribute, in symmetric scheme only
	'6983'	Authentication method blocked in symmetric scheme only
	'6984'	Reference data not usable in symmetric scheme only
	'9000'	External authentication successful

Table 67: Table 96: External authentication response description

4.3.3 PACE

ID.me allows performing a PACE establishment using the GENERAL AUTHENTICATE command. The whole process is described hereby.

PIN Objects used for PACE authentication must have an identifier comprised between ‘10’ and ‘1F’.



MRTD Chip (PICC)		*Inspection System (PCD)*
static domain parameters D_{PICC}		
choose random nonce $s \in_R Dom(E)$		
$z = \mathbf{E}(K_\pi, s)$	$\xrightarrow{z}$	$s = \mathbf{D}(K_\pi, z)$
additional data required for $\mathbf{Map}()$	$\langle - \rangle$	additional data required for $\mathbf{Map}()$
$\tilde{D} = \mathbf{Map}(D_{PICC}, s)$		$\tilde{D} = \mathbf{Map}(D_{PICC}, s)$
choose random ephemeral key pair $(\widetilde{SK}_{PICC}, \widetilde{PK}_{PICC}, \tilde{D})$		choose random ephemeral key pair $(\widetilde{SK}_{PCD}, \widetilde{PK}_{PCD}, \tilde{D})$
check that $\widetilde{PK}_{PCD} \neq \widetilde{PK}_{PICC}$	$\langle \frac{\widetilde{PK}_{PCD}}{\widetilde{PK}_{PICC}} \rangle$	check that $\widetilde{PK}_{PICC} \neq \widetilde{PK}_{PCD}$
$K = \mathbf{KA}(\widetilde{SK}_{PICC}, \widetilde{PK}_{PCD}, \tilde{D})$		$K = \mathbf{KA}(\widetilde{SK}_{PCD}, \widetilde{PK}_{PICC}, \tilde{D})$
$T_{PICC} = \mathbf{MAC}(KS_{MAC}, \widetilde{PK}_{PCD})$	$\langle \frac{T_{PCD}}{T_{PICC}} \rangle$	$T_{PCD} = \mathbf{MAC}(KS_{MAC}, \widetilde{PK}_{PICC})$
verify T_{PCD}		verify T_{PICC}

Figure 3: PACE procedure description

The SAC establishment is performed using two commands: Manage Security Environment (MSE Set), and General Authenticate, described here:

COMMAND	
CLA	ISO
INS	‘86’
P1	‘00’
P2	‘00’
Lc	Var.
Data In	Protocol specific data, described below
Le	Variable
Data Out	Protocol specific data, described below

Table 68: General authenticate command description

STEP	DESCRIPTION	COMMAND DATA	RESPONSE DATA
1	Generate and Encrypt Nonce	None	‘80’ Encrypted Nonce
2	Map Nonce	‘81’ Mapping data	‘82’ Mapping Data
3	Perform Key Agreement	‘83’ Ephemeral Public Key	‘84’ Ephemeral Public Key
4	Mutual Authentication	‘85’ Authentication token	‘86’ Authentication token

Table 69: Command and response data for General Authenticate

STEP 1: MSE SET

This first step is already described in Manage Security Environment (MSE Set), but it is important to note that this step determines the mode of operation for the password key derivation (CAN/MRZ/PIN-based).

USAGE	HEADER		DATA FIELD		
	P1	P2	Tag	Length	Value
Mutual auth. SAC	‘C1’	‘A4’	‘80’	Var.	Object Identifiers of the SAC protocol to use (see [SAC])
			‘83’	‘01’	Identifier of the derivation mode for SAC: ‘01’ - MRZ ‘02’ - CAN ‘XX’ –PIN (PIN id bit 5 must be 1)
			‘84’	‘01’	Reference of the domain parameters to be used for authentication

Table 70: CRT for PACE authentication

STEP 2: GENERAL AUTHENTICATE 1 - NONCE GENERATION

The first General Authenticate command instructs the card to generate a Nonce Value, and return its value ciphered with the password key. The password key is derivated from the element described through the MSE Set Command.

STEP 3: GENERAL AUTHENTICATE 2 – MAPPING DERIVATION

This step is used to generate ephemeral domain parameters that will be used to perform the key agreement in the next step.

STEP 4: GENERAL AUTHENTICATE 3 – PERFORM KEY AGREEMENT.

This step is a regular key asymmetric key agreement performed with the domain parameter computed in the preceding general authentication command. The secret obtained will be used to derivate symmetric session keys.

STEP 5: GENERAL AUTHENTICATE 4 – MUTUAL AUTHENTICATION.

Using the symmetric keys computed in the last step, a mutual authentication is performed to validate that the whole process is successful and that the same keys have been computed by the two parties.

5 CRYPTOGRAPHIC SERVICES

5.1 SIGNATURE GENERATION

Prerequisites: select the correct asymmetric Key and Hash algorithm using Manage Security Environment (MSE Set).

The CRT to be used for Hash algorithm is:

USAGE	HEADER		DATA FIELD		
	P1	P2	Tag	Length	Value
Hash Algorithm	'41'	'A6'	'80'	'01'	Algorithm identifier for Hash

Table 71: CRT for PSO Hash

CRT for signature Key and algorithm

USAGE	HEADER		DATA FIELD		
	P1	P2	Tag	Length	Value
Data Signature	'41'	'B6'	'80'	Var.	algorithm identifier for signature
			'84'	'01'	Asymmetric key (private portion) reference

Table 72: CRT for PSO Sign

The card has the capability to generate an asymmetric signature, with a hash that can be pre-computed off card. The steps are described hereby:

IFD		ICC
The IFD performs the partial hash calculation over M. The computation outcomes are the following: ○ PartialHash(M) ○ Counter(M) ○ RemainingMessage(M)		
Send partial hash data and require final hash round calculation. Use of the command PSO Hash Incoming data is for hash calculation **TAG** | **LENGTH** | **VALUE** **'90'** | **Var.** | PartialHash(M) || Counter(M) **'80'** | **Var. (≤ '40')** | RemainingMessage(M) Last block, to be hashed by the card	-> <-	The card initialize the hashing context with incoming data resulting from the partial hash calculation, then ends the hash over the last data block. Hash(M) is available.
The IFD requires the signature calculation- Use of the command PSO Compute Digital Signature	-> <-	The card calculates the signature with the selected private key and returns the result.

Table 73: Process for signature generation

STEP 1: HASH COMPUTATION

The hash computation is performed through the PSO Hash command.

COMMAND	
CLA	ISO
INS	'2A'
P1	'90'
P2	'A0'
Lc	Data field length
Data In	T1 || L1 || V1 || T2 || L2 || V2 T1 = '90' Intermediate hash code L1 = Intermediate hash length + 8 V1 = 'intermediate hash value' || '8 bytes (nb of bits already hashed)' T2 = '80' last block '0' < L2 ≤ '40' V2 = Text to hash without padding
Le	None

Table 74: PSO Hash command description

RESPONSE		
Data Out	None	
SW1 SW2	'6A85'	If the AlgID of the CRT HT of the current SE is incorrect
	'6A80'	TLV structure error
	'6700'	Bad length

Table 75: PSO - HASH response encoding

STEP 2: SIGNATURE COMPUTATION

The signature computation is performed through the PSO Sign command.

COMMAND	
CLA	ISO
INS	'2A'
P1	'9E'
P2	'9A' Data to sign
Lc	If Data Field is absent, Lc is absent If Data Field is present, Lc = Length of computed Hash
Data In	For on-card partial hash, Data In is absent For off-card hash, Data In = computed Hash
Le	Signature length = '00'
Data Out	Digital signature – unformatted

Table 76: PSO sign command description

RESPONSE		
Data Out	Digital signature – unformatted	
SW1 SW2	'6985'	No hash available
	'6A88'	Current SE problem
	'6A86'	Incorrect P1-P2
	'6982'	Signature key access conditions not fulfilled
	'6700'	Data length does not match with expected hash length

Table 77: PSO - Compute Digital Signature response encoding

5.2 ASYMMETRIC KEY DECIPHERING

Prerequisites: select the correct asymmetric Key and algorithm using Manage Security Environment (MSE Set).

The CRT to be used for decryption is:

USAGE	HEADER		DATA FIELD		
	P1	P2	Tag	Length	Value
Message decryption	'41'	'B8'	'80'	'01'	Data decryption algorithm reference
			'84'	'01'	Mutual authentication asymmetric key (private portion)reference

Table 78: CRT for key decipherment

The Key deciphering operation is performed through its dedicated command, described below:

COMMAND	
CLA	ISO
INS	'2A'
P1	'80'
P2	'86'
Lc	Variable
Data In	'81' || [Ciphered key padded according to PKCS#1 v1.5]
Le	Extracted key length
Data Out	Unformatted extracted key in plain text format

Table 79: Key decipherment command description

5.3 ASYMMETRIC KEY PAIR GENERATION

The asymmetric Key Pair Generation is performed through its dedicated command, described below:

COMMAND	
CLA	ISO
INS	'47'
P1	'00'
P2	'00'
Lc	Length of command data field = '05'
Data In	Inter industry template: '70' L_{70} {Tag of the SDO private portion referencing the key pair to generate}
Le	Variable

Table 80: Asymmetric key pair generation command description

6 APDU REFERENCES

6.1 COMMAND ⇔RESPONSE PAIRS

APDU command-response pairs are handled as indicated in [7816-3].

