EUROPEAN CARD FOR e-SERVICES AND NATIONAL e-ID APPLICATIONS

IAS ECC

Identification Authentication Signature

European Citizen Card

Technical Specifications

Revision: 1.0.1

Document writing – Technical team:

Company	Name	eMail
gemalto	Olivier JOFFRAY	OLIVIER.JOFFRAY@GEMALTO.COM
	Georges DEBOIS	GEORGES.DEBOIS@GEMALTO.COM
Oberthur Technologies	Benoit COLLIER	B.COLLIER@OBERTHUR.COM
	Alban FERAUD	A.FERAUD@OBERTHUR.COM
	Sophie AGRANIOU	S.AGRANIOU@OBERTHUR.COM
Sagem Orga SAFRAN Group	Antoine NAMOUR	ANTOINE.NAMOUR@SAGEM.COM
	Christophe LELANDAIS	CHRISTOPHE.LELANDAIS@SAGEM.COM
THALES	Jean-Pierre LAFON	JEAN-PIERRE.LAFON@THALESGROUP.COM

REVISION	MODIFICATION
V1.0	ORIGINAL VERSION JANUARY 31ST 2008
V1.0.1.A	RESOLUTION OF COMMENTS MARCH 21ST 2008
V1.0.1.B	RESOLUTION OF COMMENTS FEBRUARY 10TH 2009
V1.0.1	RESOLUTION OF COMMENTS FEBRUARY 23RD 2009

Table of contents

List of tables

List of figures

1 INTRODUCTION

1.1 Purpose of the document

The Gixel group is presenting in this document the interoperable technical specification for implementing national e-ID card combining e-services and national e-ID applications (volet régalien).

This technical specification is called IAS ECC (Identification Authentication Signature for European Citizen Card) – formerly called IAS v2 and it is based on International and European standards:

- CEN TS 15480 mandatory features for application (See [R17])
- prEN 14890-1:2006 and prEN 14890-2:2006 (See [R15] and [R16])
- ISO 7816 series(See [R1], [R2], [R3], [R4], [R5] and [R6]) and ISO 14443 series ([R20])
- EAC V1.1.1 (See [R19])

The Gixel members are key contributors to the WG15 TC224 workshop on ECC, being employees on key positions:

- Chairmanship of the TC 224 / WG15
- Editorship of CEN/prTS 15480 part 3

The Gixel members are also key contributors to others international standardization bodies such as: Global Platform, Resident European Card, Electronic European Health Insurance Card (TC 251), BIG organization.

This document is first release of the IAS ECC technical specification and it covers:

- Electrical part compliant with CEN TS 15480-2, the so called IAS ECC
- European Interoperable Middleware, which will be fully based on the upcoming CEN TS 15480-3 specifications - to be published in 2009
- CEN TS 15480-4, upcoming usage profile(s) from the European Members states or instances.

In order to not duplicate technical specifications:

- The present document will describe only the e-services part.
- The national e-ID part is based on EAC V1.1.1 and is not detailed here
- The European Middleware technical specification will be published at the CEN TS 15480-3
- Profiles for CEN TS 15480-4 should be provided for each use case.

Two main standards are used for the two main card applications:

- EAC V1.1.1 for national e-ID (secure storage of Biometric data)
- CEN TS 15480 series for European e-services applications

This version of the specification includes DCSSI[1] requirement for supporting ★★ and★★★ cryptographic certificates[2]. It integrates the complementary information that is required for developing the test suite to be used for product homologation.

One shall take into account that some of the here-enclosed functions are not mandatory for raising the ★★ signature creation device level, while they are mandatory to get ★★★ product certificate.

1.2 Design approaches

This specification does not mandate the use of a particular technology (e.g. native vs. Java Card).

For instance the Master File is optional (see ISO7816-4 [R2]). In this specification, some examples or figures may include a graphical representation of a Master File, but they are purely for illustration purposes.

[1] - French government agency for cryptographic algorithm using rules and recommendations (plus other defense related functions).

[2] - ★★ and ★★★ cryptographic certificates is the French method for classifying the security level, thus the associated capabilities, of a cryptographic device for signature generation.

1.3 References

ISO/IEC References

ISO/IEC 7816: Information technology - Identification cards - Integrated circuit(s) cards with contacts

[R1] ISO/IEC 7816-3 (2004) - Part 3: Electronic signals and transmission protocols

[R2] ISO/IEC 7816-4 (2003) - Part 4: Inter-industry commands for interchange.

[R3] ISO/IEC 7816-6 (2004) - Part 6: Inter-industry data elements.

[R4] ISO/IEC 7816-8 (2004) - Part 8: Security related inter-industry commands.

[R5] ISO/IEC 7816-9 (2004) - Part 9: Additional inter-industry commands and security attributes.

[R6] ISO/IEC 7816-15 (2004) - Part 15: Cryptographic Information Application.

Other References

[R7] ISO/IEC 9796-2 (2002) - Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Mechanisms using a hash-function

[R8] ISO/IEC 9797-1 (1999) - Information technology - Security techniques – Message Authentication Codes (MACs) – Part 1 : Mechanisms using a block cipher

[R9] ISO/IEC 9797-2: (1994) - Information technology-Security techniques- Data integrity mechanism using a cryptographic check function employing a block cipher algorithm

[R10] FIPS 46-3: (1999) – Triple DES.

[R11] X9.52: (1998) – Mode of operation of triple DES.

[R12] ISO 10116: (1991) – Mode of operation of an n-bits cipher.

[R13] FIPS PUB 180-1 – Secure Hash standard.

RSA References

[R14] PKCS #1 v2.1: RSA Cryptography Standard - RSA Laboratories - June 14th 2002

CEN/ISSS WS/E-Sign Area K References

Application Interface for smart cards used as Secure Signature Creation Devices

[R15] prEN 14890-1 : 2006 /Part 1: Basic requirements

[R16] prEN 14890-2 : 2006 /Part 2: Additional services

[R17] CEN/TS 15480-2 - Part 2: European Citizen Card

[R18] CEN/TS 15480-1 - Part 1: European Citizen Card

Other references

[R19] Machine Readable travel document for extended Access control - Technical guidelines . TR 03111 v1.1.1

[R20] ISO/IEC 14443 : identification cards – contactless integrated circuit (s) cards – proximity cards

1.4 Abbreviations

ADF	Application Dedicated File
AID	Application IDentifer
AT	Authentication Template
AMB	Access Mode Byte
BER	Basic Encoding Rules
CA	Certification Authority
CCT	Cryptographic Checksum Template
CRT	Control Reference Template
CSE	Current Security Environment
CT	Confidentiality Template
DES	Data Encryption Standard
DF	Dedicated File
DOCP	Data Object Control Parameters
DST	Digital Signature Template
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
KAT	Control reference template for key agreement
MAC	Message Authentication Code
MF	Master File
MSE	Manage Security Environment
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 in case of a JavaCard implementation, the MF otherwise.
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 Environnement for Security Policy
SK	Secret key – Symmetric key based algorithm
SM	Secure Messaging
TLV	Tag Length Value
UQB	Usage Qualifier Byte

2 SMART CARD CHARACTERISTICS

2.1 ISO/IEC 7816-15 consideration

At personalization phase, ISO 7816-15 data are recorded onto the card. These data are not accessed by the card application(s). They are intended for interoperability purposes to inform the IFD about the way to access files and to handle cryptographic objects present in the card including SDO. These data reflect the rules governing all of part of the security objects hosted in the card, thereby enabling the IFD to perform transactions with the card.

According to ISO 7816-15 implementation, whenever a reference object is located in another instance, its path is related to this instance.

For more information about ISO 7816-15 implementation of the cryptographic information application in the [IAS ECC], see chapter 10.4.

2.2 Cryptographic mechanisms supported by the card

[IAS ECC] application supports the following cryptographic mechanisms:

What	Name	Size / Supported characteristics		
ASYMMETRIC ALGORITHM	RSA	1024 bits	1536 bits	2048 bits
HASH FUNCTION	SHA-1	160 bits		
	SHA-2	256 bits		
SIGNATURE	PKCS#1	RSASSA – PKCS1 – V1_5		
	ISO 9796-2	Scheme 1		
SECURE MESSAGING	3DES – EDE2	Confidentiality – mode CBC		
	9797-2	Integrity – padding 2 – algorithm 3		
KEY AGREEMENT	Diffie-Hellmann	1024 bits	1536 bits	2048 bits
DECRYPTION (EMAIL KEYS)	PKCS#1	RSAES – PKCS1 – V1_5		
C/S AUTHENTICATION	PKCS#1	EMSA – PKCS1 – V1_5		

Existing combinations between hash algorithms and miscellaneous calculation are described chapter §10.5.

2.3 ATR / ATS and EF.ATR

The ATR of the card shall follow the rules indicated in [R1] and [R2]. The ATS of the card shall follow the rules specified in ISO/IEC 14443-4.

Data objects for card identification shall be provided in the historical data bytes of the ATR/ATS and in a mandatory EF.ATR. The data Objects described hereafter for both the ATR/ATS and EF.ATR is mandatory.

Card Identification Data Objects table provides the list of data objects which may be supported by the card. Data Objects are BER-TLV formatted in an EF.ATR, while they use the compact TLV format in the historical data bytes.

2.3.1 Answer To Reset / Answer To Select

The ATR/ATS contains configuration data so that ICC and IFD can communicate together (protocol, speed, etc…). The following historical bytes are mandatory:

Byte #	What	Value	Meaning
1	**Category indicator**	**‘00’**	**COMPACT-TLV data objects followed by a status indicator shall be present as the last three historical bytes**
2	**Card service data TAG**	**‘31’**	**Tag for next byte**

3	Card service data byte	’B9’	b8=1: Application selection by full DF name
			b6=1: BER-TLV DO are present in EF.DIR
			b5=1: BER-TLV DO present in EF.ATR[3]
			b4...b2=100: EF.DIR/EF.ATR is a transparent EF (use READ BINARY)
			b1 = 0: Card with MF
			b1 = 1: Card without MF

4	**Pre-issuing DO TAG**	**‘64’**	**Tag for next 4 bytes**
5	IC manufacturer	‘XX’	IC Manufacturer according ISO/IEC 7816-6
6	Type of the IC	‘XX’	defined by the IC or card manufacturer
7	OS Version	‘ECCY’	IAS ECC version encoded as follows: • ECC to indicate the card is ECC compliant • Y encodes the version over the least significant nibble Then V1.0.1 = ‘ECC1’
8	Discretionary data		
9	**Card capabilities data TAG**	**‘73’**	**Tag for next 3 bytes**

[3] A DO may be present in EF.ATR for the purposes of compatible tag allocation scheme

Byte #	What	Value	Meaning
10	Card capabilities data byte 1 → *Selection method*	‘94’	**DF selection** b8=1: DF selection full name b5=1: DF selection using file identifier **EF selection** b3=1: file selection using short file identifier is supported
11	Card capabilities data byte 2 → *Data coding byte*	‘01’	b4...b1 = 0001: data unit size is 1 byte
12	Card capabilities data byte 3 → *Miscellaneous*	‘C0’, ‘80’, ‘D0’, ‘90’	b8=1: command chaining is supported b7=0: Extended Lc and Le fields NOT supported b7=1: Extended Lc and Le fields supported b5, b4=00 : no logical channel supported b5, b4=10 : channel number assignment by the card Maximum number of channels supported :4
13	**Status indicator TAG**	**‘82’**	**Tag for next byte**
14	Status indicator	‘9000’	Status word SW1-SW2

Figure 1: Card Identification historical bytes

The list and the order of the historical bytes are compulsorily unless further items complete these historical bytes according to the smart card manufacturer’s policy (e.g. reference and version of the application).

2.3.2 EF.ATR content

Data Object form being used in EF.ATR, the length of the DO shall be used as defined in the table. Furthermore the definitions for coding the content of the data objects given in the table are mandatory.

In order to identify the compatible tag allocation scheme and the authority responsible for the scheme, the interindustry template referenced by tag '78' is used by the present specification. The allocation authority shall be identified as CEN by an OID with the following root:

- 1.3.162 made up of level-1 for ISO (1), level-2 for identified-organization (3) and level-3 for CEN (162).

The specification number to be defined according to CEN conventions shall be added to this root to figure the full OID. The corresponding BER-TLV encoded OID shall be nested in the data object '06'.

The DO referenced by tag '78' may be hosted in EF.ATR while the historical byte denoting the Card Service Data shall has bit5 set to one to indicate that a DO is available in EF.ATR.

Alternatively, the D.O. ‘78’ can be hosted by an ADF whereby denoting that the tag allocation scheme applies to data objects within this ADF and not to the whole card

Tag	Length	What	Value	Meaning

Tag	Length	What	Value	Meaning
‘80’	**N.A.**	**Category indicator**	**‘00’**	**Indicate format of next historical bytes (compact TLV)**
‘43’	**‘01’**	**Card service data tag**		**Tag for next byte**

		Card service data byte	’B9’	b8=1: Application selection by full DF name
				b6=1: BER TLV DO are present in EF.DIR
				b5=1 BER-TLV DO present in EF.ATR[4]
				b4...b2=100: EF.DIR/EF.ATR is a transparent EF (use READ BINARY)
				b1 = 0: Card with MF
				b1 = 1: Card without MF

‘46’	**‘04’**	**Pre-issuing DO**		**Tag for next 4 bytes**
		IC manufacturer	‘XX’	IC Manufacturer according ISO/IEC 7816-6
		Type of the IC	‘XX’	defined by the IC or card manufacturer
		OS Version	‘XX’	Version of the operating system defined by card manufacturer
		Discretionary data	‘XY’	IAS ECC version Encoded as follows: • X encodes the major version over the 4 most significant bits • Y encodes the minor version over the least significant bits Then V1.0.1 = ‘10’
‘47’	**‘03’**	**Card capabilities tag**		**Tag for next 3 bytes**
		Card capabilities data byte 1 → *Selection method*	’94’	**DF selection** b8=1: DF selection full name b5=1: DF selection using file identifier **EF selection** b3=1: file selection using short file identifier is supported
		Card capabilities data byte 2 → *Data coding byte*	‘01’	b4...b1 = 0001: data unit size is 1 byte
		Card capabilities	‘C0’, ‘80’, ‘D0’, ‘90’	b8=1: command chaining is supported b7=0: Extended Lc and Le fields NOT

[4] A DO(s) may be present in EF.ATR for the purposes of compatible tag allocation scheme or other purposes

Tag	Length	What	Value	Meaning
		data byte 3 → *Miscellaneous*		supported b7=1: Extended Lc and Le fields supported b5, b4=00 : no logical channel supported b5, b4=10 : channel number assignment by the card Maximum number of channels supported :4
‘4F’	**‘01’…’10’**	**Application Identifier**	**’XX…XX’**	**AID of implicitly selected application (1 to 16 bytes)**
‘E0’	’10’	IO buffer size	‘XX…XX’	4 concatenated data objects with tag ‘02’ (Universal class). The value field of each DO encodes the maximum number of bytes of the respective APDU. ‘02’ L .xx .xx. = DO maximum length of **command** APDU **without** secure messaging ‘02’ L .xx .xx. = DO maximum length of **command** APDU **with** secure messaging ‘02’ L .xx .xx. = DO maximum length of **response** APDU **without** secure messaging ‘02’ L .xx .xx. = DO maximum length of **response** APDU **with** secure messaging.
‘78’	**‘08’**	**Allocation scheme tag**		**Tag for next 8 bytes**
		OID	‘06062B8122F87802’	OID of the allocation authority to be identified as CEN.
‘82’	**‘02’**	**Status indicator**		**Tag for next 2 bytes**
		Status Word	‘9000’	SW1+ SW2

Figure 2: Card Identification Data Objects

2.3.3 Case of a multi-applicative card: coexistence with a travel document application

If a travel document application is present on the card, the relevant bytes shall be added in the ATR/ATS.

3 FILE SYSTEM

3.1 Introduction

The memory organization in the **[IAS ECC]** is designed to be compatible with [R2]. The structure of the card is shown in the following figure. The characteristics of the individual objects are described in the following chapters.

On a card reset, a default context is always selected. This context is called the root. The root is unique on the card and identifies the default directory.

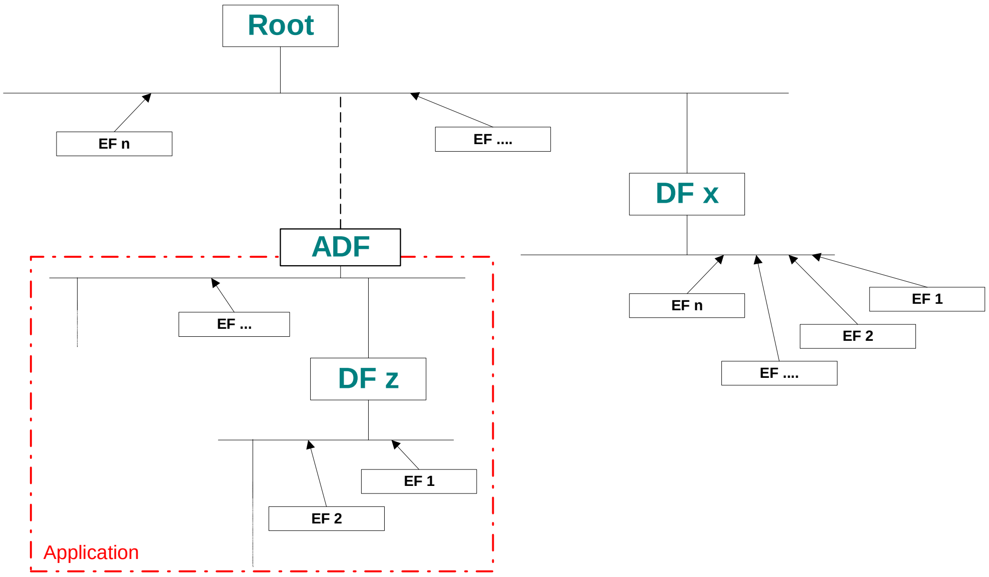


Figure 2: Directory architecture example

The card supports the following type of objects:

- One or more ADFs, each representing an application that may include dedicated files;
- Dedicated files that are attached to the root or to another DF or ADF;
- Elementary Files;
- Data objects;
- Security data objects;
- Security environment.

3.2 Objects description

3.2.1 Application Dedicated File (ADF)

An ADF represents an application and may include any other objects. An ADF contains a set of information dedicated to control the access to this ADF and to its included objects (see § 3.3.4.1).

3.2.2 Dedicated File (DF)

A DF represents a directory and may include any other objects (except an ADF). A DF contains a set of information dedicated to control the access to this DF and to its included objects (see § 3.3.4.1).

3.2.3 Elementary File (EF)

[IAS ECC] supports elementary transparent files as defined in [R2]

Transparent files are seen as a single continuous sequence of data units with granularity of one byte. The supported data unit size is one byte. Any data can be accessed by providing an offset and a length.

A file is made of:

1. A header containing the File Control Parameter (FCP).
2. Data held in the file (aka file body).

The file length cannot be changed after the file creation
The file content is initialized to ‘00’

3.2.4 Security Data Objects (SDO)

A SDO represents a secret or a public part of a secret and is always involved in cryptographic mechanism. For example a PIN/PASSWORD is a SDO. Each SDO has a class and a reference.

A SDO is made of:

1. A header containing the Data Object Control Parameter (DOCP) see §3.4.5

2. A body: containing the Data Object Usage Parameter (DOUP) see §3.4.6

See § 3.4 for more details

3.2.5 Security Environment (SE)

A SE is encapsulated within a SDO template. It is involved in the card security context setting (clarifying algorithm or SDO to use) when needed dynamically, or to determine the access rules of an object / File.

See §3.5 for more details.

3.3 File management (EF, DF and ADF)

This chapter described the management of the EF, DF and the ADF.

3.3.1 Available commands

The set of commands available, for the file management system, is the following:

- SELECT ([R16] based), see § 9.7.1.
- CREATE FILE ([R16] based), see §9.7.4
- ACTIVATE FILE (extension to [R16]), see § 9.7.6.
- DEACTIVATE FILE (extension to [R16]), see § 9.7.7.
- UPDATE BINARY ([R16] based), see § 9.7.3.
- READ BINARY ([R16] based), see § 9.7.2.
- TERMINATE (EF / DF) (extension to [R16]), see § 9.7.9 and 9.7.8.
- DELETE (extension to [R16]), see § 9.7.5.

3.3.2 Files creation and deletion

File creation of a DF, ADF or root, can only be done during personalization phase (i.e. before card issuance). This phase is out of the scope of this specification. [IAS ECC] only envisions EF creation.

The following rules should apply:

1. The root is similar to an ADF and is unique in the card.
2. Each ADF has a single unique name.

File deletion of an EF, DF, ADF or root, should conform to the following rules:

3. The deletion of an ADF or of the root is not possible.
4. The deletion of a DF also deletes child EF, DF and SDO.
5. The memory freed following a file or SDO deletion is reusable for other usage.

3.3.3 Selection of data structures

After reset (ATR/ATS), the root is automatically selected (Refer to § 3.1 for definition of the root).

As defined in [R2], the file system supports:

- A current DF (pointing to the root following card reset).
- A current EF (not initialized following a DF or an ADF selection).

SELECT command update file pointers. Following an unsuccessful SELECT command, file pointers remain unchanged.

Once a file (DF or EF) is selected by any mean, it remains selected until another one is selected, or the application is deselected or reset.

3.3.3.1 Selection by DF name (selection of an ADF or root)

This mode of selection allows the selection of an ADF or the root with an AID selection by AID may be performed in clear, even if a secure messaging session is active. It does not end the secure session.

3.3.3.2 Selection by file identifier

This mode of selection allows the selection of files in the current DF.

A file identifier consists of two bytes.

In order to unambiguously select any file by FID, all file identifier shall be unique under a given DF or ADF.

3.3.3.3 Selection by path

This mode of selection allows the selection of files in the current DF.

A path may reference any file placed under the current DF in the directory architecture. It is a concatenation of FID relative to the current DF. It starts with the identifier of a current DF's child, ending with the FID of the file to select.

Between those two identifiers, the path consists of the identifiers of the successive parent DFs, if any. The order of the file identifiers is always in the direction parent to child. The path allows an unambiguous selection of any file.

Once a file (DF or EF) is selected by any mean, it remains selected until another one is selected, or the application is deselected or reset

3.3.3.4 Selection by short EF identifier

This mode of selection allows the selection of files in the current DF.

A short EF identifier may reference any EF. It consists of five bits not all equal (i.e. any number from one to thirty). When used as short EF identifier, the number zero, i.e., 00000 in binary, references the current EF.

Short EF identifiers cannot be used in a path or as an EF identifier (e.g., in a SELECT command).

The short EF identifier selection is only used by the commands: UPDATE BINARY AND READ BINARY.

3.3.3.5 Selection of parent DF

This mode of selection allows the selection of the parent directory of the current DF. When the current DF is an ADF or the root an error is returned and the ADF or the root remains the current DF.

3.3.4 File Attributes

When a file (EF, DF, ADF or root) is selected, a buffer containing file attributes, called File Control Parameter, can be returned:

- FCP: File control parameters that describes control parameters of a file (size, security attributes…)

This specification does not support the FMD nor FCI:

Attributes returned to a SELECT command ✘ **means the parameter is mandatory** ✓ **means the parameter is optional**			**Presence**			
			EF	**DF**	**ADF**	**ROOT**
Tag	**Length**	**Value**				
'62' {...}	Var.	File Control Parameters elements (FCP)	✘	✘	✘	✘

Table 1: FCP for EF, DF, ADF or the root

3.3.4.1 File Control Parameter (FCP)

The FCP referenced by tag '62' provides the logical, structural and security attributes of a file (EF, DF, ADF or root). **Table 2** defines the subset of [R2] data objects to be supported at least by the **[IAS ECC]**. A security attribute for the "Life cycle status" (tag '8A') is always created with a default value.

The **[IAS ECC]** may return additional data objects in the FCP (i.e. data objects with other tags). Those data objects are to be ignored by the IFD if necessary.

Note: All the data of the FCP are returned to a SELECT command (regarding the APDU command parameters).

File Control Parameter ✘ **means the parameter is mandatory** ✓ **means the parameter is optional**			**Presence**		
			EF	**DF**	**ADF**
Tag	**Length**	**Value**			
'62' {...}	Var.	FCP data objects composed of the following data	✘	✘	✘
'80'	'02'	File length. Specify the file size excluding the structural information (see § 3.3.4.1.1)	✘	▨	▨
'82'	'01'	File descriptor byte (see § 3.3.4.1.2)	✘	✘	✘
'83'	'02'	File identifier (see § 3.3.4.1.3)	✘	✘	▨
'84'	'05' to '10'	DF name (AID)	▨	▨	✘

'88'		'00' or '01'	Short file identifier (see § 3.3.4.1.4)	✓	▨	▨
'8A'		'01'	Life cycle status[5] byte (see § 3.3.4.1.6)	✓	✓	▨
'A1'		Var.	Security attributes in proprietary format (see § 4.2.2)	✗	✗	✗
	'8C'	Var.	Security attribute referencing the proprietary format for contact interface	✓	✓	✓
	'9C'	Var.	Security attribute referencing the proprietary format for contactless interface	✓	✓	✓
'A5'		Var.	Issuer discretionary data in BER TLV format	✓	✓	✓
'85'		Var.	Issuer discretionary data in NON BER TLV format	✓	✓	✓

Table 2: FCP for EF, DF, ADF or the root

Note: If only one communication medium is allowed by the [IAS ECC] card (contact/contactless), the non relevant security attributes shall be omitted (e.g. if contactless is not supported, the D.O. '9C' is omitted)

3.3.4.1.1 File length

File length is coded on two bytes and specifies the file size excluding the structural information.

3.3.4.1.2 File descriptor byte

The file descriptor byte is encoded as defined in Table 3.

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
0	0	-	-	-	-	-	-	File not shareable through logical channel
		0	0	0	0	0	1	EF
		1	1	1	0	0	0	DF, ADF or the root

Table 3: File descriptor byte (Tag '82')

3.3.4.1.3 File identifier (FID)

A file identifier may reference any file and consists of two bytes. There are no restrictions on file IDs, except for those defined by [R2]:

- The value '3F00' is reserved by [R2].
- The value 'FFFF' is reserved for future use.
- The value '3FFF' is reserved for referencing the current DF.
- The value '0000' is reserved for the current file.

In order to unambiguously select any file by its identifier, all EFs and DFs immediately under a given DF, ADF or the root shall have different file identifiers. This should be checked during the file creation process.

[5] Parameter optional during the creation process but returned in response to a select command.

3.3.4.1.4 DF name (AID)

A DF name or AID is a five to sixteen byte-long string. Each DF name shall be unique for the whole card.

RID (5 bytes)	PIX (0 to 11 bytes)

- RID: Registration number of the application issuer.
- PIX: Application identifier chosen by the issuer.

3.3.4.1.5 Short File Identifier (SFI)

A short file identifier may reference any EF and consists of five bits (coded on one byte). The following rules, specified by [R2], apply for the use of tag '88' in the control parameters of any EF:

1. If tag '88' is absent, then in the second byte of the file identifier (tag '83'), bits B5 to B1 encode the short EF identifier.
2. If tag '88' is present with a length set to zero, then the EF supports no short identifier.
3. If tag '88' is present with a length set to one and if bits B8 to B4 of the data element are not all equal and if bits B3 to B1 are set to 000, then bits B8 to B4 encode the short EF identifier (a number from one to thirty).

There are no restrictions on SFIs, except for those defined by [R2] :

- The value 00000 (in binary) is reserved for referencing the current EF.
- The value 11110 (in binary) is unique under the current DF and reserved for referencing the EF.DIR (see § 10.4.3).
- The value 11111 (in binary) is forbidden

In order to unambiguously select an EF by its short identifier, all EFs immediately under a given DF, ADF or the root shall have different short file identifiers. This should be checked during the file creation process.

3.3.4.1.6 Life cycle status byte

Referenced by tag '8A' a life cycle status byte is present in the FCP of any file (EF and DF). This byte shall be interpreted compliant to [R2] and [R5] as defined in Table 4.

B8	B7	B6	B5	B4	B3	B2	B1	MEANING	APPLIES TO
0	0	0	0	0	1	0	1	Operational state (activated)	EF, DF
0	0	0	0	0	1	0	0	Operational state (deactivated)	EF, DF
0	0	0	0	1	1	0	-	Termination state	EF, DF

Table 4: Life cycle status byte (Tag '8A')

From any states, a call to the TERMINATE command turns the file to the termination state. There is no possible return to 'Creation' or 'Operational' state.

Figure 3 represents the file life cycle states and the commands that invoke a transition upon successful completion.

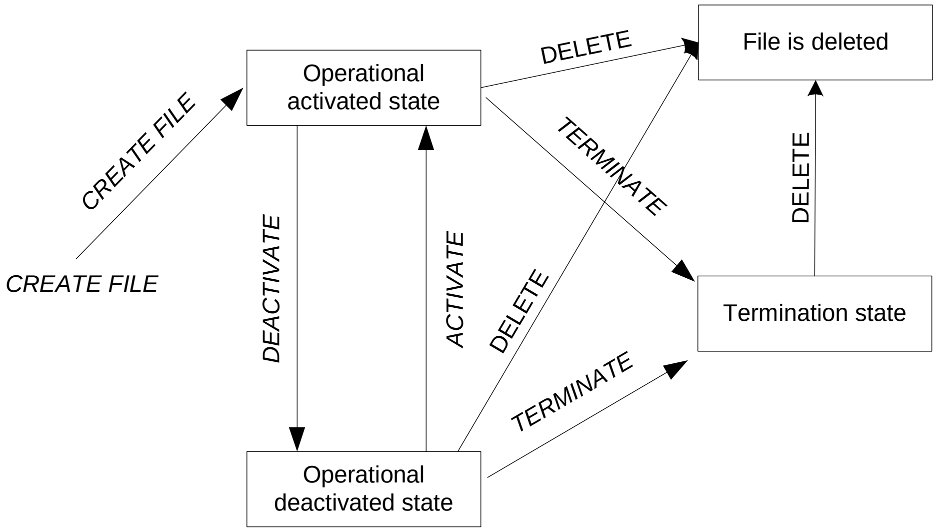


Figure 3: Diagram for file life cycle and transition

Table 6 represent the different APDU COMMANDS and the behavior, regarding the state of the current file.

Note: As indicated in [R5], an EF in terminated state can still be read if its access condition authorizes it freely during other life cycle states.

Type	Apdu Command		Operational state (activated)		Operational state (Deactivated)		Terminate state	
			Avail.	Processing	Avail.	Processing	Avail.	Processing
EF	Select		Yes	Everything is allowed (with respect to security condition).	Warn	SW = '6283', the file is disabled but selected.	Warn	SW = '6285' following the FCP, the file is terminated, but selected. All commands, except Delete and Read Binary, are blocked.
	Delete File		Yes		Yes	with respect to security condition	Yes	with respect to security condition
	Terminate File		Yes		Yes	with respect to security condition	Warn	SW = '6285', the file is already terminated.
	Activate File		Yes		Yes	with respect to security condition	No	
	Deactivate File		Yes		Warn	SW = '6283', the file is disabled.	No	
	Update Binary		Yes		No		No	
	Read Binary		Yes		No		Yes	SW = '6285', read is available with respect to security condition

Table 5: Apdu commands regarding the EF state

Type	Apdu Command	Operational state (activated)		Operational state (Deactivated)		Terminate state	
		Avail.	Processing	Avail.	Processing	Avail.	Processing
DF	Select DF	Yes	Everything is allowed (with respect to security condition).	Warn	SW = '6283', the file is disabled but selected. All commands, except SELECT/ Delete / Terminate / Activate, are blocked at the DF level	Warn	SW = '6285', the file is terminated, but selected. All commands, except Delete, are blocked.
	Delete File (Self)	Yes		Yes	with respect to security condition	Yes	Submit to security check
	Terminate File	Yes		Yes	with respect to security condition	Warn	SW = '6285', the file is already terminated.
	Activate File	Yes		Yes	with respect to security condition	No	
	Deactivate File	Yes		Warn	SW = '6283', the file is disabled.	No	
	SDO – Use	Yes		No	SDO are unusable. EF cannot be created.	No	SDO are unusable. EF cannot be created.
	SDO – Update / Create File EF	Yes		No		No	

Table 6: Apdu commands regarding the DF state

Note:

- As soon as the command does not use a local object (SDO or file), it can be realized from the DF, whatever its state is.
- When a DF is deactivated or terminated, it is not possible to access by any mean a file or a SDO within the DF (READ BINARY by SFI, SELECT EF, VERIFY PIN on a local PIN,…), for instance a global SDO is still accessible from this DF (VERIFY PIN on a Global PIN…)

3.4 Security Data Objects

This chapter described the management of the Security Data Objects (SDO).

A SDO is always associated to a DF and could be defined as “global” (only in the root directory) or “local” otherwise.

For easier management of the different data stored in a card, Static Security Environments (SSE) are closed together with SDO. However, SSE sets have specific rules and are totally describes in §3.5.3.

3.4.1 Available commands

The set of commands available, for the Security Data Objects management, is the following:

- GET DATA, see § 9.8.1.
- PUT DATA, see § 0.
- CHANGE REFERENCE DATA, (used to modify a PIN value) see §9.4.2.
- RESET RETRY COUNTER, (used to unblock and optionally change a PIN value) see §9.4.3.

3.4.2 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' || 0b1 || ObjClass (7 bits) || 0b0 || 0b00 || ObjRef (5 bits) ≡ 3 bytes

Where:

- 'BF' means context-specific class, constructed data object in the BER TLV encoding rules.
- 0b1 means that another byte is following the current one.
- ObjClass is one of the authorized classes listed here below.
- 0b0 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 0b00000 for ObjRef shall not be used

Whenever required for interoperability, a compatible tag allocation scheme according to [R17] Annex G applies to SDO data structure. Accordingly, SDO are further data objects nested within interindustry template referenced by tag '70'. In the context of the SDO structure, the European specification [R17] Annex G reserves data objects of the context-specific class referenced by tags '80' to 'BF'. Consequently, the SDO identification becomes in this case: ‘70’ || length of SDO || SDO tag, as described in the present clause. Each SDO object in the interoperability field with an ECC-compliant IFD shall be encapsulated in a data object ‘70’.

3.4.3 Security data object class architecture

Security data objects are organized according to the following structure:

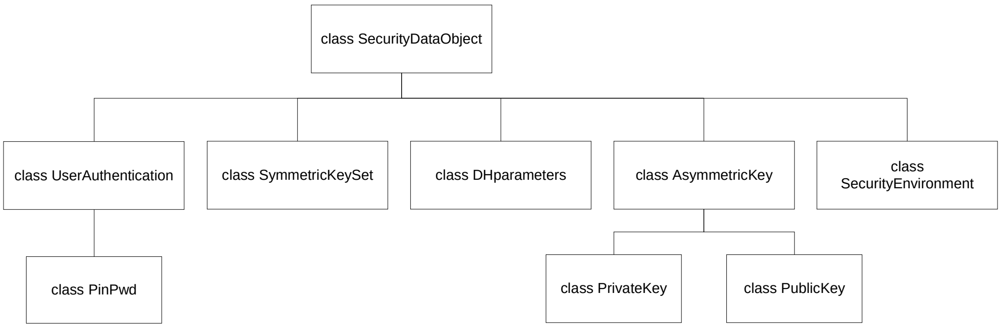


Figure 4: Security data object class structure

There are 5 main classes that can have subclasses:

- Class user authentication, with
  - PIN / Password sub classes,
- Class symmetric key set.
- Class asymmetric key, with
  - Private key (RSA) sub classes,
  - Public key (RSA) sub classes.
- Class security environment.
- Class Key Agreement (DH)

Security data object class architecture was designed in order to take into account both the object family, and in more practical consideration, the commands that may interact with these security data objects.

SDO CLASS	IDENTIFIER VALUE	TAG MIDDLE BYTE CODING
User authentication: PIN / Password	'01'	'81'
Symmetric key set (2 keys)	'0A'	'8A'
Asymmetric keys RSA – Private portion	'10'	'90'
Asymmetric keys RSA – Public portion	'20'	'A0'
Domain parameters DH	'21'	'A1'
Security environment	'7B'	'FB'
Any other identifier values are reserved for future use.		

Table 7: Security Data Objects class identifiers

3.4.4 SDO tag structure

Likewise a file, a SDO is comprised of two parts:

1. A header: the Data object control parameters (DOCP), contains a set of information dedicated to control the access to the SDO (see § 3.4.5).

2. A body: the Data Object Usage Parameters (DOUP), contains the data (see § 3.4.6).

SDO are coded according to the following scheme (Refer to [R17] Annex G):

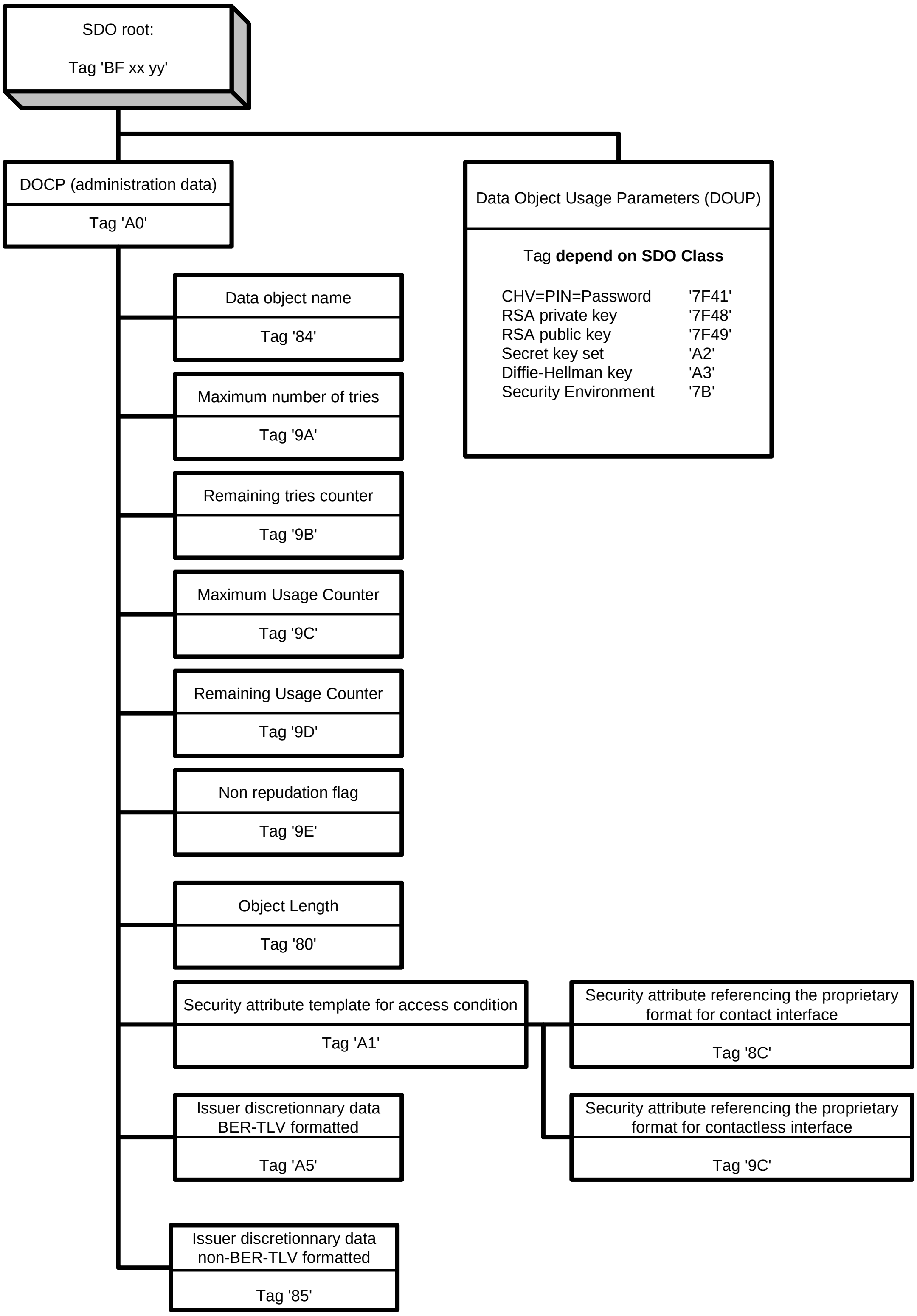

SDO root:
Tag 'BF xx yy'
DOCP (administration data)
Tag 'A0'
Data Object Usage Parameters (DOUP)
Tag **depend on SDO Class**
CHV=PIN=Password '7F41'
RSA private key '7F48'
RSA public key '7F49'
Secret key set 'A2'
Diffie-Hellman key 'A3'
Security Environment '7B'
Data object name
Tag '84'
Maximum number of tries
Tag '9A'
Remaining tries counter
Tag '9B'
Maximum Usage Counter
Tag '9C'
Remaining Usage Counter
Tag '9D'
Non repudation flag
Tag '9E'
Object Length
Tag '80'
Security attribute template for access condition
Tag 'A1'
Security attribute referencing the proprietary format for contact interface
Tag '8C'
Security attribute referencing the proprietary format for contactless interface
Tag '9C'
Issuer discretionnary data BER-TLV formatted
Tag 'A5'
Issuer discretionnary data non-BER-TLV formatted
Tag '85'

Figure 5: Tree describing the SDO structure

3.4.5 Data object control parameters (DOCP) supported

DOCP are stored in a BER coded object having the tag 'A0' (i.e. a SDO). The following table shows the DOCP that a security data objects supports:

Data object control parameters ✗ means the parameter is mandatory ✓ means the parameter is optional			Never returned by the card	Object classes					May be updated after SDO creation
				Symmetric key set	PIN / password	Asym. Key		Security environment	
						Public part	Private part		
Tag	**Length**	**Value**							
'A0' {...}	Var.	Object header containing DOCP		✗	✗	✗	✗	✗	
'84'	'01' to '10'	Data object name – Example: "PK.CH.DS"		✓	✓	✓	✓	✓	
'9A'	'01'	Maximum number of tries		✗	✗				
'9B'	'01'	Remaining tries counter		✗	✗				
'9C'	'02'	Maximum Usage counter		✓	✓	✓	✓		
'9D'	'02'	Remaining usage counter		✓	✓	✓	✓		
'9E'	'01'	Non-repudiation flag					✗		
'80'	'02'	Object Length		✓	✓	✓	✓	✓	
'A1' {...}	Var.	Security attributes template for access condition (§4.2)		✗	✗	✗	✗	✗	
'8C'	Var.	Security attributes referencing the compact format for contact interface		✓	✓	✓	✓	✓	
'9C'	Var.	Security attributes referencing the compact format for contactless interface		✓	✓	✓	✓	✓	
'A5'	Var.	Issuer discretionary data in BER TLV format		✓	✓	✓	✓	✓	
'85'[6]	Var.	Issuer discretionary data in non BER TLV format		✓	✓	✓	✓	✓	

Table 8: Security Data Objects control parameters

Here follow a description of each tag of the DOCP:

- '80' Object Length: this parameter allows to allocate the necessary memory to store the object. Here is a table with examples of length

SDO	Example of encoding
PIN container of 15 bytes	'000F'
Symmetric keys set of 128 bits each	'0010'
Asymmetric keys RSA – Private portion (2048 bits)	'0100'
Asymmetric keys RSA – Public portion (2048 bits)	'0100'
Asymmetric keys DH – Public portion (2048 bits)	'0100'

[6] The DO '85' is not defined in [R16] Annex G within DOCP and consequently will not be interpreted by the terminal

Static Security Environment	Number of CRTs 'XXXX'

Table 9: Values for Tag '80'

- '84' Data object name: this is the name of the data object and can be useful for clarifying the version
- '85' Issuer discretionary data: discretionary data coded in a byte string.
- '9A' - Maximum number of tries: Maximum number of errors before SDO use is forbidden. If set to zero, there is no maximum number of tries (infinite).
- '9B' - Remaining tries counter: Decremented on every security verification failure (e.g. PIN verification).
- '9C' - Maximum number of uses : If absent, there is no usage limit (infinite).
- '9D' - Remaining usage counter: Decremented each time the security object is used. When the counter value reaches zero, the SDO use is forbidden. (no meaning if the maximum usage counter is infinite).
- '9E' - Non-repudiation flag (Boolean): Boolean indicating if the user authentication object associated with the SDO shall be reset after each signature calculation. '00' means 'false', '01' means 'true'
- 'A1' Security attributes template for access condition : Coded in BER-TV format, indicates the different access rules for each interface
- '8C' Security attributes referencing the compact format for contact interface: see §4.2 for more details
- '9C' Security attributes referencing the compact format for contactless interface: see §4.2 for more details
- 'A5' Issuer discretionary data: discretionary data coded in BER-TLV.

Note:

The content of the D.O.C.P shall be chosen in such a way that it does always fit within a unique response to the GET DATA SDO as stated in §9.8.1.1

If only one communication medium is allowed by the [IAS ECC] card (contact/contactless), the non relevant security attributes shall be omitted (e.g. if contactless is not supported, the D.O. '9C' is omitted)

3.4.6 Data Object Usage Parameters (DOUP)

The following tables described the Data Object Usage Parameters (DOUP) for each SDO class.

For different SDO the tag '80' is used to define the "Algorithm to compulsorily use". As it is not a mandatory tag, when it is omitted or set to '00', the SDO may be used with any algorithm. When it is declared, the card should check if the requested algorithm matches the defined one, in all other cases the request is rejected.

This chapter describes the D.O. that may be updated after the SDO creation with the command PUT DATA SDO (see §9.8.2). The SDO update shall be ordered according to the sequence indicated in the template and shall be made one component after the other. For each primitive data object (within the DOUP), there should be ONE PUT DATA SDO command.