6.2 CLASS BYTE CODING

CLA indicates the class of the command. According to [7816-4], 'FF' is an invalid value.

Bit 8 of CLA distinguishes between the inter industry class and the proprietary class. Bit 8 set to 0 indicates the inter industry class.

The values 000xXXXxb are specified hereafter.

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	Support
0	0	0	**X**	-	-	-	-	**Command chaining control**	YES
			0	-	-	-	-	— The command is the last or only command of a chain	
			1	-	-	-	-	— The command is not the last command of a chain	
			-	**X**	**X**	-	-	**Secure messaging indication**	YES
			-	0	**0**	-	-	— No SM	
			-	0	1	-	-	— Proprietary SM format	NO
			-	1	0	-	-	— SM, command header not processed	
			-	1	1	-	-	— SM, command header authenticated	YES
			-	-	-	**X**	**X**	**Logical channel number from zero to three**	NO

Table 81 - CLASS byte Interindustry values

6.3 THE STATUS BYTES SW1 & SW2

SW1-SW2 indicates the processing state. Their coding follows these rules:

	SW1	SW2	MEANING
Normal processing	**‘90’**	**‘00’**	No further qualification
	‘61’	**‘XX’**	SW2 encodes the number of data bytes still available
Warning processing	**‘62’**	**‘XX’**	State of non-volatile memory is unchanged (further qualification in SW2)
	‘63’	**‘XX’**	State of non-volatile memory has changed (further qualification in SW2)
Execution error	**‘64’**	**‘XX’**	State of non-volatile memory is unchanged (further qualification in SW2)
	‘65’	**‘XX’**	State of non-volatile memory has changed (further qualification in SW2)
	‘66’	**‘XX’**	Security-related issues
Checking error	**‘67’**	**‘00’**	Wrong length; no further indication
	‘68’	**‘XX’**	Functions in CLA not supported (further qualification in SW2)
	‘69’	**‘XX’**	Command not allowed (further qualification in SW2)
	‘6A’	**‘XX’**	Wrong parameters P1-P2 (further qualification in SW2)
	‘6B’	**‘00’**	Wrong parameters P1-P2
	‘6C’	**‘XX’**	Wrong Le field; SW2 encodes the exact number of available data bytes
	‘6D’	**‘00’**	Instruction code not supported or invalid
	‘6E’	**‘00’**	Class not supported
	‘6F’	**‘00’**	No precise diagnosis

Table 82 - General meaning of the Interindustry values of SW1-SW2

SW1	SW2	MEANING
‘62’ (warning)	**‘00’**	No information given
	‘01’ to ‘80’	RFU
	‘81’	Part of returned data may be corrupted
	‘82’	End of file or record reached before reading N_e bytes
	‘83’	Selected file deactivated
	‘84’	File control information are not correctly formatted
	‘85’	Selected file in termination state
	‘86’	No input data available from a sensor on the card
‘63’ (warning)	**‘00’**	No information given
	‘81’	File filled up by the last write
	‘CX’	Counter from 0 to 15 encoded by ‘X’ (exact meaning depending on the command)
‘64’ (error)	**‘00’**	Execution error
	‘01’	Immediate response required by the card
	‘02’ to ‘80’	Triggering by the card
‘65’ (error)	**‘00’**	No information given
	‘81’	Memory failure
‘68’ (error)	**‘00’**	No information given
	‘81’	Logical channel not supported
	‘82’	Secure messaging not supported
	‘83’	Last command of the chain expected
	‘84’	Command chaining not supported
‘69’ (error)	**‘00’**	No information given
	‘81’	Command incompatible with file structure
	‘82’	Security status not satisfied
	‘83’	Authentication method blocked
	‘84’	Reference data not usable
	‘85’	Conditions of use not satisfied
	‘86’	Command not allowed (no current EF)
	‘87’	Expected secure messaging data objects missing
	‘88’	Incorrect secure messaging data objects
‘6A’	**‘00’**	No information given

SW1	SW2	MEANING
(error)	'80'	Incorrect parameters in the command data field
	'81'	Function not supported
	'82'	File or application not found
	'83'	Record not found
	'84'	Not enough memory space in the file
	'85'	L_c inconsistent with TLV structure
	'86'	Incorrect parameters P1-P2
	'87'	L_c inconsistent with parameters P1-P2
	'88'	Referenced data or reference data not found (exact meaning depending on the command)
	'89'	File already exists
	'8A'	DF name already exists

Table 83 - Specific inter industry warning and error conditions

6.3.1 Status word treatment for interoperability

Depending on each implementation, the status word values may vary. For interoperability, the status words to verify are:

1. Normal ending:

   ‘9000’ -> No error.

2. Warning, the command was executed but a concern was detected:

   '6200’; '6281' to '629F' / '6300'; '6381' to '639F'

3. Error, command not executed:

   '6400' / '6800'; '6881' à '688F' / '6900'; '6981' to '699F' / '6A00'; '6A80' to '6A9F' / ‘6581’ /‘6700’ / ‘6B00’ / ‘6D00’ / ‘6E00’

4. Error generated following a secret data verification that leaded to a ratification:

   ‘63Cx’, (‘x’ indicating the remaining number of tries).

6.4 USER AUTHENTICATION COMMANDS

The card supports any format for PIN / password presentation.

The commands defined in this chapter have the same coding for P2:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Local reference data (Application)
0	-	-	-	-	-	-	-	Global reference data (Card)
-	-	-	**X**	**X**	**X**	**X**	**X**	User authentication DO reference
0	0	0	0	0	0	0	0	Forbidden

Table 84 - coding of P2: qualifiers of the reference data

6.4.1 VERIFY

The VERIFY command specification can be found in 4.1.

6.4.2 CHANGE REFERENCE DATA

The CHANGE REFERENCE DATA command specification can be found in 0.

6.4.3 RESET RETRY COUNTER

The RESET RETRY COUNTER command specification can be found in 4.1.6

6.5 AUTHENTICATION COMMANDS

6.5.1 GET CHALLENGE

6.5.1.1 DEFINITION AND SCOPE

This command shall be used to generate a random value from the card.

6.5.1.2 CONDITIONAL USAGE AND SECURITY

No security conditions are required to perform this command.

The random is valid only for the command following GET CHALLENGE call.

6.5.1.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	‘84’
P1	‘00’
P2	‘00’
Lc	None
Data In	None
Le	‘08’

Table 85 - GET CHALLENGE command encoding

6.5.1.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	Challenge	
SW1 SW2	‘6A86’	P1 ≠ ‘00’ or P2 ≠ ‘00’
	‘6700’	Lc ≠ ‘00’

Table 86 - GET CHALLENGE response encoding

6.5.2 INTERNAL AUTHENTICATE. PK-DH scheme

The INTERNAL AUTHENTICATE command with PK-DH scheme is not supported by the ID.me applet.

6.5.3 EXTERNAL AUTHENTICATE. PK-DH scheme

The EXTERNAL AUTHENTICATE command with PK-DH scheme is not supported by the ID.me applet.

6.5.4 MUTUAL AUTHENTICATE. SK scheme

6.5.4.1 DEFINITION AND SCOPE

The mutual authentication is the process whereby the ID.me card authenticates the external application and vice versa by means of a signature based on challenge/response authentication scheme. If this verification process succeeds then the external card application will get access to the authorized data and functions in the ID.me card.

A successful mutual authentication shall cause the setting of a secure message channel between the external application and the ID.me card.

6.5.4.2 CONDITIONAL USAGE AND SECURITY

Both entities (IFD and ICC) authenticate each other.

SN.IFD size is 8 bytes.

SN.ICC size is 8 bytes. If the actual size of SN.ICC is not 8 bytes the 8 rightmost bytes are used.

MUTUAL AUTHENTICATE shall be preceded by the GET CHALLENGE command.

6.5.4.3 COMMAND MESSAGE

COMMAND	
CLA	‘00’
INS	‘82’
P1	‘00’ Algorithm reference -> no further information (information available in current SE)
P2	‘00’ Secret reference -> no further information (information available in the current SE)

Lc	Lc = ‘48’ (3DES cipher block)
	Lc = ‘50’ (AES cipher block)

Data In	IncomingEncryptedText || MAC Having: o IncomingEncryptedText = ENC[K_{ENC}](RND.IFD || SN.IFD || RND.ICC || SN.ICC || K.IFD) o MAC = MAC[K_{MAC}](IncomingEncryptedText) Where ENC = 3DES-CBC or AES-CBC MAC=ISO/IEC 9797-1 algorithm 3 padding 2 (3DES) or CMAC (AES) RND.IFD: 8 byte-long random number generated by the IFD SN.IFD: 8 byte-long serial number of the IFD (8 least significant bytes) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the ID.me application (8 least significant bytes) K.IFD: 32 byte-long random number generated by the IFD

Le	Le = ‘48’ (3DES cipher block)
	Le = ‘50’ (AES cipher block)

Table 87 - MUTAL AUTHENTICATE command encoding for SK scheme

6.5.4.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	OutgoingEncryptedText || MAC Having: OutgoingEncryptedText = ENC[K_{ENC}](RND.ICC || SN.ICC || RND.IFD || SN.IFD || K.ICC) and MAC = MAC[K_{MAC}](OutgoingEncryptedText) Where ENC = 3DES-CBC or AES-CBC MAC= ISO/IEC 9797-1 algorithm 3 padding 2 (3DES) or CMAC (AES) RND.IFD: 8 byte-long random number generated by the IFD SN.IFD: 8 byte-long serial number of the IFD (8 least significant bytes) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the ID.me application(8 least significant bytes) K.ICC: 32 byte-long random number generated by the ICC	
SW1 SW2	'6A86'	P1P2 ≠ '0000'
	'6700'	Lc ≠ '48' or Lc ≠ '50' (3DES or AES respectively)
	'6300'	No retry limit: device authentication failed
	'63Cx'	With retry limit, having x = remaining tries. device authentication failed
	'6983'	Referenced key set not successfully used AND no subsequent tries are allowed (remaining tries counter reached 0)
	'6984'	Reference data not usable – Usage counter of the key raised 0
	'6985'	Authentication process not respected (No Get Challenge just before), or SE content doesn't allow processing the command
	'6A88'	Data needed for mutual authentication not found.

Table 88 - MUTUAL AUTHENTICATE response encoding for SK scheme

6.5.5 EXTERNAL AUTHENTICATE for Role authentication

The asymmetric scheme of the EXTERNAL AUTHENTICATE command for Role Authentication is only supported in IAS PKI configuration (see ***Error! Reference source not found.****).*

6.5.5.1 DEFINITION AND SCOPE

The External Authenticate command is used by an external application, in conjunction with the Get Challenge command to allow the external application to authenticate its identity to the card.

Through the Get Challenge command, the external application receives a set of challenge data from the card (i.e., a random number generated by the card). The external application then encrypts this information with a secret key. This then forms a challenge, which is sent to the card via the External Authenticate command. If the external application knows the same secret key that is stored on the card, then when the card decrypts the challenge, it will find the same random number generated by the last Get Challenge command. Therefore, the card now knows the identity of the external application and can give it to data on the card.

The attractive characteristic of this method is that the secret key used to authenticate identity between the external application and the card was never transferred between the external application and the card.

6.5.5.2 CONDITIONAL USAGE AND SECURITY

This command may be called either in clear-text form, or within a secure channel. In any cases, the MAC calculation (when symmetric scheme is used) of this command never changes the SSC that is used for secure messaging MAC calculation (calculations are fully independent).

EXTERNAL AUTHENTICATE shall be preceded by the GET CHALLENGE command.

6.5.5.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	‘82’
P1	‘00’ Algorithm reference -> no further information (information available in current SE)
P2	‘00’ Secret reference -> no further information (information available in the current SE)
Lc	Variable
Data In	Asymmetric Role authentication: SIG = DS $_{[PuK.IFD.RA]}$ (RND.ICC||SN.ICC) Symmetric Role authentication: ENCRYPT || MAC Having: o ENCRYPT = ENC[K_{ROLE_ENC}](RND.ICC||SN.ICC) o MAC = MAC [K_{ROLE_MAC}](ENCRYPT) Where o ENC = 3DES-CBC or AES-CBC o MAC= ISO/IEC 9797-1 algorithm 3 padding 2 (3DES) or CMAC (AES) PuK.IFD.RA: Public key of the entity to be authenticated RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) RND.ICC: 8 byte-long random number generated by the card (see Get Challenge) SN.ICC: 8 byte-long serial number of the ID.me application (8 least significant bytes) RSA [PrK.IFD.AUT](…/…):PrK.IFD.ROLE_AUT.length = length of the RSA key (i.e. 1024, 1536 or 2048 bits). The key is issued from a PSO VERIFY CERTIFICATE. PrK.IFD.AUT: The IFD signs the challenge with its private key for authentication PrK.IFD.AUT
Le	None

Table 89 - EXTERNAL AUTHENTICATE command encoding for Role authentication

6.5.5.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	None	
SW1 SW2	‘6A86’	P1P2 ≠ ‘0000’
	‘6700’	Wrong length
	‘6985’	Authentication process not respected (wrong order of operations)
	‘6A88’	Data needed for external authentication not found
	‘6300’	External Authentication failed in symmetric scheme only when SDO does not have a retry counter attribute
	‘63Cx’	x = remaining tries if SDO has a retry counter attribute, in symmetric scheme only
	‘6983’	Authentication method blocked in symmetric scheme only
	‘6984’	Reference data not usable in symmetric scheme only
	‘9000’	External authentication successful

Table 90 - EXTERNAL AUTHENTICATE response encoding for Role authentication

6.5.6 INTERNAL AUTHENTICATE for Client/Server authenticate

6.5.6.1 DEFINITION AND SCOPE

The Internal Authenticate command is a command sent by an external application to the security system on the card to allow the card to prove that it possesses a secret key that is shared with the external application. To prepare this command, the external application creates a set of challenge data (i.e., essentially, the application generates a random number). This number is then encrypted with some agreed-on algorithm (with the card); this constitutes a challenge to the card.

When given the command, the card then ciphers the challenge with a secret key stored in a file on the card. The information derived from the encryption is then passed back to the external application as a response to the command. If the card really does have the correct secret key, then the information passed back will be correctly checked.

This command is used by the application to authenticate the card's identity. That is, when successfully completed, the application knows the identity of the card and can give to the card access to information or services within the external application.

6.5.6.2 CONDITIONAL USAGE AND SECURITY

No security conditions are required to perform this command.

6.5.6.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	'88'
P1	'00' Algorithm reference -> no further information (information available in current SE)
P2	'00' Secret reference -> no further information (information available in the current SE)
Lc	Lc≤ 40% of the key length in bytes (i.e. '33' for 1024 bits, '4C' for 1536 bits and '66' for 2048)
Data In	Incoming message
Le	Variable

Table 91 - INTERNAL AUTHENTICATE command encoding for client/server authenticate

6.5.6.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Field	SIG = DS [SK.ICC.CS_AUT] (Incoming message)	
SW1 SW2	'6A86'	P1P2 ≠ '0000'
	'6700'	Wrong length; no further indication
	'6A88'	Reference data needed for internal authenticate not found

Table 92 - INTERNAL AUTHENTICATE response encoding for client/server authenticate

6.5.7 GENERAL AUTHENTICATE for PACE

6.5.7.1 DEFINITION AND SCOPE

A chain of General Authenticate commands is used to perform the PACE protocol.

6.5.7.2 CONDITIONAL USAGE AND SECURITY

This command can be performed only if the security status satisfies the security attributes for this operation. The successful execution of a chain may be subject to successful completion of prior commands (e.g., VERIFY, SELECT) or selections (e.g., the relevant secret). If there is a current key and a current algorithm when issuing a chain, then the chain may implicitly use them. Such a chain may conditionally update the security status using the result (yes or no) of a control performed by the card.

The card may record the number of times the command is issued, in order to limit the number of further uses of the relevant secret or the algorithm. The card may record unsuccessful comparisons (e.g., to limit the number of further uses of the reference data).

6.5.7.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	'86'
P1	'00'
P2	'00'
Lc	None
Data In	Step specific (see below)
Le	None

Table 93 - GENERAL AUTHENTICATE command encoding

6.5.7.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	Step specific (see below)	
SW1 SW2	'9000'	Normal operation – the protocol (step) was successful
	'6300'	Authentication failed – the protocol (step) failed
	'63CX'	Authentication with PIN failed – X tries remaining.
	'6A80'	Incorrect parameters in data field – provided data is invalid
	other	Operating system dependent error – The protocol (step) failed

Table 94 - GENERAL AUTHENTICATE response encoding

6.5.7.5 EXCHANGED DATA OBJECTS

Step	Description	Command Data		Response Data	
1.	Encrypted Nonce	-	Absent[1]	'80'	Encrypted Nonce
2.	Map Nonce	'81'	Mapping Data	'82'	Mapping Data
3.	Perform Key Agreement	'83'	Ephemeral Public Key	'84'	Ephemeral Public Key
4.	Mutual Authentication	'85'	Authentication Token	'86'	Authentication Token
				'87'	Certificate Authority Reference #1[2] (conditional)
				'88'	Certificate Authority Reference #22 (conditional)

Table 95 - GENERAL AUTHENTICATE command/response exchanged data

For more details about the encoding of each element, please refer to [BSI TR-03110].

<u>Note</u>2:

The Certificate Authority Reference(s) (CAR) are present if PACE is used with a Certificate Holder Authorization Template (CHAT), i.e. if PACE is to be followed by Terminal Authentication version 2. In this case the data object '87' contains the most recent CAR. The data object '88' may contain the previous CAR. The CHAT is provided in the `MSE: Set AT` command that initiates the PACE protocol; see 6.9.2 for details.