The SDO components might be imported in different orders, or by concatenating several of them within a single PUT DATA command. However, these features are out of the scope of this specification

3.4.6.1 PIN / Password

Cardholder Verification values ≡ CHV ≡ PIN ≡ password ✘ means the parameter is mandatory ✓ means the parameter is optional			Never returned by the card	Presence	May be updated after SDO creation
Tag	Length	Value			
'7F41' {...}	Var.	Cardholder Verification values		✘	
'80'	1	PIN / password maximum size (in bytes)		✘	
'81'	1	PIN / password minimum size (in bytes)		✘	
'82'	Var.	CHV value	●	✘	

Note:

- The tag '82' length should always be consistent with values indicated by tag '80' and '81'
- The CHV value shall be changed using the command CHANGE REFERENCE DATA (see §9.4.2) or .RESET RETRY COUNTER (see §9.4.3)

3.4.6.2 Private RSA Key data object

Private RSA Key data object ✘ means the parameter is mandatory ✓ means the parameter is optional			Never returned by the card	Presence	May be updated after SDO creation
Tag	Length	Value			
'7F48' {...}	Var.	Private RSA Key data		✘	
'92'	Var.	TL pair for parameter p	●	✘	●
'93'	Var.	TL pair for parameter q	●	✘	●
'94'	Var.	TL pair for parameter (q-1) mod p	●	✘	●
'95'	Var.	TL pair for parameter d mod (p-1)	●	✘	●

Tag	Length	Value	Never returned by the card	Presence	May be updated after SDO creation
'96'	Var.	TL pair for parameter d mod (q-1)	●	✗	●
'80'	1	Algorithm to compulsorily use		✓	

3.4.6.3 Public RSA Key data object

Public RSA Key data object ✗ means the parameter is mandatory ✓ means the parameter is optional			Never returned by the card	Presence	May be updated after SDO creation
Tag	**Length**	**Value**			
'7F49' {...}	Var.	Public RSA Key data		✗	
'81'	Var.	Modulus N		✗	●
'82'	Var.	Public exponent e		✗	●
'80'	1	Algorithm to compulsorily use		✓	
'5F20'	'0C'	Certificate Holder Reference – CHR		✓	●
'5F4C'	Var.	Certificate Holder Authorization – CHA		✓	

Note: CHR and CHA shall be mandatory for root CA public key (used for certificates verification). In all other cases, when RSA public key is linked to RSA private key, those 2 tags are not needed.

3.4.6.4 Secret key set data object (symmetric algorithm)

Secret key set data object (symmetric algorithm) ✗ means the parameter is mandatory ✓ means the parameter is optional			Never returned by the card	Presence	May be updated after SDO creation
Tag	**Length**	**Value**			
'A2' {...}	Var.	Secret key set data		✗	
'90'	Var.	Key value for K_{MAC}	●	✗	●
'91'	Var.	Key value for K_{ENC}	●	✗	●
'80'	1	Algorithm to compulsorily use		✓	

3.4.6.5 Domain Parameters for Key agreement DH

Parameters of public Key data object for DH privacy algorithm ✗ means the parameter is mandatory ✓ means the parameter is optional			Never returned by the card	Presence	May be updated after SDO creation
Tag	**Length**	**Value**			
'A3' {...}	Var.	Parameters of public Key data object for DH privacy algorithm		✗	
'97'	Var.	TL pair for parameter g		✗	
'98'	Var.	TL pair for parameter p		✗	
'99'	Var.	TL pair for parameter q		✗	
'80'[7]	'01'	Algorithm to compulsorily use (see §5.2.4)		✓	

[7] The DO '80' is not defined in [R17] Annex G within structure 'A3' and consequently will not be interpreted by the terminal

Note: This SDO is the only one allowing independent parameters retrieval. Get Data DOUP can be performed the retrieve g, p,or q independently, thus sending three Get Data DOUP.

3.4.6.6 Security environment

Security environment ✘ **means the parameter is mandatory** ✓ **means the parameter is optional**			**Never returned by the card**	**Presence**	**May be updated after SDO creation**
Tag	**Length**	**Value**			
'7B' {...}	Var	Security environment		✓	
'A4'	Var.	Control reference template valid for authentication (AT)		✓	
'A6'	Var.	Control reference template valid for key agreement (KAT)		✓	
'AA'	Var.	Control reference template valid for hash-code (HT)		✓	
'B4'	Var.	Control reference template valid for cryptographic checksum (CCT)		✓	
'B6'	Var.	Control reference template valid for digital signature (DST)		✓	
'B8'	Var.	Control reference template valid for confidentiality (CT)		✓	

3.4.7 Additional information regarding the SDO personalization

3.4.7.1 Public and private RSA key SDO

The normal handling of cryptographic secrecy requires that private and public portion of an RSA key pair be consistent, if the public portion is present: their value should be linked.

If the matching public key is present, its algorithm should be set to the same value as the one set for the private key.

For security reasons, some implementations might perform double check of the signature using the matching public portion. Though this is out of the scope of the specification, they shall be consistent (size, value,.).

However, it still remains possible to use a private portion for a given use (decryption of data for instance) and the matching public portion for another use (certificate verification).

This specification does not forbid personalizing a card in that way, even though it is not envisioned in this specification.

3.5 Security Environments

This chapter described the management of the Security Environment. Two types of Security Environment are present inside the card:

- Static Security Environments (SSE): several SSE exist on the card and are objects created during card personalization and involved into the security architecture.
- Current Security Environment (CSE): only one CSE exists in the card and represents the security context of the card at a given moment.

3.5.1 Available commands

The set of commands available, for the Security Environments management, is the following:

- GET DATA, see § 9.8.1[8].
- MSE SET, see § 9.9.2.
- MSE RESTORE, see § 9.9.1.

3.5.2 Description

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).

It provides two main functionalities:

- When it is in non-volatile memory, it is involved into the security architecture. In that case, it is called “Static Security Environment” (See §3.5.3).
- When it is in RAM, it is involved for clarifying any additional data needed for the execution of some commands (eg, PSO-Hash). In that case, it is called “Current Security Environment” (see §3.5.5)

3.5.3 Static Security Environments (SSE)

The card may contain several SE sets, each of them containing at most 31 SE numbered from 1 to 31. A SE set can exist under the Root, or under an ADF. It is always identified by a local reference.

The case in which SE are stored within DF is out of the scope of [IAS ECC]

The SE #1 of a SE set is called the “default SE”. After an application selection (ROOT or ADF) it is copied in the current SE. It may be useful to put the default context (e.g., cryptographic algorithms or security operations) into the current SE in order to have few exchanges of MSE set commands.

The content of the default SE may be empty.

[8] SE are nested in SDO structure. GET DATA is proprietary as long as the DO are not encapsulated according the compatible allocation scheme: [R17] Annex G mentions GET DATA with extended header list for retrieval of SDO.

These Static Security Environments may be restored in the CSE (see §3.5.5) using the command MSE RESTORE (see §9.9.1) to load a predefined security context to perform operations.

3.5.4 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 §4.2.3). 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.

As the SE#1, is the “default SE”, that is always restored in the current SE, it should not be used as a SSESP.

It is up to application administrator that issues the **[IAS ECC]** application to decide which SSE will be SSESP.

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 (example below in §3.5.6.2).

3.5.5 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, two commands are provided:

- MSE RESTORE replaces the CSE content with the one restored from the non-volatile memory (from the requested SSE).
- MSE SET command is used to change partially or totally the CSE content.

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

- The supported CRTs are listed in §10.6.
- After an application selection (ROOT or ADF) either at card reset or using the SELECT by name command, SE #1 (“default SE”) of the newly selected root, ADF is restored to the current SE.
- At each moment, at most, there should be only one Control reference template of each type (AT, KAT,HT,CCT,DST,CT)

3.5.6 Control Reference Templates

A control reference template is a set of data objects that fully specifies the cryptographic algorithms to be executed, the SDO to be used and any additional data needed by a security mechanism.

3.5.6.1 Control Reference Templates list

Tag	Meaning
'A4'	Data objects for an Authentication Template (AT)
'A6'	Data objects for a Key Agreement Template (KAT)
'AA'	Data objects for a Hash Template (HT)
'B4'	Data objects for a Cryptographic Checksum Template (CCT)
'B6'	Data objects for an Digital Signature Template (DST)
'B8'	Data objects for an Confidentiality Template (CT)

Table 10: Tags of Control Reference Templates

3.5.6.2 SDO multiple referencing in SSE for security policy

When a OR condition is needed for access rule, an easy way to execute such condition when SDO have the same type is to multiply the tag “SDO reference” (tag 83 or 84) into a CRT.

A CRT (tag ‘XX’)is constructed as follows:

Tag	Lg.	Value		
‘XX’	‘09’			
		Tag	***L***	***Value***
		‘95’	‘01’	UQB
		‘83’ or ‘84’	‘01’	SDO reference
		‘80’	‘01’	Algorithm ID

Depending on the CRT, several SDO references may be present (tag ‘83’ for a secret key, a PIN or a public key, tag ‘84’ for a private key).

Example for a CRT referencing two SDOs:

Tag	Lg.	Value		
‘XX’	‘0C’			
		Tag	***L***	**Value**
		‘95’	‘01’	UQB
		‘83’ ou ‘84’	‘01’	1st SDO reference
		‘83’ ou ‘84’	‘01’	2nd SDO reference
		‘80’	‘01’	Algo ID

Not all the CRT supports several SDO references. For more information, see each CRT description.

This mechanism has a sense into SSE for security policy, because there are involved into security architecture. When restored into CSE, no cryptographic mechanism should be processed. To prevent from restoring this kind of SSE for security policy access right for MSE RESTORE shall be set to "Never" and MSE SET command shall be used instead.

3.5.6.3 Usage Qualifier Byte (UQB)

Usage Qualifier Byte is not explicitly used in **[IAS ECC]** application since it is not present in CRT of MSE SET command. However UQB is implicitly conveyed in P1 value of MSE SET command and concordance between P1 and UQB can easily retrieved using a bit mask. However UQB is still used in SSE.

The only possible values for the UQB are:

CRT	UQB	Meaning
DST (B6h)	80h	Verify Certificate
	40h	Compute Digital Signature
AT (A4h)	80h	External Authentication
	40h	Internal Authentication
	C0h	Mutual Authentication
	08h	User authentication
CT (B8h)	30h	Secure messaging for encryption in command and response data fields
	40h	Decipherment
CCT (B4h)	30h	Secure messaging for integrity in command and response data fields
KAT (A6h)	C0h	Key Agreement (required for DH key agreement)
HT (AAh)	-	No Usage Qualifier Byte supported

Table 11: UQB values summary

3.5.6.4 Digital Signature Template (DST)

A DST references data for digital signature computation and certificate verification.

This CRT is only involved in cryptographic mechanisms even if sets into a SSE.

3.5.6.5 Authentication Template (AT)

When the AT is set in the CSE, it is possible to use authentication-related commands that implicitly use data referenced in the AT.

When the AT is set in a SSESP for access rules definition, it is possible to require the authentication of the provided reference data (SDO) for accessing or using an object in the memory architecture.

3.5.6.6 Confidentiality Template (CT)

When the CT is set in the CSE, it is possible to use Encryption key decipherment related command that implicitly use data referenced in the CT.

When the CT is set in a SSESP for access rules definition, this SSESP can be referenced to protect the use of an object with secure messaging in confidentiality. In this case CCT shall also be set with the same content to comply with ECC requirement.

3.5.6.7 Cryptographic Checksum Template (CCT)

CRT CCT cannot be set in the CSE since there is no use and no meaning of having this template in the CSE.

However the CCT can be set in a SSESP for access rules definition, this SSESP can be then referenced to protect the use of an object with secure messaging in Integrity. . In this case CT shall also be set with the same content to comply with ECC requirement.

3.5.6.8 Key Agreement Template (KAT)

When the CRT KAT is set, it is possible to use key agreement related commands can be used. These commands implicitly use data referenced in the KAT.

This CRT is only involved in cryptographic mechanisms even if sets into a SSE.

3.5.6.9 Hash Template (HT)

HT references the hash algorithm to use.

This CRT is only involved in cryptographic mechanisms even if sets into a SSE.

4 SECURITY ARCHITECTURE

4.1 Overview - Basic security principles

This clause describes security status, security attributes and security mechanisms.

4.1.1 Security status

The security status represents the current state possibly achieved after completion of the answer to reset and a possible protocol and parameter selection and / or a single command or a Protocol stepss possibly performing authentication procedures. The security status may also result from completion of a security procedure related to the identification of the involved entities, if any, e.g., by proving knowledge of a password (e.g., using a VERIFY command) or knowledge of a key (e.g., using a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command, or using a sequence of PERFORM SECURITY OPERATION – VERIFY CERTIFICATE commands), or by secure messaging (e.g., message authentication).

Four security statuses are considered:

- **Global security status:** In a card using a hierarchy of DFs, it may be modified by completion of a ROOT related authentication procedure (e.g., entity authentication by a password or a key attached to the ROOT).
- **Application-specific security status:** It may be modified by completion of an application-related authentication procedure (e.g., entity authentication by a password or a key attached to the application); it may be maintained or recovered by application selection; this modification may be relevant only for the application to which the authentication procedure belongs.
- **File-specific security status:** It may be modified by completion of a DF-related authentication procedure (e.g., entity authentication by a password or by a key attached to the specific DF); it may be maintained or recovered by file selection; this modification may be relevant only for the application to which the authentication procedure belongs.
- **Command-specific security status:** It only exists while processing a command using secure messaging and involving authentication; such a command may leave the other security status unchanged.

In an application (ADF), security status flags are managed as follows:

- An access right granted in the application ADF is valid for all DF of the application.
- Security status relevance follows the SDO relevance as indicated in chapter §4.3.
- The security statuses are kept when browsing the file system (including the ADF(s))

The security statuses are set upon successful:

- EXTERNAL AUTHENTICATION (symmetric and asymmetric role authentication), MUTUAL AUTHENTICATION (symmetric device authentication) and INTERNAL AUTHENTICATION (device authentication with privacy protection)
- VERIFY for user authentication.
- Secure Messaging opening session

4.1.2 Security attributes

The security attributes define which commands are allowed, and under which conditions. The security attributes of a file depend on its category (DF or EF) and on parameters in its FCP. The security attributes of a SDO depend on parameters in its DOCP. In particular, security attributes may specify the security requirements associated to the SDO/File before accessing or using the data. If not explicitly authorized in the FCP or in the DOCP, an access is always forbidden.

The security attributes might be different, depending on the communication link, like in contact or contactless mode.

Access Mode Byte (AMB) and Security Condition Byte (SCB) describe those security attributes (§4.2).

4.1.3 Security mechanisms

This clause considers the following security mechanisms:

- **Entity authentication with password:** The card compares data received from the outside world with secret internal data (e.g., VERIFY). This mechanism may be used for protecting the rights of the user.
- **Entity authentication with key:** The entity to authenticate has to prove the knowledge of the relevant secret or private key in an authentication procedure (e.g., a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command).
- **Device authentication:** The IFD is identified and entitles to send data to the ICC (MUTUAL AUTHENTICATE,..)

4.2 Security attributes

4.2.1 Interpretation rules

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

'8C or '9C', $L_{8C\text{ or }9C}$ {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.

Example:

AMB = '81' = 0b10000001	there is one SCB following the AMB
AMB = '47' = 0b01000111	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:

- '8C' or '9C', $L_{8C\text{ or }9C}$ {AMB1, $SCB1_1$ to $SCB1_n$, AMB2, $SCB2_1$ to $SCB2_n$, AMB3, $SCB3_1$ to $SCB3_n$ }

In such a case, the OR condition applies between access rights that may be duplicated in two or more {AMB, SCB1 to SCBn} security attributes. Example:

{AMB1, $SCB1_1$ to $SCB1_n$} defines reading is protected under PIN referenced in SE #3.

{AMB2, SCB2$_1$ to SCB2$_n$ } defines reading is protected under PIN AND requires an external authentication (object references in SE #2).

This shall be interpreted as follows: (PIN[SE#3]) OR (PIN[SE#2] AND KEY.AUTH[SE#2]).

4.2.2 AMB coding

4.2.2.1 AMB coding for files

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

The following tables are full [R2] compliant.

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 12: 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 13: Access mode byte for EFs

4.2.2.2 AMB coding for Security Data Objects[9]

Whenever required for interoperability, the AMB for Security Data Objects shall be encoded in a cryptographic information application that may use the extended AccessMode definition as per ISO/IEC 7816-15:2007 AM2.

[9] Proprietary AMB are not likely to be interpreted by a ECC-compliant terminal unless the AMB coding is reflected in a standardized way as per cryptographic information application [R6]

The following tables show the AMB coding according to the SDO type. The bit B8 is always set to 1. This coding is compatible to [R2] standard and indicates the bits B7 to B4 are proprietary coded.

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

Table 14: User authentication (Pin/Password)

B8	B7	B6	B5	B4	B3	B2	B1	Meaning
1	1	-	-	-	-	-	-	PSO Compute Digital Signature
	-	1	-	-	-	-	-	Internal Authenticate
	-	-	1	-	-	-	-	PSO Decipher
	-	-	-	1	-	-	-	Generate Asymmetric Key Pair
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1		Put Data
	-	-	-	-	-	-	1	Get Data

Table 15: 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 16: RSA Key, public 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 17: Symmetric key set

B8	B7	B6	B5	B4	B3	B2	B1	Meaning
1	0	-	-	-	-	-	-	RFU
	-	0	-	-	-	-	-	RFU
	-	-	0	-	-	-	-	RFU
	-	-	-	0	-	-	-	RFU
	-	-	-	-	0	-	-	RFU
	-	-	-	-	-	1		Put Data
	-	-	-	-	-	-	1	Get Data

Table 18: Diffie-Hellman key

B8	B7	B6	B5	B4	B3	B2	B1	Meaning
1	0	-	-	-	-	-	-	RFU
	-	0	-	-	-	-	-	RFU
	-	-	0	-	-	-	-	RFU
	-	-	-	0	-	-	-	RFU
	-	-	-	-	1	-	-	Mse Restore
	-	-	-	-	-	0		Put Data (not supported)
	-	-	-	-	-	-	1	Get Data

Table 19: Security Environment

4.2.3 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 20: SCB - 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 (see §10.6) 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)

Note:

The Static Security environment pointed out in the least significant bits are called “Static Security Environment for security policies” (see §3.5.4)

4.3 SDO use and SE set organization and relevance

Figure 6 presents an overview of the way SDO and SE are searched and accessed when the card needs to use them.

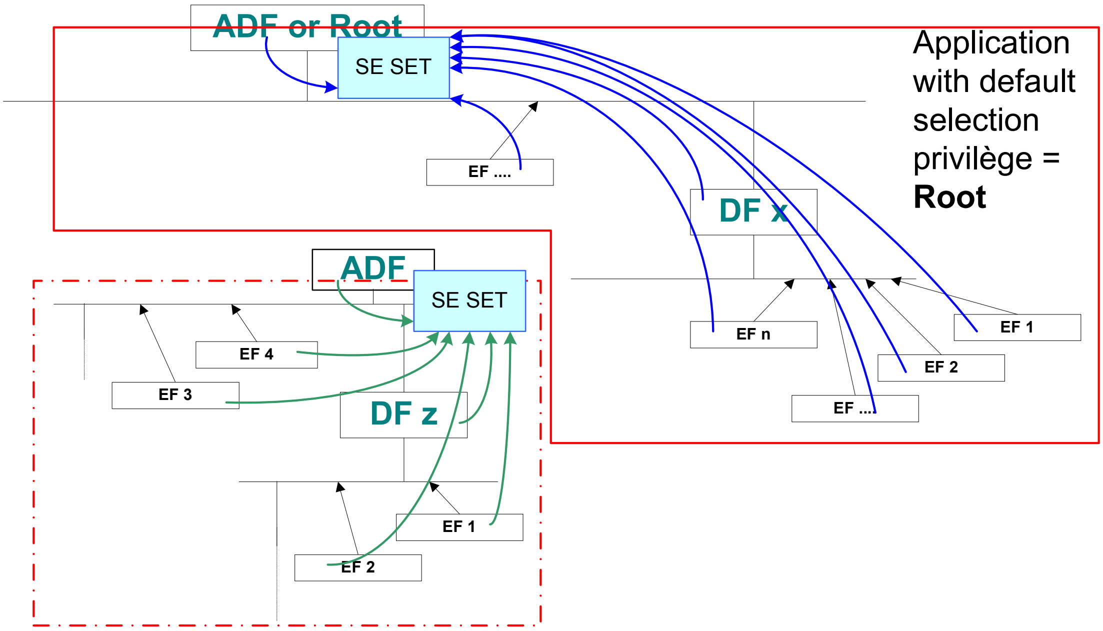


Figure 6: Security Data Object relevance

The relevance for using a SDO (except SE) is determined as follows:

SDO searching is always performed:

- within ADF boundaries for a local SDO (a search never crosses the boundaries delimited with the dashed red lines)
- Or directly in the root if a global SDO is requested (in the AREA ROOT only).

When SSEs are searched, no relevance applies.

Two cases may occur:

- In case of Access condition analysis, the SE are searched from the ADF containing the SDO or files whose Access conditions are analyzed."
- When trying to restore a SSE in the CSE, the SSE are searched in the current ADF.

4.4 Local vs. global SDO referencing

Two SDO referencing cases shall be differentiated:

1. When executing a command, the SDO reference on which it applies:
   a. The SDO reference is present in the P2 parameter of the APDU.
   b. The SDO reference is extracted from the applicable CRT retrieved from the current SE
2. During access rights verification. The Compact Security Condition points on a static SE containing the CRT from which the SDO references are retrieved.

For both cases, the byte coding is the same:

b8	b7	b6	b5	b4	b3	B2	b1
1: Local 0: Global	RFU ≡ 0b00		Object reference [1 .. 31]				

The card extracts bits b1 to b5 and rebuilds the complete SDO identifier by calculating the SDO tag middle byte according to the context (BF xx).

From this complete SDO reference, the card can search for the object in its memory architecture.

According to b8 value, the reference does not point on the same object:

- b8 = 0 → the referenced object is stored in the root.
- b8 = 1 → the referenced object is stored in the current application (may be the root application)

All the SDO stored in the root might be accessed in a global way by any ADF (PIN, DH,..)

Whenever required for interoperability, the global/local qualifier shall be indicated in the cryptographic information application PKCS#15 (e.g. password attributes)[10]

Note:

- The feature of the relevance depends on the use case to consider for the [IAS ECC] specification. If the use case only considers a Root application without any DF, the notion of global vs local will be pointless.

[10] Uniquely the VERIFY command is enabled to convey in its parameter P2 the indication of whether the key/pwd reference is local or global. This indication is reflected as well in related ISO/IEC 7816-15 TypeAttributes

- For the implementations of an ECC compliant application, it is recommended to store all the objects within the Root and ADF.