6.6 PERFORM SECURITY OPERATION COMMANDS

6.6.1 PSO – Hash

6.6.1.1 DEFINITION AND SCOPE

This command computes a hash sum. The algorithm to be used must be specified using the MSE SET command. This command supports command chaining mechanism, which utilizes the CLA value to indicate the end of the command chain.

6.6.1.2 CONDITIONAL USAGE AND SECURITY

The calculation result (i.e. the hash) is saved until the next command call that should be a PSO CDS.

If the message to hash is less or equal to 64 bytes, L1 value is set to zero and the complete message to hash is computed by the card.

The command chain has CLA = 10h for all but the last command of the chain, which has CLA = 00h. In chained commands the commands with CLA = 10h shall carry only data which length is multiple of the block size of the hashing algorithm. The last command of the chain has no data length limitations. In order to be able to sign or verify the generated hash sum, the CLA must be 00h (end of chain) in the PSO: HASH command given immediately before the PSO: COMPUTE DIGITAL SIGNATURE command.

6.6.1.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	‘2A’
P1	‘90’
P2	‘A0’
Lc	Data field length shall be: In SHA1 ≤ ‘60’ In SHA224 ≤ ‘68’ In SHA256 ≤ ‘6C’ In SHA384 ≤ ‘7C’ In SHA512 ≤ ‘8C’
Data In	T1 || L1 || V1 || T2 || L2 || V2 T1 = ‘90’ Intermediate hash code L1 = ‘00’ or 28 bytes for SHA-1,36 bytes for SHA-224, 40 bytes for SHA-256, 56 bytes for SHA-384, 72 bytes for SHA-512 V1 = ’intermediate hash value’ || ‘8 bytes (nb of bits already hashed)’ T2 = ‘80’ last block ‘0’ < L2 ≤ '72' V2 = Text to hash without padding
Le	None

Table 96 - PSO - HASH command encoding

6.6.1.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	None	
SW1 SW2	‘6A85’	If the AlgID of the CRT HT of the current SE is incorrect
	‘6A80’	TLV structure error
	‘6700’	Bad length

Table 97 - PSO - HASH response encoding

6.6.2 PSO – Compute Digital Signature

6.6.2.1 DEFINITION AND SCOPE

This command computes a digital signature. The private key and algorithm to be used must be specified using the MSE SET command.

Only signature keys apply to signature algorithms:

- Authentication key
- Non-Repudiation key

6.6.2.2 CONDITIONAL USAGE AND SECURITY

The Access Condition of the referenced key related to this command must be fulfilled prior using the command:

- The PIN/Biometry access right must have been granted at any time before using the signature authentication key.
- Each time the signature non-repudiation key is used, a successful Verify APDU command has to be executed just before performing this command.

- For Signature generation with partially hash computed on-card, PSO-HASH command shall be performed before PSO-Compute Digital Signature.
- For Signature generation with hash computed off-card, PSO-Compute Digital Signature command data field will contain the computed hash. If PSO-HASH command is performed before PSO-Compute Digital Signature command then hash calculated by PSO-HASH command is discarded and hash provided in command data will take precedence over the previously generated hash.

6.6.2.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	'2A'
P1	'9E'
P2	'9A' Data to sign
Lc	If Data Field is absent, Lc is absent If Data Field is present,Lc = Length of computed Hash
Data In	For on-card partial hash, DataIn is absent For off-card hash, DataIn = computed Hash
Le	Signature length = '00'

Table 98 - PSO - Compute Digital Signature command encoding

6.6.2.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	Digital signature – unformatted	
SW1 SW2	'6985'	No hash available
	'6A88'	Current SE problem
	'6A86'	Incorrect P1-P2
	'6982'	Signature key access conditions not fulfilled
	'6700'	Data length does not match with expected hash length

Table 99 - PSO - Compute Digital Signature response encoding

The data Out (Digital signature) format is described in the following table:

Signature data format	
Algorithm	Format
RSA PKCS#1 v1.5	Plain data corresponding to the RSA signature.
RSA PKCS#1 v2.1	Plain data corresponding to the RSA signature.
ECDSA	The returned data have a size equal to the related EC key size * 2. The output signature is returned in plaintext without any TLV formatting. It is composed of two concatenated elements: r and s, each one on the length of the elliptic curve base point order.

Table 100 - PSO - Compute Digital Signature data out format

6.6.3 PSO – Decipher (RSA)

6.6.3.1 DEFINITION AND SCOPE

DECIPHER command decrypts an encrypted message (cryptogram). The key and algorithm to be used must be specified using the MSE SET command.

PSO Decipher can be used from RSA Key length from 1024 bits to 3072 bits.

6.6.3.2 CONDITIONAL USAGE AND SECURITY

The access right of the referenced key related to this command must be fulfilled prior using the command.

6.6.3.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	‘2A’
P1	‘80’
P2	‘86’
Lc	Variable
Data In	'81' || [Ciphered key padded according to PKCS#1 v1.5]
Le	Extracted key length

Table 101 - PSO – Decipher command encoding using RSA

6.6.3.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	Unformatted extracted key in plain text format (padding removed) It shall be ≤ 40% of the key size (i.e. ‘33’ for 1024 bits, ‘4C’ for 1536 bits and ‘66’ for 2048)	
SW1 SW2	‘6700’	The enciphered data length doesn’t match with the decipher key length
	‘6A80’	Padding verification error or deciphered key length is longer than 40% of the key length
	‘6A88’	Current SE problem
	‘6A86’	Incorrect P1-P2
	‘6982’	Decipher key access conditions not fulfilled, or extraction error

Table 102 - PSO – Decipher response encoding using RSA

6.6.4 PSO – Decipher (ECC)

6.6.4.1 DEFINITION AND SCOPE

ECC based key decipherment is used for a key agreement at the receiver and sender end. The key and algorithm to be used must be specified using the MSE SET command.

PSO Decipher can be used with the following key lengths:

1. ANSSI FRP256v1 (specified in [PRMD1123151V])
2. Brainpool192r1
3. Brainpool224r1
4. Brainpool256r1
5. Brainpool320r1
6. Brainpool384r1
7. Brainpool512r1
8. NIST P-192 (secp192r1)
9. NIST P-224 (secp224r1)
10. NIST P-256 (secp256r1)
11. NIST P-384 (secp384r1)
12. NIST P-521 (secp521r1)

6.6.4.2 CONDITIONAL USAGE AND SECURITY

The access right of the referenced key related to this command must be fulfilled prior using the command.

6.6.4.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	‘2A’
P1	‘80’
P2	‘86’
Lc	Variable
Data In	EC: ‘xx…xx’ cryptogram = Y_G – no padding(uncompressed according to section 12.7.5 in [EN-419212-1])
Le	‘00’

Table 103 - PSO – Decipher command encoding using ECC

Note: Cryptogram in the data field is encoded as 0x04 || X || Y where X and Y are the coordinates of the point on the curve.

6.6.4.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE		
Data Out	Common secret (ZZ)	
SW1 SW2	‘6700’	The enciphered data length doesn’t match with the decipher key length
	‘6A80’	Bad data format in the data field
	‘6A88’	Current SE problem
	‘6A86’	Incorrect P1-P2
	‘6982’	Decipher key access conditions not fulfilled, or extraction error
	‘6985’	MSE SET might not have performed with correct data

Table 104 - PSO – Decipher response encoding using ECC

6.7 FILE MANAGEMENT SYSTEM

The commands specification can be found in 3.1.

6.8 DATA OBJECT COMMAND

6.8.1 GET DATA

This section contains ID.me specific GET DATA extended header list for the following functionalities:

- GET DATA for ID.me authentication services configuration
- GET DATA for ID.me cryptographic services configuration
- GET DATA for ID.me version
- GET DATA for ID.me security status retrieval

6.8.1.1 DEFINITION AND SCOPE

This command allows retrieving the whole DOUP or DOCP of a SDO or a SE. Incoming data for this command is an extended header list, being a constructed TAG only. Some SDO parameters can be retrieved while some other cannot, depending on the type of the object, refer to **Error! Reference source not found.** for more information.

This command includes an important modification from [IAS-ECC], as it also allows retrieving the card security status.

In the case where the command is recognized as security status retrieval, the response APDU will contain the list of CRTs related to the performed security operations.

All the SDO references (BFXXYY) in the command data field has to be written with the global/local bit set to 0. The SDO referenced will be searched only under the current ADF/DF.

6.8.1.2 CONDITIONAL USAGE AND SECURITY

The GET DATA command can be performed only if the security status satisfies the security conditions defined by the application within the context for the function.

6.8.1.3 COMMAND MESSAGE

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'CB’
P1	‘3F’ – current DF
P2	‘FF’ – current DF
Lc field	Length of command data field
Data field	Extended header list: ‘4D’ L_{4D} {References of the data to retrieve}
Le field	Variable

Table 105: SDO Get Data command coding

NOTE: The extended header list is to be formed in a way that complete DOCP & DOUP is to be returned in response of get data command for SDO's. Partial retrieval of specific tags within DOCP & DOUP is not supported.