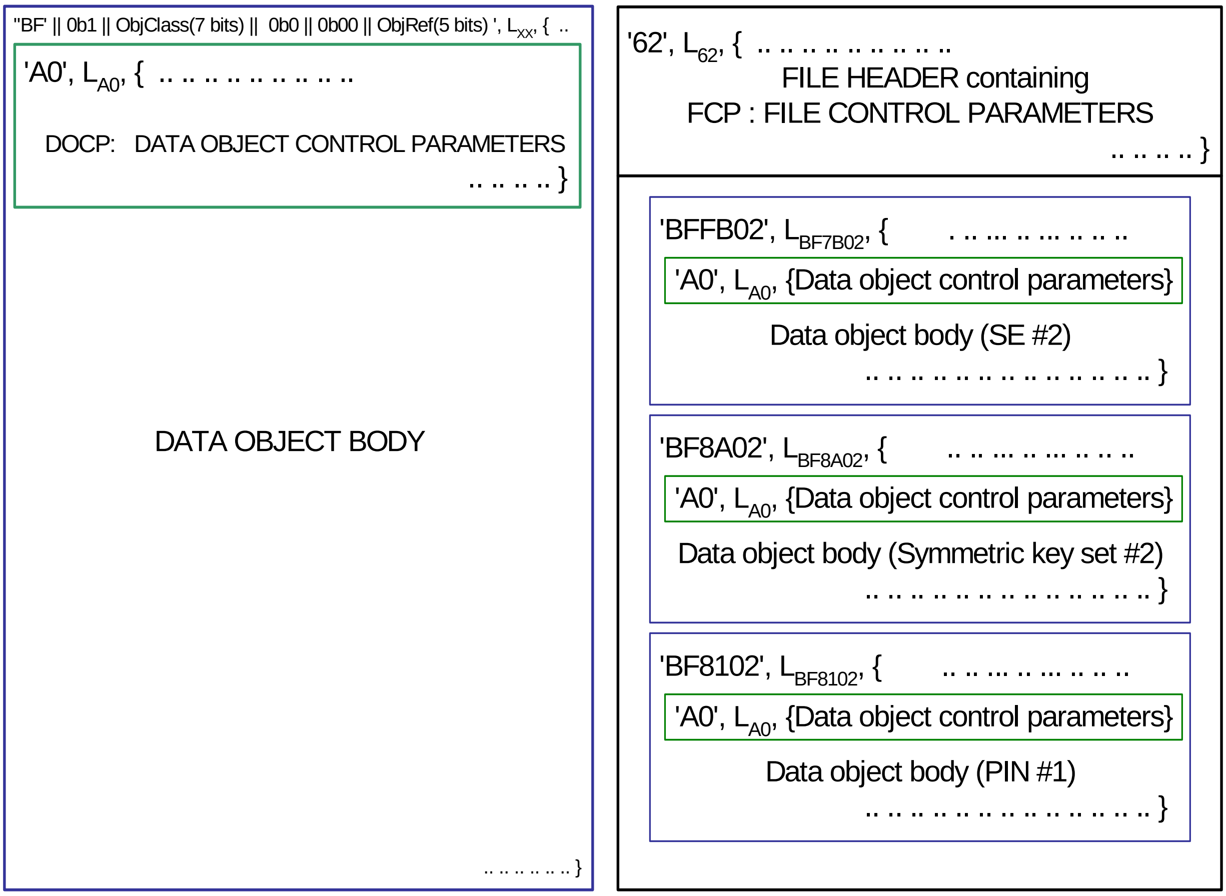

"BF' || 0b1 || ObjClass(7 bits) || 0b0 || 0b00 || ObjRef(5 bits) ', L_{XX}, { ..
'A0', L_{A0}, { .. ..
DOCP: DATA OBJECT CONTROL PARAMETERS
.. .. }
DATA OBJECT BODY
.. .. }
'62', L_{62}, { .. ..
FILE HEADER containing
FCP : FILE CONTROL PARAMETERS
.. .. }
'BFFB02', L_{BF7B02}, { . .. .
'A0', L_{A0}, {Data object control parameters}
Data object body (SE #2)
.. .. }
'BF8A02', L_{BF8A02}, { .. ..
'A0', L_{A0}, {Data object control parameters}
Data object body (Symmetric key set #2)
.. .. }
'BF8102', L_{BF8102}, { .. ..
'A0', L_{A0}, {Data object control parameters}
Data object body (PIN #1)
.. .. }

Figure 5 : security data objects and file structure[11]

[11] The SDO header shall be encapsulated in a '70' data object for the purposes of tag compatible allocation scheme as per [R2].

4.5 Dynamic control of files and Security Data Objects accesses

When an incoming command is processed and needs to access or use a SDO/File, the card verifies the security conditions are fulfilled according to the command requirement (i.e. match the current security status).

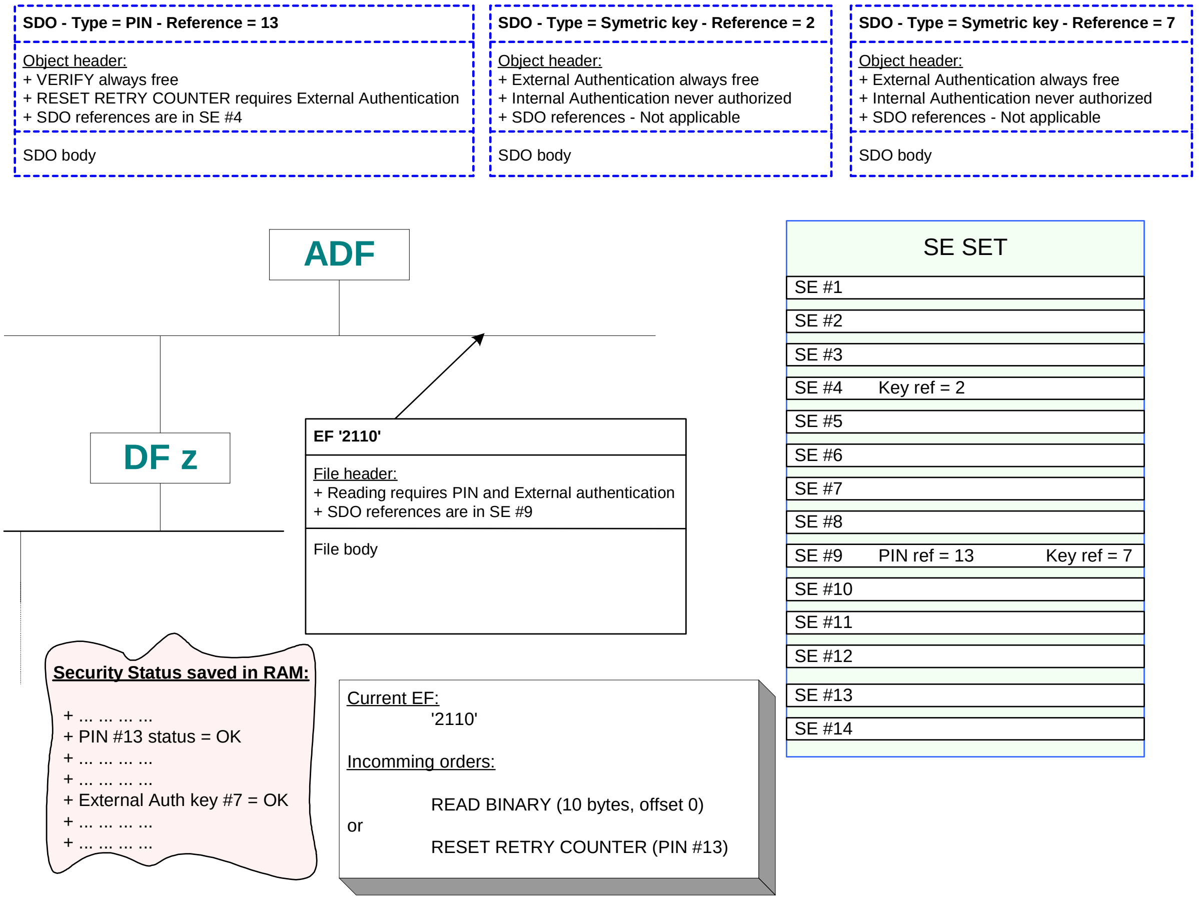

SDO - Type = PIN - Reference = 13
Object header:
+ VERIFY always free
+ RESET RETRY COUNTER requires External Authentication
+ SDO references are in SE #4
SDO body
SDO - Type = Symetric key - Reference = 2
Object header:
+ External Authentication always free
+ Internal Authentication never authorized
+ SDO references - Not applicable
SDO body
SDO - Type = Symetric key - Reference = 7
Object header:
+ External Authentication always free
+ Internal Authentication never authorized
+ SDO references - Not applicable
SDO body
ADF
DF z
EF '2110'
File header:
+ Reading requires PIN and External authentication
+ SDO references are in SE #9
File body
SE SET
SE #1
SE #2
SE #3
SE #4 Key ref = 2
SE #5
SE #6
SE #7
SE #8
SE #9 PIN ref = 13 Key ref = 7
SE #10
SE #11
SE #12
SE #13
SE #14
Security Status saved in RAM:
+ ... ...
+ PIN #13 status = OK
+ ... ...
+ ... ...
+ External Auth key #7 = OK
+ ... ...
+ ... ...
Current EF:
'2110'
Incomming orders:
READ BINARY (10 bytes, offset 0)
or
RESET RETRY COUNTER (PIN #13)

Figure 7: Smart card dynamic control example

In the Figure 7: Smart card dynamic control example

Example, the following is performed to determine if the accesses shall be granted or not:

<u>Case #1</u>: a READ command is requested on EF '2110'.

- The application retrieves the access conditions to read the file:
    - PIN and External authentication are required.
    - References on PIN and key related to the condition are available in SE#9
- The application accesses SE #9 from the relevant set:
    - The reference of the PIN to present before reading the file is 13.
    - The reference of the key to use to perform the External authentication required before reading the file content is 7.

- The card compares the requirement with its current security status in RAM. In this example, all conditions are fulfilled.
  Consequently, the READ command is authorized.

Case #2: a RESET RETRY COUNTER of PIN #13 is requested.

- The SDO header of PIN #13 is read to get the access condition to fulfill to be allowed to reset its retry counter.
  - An external authentication is required.
  - References on key related to the condition are available in SE#4.
- SE #4 is read from the relevant set:
  - The reference of the key used to perform the External authentication required before issuing reset retry counter command is 2.
- The card compares the requirement with its current security status in RAM. In this example, the condition is not fulfilled.→ Consequently, the RESET RETRY COUNTER request is rejected.

The security conditions to fulfill are always challenged against the current security status. It is up to the IFD to pass the adequate security conditions before calling the command.

The two examples here above shall be generalized.

5 AUTHENTICATION MECHANISMS

The [IAS ECC] offers several authentication schemes enabling to authenticate different roles, such as

- The card holder entitled to use the services offered by the card. It is called "User Authentication"
- The device communicating with the card, to establish a trusted channel (secure messaging) and protect the communication. It is called 'Device authentication"
- The administrator of a service, to administrate some features. It is called "Role authentication"

5.1 User authentication

The User authentication is knowledge based, i.e. submission of a PIN/password.

5.1.1 Knowledge based

The Authentication of the user relies on a shared secret, known by both the holder and the smartcard. The entities claiming it is the entitled card holder submits its secret to the smartcard. If the secret is the one the smart card expects, the access rights are granted. This secret may take any hexadecimal value, of any length (see § 3.4.6.1), i.e. [IAS ECC] does not mandate any specific format for the PIN.

The Card holder is authenticated by the means of the "VERIFY" command (see §9.4.1). The verification process uses a velocity checking mechanism, thus a remaining tries counter and a maximum error counter are defined for each PIN. If the verification fails, the tries counter is decremented by one and an error status that contains the remaining attempts is returned by the application.

When all available tries have failed, the PIN is blocked and can no longer be used. Note that a successful verification of the PIN resets its remaining tries counter to the maximum error counter.

Each PIN could be assigned a "usage counter", decremented each time a correct verification is performed. When the limit (set at the PIN creation) is reached, the PIN cannot be used anymore.

Once a PIN is blocked, the user should use a resetting code (PUK) to unblock that PIN (or any other operations requested by the access conditions protecting the PIN unblocking features of the PIN). During that process the PIN may optionally be set to a new value. This operation is performed thanks to the "RESET RETRY COUNTER" command (see §9.4.3)

After a successful unblocking, the remaining tries counter is reset to the maximum error counter.

The value may be changed by means of the 'CHANGE REFERENCE DATA" command (see §9.4.2)

5.1.1.1 Available services

The PIN verification is the only service available. The PIN in which the action applies is indicated in the P1P2 parameters of the command to use.

The PIN is submitted through to the command "VERIFY" (see §9.4.1)

5.1.1.2 Protection of an asset by this authentication

Once the User authentication was successfully performed, its internal status is set to "authenticated" in the current security status.

The checking of this status may allows operation on files or SDO as described in §3.4.

The Access conditions meaning “assets protected by the user authentication performed according to user authentication X”, is encoded using a CRT AT set in the Static Security Environment pointed out in from the SCB containing the conditions “User authentication” (see §4.2.3) encoded as follows:

Tag	Length	Tag	L	Valeur
'A4'	L_{A4}			
		'80'	'01'	Identifier of the user authentication algorithm (default value set to 0x00)
		'83'	'01'	PIN reference
		'95'	'01'	Usage Qualifier byte for User Authentication (0x08)

Note

- If several user authentication are allowed to grant the access condition, the template might gather several references to a PIN (see §3.5.6.2)
- The UQB is mandatory in this template, unlike the CRT AT template loaded in the current SE

5.2 Device Authentication

Device authentication aims at authenticating both entities willing to communicate and securing the communication between the card and a service provider (it might be a terminal, a server,..)

It allows authenticating both parties: the IFD authenticates the card, and the card authenticates the IFD.

Once the mutual authentication is performed, a shared secret called K.ICC/IFD is computed. The session keys for protecting the communication are generated from K.ICC/IFD.

Theses session keys enable to ensure the confidentiality of the data exchange (protection against disclosure, eavesdropping) and integrity of the data (protection against modification).

Furthermore, all the data exchanged within a session are tightly linked by the means of a counter (SSC). It is used to ensure the integrity over each exchange. This feature enables to protect the exchange against data insertion or deletion, and to ensure the data received are actually issued by the entity authenticated beforehand.

The session keys and the SSC that are computed during this operation are global to all the [IAS ECC] applications. They shall be used by all other [IAS ECC] application (provided they were not lost by a session breaking).

5.2.1 Authentication environments

Two kinds of environments shall be distinguished, considering their security level: trusted and untrusted environments. During the personalization phase, the issuer defines the access rules to the application resources. Different access rules can be defined for trusted and untrusted environments. The terminal shall adapt the authentication to the security level required by the resource it wishes to access

5.2.2 Symmetric authentication scheme

The smart card implements the mutual authentication scheme using symmetric keys using DES.

The MUTUAL AUTHENTICATE command is used to:

- Authenticate the terminal and the card.
- Generate two temporary keys K.ICC and K.IFD that will be further used to compute session keys for the secure messaging in the subsequent commands.
- Initialize the SSC, which is a counter used at each checksum computation.

Note: a successful smart card ⇔ IFD mutual authentication with session keys establishment does not mandate the use of secure messaging for all subsequent commands.

5.2.2.1 Cryptography involved

The authentication procedure is based on a Symmetric Key Set (KS) that is composed of two 3DES EDE2 Keys:

- KS.KENC: Key set encryption key – 16 bytes long
- KS.KMAC: Key set MAC computation key – 16 bytes long

The Key Set is managed as a Security Data Objects and shared by both parts (ICC & IFD)

From an input S, the authentication token is computed as follows:

- S is encrypted to get ENC(S) in 3DES in CBC mode with IV = "00 00 00 00 00 00 00 00" under the Key KS.KENC of the considered Key Set for Authentication.
- A Checksum is computed over ENC(S) to get MAC(S) regarding the MAC as specified §10 under the Key KS.KMAC of the considered Key Set for Authentication. IV used during computation is IV = "00 00 00 00 00 00 00 00".

The authentication token is made up with ENC(S) || MAC(S)

5.2.2.2 Authentication steps

A successfully authentication mandates the both parts share the same triple DES keys: K_{ENC} & K_{MAC}

Notation:

SN.ICC:	8 least significant bytes of the card serial number.
SNIFD:	8 least significant bytes of the terminal serial number.
RND.ICC:	8-byte-long random number generated by the card.
RND.IFD:	8-bytes-long random number generated by the terminal.
K.ICC:	32-bytes-long random number generated by the card.
K.IFD:	32-bytes-long random number generated by the terminal.

A successful Smart card ⇔ IFD mutual authentication leads to the generation of the shared secret
K.ICC/IFD = K.ICC XOR K.IFD

The device authentication continues with the computation of the secure messaging session keys SK_{ENC} and SK_{MAC} based on K.ICC/IFD and of the SSC (see 7.1.2).

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)
.../...		.../...

5.2.2.3 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	Tag	Length	Value
'A4'	L_{A4}			
		80	'01'	Algorithm identifier for symmetric device authentication (see §5.2.4)
		83	'01'	Key Reference of the associated Key Set

Note

- This template does not contain any UQB.
- Only one Key Reference is allowed

This template may be loaded within the CSE either through a 'MSE RESTORE' command (from a SSE holding it) or a 'MSE Set' command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2

5.2.2.4 Protection of an asset by this authentication

Once the device authentication was successfully performed, its internal status is set to "authenticated" in the current security status.

The checking of this status might allows operation on files or Security Data Objects as described in §4.2.

The Access conditions "assets protected by the symmetric device authentication performed with keyset X", is encoded using a CRT AT set in the Static Security Environment pointed out in the SCB containing the conditions "Secure Messaging" (see §4.2.3) encoded as follows:

Tag	Length	Tag	L	Valeur
'A4'	L_{A4}			
		'80'	'01'	Identifier of the device authentication algorithm (see §5.2.4)
		'83'	'01'	Key Reference of the associated Key Set
		'95'	'01'	Usage Qualifier byte for Mutual Authentication (0xC0)

Note

- If several symmetric device authentication are allowed to grant the access condition, the template might gather several references to a symmetric key set
- The UQB is mandatory in this template, unlike the CRT AT template loaded in the current SE
- In [IAS ECC], the Access conditions "Secure Messaging" mandates both a successful device authentication and an active secure messaging session as described in §7.1.3 The case in which only device authentication is mandated is out of the scope of [IAS ECC].

5.2.3 Device authentication with privacy protection

The device authentication with privacy protection scheme is described in Figure 8 - Diffie-Hellman mutual authentication scheme.

This asymmetric scheme relies on the Card Verifiable Certificate (CVC) PKI to authenticate the terminal (see §7.2) and makes use of the RSA cryptography. These certificates are non self descriptive certificates

To avoid the card disclosing private information, such as identity, a secure channel session is established prior to any other operation. To do so, the protocol starts with an unauthenticated Anonymous Diffie-Hellman key exchange and then authenticates the IFD before the ICC.

Note: a successful smart card ⬄ IFD mutual authentication with session keys establishment does not mandate the use of secure messaging for all subsequent commands.

Five main phases may be distinguished:

1. Set the privacy protection.
2. Transport of the IFD public key to the ICC.
3. External authentication of the IFD.
4. Transport of the ICC public key to the IFD.
5. Internal authentication of the ICC.

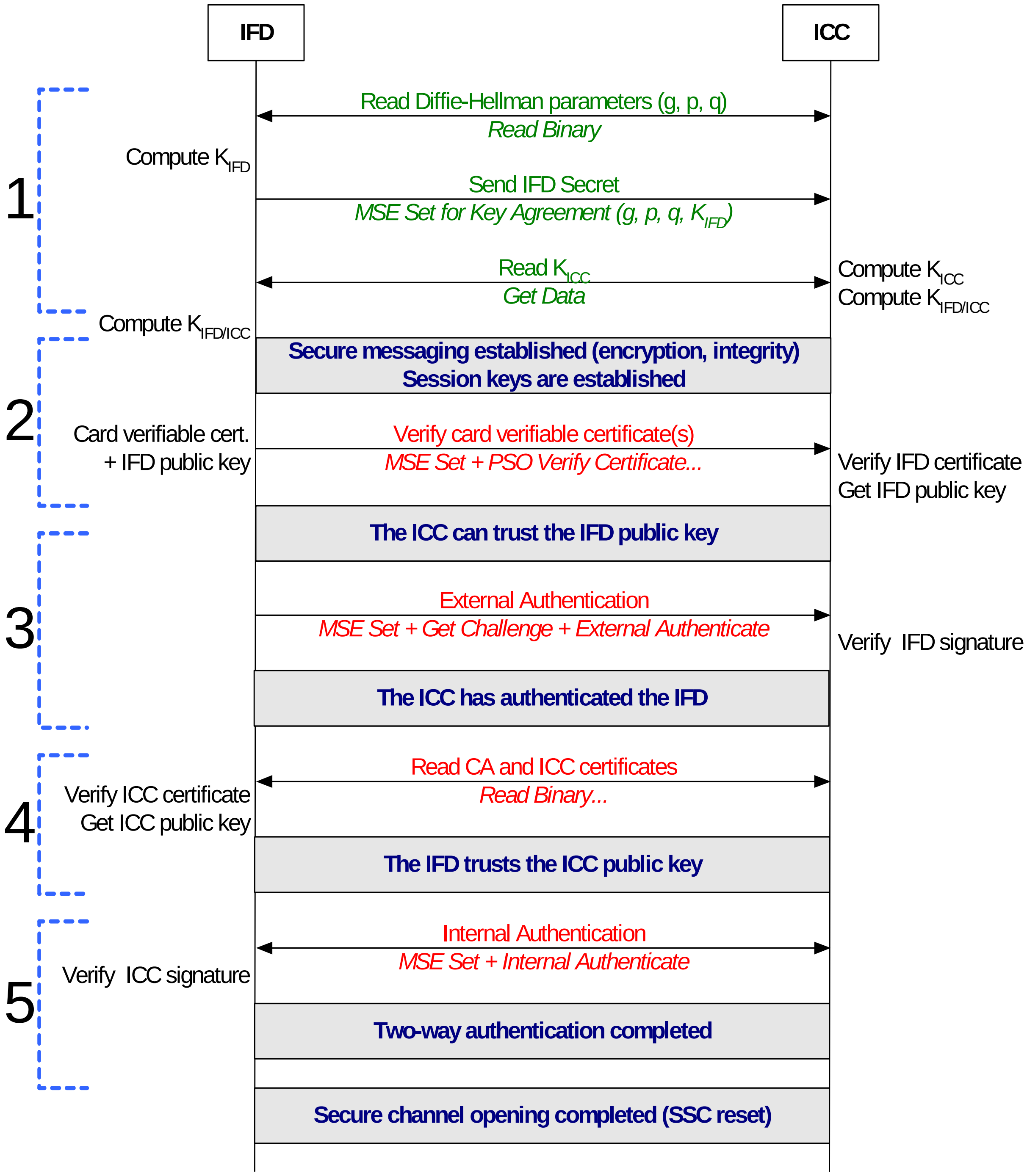

IFD
ICC
1
Compute K_{IFD}
Read Diffie-Hellman parameters (g, p, q)
Read Binary
Send IFD Secret
MSE Set for Key Agreement (g, p, q, K_{IFD})
Read K_{ICC}
Get Data
Compute K_{ICC}
Compute $K_{IFD/ICC}$
Compute $K_{IFD/ICC}$
2
Secure messaging established (encryption, integrity)
Session keys are established
Card verifiable cert.
+ IFD public key
Verify card verifiable certificate(s)
MSE Set + PSO Verify Certificate...
Verify IFD certificate
Get IFD public key
3
The ICC can trust the IFD public key
External Authentication
MSE Set + Get Challenge + External Authenticate
Verify IFD signature
The ICC has authenticated the IFD
4
Read CA and ICC certificates
Read Binary...
Verify ICC certificate
Get ICC public key
The IFD trusts the ICC public key
5
Internal Authentication
MSE Set + Internal Authentication
Verify ICC signature
Two-way authentication completed
Secure channel opening completed (SSC reset)

Figure 8 - Diffie-Hellman mutual authentication scheme

5.2.3.1 Set the privacy protection

To avoid the card disclosing private information, such as identity, a secure channel session is established prior to any other operation. To do so, the protocol starts with an unauthenticated anonymous Diffie-Hellman key exchange.

The IFD retrieves the Domain parameters the ICC uses D(g,p,q). These data are retrieved with

- 'READ BINARY' command (see §9.7.2). the file hosting DH public parameters shall be declared (EFID & SFI) in the cryptographic information application structure (PKCS#15).

  The public DH key parameters are:

  - p is the prime
  - g is a generator of the cyclic group, chosen in such a way that $g^q = 1 \bmod p$.
  - q is a order of the generator

The IFD generates an ephemeral key pair (PrK.IFD.DH,PuK.IFD,DH) over theses domain parameters as follows:

- The IFD randomly generates a number PrK.IFD.DH between 1 and (q-1) (may be equal to theses limits).
- The IFD computes PuK.IFD;DH = $g^{PrK.IFD.DH}$ mod p. If the computation leads to the result '1', a new random number shall be generated for PrK.IFD.DH

The IFD sends its ephemeral public Key for Key agreement PuK.IFD.DH to the ICC and selects the DH domain parameters to use with the command 'MSE Set KAT'.

Then, the IFD retrieves the ephemeral public key for key agreement PuK.ICC.DH generated by the ICC with the command 'GET DATA KICC'

Once the ICC received this command, it generates as well an ephemeral key pair (PrK.ICC.DH,PuK.ICC,DH) over theses domain parameters in the same way as the IFD did :

- The ICC randomly generates a number PrK.ICC.DH between 1 and (q-1) (may be equal to theses limits).
- The ICC computes PuK.ICC;DH = $g^{PrK.ICC.DH}$ mod p. If the computation leads to the result '1', a new random number shall be generated for PrK.ICC.DH

Once both parts know the Public Key of the other part, they can compute the quantity the shared secret ZZ.

The IFD computes it as follows:

$$ZZ = PuK.ICC.DH * PrK.IFD.DH = PuK.ICC.DH^{PrK.IFD.DH} \bmod p$$

The ICC computes it as follow:

$$ZZ = PuK.IFD.DH * PrK.ICC.DH = PuK.IFD.DH^{PrK.ICC.DH} \bmod p$$

This shared secret ZZ is used by the ICC and the IFD as an input for the session key computation. For more details see §7.1.4. The SSC used for the integrity of the exchange is initialized to '0000000000000001'.

All the following steps of the device authentication protocol is now protected thanks to a secure messaging for confidentiality & integrity.

The data are protected with a secure messaging for confidentiality & integrity computed with the dedicated keys (K.MAC & K.ENC).

5.2.3.1.1 Protocol steps

Smart card ⬄ IFD –privacy protection		
CARD ACCEPTING DEVICE (IFD)		**Smart card**
…/…		…/…
READ BINARY Reading of the Domain parameters used by the card to perform the anonymous Diffie Hellman.	→ ←	Send Domain parameters D (p,g,q)
Generate a Diffie Hellman KeyPair over D: (PrK.IFD.DH, PuK.IFD.DH)		
MSE Set KAT Select the Domain parameters to use Transmits the Public Diffie Hellman Key (PuK.IFD.DH)	→ ←	Save PuK.IFD.DH in the CSE
GET DATA K.ICC Retrieve the public Diffie Hellman Key used by the card (K.ICC PuK.ICC.DH).	→ ←	Generate a Diffie Hellman KeyPair over D: (PrK.ICC.DH, PuK.ICC.DH) Compute the common secret ZZ = PuK.IFD.DH * PrK.ICC.DH
Compute the common secret ZZ = PuK.ICC.DH * PrK.IFD.DH		

5.2.3.1.2 Setting the cryptographic context

The domain parameters are selected using CRT KAT encoded as follows:

Tag	Length	Tag	Length	Value
'A6'	L_{A6}			
		'83'	'01'	Domain parameters reference to use
		'91'	'Var'	Diffie Hellman public key of PuK.IFD.DH
		'80'	'01'	Algorithm identifier for the device authentication with privacy protection (see §5.2.4)

Note

- This template does not contain any UQB.
- Only one Domain parameters reference is allowed
- Only one Diffie Hellman Public Key (for the IFD) is allowed
- The Algorithm identifier is used to indicate the KDF to use for the SM session key generation

This template is loaded in the Current Security environment (CSE) with the MSE Set Command for Key agreement. For more details about the behavior of this commands, see §9.9.2.

5.2.3.1.3 Particular issues regarding the retrieval of the domain parameters

The Protocol steps described in §5.2.3.1.1 is fully compatible with the protocol described in [R18].

If the compliancy with [R18] is claimed by the ICC and the IFD, the domain parameters shall be retrieved using a 'READ BINARY' command (see §9.7.2). This operation can be performed freely on the mandatory file EF.DH since it is located under the Root and always readable.

Nevertheless, [IAS ECC] specification handles the domain parameters as Security Data Object as well. Therefore, the domain parameters could also be retrieved using a 'GET DATA SDO' command (see §9.8.1.1), provided the Access Conditions allow it.

When both data structures are present (EF.DH and SDO(s)), Each Domain parameters with the same identifier (in EF.DH and within the relevant SDO) shall be the same.

The EF.DH as described in [R18] shall be used when a device authentication with privacy protection is performed from the Root, as it is the only case considered by this standard.

However, if a device authentication with privacy protection is performed from a location different from the Root, the command 'READ BINARY' on the EF.DH or the 'GET DATA SDO' on the matching SDO could be indistinctly used, as this case it out of the scope of [R18].

When the 'GET DATA SDO' command is used to retrieve the domain parameters used by the ICC, the IFD shall retrieve each parameter, component by component (a 'GET DATA SDO' shall be used for each component)[12],

5.2.3.2 Transport of the IFD public key to the ICC

The first device to send its credentials is the IFD. The ICC sends its private information only once the IFD is fully authenticated (to ensure Privacy protection).

The IFD proves it genuiness to the ICC with a chain of certificates. It starts from the common key (PuK.IFD.RCA.CS_AUT) certification authority (R.C.A) both know. Each time a certificate is verified, a new (public) key (PuK.IFD.CS_AUT) is exported within the ICC. This key is trusted by the ICC and is used to verify another certificate/authenticate the IFD.

The IFD keeps on performing these steps until it exports its authentication key PuK.IFD.AUT

Thanks to the chain of certificate, the ICC can ensure the Root Certification authority (PuK.IFD.RCA.CS_AUT) certifies the public key of the IFD (PuK.IFD.AUT).

The process of key export is depicted hereafter

PuK.IFD.RCA.CS_AUT -> PuK.IFD.CS_AUT -> …-> PuK.IFD.CS_AUT -> PuK.IFD. AUT

The "Digital signature template" governs the process of certificate verification of the current SE, that describes all the parameters needed to verify certificates.

[12] No instruction with this respect is presently provided in ECC2. The retrieval of domain parameters with GET DATA SDO may run in a trouble within ECC-compliant terminal.

Once the public key of the IFD is known and can be trusted by the ICC, the IFD can be authenticated with an external authentication.

The format of the credentials (certificates) is described in §7.2

5.2.3.2.1 Protocol steps

One has to distinguish between the first step, used to initiate the certificate chain. The certificate contains a CA key issued by a Root Certification Authority PuK.RCA.IFD.CS_AUT loaded on card.

The subsequent steps may be repeated as many times as needed. The certificate contains a CA key PuK.IFD.CS_AUT issued by the CA key previously extracted from a certificate.

First step: initialization of a certification chain

Smart card ⇔ IFD –transport of the IFD public key **First step**		
CARD ACCEPTING DEVICE (IFD)		**Smart card**
.../...		.../...
MSE Set DST Select the RCA key PuK.RCA.IFD.CS_AUT	→ ←	
PSO Verify Certificate Send the certificate certified by the RCA	→ ←	The CA key is loaded in memory and might be used to verify other certificates PuK.IFD.CS_AUT is available

Further step

Smart card ⇔ IFD –transport of the IFD public key **Subsequent step(s)**		
CARD ACCEPTING DEVICE (IFD)		**Smart card**
.../...		.../...
MSE Set DST Select the CA key PuK.IFD.CS_AUT	→ ←	
PSO Verify Certificate Send the certificate certified by the former CA	→ ←	The CA key is loaded in memory and might be used to verify other certificates PuK.IFD.CS_AUT/PuK.IFD.AUT is available

5.2.3.2.2 Setting the cryptographic context

The service is realized by setting the relevant template in the Current security Environment (CSE). The template is a CRT DST encoded as follows:

Tag	Length	Tag	Length	Value
' B6'	L_{B6}			
		‘83’	'01'	Key Reference of the SDO PuK.RCA.IFD.CS_AUT (for the first step only)
		OR		
		‘83’	'0C'	Name of PuK.IFD.CS_AUT (C.H.R. see §7.2.5.3)
		‘80’	‘01’	Algorithm identifier for certificate verification (see below)

Note

- This template does not contain any UQB.
- Only one Key Reference or name is allowed

This template may be loaded within the CSE either through a ‘MSE RESTORE’ command (from a SSE holding it – only for the RCA/selection of the key in the first step) or a ‘MSE Set’ command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

5.2.3.2.3 Algorithm identifier

Here is a summary of the Algorithm Identifiers used in the SDO to identify a certificate verification algorithm.

Certificate verification	Value
ISO/IEC 9796-2 (DS-scheme 1) – SHA-1	‘11’
ISO/IEC 9796-2 (DS-scheme 1) – SHA-256	‘41’

This algorithm identifier shall be assigned to the root key used to initiate the certificate verification. If it is absent, the default certificate verification is made with RSA 9796-2 SHA-1

It is not attached to ephemeral public key as such. It is figured by an O.I.D as described in §7.2.5.6.

5.2.3.3 External authentication of the IFD

The external authentication is initiated by sending to the card the “authentication template” with the command MSE set (see 9.9.2 for data structure). It describes the key to be used for the external authentication. This key (PuK.IFD.AUT) is made available after the last certificate verification. The Key shall be selected by using its name (C.H.R. used in last certificate).

Now that the ICC trusts the IFD public key, a challenge/response authentication takes place.

- The IFD asks the ICC to generate a challenge RND.ICC;

- The IFD signs the challenge with its private key for authentication PrK.IFD.AUT;
- The ICC verifies the signature SIG.IFD with the IFD public key PuK.IFD.AUT

The IFD is then fully authenticated by the ICC.

Note: The secure messaging for confidentiality & integrity (using the session keys computed during the Diffie-Hellman key exchange) protects this challenge/response sequence.

If the external authentication fails, the process of device authentication is stopped. Otherwise it carries on the next step.

The algorithm used for the external authentication is RSA signature padded according to ISO9796-2.

5.2.3.3.1 Protocol steps

The external authentication of the IFD is performed as follows:

- SN.IFD: terminal serial number (shall be 8 bytes long)
- RND.ICC: 8-byte-long random number generated by the ICC.
- PuK.IFD.AUTH: Public Authentication key of the IFD.
- PRND: Pseudo Random number used as padding in the authentication cryptogram.
- PuK.IFD.AUTH: DH public key generated by the IFD
- PuK.ICC.AUTH: DH public key generated by the IFD
- (g,p,q): DH Domain parameters used by the ICC and IFD.

Entity to authenticate		ICC
MSE Set AT Select the Authentication key PuK.IFD.AUT	→ ←	
GET CHALLENGE	→ ←	Generate Random number RND.ICC
Compute an authentication token SIG SIG.IFD = DS(RND.ICC)		
EXTERNAL AUTHENTICATE Data =SN.IFD | SIG.IFD Where SIG.IFD = ENC(6A|PRND|h(PRND|PuK.IFD.DH|SN.IFD|RND.ICC|PuK.ICC.DH|g|p|q)|BC) PRND is a pseudo random number chosen such as the data to cipher has the correct size. h(z) designates hashing function SN.IFD is the serial number of the IFD	→ ←	Verify the signature, RND;ICC,SN.IFD,PuK.ICC.DH, PuK.IFD.DH, the Domain parameters D

5.2.3.3.2 Setting the cryptographic context

The service is realized by setting the relevant template in the Current security Environment (CSE). The template might be set in the current security environment prior to performing the asymmetric external device authentication.

The template is a CRT AT encoded as follows:

Tag	Length	Tag	Length	Value
'A4'	L_{A4}			
		‘83’	'0C'	Name of the authentication public key PuK.IFD.AUT

Note

- This template does not contain any UQB.
- Only one Key Name is allowed

This template may be loaded within the CSE through a ‘MSE Set’ command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

The algorithm to use with the public key is attached to ephemeral public key by its O.I.D as described in §7.2.5.6. (conveyed in the certificate).

5.2.3.4 Transport of the ICC public key to the IFD

Once the IFD is authenticated and can be trusted, the ICC can safely transmit its credentials to the IFD. In the same way as the ICC check the chain of certificates for the public key of the IFD, the IFD does the same to ensure the public key of the ICC can be trusted.

5.2.3.5 Internal authentication of the ICC

Once the IFD is authenticated, the ICC can authenticate itself to the IFD. The IFD selects the authentication secret key of the ICC PrK.ICC.AUT. The IFD generates a challenge that the ICC signs and returns to the IFD. This internal authentication steps is preceeded by the selection of the ICC private key to use. This selection is realized by sending the card the MSE Set command containing the "authentication template" with the ID of the authentication key to use. This template describes the parameters are to be used for the internal authentication. In particular, it specifies

- The ID of the private key to use, matching the public key the IFD retrieved.
- The Name of the Authentication key the IFD used (PuK.IFD.AUT)
- The algorithm to use for the ICC authentication

The internal authentication is realized through a challenge/response mechanism:

- The IFD sends a challenge RND.IFD to the ICC;
- The ICC signs the challenge with its private key for authentication PrK.ICC.AUT;
- The IFD verifies the signature SIG.ICC with the ICC public key PuK.ICC.AUT

The ICC is then fully authenticated by the IFD.

The algorithm used for the internal authentication is RSA signature padded according to ISO9796-2 (see 9.9.2 for data structure.).

As a result, both side have been authenticated (credentials have been exchanged) and privacy has been protected under a Secure Messaging for confidentiality & integrity.

For the next exchange, the commands/response data may be protected with the secure messaging in for confidentiality & integrity using the session keys generated at the beginning of the process: SKMAC and SKENC. The secure messaging for integrity for the subsequent command is computed with the correct starting value for

SSC = RND.ICC (4 least significant bytes) || RND.IFD (4 least significant bytes)

5.2.3.5.1 Protocol steps

The internal authentication of the ICC is performed as follows:

- SN.ICC: 8 least significant bytes of the card serial number
- RND.IFD: 8-byte-long random number generated by the IFD.
- PrK.ICC.AUTH: Private Authentication key of the ICC.
- PRND: Pseudo Random number used as padding in the authentication cryptogram.
- PuK.IFD.AUTH: DH public key generated by the IFD
- PuK.ICC.AUTH: DH public key generated by the IFD
- (g,p,q): DH Domain parameters used by the ICC and IFD.

Terminal		Carte
MSE Set AT Select the Authentication key PrK.ICC.AUT	→ ←	
INTERNAL AUTHENTICATE Data=RND.IFD	→ ←	The card returns SN.ICC | SIG.ICC Where SIG.ICC = ENC(6A|PRND|h(PRND|PuK.ICC.DH|SN.ICC|RND.IFD|PuK.IFD.DH|g|p|q))BC) PRND is a pseudo random number chosen such as the data to cipher has the correct size. h(z) designates the hashing function SN.ICC is the serial number of the ICC
Verify the signature, RND;IFD,SN.ICC,PuK.IFD.DH,PuK.ICC.DH,the Domain parameters		

5.2.3.5.2 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	Tag	Length	Value
'A4'	L_{A4}			
		‘80’	'01'	Algorithm identifier for the device authentication with privacy protection (see §5.2.4)
		‘84’	'01'	Private Key SDO ID of PrK.ICC.AUT

Note

- This template does not contain any UQB.
- Only one key Reference is allowed (for the ICC private key for authentication)

This template may be loaded within the CSE either through a ‘MSE RESTORE’ command (from a SSE holding it) or a ‘MSE Set’ command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

5.2.3.6 Protection of an asset by this authentication protocol

Once the authentication protocol was successfully performed, the role associated to the IFD public key PuK.IFD.AUT is validated. It means the role of IFD shall be validated (‘01’).

The checking of this status might allows operation on files or Security Data Objects.

The Access conditions “assets protected by the device authentication”, is encoded using a CRT AT set in the Static Security Environment for security policy pointed out in the SCB containing the conditions “Secure Messaging” (see §4.2.3) encoded as follow:

Tag	Length	Value		
‘A4’	L			
		Tag	**L**	**Value**
		‘80’	‘01’	Algorithm identifier for the device authentication with privacy protection (see §5.2.4)
		‘84’	‘01’	Private key reference
		‘95’	‘01’	'40' ≡ UQB for internal authentication

Note

- The UQB is mandatory in this template, unlike the CRT AT template loaded in the current SE
- In [IAS ECC], the Access conditions “Secure Messaging” mandates both a successful device authentication and an active secure messaging session as described in §7.1.3 The case in which only device authentication is mandated is out of the scope of [IAS ECC].

5.2.4 Algorithm identifier

Here is a summary of the Algorithm Identifiers used in the SDOs and in the Control reference template to identify a device authentication protocol.

The Algorithm identifier byte for the device authentication is encoded as follows:

b8	b7	b6	b5	B4	b3	b2	b1	Hexa value	Signification
x	-	-	-	-	-	-	-		Session key derivation[13] 0 -> SHA-1 1 -> SHA-256
-	x	-	-	-	-	-	-		Type of Secure messaging 0 -> Based on TDES
-	-	x	x	x	x	x	x		Encoding of the device authentication protocol
0	0	0	1	1	0	1	1	‘1B’	Asymmetric Device authentication with privacy protection SHA1
1	0	0	1	1	0	1	1	‘9B’	Asymmetric Device authentication with privacy protection SHA256
0	0	0	0	1	1	0	0	‘0C’	Symmetric Device authentication with symmetric scheme using TDES SHA1
1	0	0	0	1	1	0	0	‘8C’	Symmetric Device authentication with symmetric scheme using TDES SHA256

These algorithms Identifiers are set in the SDOs involved in the device authentication protocol, i.e.

- Symmetric key set used the symmetric authentication scheme
- ICC Authentication private key & Diffie Hellmann used for the device authentication with privacy protection

5.3 Role Authentication

This chapter presents the procedure to authenticate an external entity to the card in order to associate to it a specific role (e.g. access rights). The [IAS ECC] considers these schemes:

- A symmetric role authentication
- An asymmetric role authentication based on RSA

5.3.1 Symmetric Role authentication

The following procedure describes:

- The cryptographic operation that allows the authentication
- The specification of the associated role in the card

This feature is described in [R16] §7.3

[13] Not mandated by [R16]

5.3.1.1 Cryptography involved

The authentication procedure is based on a Symmetric Key Set (KS) that is composed of two 3DES EDE2 Keys:

- KS.KENC: Key set encryption key – 16 bytes long
- KS.KMAC: Key set MAC computation key – 16 bytes long

The Key Set is managed as a Security Data Objects and shared by both parts (ICC & IFD)

From an input S, the authentication token is computed as follows:

- S is encrypted to get ENC(S) in 3DES in CBC mode with IV = "00 00 00 00 00 00 00 00" under the Key KS.KENC of the considered Key Set for Authentication.
- A Checksum is computed over ENC(S) to get MAC(S) regarding the MAC as specified §10 under the Key KS.KMAC of the considered Key Set for Authentication. IV used during computation is IV = "00 00 00 00 00 00 00 00".

The authentication token is made up with ENC(S) || MAC(S)

5.3.1.2 Authentication sequence

Notation

- SN.ICC: 8 least significant bytes of the card serial number.
- RND.ICC: Random Number generated by the card - 8 bytes long

Terminal / 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)**

At the end of these Protocol steps, the IFD is authenticated.

5.3.1.3 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	Tag	Length	Value
'A4'	L_{A4}			
		‘80’	'01'	‘1C’ ≡ Algorithm identifier for symmetric role authentication (see §5.3.3)
		‘83’	'01'	Key Reference of the associated Key Set

Note

- This template does not contain any UQB.
- Only one Key Reference is allowed

This template may be loaded within the CSE either through a ‘MSE RESTORE’ command (from a SSE holding it) or a ‘MSE Set’ command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

5.3.1.4 Protection of an asset by this authentication protocol

Once the role authentication was successfully performed, its internal status is set to “authenticated”.

The checking of this status might allows operation on files or Security Data Objects.

The Access conditions “assets protected by the symmetric role authentication performed with keyset X”, is encoded using a CRT AT set in the Static Security Environment pointed out in the SCB containing the conditions “External authentication” (see §4.2.3) encoded as follows:

Tag	Length	Tag	L	Valeur
'A4'	L_{A4}			
		'80'	'01'	‘1C’ ≡ Algorithm identifier for symmetric role authentication (see §5.3.3)
		'83'	'01'	Key Reference of the associated Key Set
		'95'	'01'	Usage Qualifier byte for External Authentication (0x80)

Note:

- If several role authentication are allowed to grant the access condition, the template might gather several references to a symmetric key set
- The UQB is mandatory in this template, unlike the CRT AT template loaded in the current SE

5.3.2 Asymmetric authentication based on RSA

Public key are transported through Card Verifiable Certificates (CVC) which are fully described in chapter 7.2.

This step can only be realized after a certificate chain is initiated (as described in §5.2.3.2). In this case, the last key to be imported is a role authentication key called PuK.IFD.RA, and not a device authentication key PuK.IFD.AUT. This chapter only describes the subsequent steps

This feature is described in [R16] §7.4

5.3.2.1 Cryptography involved

From an input S, the authentication token is computed as follows:

- S is padded as follows

  M = 0x6A || PRND || hash(PRND || S) || 0xBC

  Where PRND is a pseudo random number (no consecutives bytes to zero) chosen in such a way the size of the whole data set is equal to the size of the key used for the role authentication

- The hash function may be either SHA-1 or SHA-256 (it depends on the role Algorithm chosen for the role authentication – see §10.5)

- M is subsequently encrypted using the private key to get DS(S)

The authentication token is DS(S)

5.3.2.2 Authentication sequence

Notation:

- o CERT.PATH: Certificate chain constructed by the entity as described in chapter 7.2.2
- o PuK.IFD.RA: Public Key of the entity to be authenticated
- o RND.ICC: Random Number generated by the card - 8 bytes long
- o SN.ICC: 8 least significant bytes of the card serial number
- o DS: Signature algorithm according to ISO9796-2 with SHA-1 or SHA-256
- o DS-1: Signature verification algorithm according to ISO9796-2 with SHA-1 or SHA-256

Entity to authenticate		ICC
Retrieve the Certificate chain CERT.PATH from the CA implicitly trusted by the ICC from RootCA down to the Entity		
Present the Certificate chain as described §7.2.6	→ ←	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

5.3.2.3 Setting the cryptographic context

The service is realized by setting the relevant template in the CSE. The template might be set in the current security environment prior to performing the asymmetric external role authentication.