NOTE: The compulsory algorithm TLV (tag '80') will always be present in the DOCP returned by this command even if it was not originally inserted in the SDO (tag '80' is optional). In the case where this TAG was not present in the SDO creation this command will return '00' as a value for the TAG '80' (meaning no compulsory algorithm is set).

NOTE: In case some of the bits of the compulsory algorithm identifier are reserved for future use, they all will be set to the recommended value `0b` within the returned algorithm identifier value.

6.8.1.4 GET DATA FOR ID.ME CRYPTOGRAPHIC CONFIGURATION

Lc command is '06'.

TAG			LENGTH	VALUE
'4D'			'04'	Extended Header List
	'70'		'02'	Inter industry template for further objects
		'C1'	'80'	Cryptographic services

Table 106 - Tag 4D for activated cryptographic services reading

The response (3 bytes) gives the supported algorithms, RSA support, and EC support.

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
1	-	-	-	-	-	-	-	Support 3DES Cipher
-	1	-	-	-	-	-	-	Support AES Cipher
-	-	1	-	-	-	-	-	Support RSA Cipher
-	-	-	1	-	-	-	-	Support SHA512 Digest
-	-	-	-	1	-	-	-	Support SHA384 Digest
-	-	-	-	-	1	-	-	Support SHA256 Digest
-	-	-	-	-	-	1	-	Support SHA224 Digest
-	-	-	-	-	-	-	1	Support SHA1 Digest

Table 107 - Algorithm support coding

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
X	-	-	-	-	-	-	-	RFU
-	X	-	-	-	-	-	-	RFU
-	-	X	-	-	-	-	-	RFU
-	-	-	1	-	-	-	-	Support RSA SHA512 Signature
-	-	-	-	1	-	-	-	Support RSA SHA384 Signature
-	-	-	-	-	1	-	-	Support RSA SHA256 Signature
-	-	-	-	-	-	1	-	Support RSA SHA224 Signature
-	-	-	-	-	-	-	1	Support RSA SHA1 Signature

Table 108 - RSA signature support coding

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
X	-	-	-	-	-	-	-	RFU
-	X	-	-	-	-	-	-	RFU
-	-	X	-	-	-	-	-	RFU
-	-	-	1	-	-	-	-	Support EC SHA512 Signature
-	-	-	-	1	-	-	-	Support EC SHA384 Signature
-	-	-	-	-	1	-	-	Support ECSHA256 Signature
-	-	-	-	-	-	1	-	Support EC SHA224 Signature
-	-	-	-	-	-	-	1	Support ECSHA1 Signature

Table 109 - EC Signature support coding

6.8.1.5 GET DATA FOR ID.ME AUTHENTICATION CONFIGURATION

Lc command is ‘06’.

TAG			LENGTH	VALUE
‘4D’			‘04’	Extended Header List
	‘70’		‘02’	Inter industry template for further objects
		‘C2’	‘80’	Authentication services

Table 110 - Tag 4D for activated authentication services reading

Response is on one byte, with the following coding:

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
X	-	-	-	-	-	-	-	RFU
-	X	-	-	-	-	-	-	RFU
-	-	X	-	-	-	-	-	RFU
-	-	-	X	-	-	-	-	RFU
-	-	-	-	X	-	-	-	RFU
-	-	-	-	-	1	-	-	Support SAC Generic Mapping
-	-	-	-	-	-	1	-	Support SAC Integrated Mapping
-	-	-	-	-	-	-	1	Support MOC

Table 111 - Authentication support byte 1 coding

6.8.1.6 GET DATA FOR SECURITY STATE

Lc command is ‘06’.

TAG			LENGTH	VALUE
‘4D’			‘04’	Extended Header List
	‘70’		‘02’	Interindustry template for further objects
		‘C4’	‘80’	Security status

Table 112: Tag 4D for security status retrieval

6.8.2 PUT DATA

6.8.2.1 DEFINITION AND SCOPE

PUT DATA command is applicable only for SDO, and can be used for updating an existing SDO, or creating a new one, except SE which cannot be created.

This command includes an important modification from [IAS-ECC], as it also allows unsetting the card security status.

In the case where the command is recognized as security status reset, the data APDU will contain the CRT related to the security operations to reset.

Note: The PUT DATA command to update the value of a biometric SDO will reset its retry counter to its maximum value.

All the SDO references (BFXXYY) in the command data field has to be written with the global/local bit set to 0. The SDO referenced will be searched only under the current ADF/DF.

SDO of type key cannot be updated with key of a different size than the present SDO. Nevertheless the SDO can be deleted and recreated with a different key length.

6.8.2.2 CONDITIONAL USAGE AND SECURITY

The PUT DATA command can be performed only if the security status satisfies the security conditions defined by the application within the context for the function.

6.8.2.3 COMMAND MESSAGE

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'DB'
P1	'3F' – current DF
P2	'FF' – current DF
Lc field	Length of command data field
Data field	Data to put BER-TLV formatted for classic get data, list of CRTs to unset for reset security status
Le field	None

Table 113: SDO Put Data command coding

Example of valid data fields for classic put data:

- **Updating a RSA public key**:

TAG				LENGTH	VALUE
'70'				'L70'	Interindustry template for further objects
	'BFA001'			'LBFA001'	RSA Public Key
		'7F49'		'L7F49'	DOUP
			'81'	'LN'	Modulus
			'82'	'Le'	Public Exponent
			'5F20'	'LCHR'	Certificate Holder Reference

Table 114: Command data field for public RSA key put data

- **Updating a RSA private key**:

TAG				LENGTH	VALUE
'70'				'L70'	Interindustry template for further objects
	'BF9001'			'LBF9001'	RSA Private Key
		'7F48'		'L7F48'	DOUP
			'92'	'Lp'	P
			'93'	'Lq'	Q
			'94'	'Lq-1'	Q-1 mod P
			'95'	'Ldp'	dP
			'96'	'Ldq'	dQ

Table 115: Command data field for private RSA key put data

- **Unsetting a security status:**

Unsetting the security status with the class byte of the SDO ID set to '84' will also unset the SM associated security status.

See 0 for CRT code Algorithm code.

TAG			LENGTH	VALUE
'70'			'L70'	Interindustry template for further objects
	'C4'		'Lc4'	Security Status
		'80'	01	CRT code Algorithm code
		'83'	02	SDO ID

Table 116: Command data field for Unsetting a security status

➢ **Delete SDO**:

TAG				LENGTH	VALUE
'70'				'06'	Interindustry template for further objects
	'BFXXXX'			'02'	Reference of the SDO to DELETE see 3.2.1
		A0		'00'	Length '00' Means delete SDO

Table 117: Command data field delete SDO command

Note: in case the non-repudiation flag is set for the SDO to be deleted, the deletion of the SDO will not cause the authentication state to be unset.

6.8.2.4 RESPONSE MESSAGE AND STATUS CONDITIONS

RESPONSE PARAMETER	MEANING
Data Field	None
SW1 - SW2	'6700' - Wrong length; no further indication '6982' - Security status in incoming data not satisfied '6A86' - Incorrect parameters P1-P2 '6A80' - Incorrect parameters in the command data field '6A88' - Reference data not found '6985' - Conditions of use not satisfied '6A81' - Function not supported – data cannot be put

Table 118: SDO Put Data command response

6.8.3 GENERATE ASYMMETRIC KEY PAIR

6.8.3.1 DEFINITION AND SCOPE

This command allows generating an asymmetric key pair. Incoming data for this command is an extended header list, being a constructed TAG only.

6.8.3.2 CONDITIONAL USAGE AND SECURITY

Only the private key shall be stored into the card, the public key should be retrieved by the Get Data APDU command just after the generation.

6.8.3.3 COMMAND MESSAGE

COMMAND	
CLA	ISO
INS	'47'
P1	'00'
P2	'00'
Lc	Length of command data field = '05'
Data In	Inter industry template: '70' L_{70} {Tag of the SDO private portion referencing the key pair to generate}
Le	Variable

Table 119 - GENERATE ASYMMETRIC KEY PAIR command encoding

RESPONSE		
Data Out	None	
SW1 SW2	'6700'	The enciphered data length doesn't match with the decipher key length
	'6A80'	Padding verification error or deciphered key length is longer than 40% of the key length
	'6A88'	Current SE problem
	'6A86'	Incorrect P1-P2
	'6982'	Decipher key access conditions not fulfilled, or extraction error

Table 120 - GENERATE ASYMMETRIC KEY PAIR response encoding

6.9 MANAGE SECURITY ENVIRONMENT COMMANDS

6.9.1 MSE RESTORE

The MSE RESTORE command is not supported by the ID.me applet.

6.9.2 MSE SET

The MSE SET command specification can be found in 3.3.2.