The template is a CRT AT encoded as follows:

Tag	Length	Tag	Length	Value
'A4'	L_{A4}			
		'83'	'0C'	Name of the Public Key for role Authentication PuK.IFD.RA
		'80'	'01'	Algorithm identifier for asymmetric role authentication based on RSA (see §5.3.3)

Note

- This template does not contain any UQB.
- Only one Key Name is allowed

This template may be loaded within the CSE through a 'MSE Set' command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

5.3.2.4 Protection of an asset by this authentication protocol

The roles associated with the Key are mapped in the CHA field

Upon successful authentication, the role of PuK.IFD.RA is validated.

The checking of this status might allow operation on files or Security Data Objects.

The Access conditions "assets protected by the asymmetric role authentication performed with the role X", is encoded using a CRT AT set in the Static Security Environment pointed out in the SCB containing the conditions "External authentication" (see §4.2.3) encoded as follows:

Tag	Length	Tag	L	Value
'A4'	L_{A4}			
		'80'	'01'	Algorithm identifier for asymmetric role authentication based on RSA (see §5.3.3)
		'5F4C'	'Var'	CHA
		'8A'	'01'	Identifier of the comparison algorithm (see §7.2.5.4.2)
		'95'	'01'	Usage qualifier byte External Authentication ('0x80')

Note

- If several role authentications are allowed to grant the access condition, the template might gather several CHA.
- The UQB is mandatory in this template, unlike the CRT AT template loaded in the current SE

When a CHA is validated through the external role authentication, its role id is compared with the reference Role id. The comparison is performed upon the following conditions:

- The validated privilege shall only be compared to reference CHA (s) of the same length
- While the conditions is 'false' the comparison is carried on with the next allowed reference Role Id

The field 'Identifier of the comparison algorithm' is used to decide whether the right is granted or not.

Once a CHA is valid, it is compared to the reference CHA(s) present in the template to grant the access or not

5.3.3 Algorithm identifier

Here is a summary of the Algorithm Identifier used in the Control reference template to identify a role authentication method.

Role authentication method	Value
Symmetric scheme	'1C'
Asymmetric scheme based on RSA SHA1	'1E'
Asymmetric scheme based on RSA SHA256	'9E'[14]

[14] Not mandated by [R16]

6 CRYPTOGRAPHIC SERVICES

Based upon these authentication features, the [IAS ECC] offers several cryptographic services controlled by these roles it authenticated.

These features use cryptographic objects (i.e. a SDO – see §3.4). When they are successfully performed, usage counter of the cryptographic object – when exists - is decremented. When this counter reaches its limit (zero), the cryptographic feature cannot be performed anymore. (For more details see §3.4.5)

6.1 Digital Signature

The Digital signature computation is the basic service each [IAS ECC] card should offer to be compliant with [R17].

The card computes a digital signature over data it receives. The data it receives are partially hashed by the SCA. It computes a hash over the data to be signed so that only remains a data block whose size is lower than 64 bytes. The partial hash computed, the number of data block hashed (the hash is performed over data block of 64 bytes) and the remaining data block is sent to the [IAS ECC].

Once the [IAS ECC] card received these data, it computes the last round of the hash and signs it. These two steps should be consecutives.

6.1.1.1 Cryptography involved

The [IAS ECC] may sign the data using an RSA scheme.

The algorithms available are the following :

- PKCS#1 v1.5 SHA-1
- PKCS#1 v1.5 SHA-256

The encryption block during signing is built as follows:

EB = 00 || 01 || PS || 00 || T
where T is the DER encoding of digestInfo:

digestInfo::= SEQUENCE {

digestAlgorithm AlgorithmIdentifier of SHA-X,

digest OCTET STRING

}

AlgorithmIdentifier ::= SEQUENCE {

ObjectIdentifier of SHA-X,

Parameter PARAMETER

}

PARAMETER = NULL

The algorithm identifier is the following:

- For SHA-1 : **1.3.14.3.2.26**
- For SHA-256 : **2.16.840.1.101.3.4.2.1**

For PKCS#1 v1.5 SHA-1, the exact sequence coding of *digestInfo* is:

```
‘30’ ‘21’
          ‘30’ ‘09’
                    ‘06’ ‘05’ ‘2B 0E 03 02 1A’
                    ‘05’ ‘00’
          ‘04’ ‘14’ SHA-1 digest (20 bytes)
```

Table 21: *digestInfo* encoding for SHA-1

For PKCS#1 v1.5 SHA-256, the exact sequence coding of *digestInfo* is:

```
‘30’ ‘31’
          ‘30’ ‘0D’
                    ‘06’ ‘09’ ’60 86 48 01 65 03 04 02 01’
                    ‘05’ ‘00’
          ‘04’ ‘20’ SHA-256 digest (32 bytes)
```

Table 22: *digestInfo* encoding for SHA-256

6.1.1.2 Particular issues for qualified signature

If a qualified signature is needed, i.e. compliance with the requirement of the PP SSCD the private key SDO used for the signature should have the non repudiation flag set to true . This flag ensures that any user authentication protecting the SDO private key is invalidated, each time one of the following commands is issued:

- “PSO Compute Digital signature” (when compliant with PP SSCD type 2 & 3)
- “Generate Asymmetric Key Pair” (when compliant with PP SSCD type 3)
- “Put Data” (when compliant with PP SSCD type 2)

The behavior when the non repudiation flag is different from true or false is out of the scope

The use of the digital signature imposes as well to store the certificate of the matching public key in a dedicated file (SVD). This certificate is signed by the service issuer and ensures the identity of the signatory. It does not mandate to store all the certificates within the same file nor at any particular location, since their location can be retrieved through PKCS#15 structures.

Qualified signature according to PP SSCD requests as well to personalize in a given manner the private key SDO.

Moreover, depending on the environment considered to perform the qualified signature, a secure channel (secure messaging) might be used to ensure the following objectives:

- Integrity of the DTBS (Data to Be Signed)
- Confidentiality of the digital signature to protect the private key from any forgery attempt
- Confidentiality and integrity of the VAD (user authentication data)

The user authentication assets used to use the private key for signature shall use an error counter.

The PRIS identifies two different levels for the qualified signature generation:

- 2* - In that case, the signature key could be used for another purpose (authentication for instance).
- 3* - In that case, the signature key can only be used for qualified signature.

6.1.1.3 Protocol steps

Notation:

- M: Message to sign
- PartialHash(M) : Partial hash of the message M computed over M excluding the last data block whose size should be lower than 64 bytes (size of the data block for hashing)
- Counter(M) : Counter coded over 8 bytes which indicates the number of **bits** hashed by the outside: e.g. 2 data block (64 bytes each) hashed by the SCA would be encoded as '0000000000000400'
- RemainingMessage(M) : Remaining data block (LSB of M) whose length is lower than the hashing data block size (64 bytes). The case in which this field is empty is out of the scope of [IAS ECC].
- Hash(M) : Hash of the message M, which is used as input for the digital signature computation

The service is performed in two steps. The first one performs the last round of the partial hash, and the second one realizes the digital signature computation over the entire hash computed thanks to the last round computation.

The computation of the (partial hash) is performed thanks to the command 'PSO hash' (see §9.6.1) as described in [R15]. The two hashing method (on card & off card) described in [R15] are not mandated by this [IAS ECC] version.

The computation of the digital signature is performed thanks to the command 'PSO compute digital signature" (see §9.6.2)

IFD		ICC
The IFD performs the partial hash calculation over M. The computation outcomes are the following: o PartialHash(M)		

<table>
<tr><th>IFD</th><th></th><th>ICC</th></tr>
<tr><td>o Counter(M)
o RemainingMessage(M)</td><td></td><td></td></tr>
<tr><td>Send partial hash data and require final hash round calculation. Use of the command ‘PSO Hash’ (see §9.6.1)
Incoming data are for hash calculation
<table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'90'</td><td>'Var'</td><td>PartialHash(M) || Counter(M)</td></tr>
<tr><td>'80'</td><td>Var. (≤'40')</td><td>RemainingMessage(M)
Last block, to be hashed by the card</td></tr>
</table></td><td>→
←</td><td>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</td></tr>
<tr><td>The IFD requires the signature calculation- Use of the command ‘PSO Compute Digital Signature’ (see §9.6.2)</td><td>→
←</td><td>The card calculates the signature with the selected private key and returns the result.</td></tr>
</table>

6.1.1.4 Setting the cryptographic context

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

- A template to set the hash context (SHA-1 or SHA-256)
- A template to set the digital signature computation (PKCS#1 v1.5 SHA-1 or SHA-256)

The hash template is a CRT HT encoded as follows:

Tag	Length	Tag	Length	Value
'AA'	L_{AA}			
		‘80’	'01'	Hash Algorithm (see §6.1.2)

Note

- This template does not contain any UQB nor key reference

This template may be loaded within the CSE either through a ‘MSE RESTORE’ command (from a SSE holding it) or a ‘MSE Set’ command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

The signature computation template is a CRT DST encoded as follows:

Tag	Length	Tag	Length	Value
'B6'	L_{B6}			
		'84'	'01'	Private Key Reference
		'80'	'01'	Algorithm Identifier for Digital signature computation (see §6.1.3)

Note

- This template does not contain any UQB.
- Only one Key Reference is allowed

This template may be loaded within the CSE either through a 'MSE RESTORE' command (from a SSE holding it) or a 'MSE Set' command (explicitly loads it in the CSE). The commands are formatted as follows:

MSE RESTORE

COMMAND PARAMETER	MEANING
CLA INS P1 P2	ISO '22' 'F3' Identifier of the SSE (SEID) to restore containing the relevant CRT DST
L_c field	'00'
Data field	Absent
L_e field	Absent

MSE SET

COMMAND PARAMETER	MEANING
CLA INS P1 P2	ISO '22' '41' 'B6'
L_c field	Input data length
Data field	Content of the CRT DST for digital signature as described above
L_e field	Absent

For more details about the behavior of this commands, see §9.9.2

Combination of allowed algorithms:

Signature Scheme	Hash Algorithm to set in the algorithm field of HT	Digital signature algorithm to set in the algorithm field of DST
PKCS#1 v1.5 SHA-1	SHA-1	PKCS#1 v1.5 SHA-1
PKCS#1 v1.5 SHA-256	SHA-256	PKCS#1 v1.5 SHA-256

6.1.2 Algorithm identifier for Hash algorithm

Here is a summary of the Algorithm Identifier used in the Control reference template to identify a hash algorithm.

Hashing method	Value
SHA-1	‘10’
SHA-256	‘40’

Note:

The hash algorithm chosen MUST match the digital signature algorithm

6.1.3 Algorithm identifier for Digital signature

Here is a summary of the Algorithm Identifier used in the SDO and in the Control reference template to identify a digital signature algorithm.

Digital signature	Value
PKCS#1 v1.5 – SHA-1	‘12’
PKCS#1 v1.5 – SHA-256	‘42’

Note:

The hash algorithm chosen with a dedicated algorithm ID MUST match the digital signature algorithm

6.2 Client / server authentication

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

The IFD would probably call the READ BINARY command to read the certificate of the RSA key dedicated to the authentication mechanism (the certificate format might be X509, CVC…depending on the card personalization).

This feature is described in [R16] §6.4

6.2.1.1 Cryptography involved

For the RSA scheme:

The [IAS ECC] 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 equal to the key length.

The Message M' is then encrypted using the private key.

It **assumes** the incoming data M are formatted as described by PKCS#1 v1.5:

M = digestInfo

where

digestInfo::= SEQUENCE {

digestAlgorithm AlgorithmIdentifier of hash function,

digest OCTET STRING

}

AlgorithmIdentifier ::= SEQUENCE {

ObjectIdentifier of hash function,

Parameter PARAMETER

}

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

6.2.1.2 Protocol steps

The service is realized through the command 'INTERNAL AUTHENTICATE' (see §9.5.6)

Notation:

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

Terminal		Carte

Send command 'INTERNAL AUTHENTICATE' (see §9.5.6) o Data=digestinfo	→ ←	The card calculates the authentication token according to the algorithm described here below C/S(M)

6.2.1.3 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	Tag	Length	Value
'A4'	L_{A4}			
		'84'	'01'	Private Key Reference
		'80'	'01'	Algorithm identifier for C/S authentication (see §6.2.2)

Note

- This template does not contain any UQB.
- Only one Key Reference is allowed

This template may be loaded within the CSE either through a 'MSE RESTORE' command (from a SSE holding it) or a 'MSE Set' command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

6.2.2 Algorithm identifier

Here is a summary of the Algorithm Identifier used in the Control reference template to identify the C/S authentication

C/S authentication method	Value
C/S RSA with DSI according to PKCS#1, parameter Digestinfo	'02'

6.2.3 Card authentication

Client / server authentication (§6.2) is usable to authenticate the card to other type of systems. It is not compulsorily dedicated to client / server authentication. Nevertheless, in such cases, it does not mandate to reset the security statuses of the user authentication data protecting the private key used for signature. No non-repudiation flag for the linked SDO.

6.3 Encryption key decipherment

Key decryption consists in using the Card as a crypto box. It is used in SSL session establishment for web services. The purpose is to send data ciphered by a remote server to a computer a person uses to access the web services in a secure manner:

[IAS ECC] application receives the symmetric key to use (for decipherment) within a cryptogram. This cryptogram was computed by the remote server with a dedicated public key. The ICC recovers the symmetric key by making use of its matching private key. The key is sent to the computer so that this latter deciphers the message sent by the remote server. This method makes use of the RSA cryptography.

This feature is described in [R16] §8

6.3.1.1 Cryptography involved

The remote server computes the cryptogram it sends to the [IAS ECC] as follows:

The Symmetric Key K is padded according to PKCS#1

M' = '00' || '02' || PS || '00' || K

Where PS is a pseudo random number (consecutive bytes different from 0) whose length is chosen so that M' length equals the key length.

The Message M' is then encrypted using the public key of the Encryption Key decryption services.

For security reasons, the Key length to encrypt should not be larger than 40% of the encryption key size. If not, the service should be rejected.

6.3.1.2 Protocol steps

The service is realized through the command 'PSO Decipher' (see §9.6.4)

Notation:

- M: Cryptogram received from the remote server
- K: Symmetric Key deciphered

IFD		ICC
Send the cryptogram received from the remote server. Use of the command 'PSO Decipher' (see §9.6.4) Data = PI || Cryptogram (for RSA scheme) PI is a byte indicating the padding used for the cryptogram computation. It is set to '81' (for the RSA scheme).	→ ←	The card deciphers the cryptogram and returns K

6.3.1.3 Setting the cryptographic context

The service is realized by setting the relevant template in the Current security Environment (CSE). The template is a CRT CT encoded as follows:

Tag	Length	Tag	Length	Value
'B8'	'06'			
		84	'01'	Private Key Reference
		80	'01'	Algorithm identifier for Encryption key decipherment (see §6.3.2)

Note

- This template does not contain any UQB.
- Only one Key Reference is allowed

This template may be loaded within the CSE either through a 'MSE RESTORE' command (from a SSE holding it) or a 'MSE Set' command (explicitly loads it in the CSE). For more details about the behavior of this commands, see §9.9.2.

6.3.2 Algorithm identifier

Here is a summary of the Algorithm Identifier used in the Control reference template to identify the Encryption key decipherment

Encryption key decipherment method	Value
RSA with DSI according to PKCS#1 v2.1 3EME PKCS#1 V1.5	'1A'

6.4 Asymmetric Key generation

Asymmetric Key generation may be used when a new asymmetric key (pair) is needed, e.g. for authentication, digital signature or encryption key decipherment. This is a very useful feature.

6.4.1 Particular issue

The asymmetric key generation leads the **[IAS ECC]** to generate an asymmetric key pair. Prior to any use, the public key should be certified by an entitled authority, called CGA (certificate generation authority) : this authority is the issuing authority of the service for whom the key pair is generated (signature, authentication,..). The CGA certifies the key pair by issuing a certificate for the **[IAS ECC]** (whatever its format is). This certificate should be stored on the card and declared in the ADF CIA matching the application.

6.4.2 Prerequisite

The asymmetric key generation can only be used to generate an asymmetric key pair, gathering both a private and public portion.

This action is performed on existing keys that shall be created (private and public portion): The following conditions shall be fulfilled prior to any key generation:

- A key pair shall be available: a set of two SDOs with the same Identifier byte, of the same kind (e;g. RSA), one being the private portion, and the other the public portion (see §XXX) shall exist.
- Both SDOs shall be located within the same DF
- Both SDOs shall be created, i.e their DOCP shall be fully filled
- Both the public and private portion shall have the algorithm identifier field of their body filled
- Both SDOs shall have the same key size
- The access conditions for key generation shall be fulfilled for both SDOs (private and public portion)
- For RSA key pair generation, if the public exponent of the key pair is empty, the generation is performed with the public exponent set to '010001'. Otherwise, the value set for the public exponent is taken into consideration for the key pair generation

The service is realized throught the command 'Generate Asymmetric Key Pair'

IFD		ICC
Send the command for asymmetric key pair generation Incoming data for asymmetric key pair generation (see below)	→ ←	The **[IAS ECC]** generate an asymmetric key pair

The key pair to generate is identified by an extended header list built as follows :

Tag	Length	Value
'70'	'03'	'BFXXYY' Private portion with • Type indicated by XX • Identifier YY

- The type of the key pair to generate is indicated by the byte XX, which is the class of the SDO of the private portion.
- The identifier of the key pair to generate is indicated by YY. It is the identifier of both portions (private & public).

Note:

- Once the generation is successfully realized, the public portion may be retrieved by reading the SDO body of the public portion of the key pair.

Upon successful key pair generation, the usage counter of the private and public portions are reset to their maximum values – content of tag '9C' (if they have one).

7 OTHER FEATURES

7.1 The Secure Messaging layer

7.1.1 Overview

When the ICC is used within an untrusted environment, the secure Messaging layer aims at ensuring that a communication between the ICC and the IFD is protected in integrity (at each moment both parties are identified) and in confidentiality (protection against eavesdropping).

Once the secure channel is established (through a device authentication – see §5.2). It remains open as long as the incoming commands are correctly wrapped with the secure messaging session keys and SSC.

SM session break includes the following:

- SM session keys are invalidated.
- SSC is invalidated
- The SM status is reset into the card security state

The following is mandatory for the IAS_ECC2 application:

- A SM session always includes data confidentiality and integrity (SMI + SMC).

7.1.2 Setting the cryptographic context

No specific operation is required for setting the cryptographic context (ie. Neither MSE SET CT nor MSE SET CCT is required) since integrity and confidentiality is mandatory.

7.1.3 Protection of an asset with secure messaging

The Access conditions assets protected by the secure messaging for integrity and confidentiality is encoded using a CRT CCT and CRT CT set in the Static Security Environment pointed out in the SCB containing the conditions “Secure messaging” (see §4.2.3) encoded as follows:

The template CCT enforcing integrity on incoming and outgoing commands:

Tag	Length	Tag	L	Value
'B4'	L_{B4}			
		'80'	'01'	Identifier of the device authentication algorithm that initiated the secure channel
		'83' or ‘84’	'01'	Key Reference used by the ICC to initiate the secure channel
		'95'	'01'	Usage Qualifier byte for Secure Messaging (0x30)

The template CT enforcing confidentiality on incoming and outgoing commands:

Tag	Length	Tag	L	Value
'B8'	L_{B8}			
		'80'	'01'	Identifier of the device authentication algorithm that initiated the secure channel
		'83' or ‘84’	'01'	Key Reference used by the ICC to initiate the secure channel
		'95'	'01'	Usage Qualifier byte for Secure Messaging (0x30)

Note

- These templates do only accept one field of each type, i.e. UQB, Key Reference, and identifier of the device authentication that initiated the secure channel.
- The enforcement of the Access condition integrity and confidentiality requires to store both templates in the SSE
- If the field Key Reference AND identifier of the device authentication that initiated the secure channel are both set to ‘00’, it means the access condition only requires a secure channel, regardless the device authentication type and key identifier that initiated it. In such cases, the Key reference shall be pointed out using the tag ‘83’.
- If the field Key Reference AND identifier of the device authentication that initiated the secure channel are both different from ‘00’. This means the access condition requires a secure channel, initiated with the device authentication specified by the field “identifier” (encoded as indicated in §5.2.4) and the key id indicated in “Key Reference” (In case of a symmetric device authentication, the symmetric key set shall be pointed out with tag ‘83’, while in case of the device authentication with privacy protection, the ICC authentication private key shall be pointed out with tag ‘84’).
- The UQB is mandatory in this template.
- As these access conditions imposes on the IFD to provide secure messaging with both integrity and confidentiality (provided there are incoming data), it guarantees the ICC will return its response with the same security level.

7.1.4 Secure messaging - Session keys computation

The secure messaging between the card and the IFD requires two 16 bytes long session keys (128 bits-long).