7 ANNEX

7.1 SUMMARY OF AVAILABLE CRYPTOGRAPHIC SERVICES

All the available cryptographic services are summarized in the following table, with its matching algorithm identifier to use:

Cryptographic service	Matching APDU	Cryptographic features	Algo. ID
Symmetric authentication for secure channel opening.	GET CHALLENGE / MUTUAL AUTHENTICATE	eSign-K symmetric authentication scheme. Key derivation function using SHA-1	‘0C’
		eSign-K symmetric authentication scheme. Key derivation function using SHA-256	‘8C’
		eSign-K symmetric authentication scheme based on AES. Key derivation function using SHA-256	‘FF40104C’
Client/server authentication	INTERNAL AUTHENTICATE Client / Server	PKCS#1 v1.5 RSA signature calculation – No internal hash computation.	‘02’
		PKCS#1 v2.1 RSA signature calculation – SHA-1	‘03’
		ECDSA signature calculation – SHA-1	‘04’
Digital signature computation	PSO Hash followed by PSO COMPUTE DIGITAL SIGNATURE	RSA PKCS#1v1.5 – SHA-1.signature Digital signature computation	‘12’
		RSA PKCS#1v1.5 – SHA-224.signature Digital signature computation	‘32’
		RSA PKCS#1v1.5 – SHA-256.signature Digital signature computation	‘42’
		RSA PKCS#1v1.5 – SHA-384.signature Digital signature computation	‘52’
		RSA PKCS#1v1.5 – SHA-512.signature Digital signature computation	‘62’
		RSA PKCS#1v2.1– SHA-1.signature Digital signature computation	‘15’
		RSA PKCS#1v2.1– SHA-224.signature Digital signature computation	‘35’
		RSA PKCS#1v2.1– SHA-256.signature Digital signature computation	‘45’
		RSA PKCS#1v2.1– SHA-384.signature Digital signature computation	‘55’
		RSA PKCS#1v2.1– SHA-512.signature Digital signature computation	‘65’
		ECDSA with SHA-1	‘FF110800’
		ECDSA with SHA-224	‘FF130800’
		ECDSA with SHA-256	‘FF140800’

Cryptographic service	Matching APDU	Cryptographic features	Algo. ID
		ECDSA with SHA-384	'FF150800'
		ECDSA with SHA-512	'FF160800'
Ciphering key decryption	PSO DECIPHER	RSA PKCS#1 v1.5 message decryption	'1A'
		RSA PKCS#1 v2.1OAEP message decryption	'2A'
		ECDH	'FF300400'
Role authentication (symmetric)	GET CHALLENGE / EXTERNAL AUTHENTICATE FOR ROLE AUTHENTICATION	3DES CBC EDE 128 bits (encipherment) + Retail MAC	'1C'
		AES CBC 128 bits (encipherment) + C-MAC	'FFA01200'
Hash calculation SHA-1 '10' within the digital signature sequence	PSO Hash (prior to PSO COMPUTE DIGITAL SIGNATURE)	SHA-1	'10'
		SHA-224	'30'
		SHA-256	'40'
		SHA-384	'50'
		SHA-512	'60'
Device Authentication with privacy protection	GENERAL AUTHENTICATE for Device Authentication with privacy protection / GET CHALLENGE / EXTERNAL AUTHENTICATE for Device Authentication with privacy protection / INTERNAL AUTHENTICATE for Device Authentication with privacy protection /	ECDH key agreement Certificate verification Digital signature generation Digital signature verification	**Error! Reference source not found.**

Table 121: Available services

7.2 ALGORITHM VALUES

VALUE	USAGE	SDO
‘0C’	symmetric mutual authentication based on 3DES with SHA-1	Symmetric key sets
‘8C’	symmetric mutual authentication based on 3DES with SHA-256	Symmetric key sets
'40'	symmetric mutual authentication based on AES with SHA-256	Symmetric key sets
‘1C’	Role authentication based on 3DES	Symmetric key sets
‘A0’	Role authentication based on AES	Symmetric key sets
‘1E’/’FFA14000’	Role authentication based on RSA SHA-1	Asymmetric key sets
‘FFA34000’	Role authentication based on RSA SHA-224	Asymmetric key sets
‘9E’/’FFA44000’	Role authentication based on RSA SHA-256	Asymmetric key sets
‘FFA54000’	Role authentication based on RSA SHA-384	Asymmetric key sets
‘FFA64000’	Role authentication based on RSA SHA-512	Asymmetric key sets
‘A1’	Mutual authentication based on SAC ECDH GM 3DES	EC private keys
‘A2’	Mutual authentication based on SAC ECDH GM AES192	EC private keys
‘A3’	Mutual authentication based on SAC ECDH GM AES192	EC private keys
‘A4’	Mutual authentication based on SAC ECDH GM AES256	EC private keys
‘C1’	Mutual authentication based on SAC ECDH IM 3DES	EC private keys
‘C2’	Mutual authentication based on SAC ECDH IM AES128	EC private keys
‘C3’	Mutual authentication based on SAC ECDH IM AES192	EC private keys
‘C4’	Mutual authentication based on SAC ECDH IM AES256	EC private keys
‘00’	algorithm identifier for PIN	PIN
‘02’	algorithm identifier for BIO	BIO
Var.	algorithm identifier for Device Authentication with privacy protection as specified in Table 123	EC/RSA keys
Var.	algorithm identifier for Terminal Authentication version 2 as specified in Table 124	EC public keys
Var.	algorithm identifier for Chip Authentication version 2 as specified in Table 125	EC private keys

Table 122: Algorithm Values

Byte 1 (leftmost)								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Signification**
1	1	1	1	1	1	1	1	Escape value
Byte 2								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Signification**
X	**X**	**X**	**X**	-	-	-	-	**Function**
0	1	0	0	-	-	-	-	Device Authentication protocol
-	-	-	-	**X**	**X**	**X**	**X**	**Hash type**
-	-	-	-	0	0	0	1	SHA-1
-	-	-	-	0	0	1	1	SHA-224
-	-	-	-	0	1	0	0	SHA-256
-	-	-	-	0	1	0	1	SHA-384
-	-	-	-	0	1	1	0	SHA-512
Byte 3								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Signification**
X	**X**	**X**	**X**	-	-	-	-	**Authentication protocol and related mechanisms**
0	1	0	1	-	-	-	-	Device Authentication with privacy protection (ECDH)
Byte 4 (rightmost)								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Signification**
X	**X**	**X**	**X**	-	-	-	-	**Type and length of SM keys to be generated**
0	0	1	0	-	-	-	-	TDES – Retail MAC
0	1	0	0	-	-	-	-	AES 128 - CMAC
0	1	1	0	-	-	-	-	AES 192 - CMAC
1	0	0	0	-	-	-	-	AES-256 - CMAC
-	-	-	-	**X**	-	-	-	**SSC usage for encryption**
-	-	-	-	1	-	-	-	Include SSC for encryption
-	-	-	-	-	**X**	**X**	**X**	**Function to be used for key derivation**
-	-	-	-	-	0	0	1	SHA-1
-	-	-	-	-	1	0	0	SHA-256

Table 123: - Device Authentication with privacy protection–Algorithm identifiers

Object Identifier	Value
id-TA-ECDSA-SHA-1	'04007F00070202020201'
id-TA-ECDSA-SHA-224	'04007F00070202020202'
id-TA-ECDSA-SHA-256	'04007F00070202020203'
id-TA-ECDSA-SHA-384	'04007F00070202020204'
id-TA-ECDSA-SHA-512	'04007F00070202020205'

Table 124: Algorithm identifiers for Terminal Authentication version 2

Object Identifier	Value
id-CA-ECDH-3DES-CBC-CBC	'04007F00070202030201'
id-CA-ECDH-AES-CBC-CMAC-128	'04007F00070202030202'
id-CA-ECDH-AES-CBC-CMAC-192	'04007F00070202030203'
id-CA-ECDH-AES-CBC-CMAC-256	'04007F00070202030204'

Table 125: Algorithm identifiers for Chip Authentication version 2

7.3 SUPPORTED EC DOMAIN PARAMETERS

The following table describe the supported EC domain parameters:

Domain parameters
ANSSI FRP256v1 (specified in [PRMD1123151V])
Brainpool192r1
Brainpool224r1
Brainpool256r1
Brainpool320r1
Brainpool384r1
Brainpool512r1
NIST P-192 (secp192r1)
NIST P-224 (secp224r1)
NIST P-256 (secp256r1)
NIST P-384 (secp384r1)
NIST P-521 (secp521r1)

Table 126 - Domain parameters

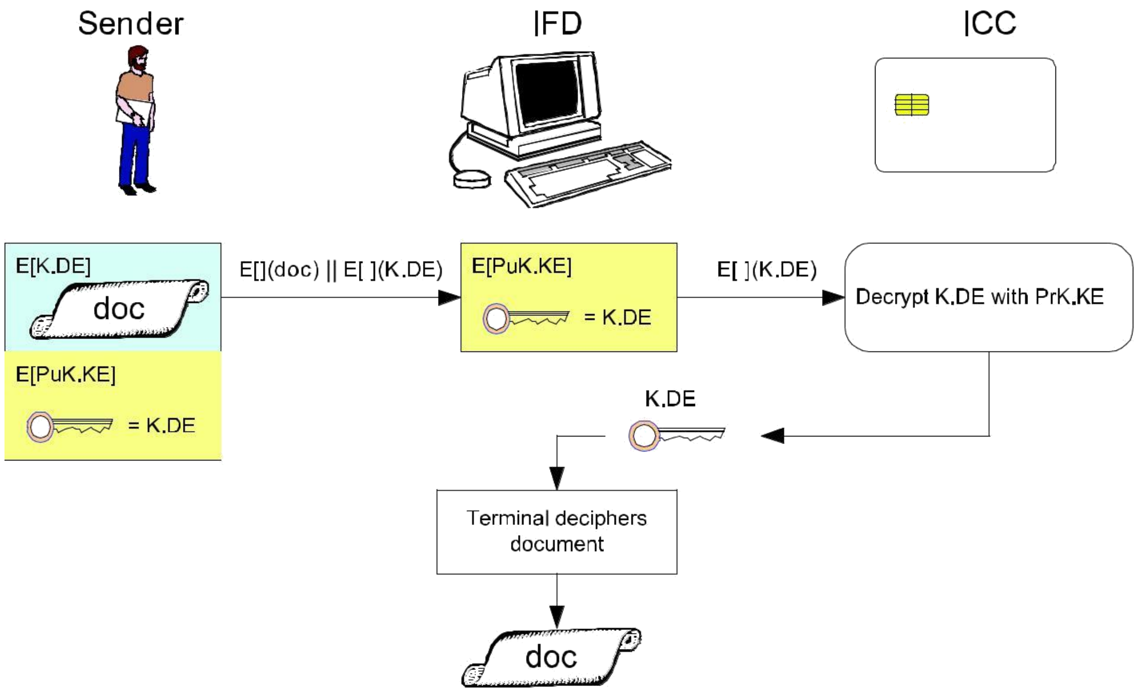
7.4 SYMMETRIC KEY TRANSMISSION BETWEEN A REMOTE SERVER AND THE ICC

7.4.1 Steps preceding the key transport

The steps preceding a key transport are application specific. Hence this section does not mandate the existence of those steps.

7.4.2 Key encryption with RSA

Encryption key decipherment is needed when an application receives a (symmetric) message key enciphered with a public key (PuK.CH.KE) that corresponds to a private key in the ICC. The IFD sends the encrypted symmetric key to the ICC. The ICC decipher it using the private key and return the plain symmetric key. The IFD may then use the



decrypted symmetric key to decipher the attached message.

Figure 4: Key decipherment and document decipherment

The figure shows that the process of key decipherment (performed by the ICC) for the purpose of a consecutive document decipherment (not performed by ICC). The following roles are involved in the document decipherment process.

Sender: wants to send a document to the IFD (receiver) The sender knows the document encryption key K.DE and uses it to encrypt the document. He also encrypts that key with the ICC’s public key (PuK.KE)

Receiver: (IFD) wants to decipher the document. The IFD does not have the encryption key K.DE not can it decrypt the key. Therefore the IFD strips the encrypted E[PuK.KE](K.DE) from the package and sends it to the ICC.

After it has received the encryption key K.DE from the ICC, it uses K.DE in order to decipher the document.

ICC: owns the PrK.KE and uses it in order to decrypt the K.DE sent by the IFD
The key decipherment service is used for the following security protocol.

- Application level encryption key decipherment
- Application level agreement
- Key agreement or key exchange in PK Kerberos' pre authentication phase
- Electronic vault

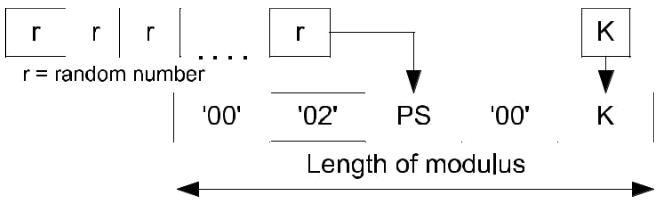
For the confidential document exchange, the following scheme is applied:

- Key transport is organised by enciphering the symmetrical content encryption key with the corresponding ICC's PuK.CH.KE
- Document enciphering with a symmetrical algorithm (e.g. TDES or DES)

If an enciphered document is sent, the ICC is not involved: the sender's software computes the content encryption key, encipher the document and finally enciphers the content encryption key by applying the ICC's public key taken from the ICC's KE certificate.

The ICC's part in confidential document exchange is only the decryption of the symmetrical content encryption keys using the PSO: DECIPHER command.

7.4.3 PKCS#1 V1.5 padding



The encryption key K shall be formed according to PKCS#1, version 2.1, Chapter 7.2.1 "EME –PKCS1-V1_5".The figures shows the PKCS encoding.

Figure 5: PKCS encoding of symmetric key K

Formatting according to PKCS#1

'00 02' || PS || '00' || K

Where PS is a byte string consisting of pseudo randomly generated non-zero bytes. The formatted byte string shall consist of n bytes where n is the length in bytes of the modulus of the private key for decryption.

7.4.4 OAEP padding

OAEP padding may be used to transport the key K. OAEP padding is described in PKCS#1, v2.1, Chapter 9.1.1.1[2] and computes as follows

OAEP(M) = '00' || (**MGF1**(maskedDB, seed.Length) **xor** *seed*) || maskedDB

With

maskedDB = (Hash(L) || '00' ...'00' '01' || K) **xor** **MGF1**(*seed*, DB.length)

The encoding format according to PKCS#1 V2.1 has the structure according the figure below

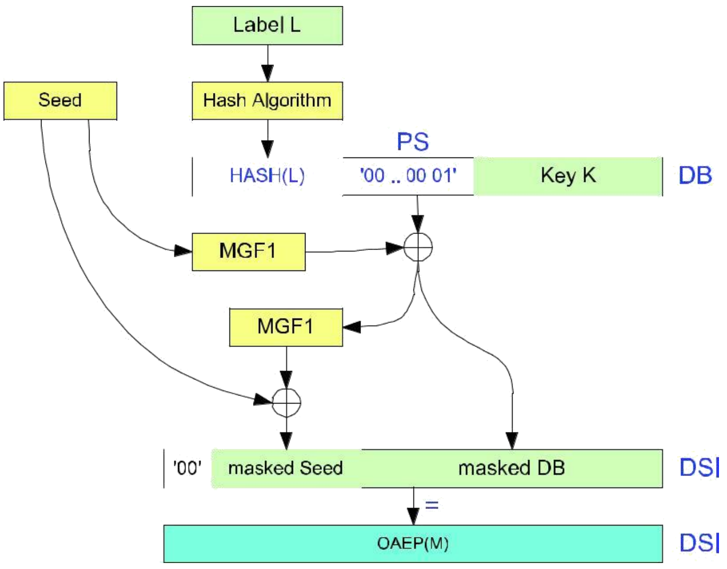


Figure 6: Example of 2048 bit encoding according to PKCS#1 v2.1

Where:

L: is an optional label to be assigned with the message; the default value for L – if L is not provided – is an empty string. For ESIGN use L is always the empty string "".

Seed: is a random number of length Length[Hash(L)]

Key: is the key to be transmitted.

the actual coding and data setup shall be taken from PKCS#1, V2.1, chapter 7.1[2]

7.4.5 Execution Flow

The figure below shows the execution flow of the key decipherment protocol.

Step	IFD	Trans-mission	ICC
1	MSE:SET PrK.CH.KE	→ ←	OK
2	PSO:DECIPHER	→ ←	decipher remove padding return key

Figure 7: Key decipherment flow

7.4.5.1 STEP1 – SET DECIPHERING KEY

Before key decipherment can be performed, the secret keys has to be selected with the MSE command.

Command Parameter	**Meaning**
CLA INS	according to ISO/IEC 7816-4 '22' MANAGE SECURITY ENVIRONMENT
P1 P2	'41' MSE SET 'B8' CT (confidentiality template)
Lc field	'xx'
Data field	CRT data field with KeyRef (PrK.CH.KE)

Figure 8: Select Key for decipher operation – Command APDU

For the structure and content of the APDU refer to ISO/IEC 7816-4:2013.

The response data field is empty

Response Parameter	**Meaning**
Data field	empty
SW1-SW2	Refer to ISO/IEC 7816-4

Figure 9: Select Key for decipher operation – Response

7.4.5.2 STEP 2 – DECIPHER KEY

After the key is seleceted, the decipher opertaion can be executed

Command Parameter	**Meaning**
CLA INS	according to ISO/IEC 7816-4 '2A' PSO: DECIPHER
P1 P2	'80' return plain value '86' Enciphered data present
Lc field	'xx'
Data field	'xx' PI 'xx..xx' cryptogram padded according to PI
Le field	'00'

Figure 10: Decipher operation – Command APDU

for the structure and content of the APDU refer ISO/IEC 7816-8:2004

Response Parameter	Meaning
Data field	'xx..xx' content-encryption key (padding already removed)
SW1-SW2	Refer to ISO/IEC 7816-4

Figure 11: Decipher operation – Response

7.5 DIFFIE-HELLMAN KEY EXCHANGE FOR KEY ENCIPHERMENT

This section will consider both Diffie-Hellman key exchange (DH) according to IETF RFC 2631 and Elliptic curve Diffie-Hellman key exchange (ECDH) according to IETF RFC 5753.