SK$_{\textbf{ENC}}$ and **SK**$_{\textbf{MAC}}$ are calculated from K.ICC and K.IFD provided by each part during the device authentication (see §5.2) as follows:

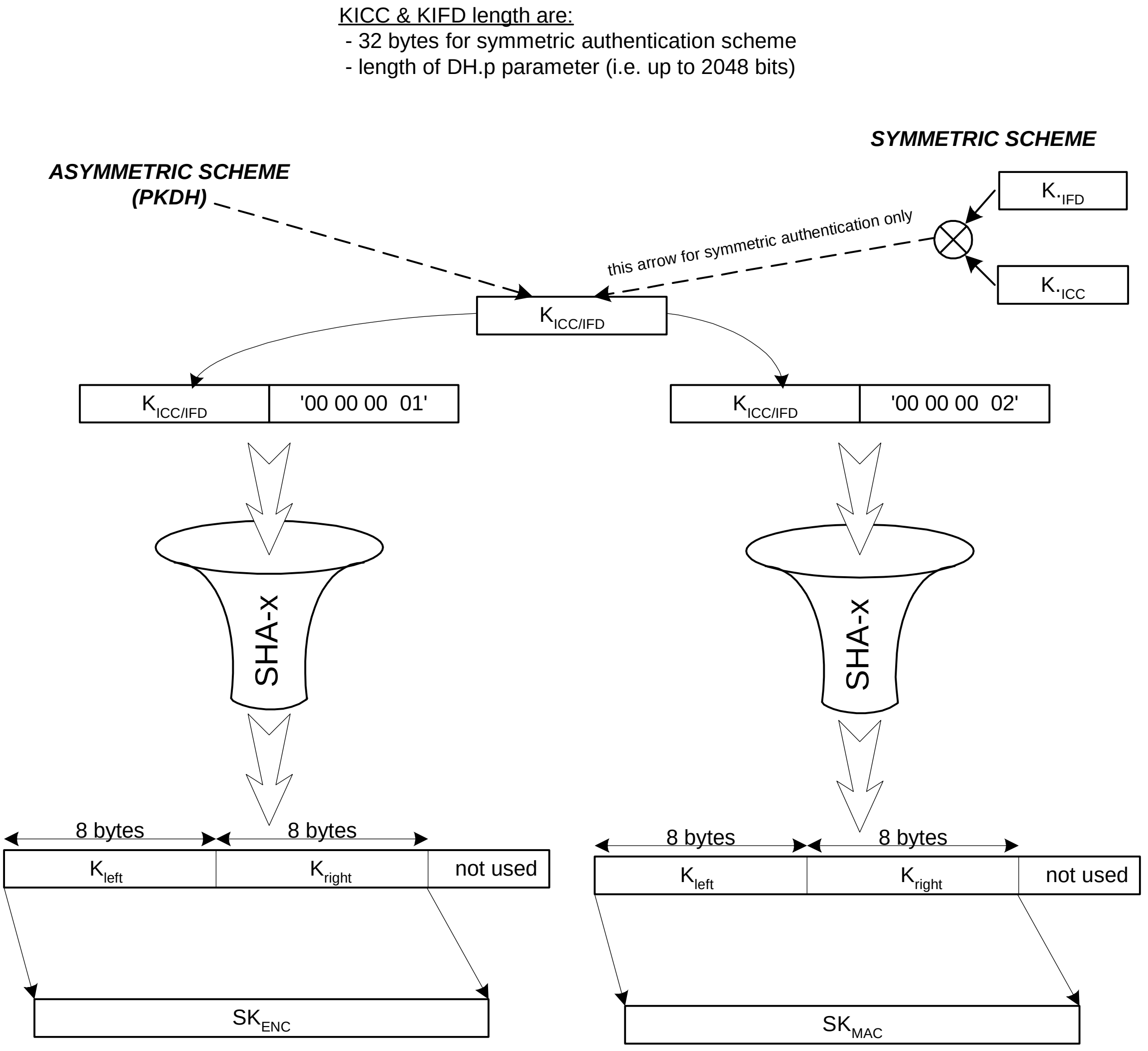


Figure 9 - SKENC and SKMAC calculation from K.ICC and K.IFD values

This method for session key generation is the one described in ANSI X9.31 with the SHA-1 or SHA-256 as hash algorithm.

7.1.5 Send Sequence Counter (SSC)

The SSC is an 8-byte number used for initial chaining vector (ICV) calculation before MAC computations. The SSC is incremented by 1 before each MAC computation. The SSC wraps around, i.e. when the value 'FF FF FF FF FF FF FF FF' is reached, the next SSC value is '00 00 00 00 00 00 00 00'.

The authentication phase ends with the sharing of RND.ICC and RND.IFD between the card and the terminal.

RND.ICC and RND.IFD are both are 8 byte-long random numbers. They are used to calculate the initial SSC value by concatenating the 4 least significant bytes of RND.ICC and RND.IFD:

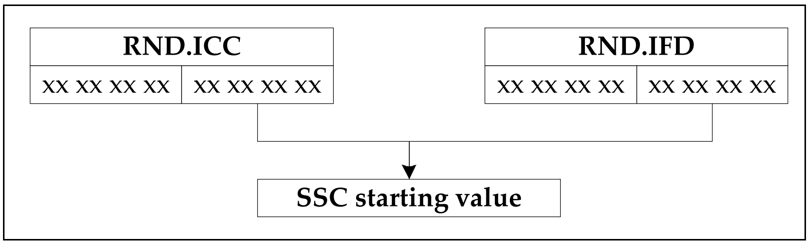


Figure 10 - SSC starting value calculation

7.1.6 SM Data Objects

The following table lists the DOs used by the IAS application

Tag	Meaning	Name
'97'	Le (to protect using CC)	Tle
'99'	Status-Info (to protect using CC)	Tsw
'8E'	Cryptographic Checksum	Tcc
'87'	PI || Cryptogram (to protect using CC) – For even INS code	Tcg
'85'	Cryptogram (to protect using CC) – For odd INS code	Tbr

7.1.7 SM checks

The following tests are performed by the card when receiving an APDU protected by secure messaging:

- If no session keys are available (following a device authentication), the card returns '6985' or '6982'.
- For D.O. '87' and '85', the padding of the cryptogram shall be according to ISO 9797-2. If not, the card returns '6988' and the secure session is closed.
- For D.O. '87', the padding indicator shall be equal to '01'. If not, the card returns '6988' and the secure session is closed.

- The structure of the D.O.s is checked (D.O.s are correctly BER encoded, length is correct). If the structure is incorrect the card returns ‘6988’ and the secure session is closed.
- If a SM DO is missing (such as the MAC D.O.), the card returns ‘6987’ or ‘6982’ and the secure session is closed.
- If the Checksum nested in the MAC D.O. is incorrect, the card returns ‘6988’ and the secure session is closed.

When an error occurs in the secure messaging layer, the Status word is only returned in plain text, as SM session is aborted.

7.1.8 Commands and Responses under SM

A command may be sent using secure messaging even if it is not required by the card. A command sent protected with SM while no SM session is opened will return a SW.

The CC is first checked, and then the data is decrypted.

Bits b4 and b3 of the CLA byte shall be set to '1' (i.e. CLA ≡ 'xC'). It means the command header is integrated into the CC calculation.

7.1.9 Secure messaging – Command APDU protection

APDU commands are structured as follows:

Header	Lc	Data	Le
Class, Ins, P1, P2	Lc	Incoming Data	Le

An IFD sending an APDU protected by secure messaging performs the following steps:

1. Encrypt the incoming data field as shown Figure 11[15].
2. Calculate the cryptographic checksum as shown Figure 12[15].
3. Construct the final command APDU string for sending to the smart card as shown Figure 13.

When no data is sent (Lc = 0), the first step is avoided, and any further reference to EDFB (step 2 and 3) shall not be taken into account (consider the block does not exist).

Note :

When the ICC receives an incoming command, the Secure messaging layer only unwraps the command and performs the basic checks as stated in §7.1.7. In particular, it does not preclude any presence or absence of data SM DO (‘85’ or ‘87’) nor Le SM DO (‘97’). The interpretation of whether or not data or the Le is missing is up to the application layer, using the APDU unwrapped. Thus, it does not lead to a SM break but to an applicative error (if requested by the applicative layer)

[15] - Padding with '80 …' is systematic even if the number data byte is modulo 8.

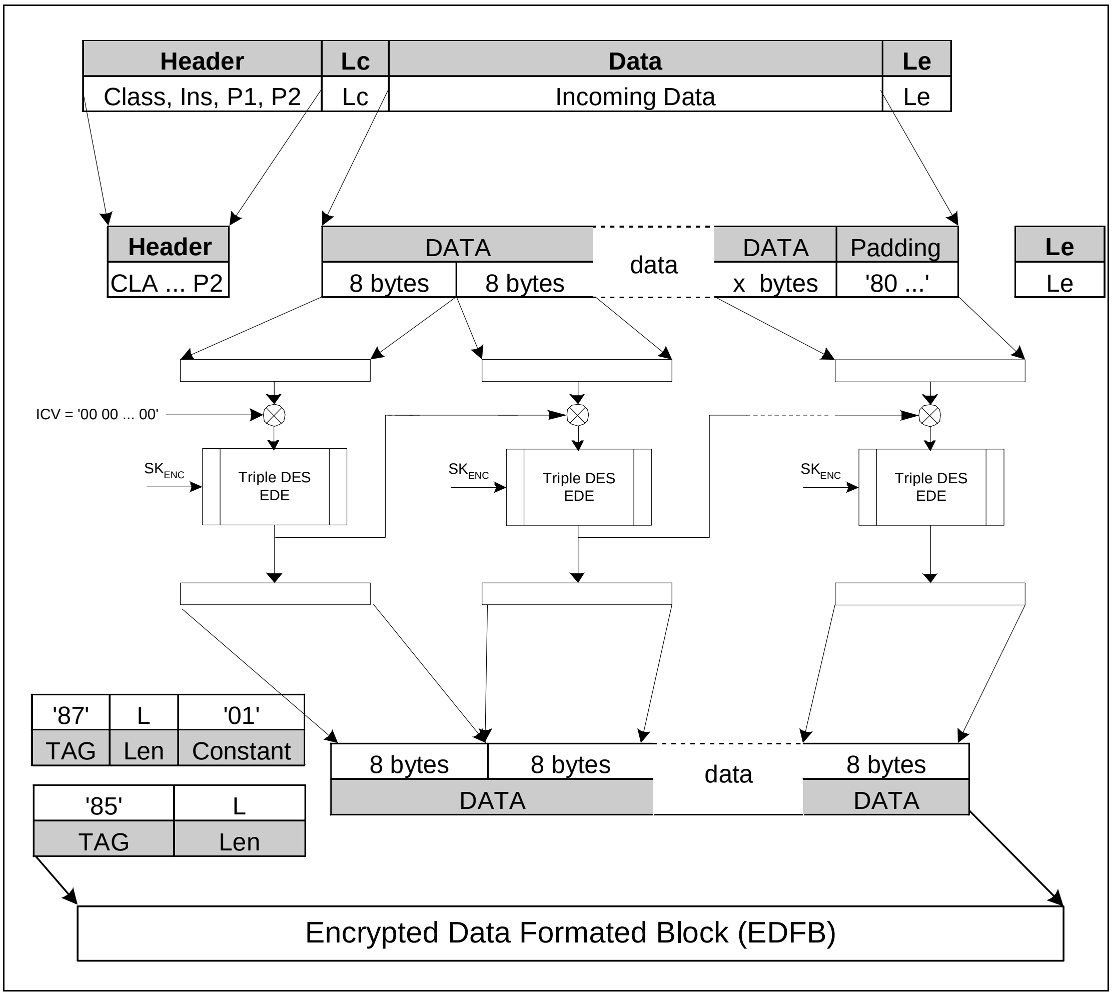


Figure 11 - APDU command data encryption

If incoming data has an odd INS code, tag ‘85’ is used to encapsulate encrypted data in EDFB, while tag ‘87’ with padding indicator ‘01’ are used if incoming command has an odd INS code.

For instance, sending a Putdata APDU command under secure messaging will mandate the use of the tag ‘85’, since INS code is odd.

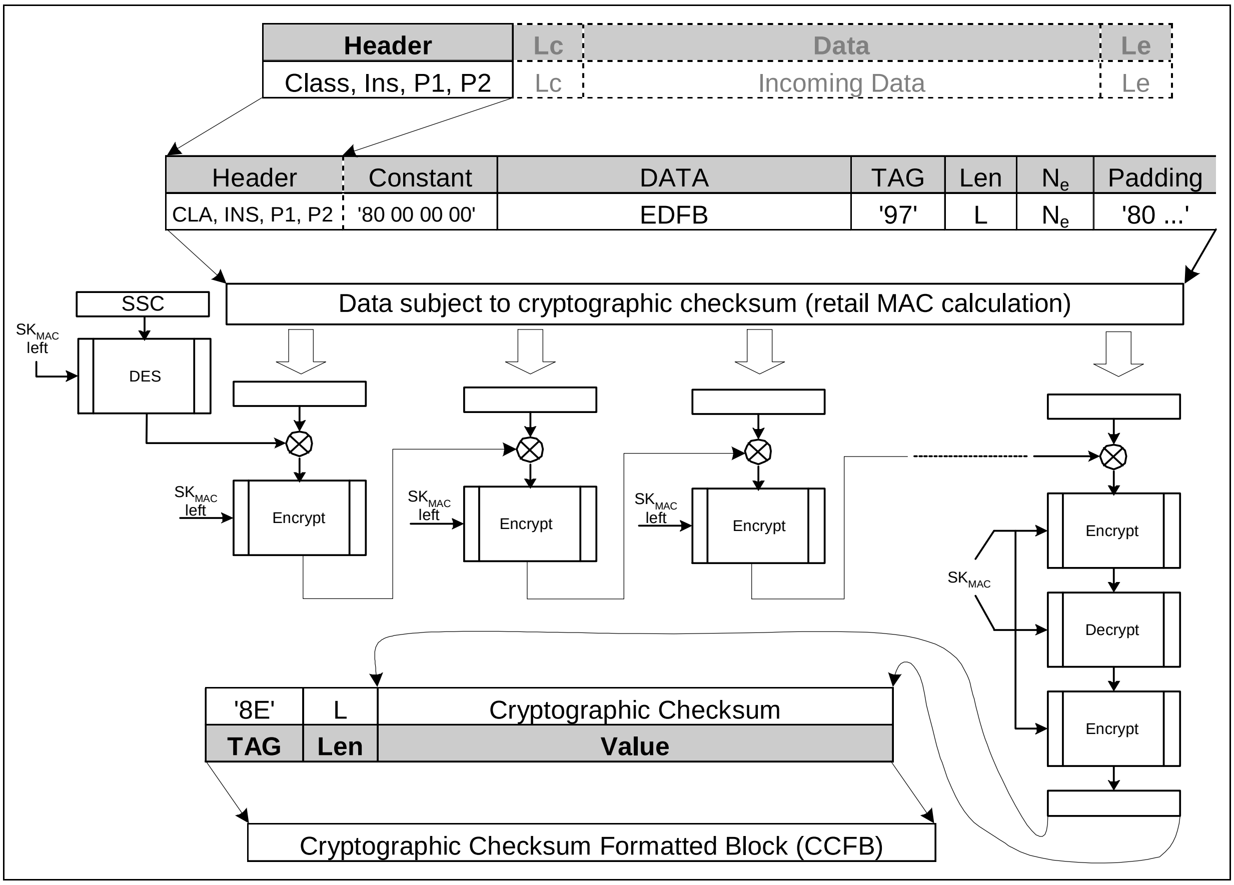


Figure 12 - APDU command cryptographic checksum calculation

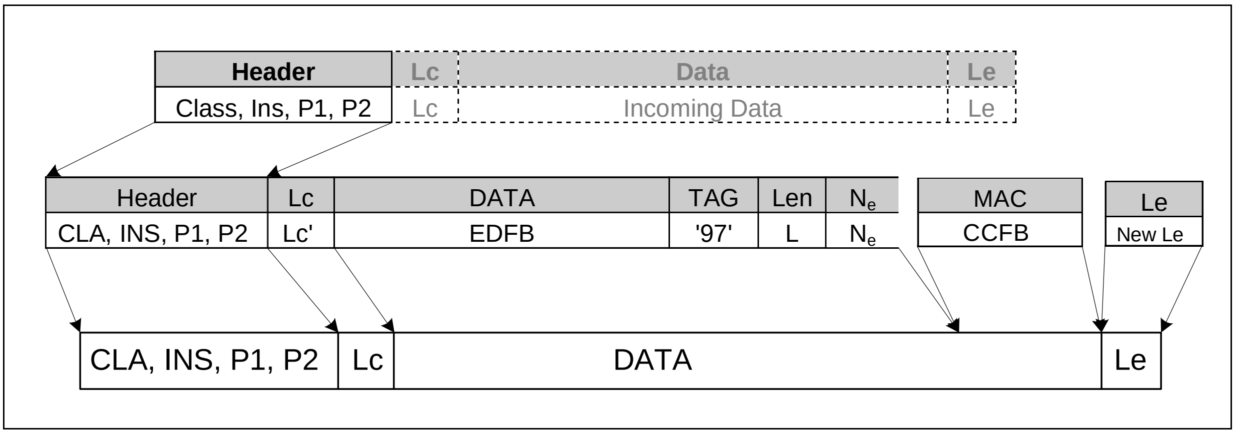


Figure 13 - Final APDU command construction

7.1.10 Secure messaging – Response APDU protection

Each APDU response is structured as follows:

Data	Status word
Outgoing Data (n bytes)	SW12 (bytes)

When returning data to the IFD, the card the following steps:

1. Encrypt the outgoing data field as shown Figure 14[15].
2. Calculate the outgoing cryptographic checksum as shown Figure 15[15].
3. Construct the final response APDU to return to the IFD as shown Figure 16.

If the command does not send any data to the card, then the first step is avoided, and any reference to EDFB during step 2 and 3 shall not be taken into account (consider the block does not exist).

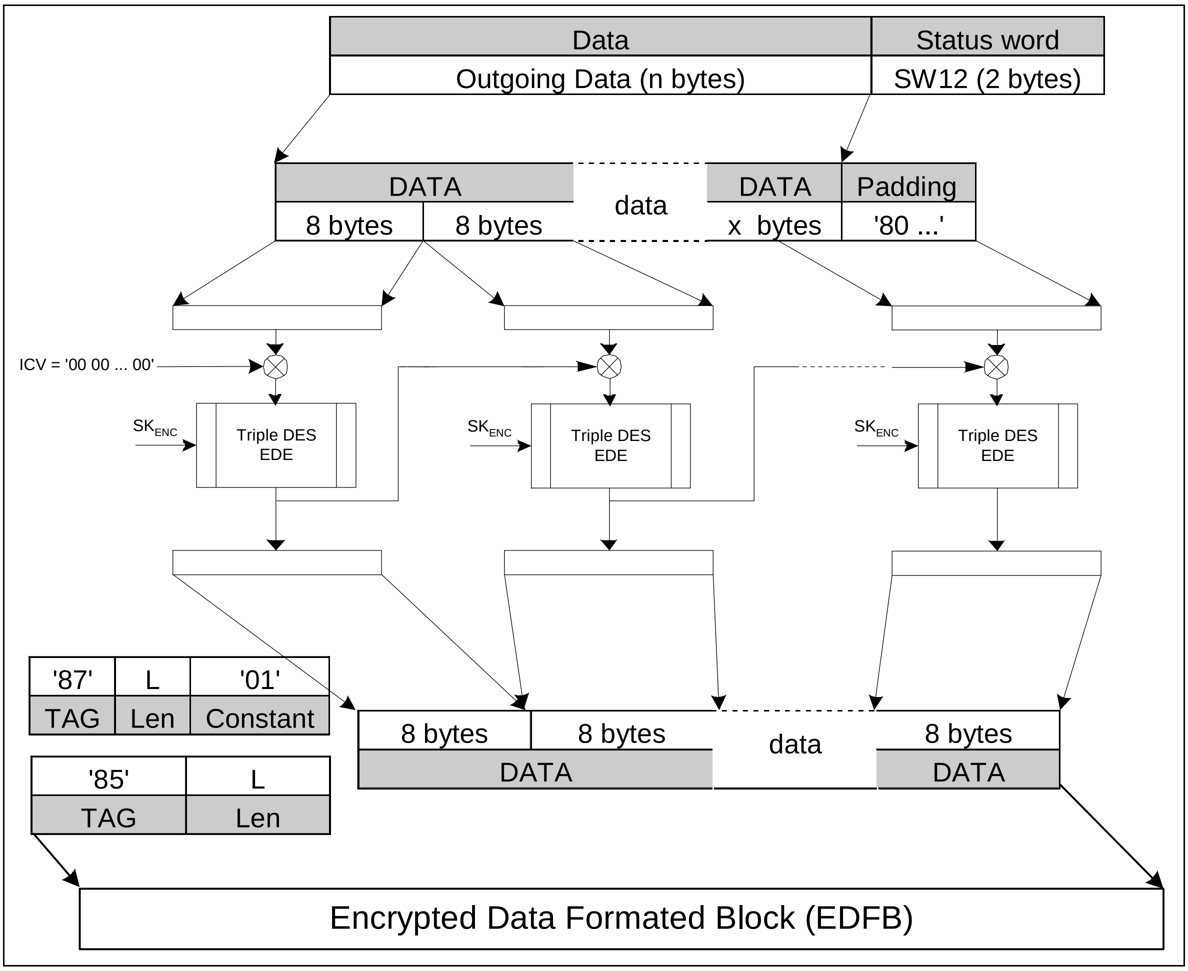


Figure 14 - APDU Response with data encryption

If incoming data has an odd INS code, tag '85' is used to encapsulate encrypted data in EDFB, while tag '87' with padding indicator '01' are used if incoming command has an even INS code.

For instance, requesting a GET DATA SDO APDU command under secure messaging will mandate the use of the tag ‘85’, since INS code is odd.

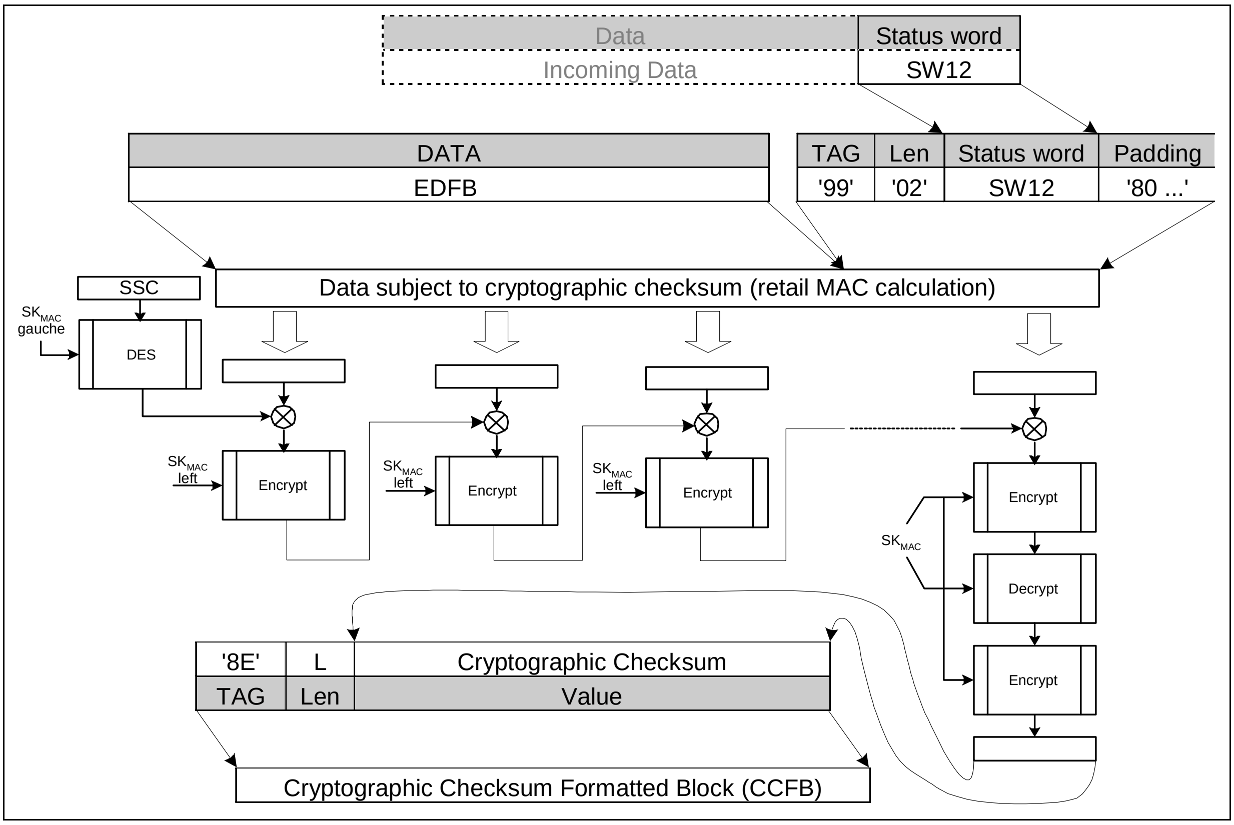


Figure 15 - APDU Response cryptographic checksum calculation

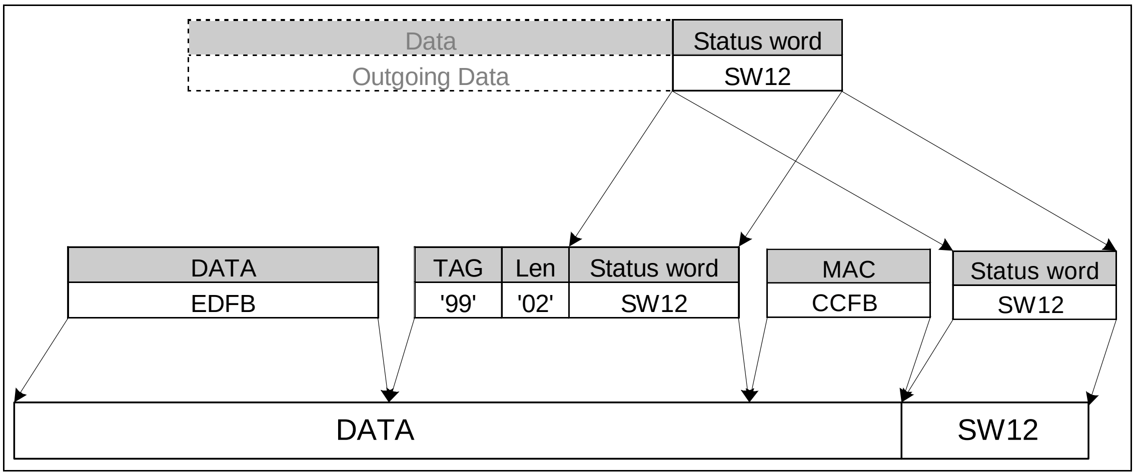


Figure 16 - Final APDU Response construction

7.1.11 SM computation

The CC is 8 bytes long and is built according to [R2] as shown (basic mechanism builds a MAC according to [R9] using DES).