There is no corresponding mechanism available in the existing ISO/IEC 7816 command set the represent the DH key exchange in this context. The DECIPHER command has been chosen for the mechanism.

The figure below shows the process for a document decipherment.

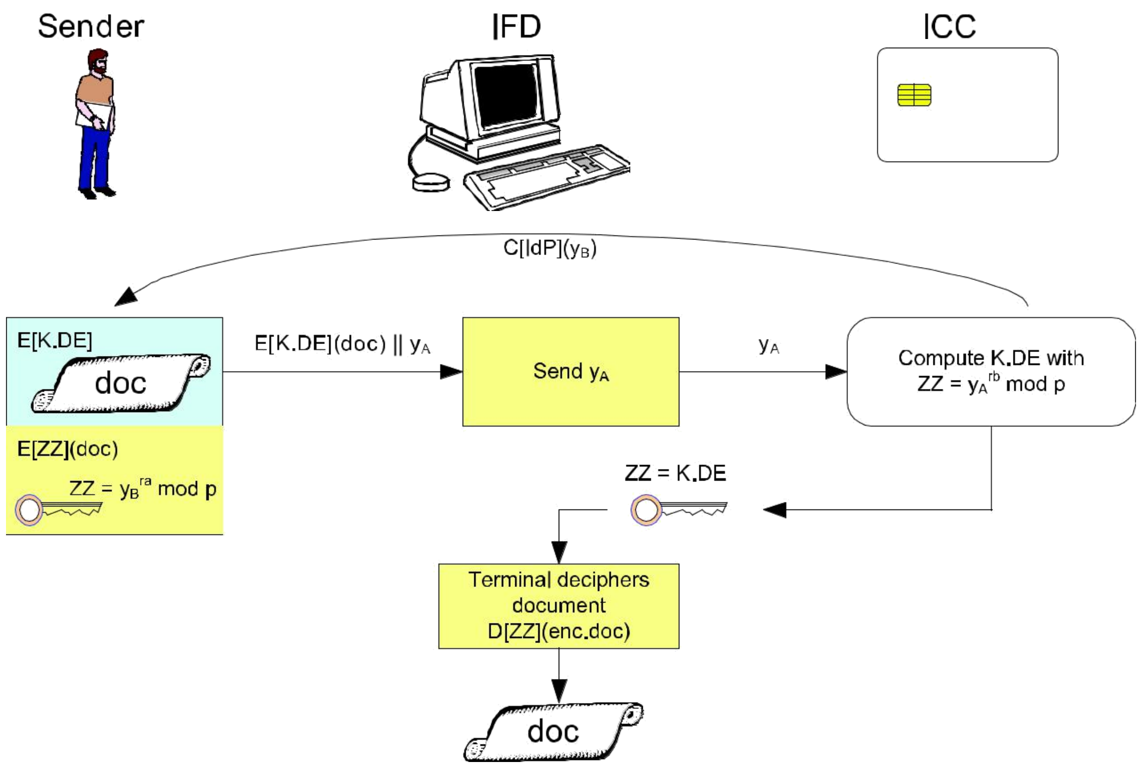


Figure 12: DH/ECDH key freshness generation and document decipherment (DH case)

Sender: wants to send a document to the IFD (receiver). The sender knows the ICC's certifies public key yb (Yb for EC) and uses it to derive ZZ to be used for the encryption of the document.

Together with the encypted document he sends his public key ya (Ya for EC) which was newly generated through ra for freshness reasons.

Receiver: (IFD) wants to decipher the document. Since the ICC does not have the sender's public key ya (Ya) it cannot generate ZZ. Therefore IF sends the sender's public key ya (Ya) to the ICC.

ICC: owns the number rb and therefore can calculate the common secret ZZ in order to decipher the document. It returns the common secret ZZ to the IFD.

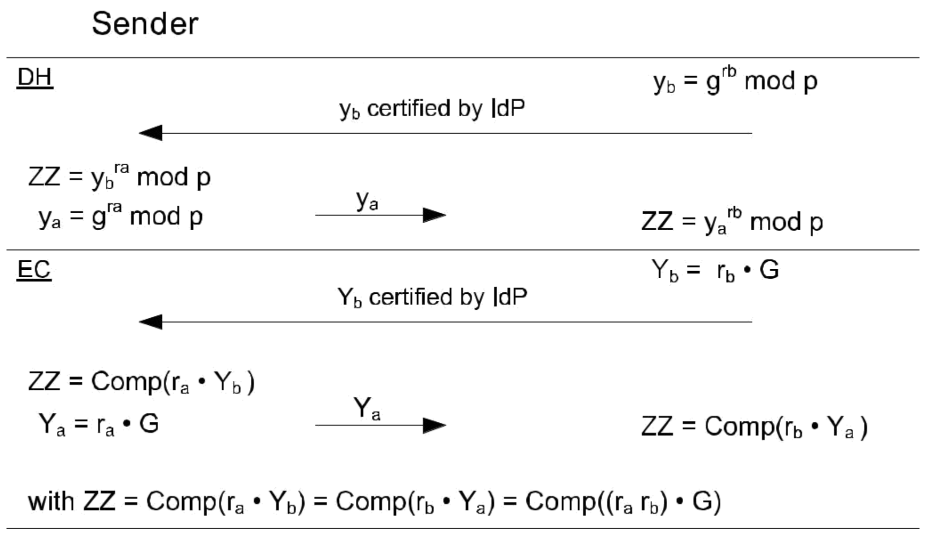


Figure 13: (EC) DH key exchange

The Figure 13 shows the life cycle of the key tokens y_a, y_b, (Y_a, Y_b). The ICC's key token $y_b(Y_b)$ is generated once. y_b (Y_b) is exported and certified and is made available to any sender that wants to send something to an IFD that uses the ICC.

For the document deciphering process freshness is provided by the sender's r_a on each document encryption.

In Figure 13, the crypto logical perspective is repeated for the process where
The sender's public key is made of

DH: $y_a = g^{ra} \bmod p$

EC: $Y_a = r_a \bullet G$

The ICC's certified public key is made of

DH: $y_b = g^{rb} \bmod p$

EC*:* $Y_b = r_b \bullet G$

It does not necessarily need to exist in the ICC after creation. The ICC keep r_a to allow the creation of ZZ.

ra: is the sender's ephemeral private (EC)DH key (randomly generated for freshness)
rb: is the ICC's static private (EC)DH key (generated once)

It can be summarized as follows:
The document is encrypted by a content encryption key, which is generated at random.
The recipient public key and the sender's private key are used to generate a symmetric key. Then the content-encryption key is encrypted in this symmetric key. The result ZZ is sent to the IFD for further derivation of keys.

Care shall be taken in order to protect the recipient's secret key against the so called "small subgroup" attacks. Refer to RFC 2785.

7.5.1 Execution Flow

The figure below shows the execution flow of the key decipherment protocol.

Step	IFD	Trans-mission	ICC
1	MSE:SET PrK.CH.KE	→ ←	OK
2	PSO:DECIPHER	→ ←	establish common secret ZZ OK

Figure 14: Key Decipherment flow

7.5.1.1.1 STEP 1: SELECT DH ENCRYPTION KEY

First the secret key PrK.CH.KE has to be selected with the MSE command.

Command Parameter	Meaning
CLA	according to ISO/IEC 7816-4
INS	'22' MANAGE SECURITY ENVIRONMENT
P1	'41' MSE SET
P2	'B8' CT (confidentiality template)
Lc field	'xx'
Data field	CRT data field with KeyRef (PrK.CH.KE)

Figure 15: Select key for DH key exchange – Command APDU

For the structure and content of the APDU refer to ISO/IEC 7816-4:2013.

The response data field is empty.

Response Parameter	Meaning
Data field	empty
SW1-SW2	Refer to ISO/IEC 7816-4

Figure 16: Select key for DH key exchange – Response

7.5.1.1.2 STEP 2: DERIVATION OF THE SHARED KEY

After the key is set, the derivation of the shared secret can be done. The computed value ZZ is sent in the response.

eNOTE: The DECIPHER operation does not decrypt the document, it neither decrypts the key that decrypts the document or generates a shared secret, that is used to derive the decryption key.

Command Parameter	Meaning
CLA	according to ISO/IEC 7816-4
INS	‘2A’ PSO:DECIPHER
P1	‘80’ return plain value
P2	‘86’ cryptogram (ISO/IEC 7816-4)
Lc field	‘xx’
Data field	‘00’ PI **DH**: ‘xx..xx’ cryptogram = y_G — no padding **EC**: ‘xx..xx’ cryptogram = Y_G — no padding (uncompressed see 12.6 in EN 419212-1:2014)
Le field	‘00’

Figure 17: Decipher operation – Command APDU

For the structure and content of the APDU refer to ISO/IEC 7816-8:2004.
The response data field contains the result ZZ of the deciphering operation

Response Parameter	Meaning
Data field	‘xx..xx’ ZZ
SW1-SW2	Refer to ISO/IEC 7816-4

Figure 18: Decipher operation - Response