7.1.12 Cryptographic checksum calculation

According to the command case, DO are not all included in the CC calculation.

- CH → Command Header (CLA INS P1 P2 where the CLA byte is the SM one).
- T_{XX} → TAG 'xx"
- Lxx → Length for tag 'xx'
- PB → Padding Bytes ('80 00 … 00')
- CG → Cryptogram. Depending on the format of the data field, TAG '85' should be used for BER TLV data and TAG '87" should be used for non BER TLV data with padding indicator '01'.

DO included in the CC calculation:

- Case 1 command: **CH || '80 00 00 00'**
- Case 2 command: **CH || '80 00 00 00' || T_{LE} || L_{LE} || L_E || PB**
- Case 3 command: **CH || '80 00 00 00' || T_{CG} || L_{CG} || CG || PB**
- Case 4 command: **CH || '80 00 00 00' || T_{CG} || L_{CG} || CG || T_{LE} || L_{LE} || L_E || PB** (T_{LE} || L_{LE} || L_E can be absent)

For SM responses, data covered by CC are:

- For case 1 or 3 responses: **T_{SW} || L_{SW} || SW || PB**
- For case 2 or 4 responses: **T_{CG} || L_{CG} || CG || T_{SW} || L_{SW} || SW || PB**

The sent sequence counter SSC is increased by 1 each time before a CC is calculated.

7.1.13 SM session establishment events

Secure messaging session establishment-beginning event varies according to the device authentication performed:

Mutual authentication	Event beginning the SM session
Device authentication with privacy protection	GET DATA for K.ICC retrieval.
Symmetric mutual authentication	MUTUAL AUTHENTICATE command. See §9.5.4

7.1.14 SM session breaking events

The SM session is broken when:

- The card is reset.
- A new device authentication sequence starts.
- One of SM checks fails (See 7.1.7).
- A command is send in clear (except for SELECT BY NAME).

7.1.15 Secure channel support

When opening a secure channel, integrity and confidentiality keys are always calculated. These keys are then always used since all exchanges between ICC and IFD have to be protected with confidentiality and Integrity, whatever the security conditions of the SDO/File are.

7.1.15.1 Secure session opening sequence using symmetric scheme

Precondition: it is assumed that the SN.ICC has been retrieved by the IFD using a READ BINARY command, if the IFD does not already know it.

Stage		COMMAND SENT
A	1	MSE SET CRT AT for the MUTUAL AUTHENTICATE
	2	GET CHALLENGE
B	3	MUTUAL AUTHENTICATE
C	4	*Secure channel established is Integrity and confidentiality until the end of the session*

7.1.15.2 Secure session opening sequence using PKDH asymmetric scheme

Preconditions: it is assumed that:

- The DH parameters have been retrieved by the IFD using a READ BINARY command, if the IFD does not already know it. It shall be performed before A1
- CA's public key of the ICC has been retrieved by the IFD using a READ BINARY command, if the IFD does not already know it. If the IFD does not know the PuK.ICC.AUT or its certificates, it should retrieve it under Secure messaging between C8 and D9.

Stage		COMMAND SENT
A	1	MSE SET KAT (K.IFD is transmitted to the ICC)
	2	GET DATA KICC
	3	*Secure channel established is Integrity and confidentiality until the end of the session*
B	4	MSE SET CHR – CRT DST – Clue for the root CA key to use.
	5	PSO VERIFY CERTIFICATE
C	6	MSE SET CHR – CRT AT – Clue for selectiing the next public key to use. CHR of the public key extracted during the previous successful pso verify certificate
	7	GET CHALLENGE
	8	EXTERNAL AUTHENTICATE PKDH
D	9	MSE SET – CRT AT – Selection of the private key for internal authenticate
	10	INTERNAL AUTHENTICATE PKDH

7.2 PKI Management: Card Verifiable Certificates

7.2.1 About ISO 9796-2 certificates

A CVC is BER-TLV coded data container issued by a trusted authority (i.e. RootCA). It provides information on a RSA public key by demonstrating it is trustable, and it possibly validates some role attributed to the certificate owner that may grant him some accesses to services or file in the card.

ISO 9796-2 certificate is performed in two steps:

1. The card verifies the CVC seal using the applicable RSA public key (i.e. RootCA key for the first step).
2. The card extracts each field constituting the certificate and checks their values. It also extracts the public key enclosed in the CVC to replace the one used during step 1.This key can be used to continue the certificate chain verification, or to perform an external authentication.

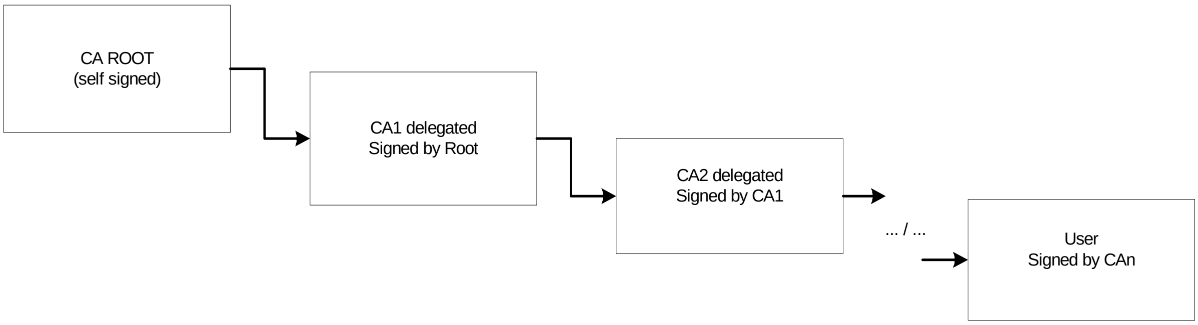
7.2.2 Certificate trusted path

A user certificate is always signed by an authority (Certification Authority CA). This CA also has a certificate that cans itself also be signed by another CA, and so on.

The top level CA has a certificate signed with its own key. Its public key shall be loaded on every smart card during personalization.

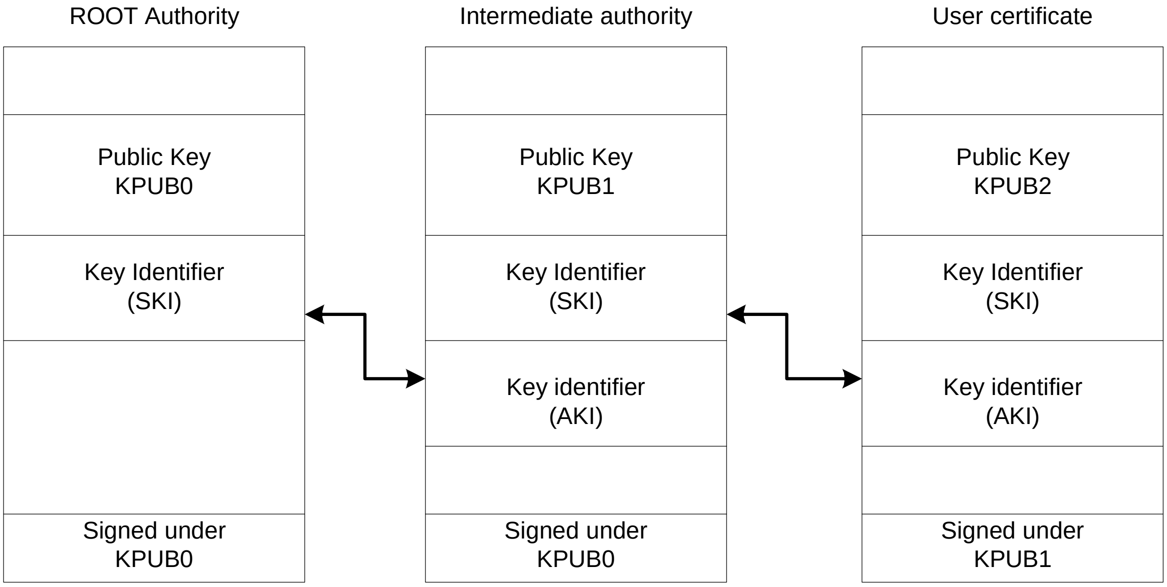
When an IFD wants a user certificate to be recognized by the card, it shall check the CA the card already knows (by browsing the CIA application). If the user certificate was not signed by a CA already known by the card, it shall present all required certificate to the card from the first one (that was signed by the CA known by the card) down to the user certificate the IFD want the card to recognize.

Doing this on establishing the certificate trusted path.



Links between certificates in a certificate path are made through identifiers contained in the certificates:

- Authority Key Identifier (AKI): allows finding the certificate of the authority that has signed the certificate.
- Subject Key Identifier (SKI): is the identifier of the public key that is contained in the certificate.



7.2.3 Context Use

The CVC authentications are used on the following occasions:

- IFD/ICC mutual authentication PK-DH scheme. CVC are used to transport the public key.
- Asymmetric role authentication. CVC are used to transport the public key as described §5.3.2.
- Certificate trusted path establishment:
  By default, the card knows the root CA public key that was written during the personalization step. During the certificate trusted path establishment, this public key is used to initiate the verification process.

7.2.4 Certificate format provided to the card

A certificate presented to the card is structured as follow (non self-descriptive):

Tag	L	Value		
'7F21'	Var.			
		Tag	**L**	**Value**
		'5F37'	Var.	CERTSIGN as described § 7.2.5.7
		'5F38'	Var.	PKREM Remainder as described §7.2.5.8 (if required, depends on the key length)
		'42'	'08'	CAR (Mandatory to retrieve the Authority Key Identifier which has signed the certificate)

The certificate is first verified, and then rebuilt in a byte string matching the value fields of the extended header list of the CPI field.

7.2.5 Format

CVC are based on [R4] and [R9], and are fully described in [R15].

These certificates are non self descriptive certificates.

The certificate is encoded as follows:

Note: [IAS ECC] will only support the non self-descriptive format of CVC.

Certificate Object '7F21'

Tag	L	Value
'7F21'	Var	Concatenation of certificate data elements

Mandatory certificate data elements shall be ordered that way :

Tag	Length	Object
'5F29'	1	Certificate Profile Identifier (CPI)
'42'	8	Certification Authority Reference (CAR)
'5F20'	12	Certificate Holder reference (CHR)
'5F4C'	Var.	Certificate holder Authorization (CHA)
'06'	Var.	Algorithm OID (ALGID)
'7F49'	Var.	Public Key of the certificate (KPUB) constructed in BER

Others fields may be present but are not mandatory in the [IAS ECC] (e.g.: support of date).

7.2.5.1 CPI

This allow to identify a profile previously stored in the card so that the card can decode the certificate without carrying the associated extended header list and accept or reject particular types of certificates. A certificate profile is composed of a complete self-descriptive extended header list

Ex:

CPI		CAR		CHR		CHA		OID		PK					
'5F29'	'01'	'42'	'08'	'5F20'	'0C'	'5F4C'	'07'	'06'	'08'	'7F49'	L_{7F49}[16]	'81'	L_{81}	'82'	L_{82}

The associated certificate is composed of the associated byte string to the extended header list.

CPI encoding is fully described in [R15]

CPI	Meaning
'00'	RFU
'01'-'6F'	CPI for non self descriptive certificates standardized in [R15]
'70'-'7F'	CPI for self descriptive certificates standardized in [R15] – N/A
'80'-'FE'	Applications specific CPIs
'FF'	RFU

[16] - In extended header list, L_{7F49} = long('81') + long('L_{81}') + long('82') + long('L_{82}') = '8181808204' = 5 bytes for 1024 bit-long key.

The [IAS ECC] applications uses the following CPI described in [R15] §14.11.1

- CPI = '03' for RCA & CA key used for certificate verification with RSA SHA-1 & SHA-256 – ISO 9796-2 in 1024 bits
- CPI = '0F' for RCA & CA key used for certificate verification with RSA SHA-1 & SHA-256 – ISO 9796-2 in 1536 bits
- CPI = '13' for RCA & CA key used for certificate verification with RSA SHA-1 & SHA-256 – ISO 9796-2 in 2048 bits
- CPI = '04' for end of chain key used for device authentication with privacy protection with RSA SHA-1 & SHA-256 – ISO 9796-2

The CPI depends on the

- The Key type (Root Key, Delegate certification authority, IFD key for device authentication, role authentication key)
- The format of the certificate, depending of the size of the Public Key, and its OID length & value

[IAS ECC] defines as well its own set of CPI. In such cases, they are assigned within the range ['80'-'FE'].

The CPI [IAS ECC] uses are built as follows:

b8	b7	b6	b5	b4	B3	B2	b1	Meaning
1								Application dependant : [IAS ECC]
	X	**X**	**X**					**RFU**
				X	**X**			**Use**
				1				Device authentication with privacy protection
					1			Asymmetric Role authentication
						X	**X**	**key size**
						0	0	1024 bits
						0	1	1536 bits
						1	0	2048 bits

7.2.5.2 CAR

This field carries the Authority Key Identifier and is used as described in the certificate path. It is structured as followed:

CA Name (5 bytes)	**Extension for key referencing (3 bytes)**

- CA Name structured as described in [R15] § 14.7.3
  - 2 ISO 3166 bytes: country (eg: France = FR).
  - 3 ASCII bytes: acronym.
- Extension for key referencing

Service Indicator (1 BCD)	Discretionary data (1 BCD)	Algorithm reference (2BCD)	Date (2BCD)

- Service Indicator: specifying the use of the key (Values indicated in [R15] §14.7.1).
- Discretionary data: distinguishing two keys issued the same year by the same CA (e.g.: following a renewal of the key).
- Algorithm reference: identifying the algorithm to use with the key if the CA supports several.
- Date: two last digits of the year the key was generated.

7.2.5.3 CHR

CHR identifies the public key stored in the certificate Subject Key Identifier (SKI).

It may be used as the key reference and has two coding manners:

- For a CA (delegated or not) certificate:

'00 00 00 00'	CAR (§7.2.5.2)

- For an end entity (i.e. the latest certificate of the verification chain):

n × '00' padding bytes to have CHR length = 12 bytes (0 ≤ n ≤ 4)	Serial number of the identity (8 to 12 bytes)

The IFD may select a public Key using the CHR indicated in the CVC imported on the ICC. In that case, this CHR is nested in the relevant CRT using the MSE SET command. It may arise in the following cases:

1. Public key selection for the External Authentication during the Device authentication with privacy protection or the asymmetric role authentication
2. Public key selection for certificate verification (provided it is not the first key initiating the certificate chain)

7.2.5.4 CHA

The CHA field indicates the privileges that may be granted following an authentication based on the public key extracted from the certificate. It contains the role validated by the certificate:

Prefix	Value
6 MSB OF THE AID OF THE APPLICATION	Role

From the 7th to the last byte, CHA contains the role ID coding the operations that are authorized using the public key extracted from the certificate.

7.2.5.4.1 Standard Role encoding for certificate verification & IFD authentication

Refer to [R15] §14.9.5.1

In the case of certificate verification & IFD authentication (within the device authentication with privacy protection), the Role ID is used to identify the entity that issued the certificate. It may be either :

- Root certification authority
- (delegate) certification Authority
- End entity that will authenticate itself (for device authentication with privacy protection)

The Role Id is encoded over one byte as follows:

Certificates designed to be transmitted to the card (CVC):

Role ID	Meaning	Notes
'21' '22'	Role of IFD Root Certification authority and (delegated) certification authority (for intermediate certificate verification)	The role is checked when verifying a certificate in the IFD certificate chain.
'01'	Role of IFD Key for device authentication.	Role checked when calling an applicable external authentication.

Certificates designed to be transmitted to the terminal (i.e. not relevant for the IAS card):

Role ID	Meaning	Notes
'31' '32'	Role of ICC Root Certification authority and (delegated) certification authority (for intermediate certificate verification)	The role is checked when verifying a certificate in the ICC certificate chain.
'00'	Role of ICC Key for device authentication.	The role is checked when starting an authentication procedure of the ICC.

Note: the second array is only informative. It is relevant for the certificates stored in the ICC, dedicated to the IFD, if they are CVC ones. Usually, the IT system would rather require the use of X509 certificates (PRIS usage for instance)

7.2.5.4.2 [IAS ECC] role encoding for external role authentication

This section is only relevant with the Public key used for external role authentication.

In that case, its Role ID is encoded in a slightly different way. It is encoded over several bytes and describes the privileges associated to this key, imported on the ICC, which will be used to identify a third party (IFD, server, administrator of a service). Upon successful validation of the role ID, the [IAS ECC] card, grants the relevant privileges

The ICC does not know a priori, the length of the Role ID. That length is not linked to the CPI. The Role ID is self-descriptive, in the way it indicates the length of the privilege.

The First byte of the Role ID is encoded as follows:

b8	b7	b6	b5	b4	B3	B2	b1	Signification
1	0	0	0	0	0	0	0	Role identifier coded as number in the subsequent byte as [R15]Table 14-15

The [IAS ECC] application supports the following encoding for the subsequent byte.

The Second byte of the Role ID is encoded as follows:

b8	b7	b6	b5	b4	B3	B2	B1	Signification
1	0	0	0	x	x	x	x	Length of subsequent bytes encoding the proprietary privilege is encoded over 4 bits.
0	x	x	x	x	x	x	x	Privilege encoded over 7 bits

Depending on the encoding on the second bytes of the role ID, the CHA may be encoded as follows:

1-The privilege is encoded over seven bits:

Privilege = ‘0xxxxxxx’

CHA		
Prefix	Role ID	
6 MSB OF THE AID OF THE APPLICATION	‘80’	Privileges

2-The privilege is encoded over several bytes:

The privilege length L_P shall be up to 15 bytes (included)

CHA			
Prefix	Role ID		
6 MSB OF THE AID OF THE APPLICATION	‘80’	‘8||L_p’	Privileges

When trying to use/access an asset protected by a given Privilege (one or several), the Privilege associated to the ephemeral public key extracted from the latest CVC and used to perform an external authentication is compared with the one (s) defined in the template managing the access conditions.

The method used to perform the comparison between the reference Privilege(s) and the validated privilege set in the CVC is set in D.O. ‘8A’ of the template defining the Access condition.

The comparison can only be performed between Privileges having the same length. If there are no Privileges with the length of the validated one, the access right is not granted.

Assuming **P_C** is the validated privilege and **P_R** the reference one to be considered for comparison (the one coded with the tag ‘8A’ in the security policy), the right is validated if:

B8	b7	b6	b5	b4	b3	B2	B1	Signification
1	0	0	0	0	0	0	0	All bits in P_C and equal to the bits in P_R
0	1	0	0	0	0	0	0	At least one bit in P_C is also set in P_R
0	0	1	0	0	0	0	0	RFU
0	0	0	1	0	0	0	0	
0	0	0	0	1	0	0	0	
0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	1	

Table 23 – Encoding of the privilege comparison

DO with tag '8A' has a meaning only in CRT that are used for access control verification (i.e. in SSE in EEPROM).

<u>The length of the privilege field is X bytes encoded on the second byte of the Role ID:</u>

The '8X' byte is followed by X bytes. Consequently, the CHA length is (8 + x) bytes. This specification does not define a maximum length for the CHA.

<u>Remark:</u> For IAS interoperability testing, verification will be performed using 1, 2 and 3 bytes long role (i.e. 8, 9 & 10 byte-long CHA respectively).

7.2.5.5 KPUB

KPUB has BER TLV structure describing the certificate public key and encoded as follows:

Tag	L			
'7F49'	Var.	T	L	V
		'81'	Var.	Public key modulus n (MSB … LSB)
		'82'	Var.	Public Key exponent e (MSB … LSB)

7.2.5.6 Object Identifier (OID)

This field describes the associated algorithm to the key nested within the certificates, authorized values are:

- Certificate verification:
  - OID = '2B 24 03 04 02 02 0X' for the certificate verification using RSA 9796-2 with hashing algorithm according to x (see below).
  - The value is present in certificates containing CA and RCA keys only.
- Device authentication:
  - OID = '2A 81 22 F4 2A 02 04 01 0X' for the device authentication using RSA 9796-2 with hashing algorithm according to x (see below).
  - The value is present in certificates containing an end entity public Key for device authentication.

- Role authentication:
  - OID = ‘2A 81 7A 01 81 2F 02 01 0X' for the certificate verification using RSA 9796-2 with hashing algorithm according to x (see below).
  - . The value is present in certificates containing an end entity public key for role authentication.

X is chosen as follows:

- If SHA-1 is chosen, ‘x’=’1’
- If SHA-256 is chosen, ‘x’=’4’

7.2.5.7 CERTSIGN

CERTSIGN is the certificate's signature; it calculated according to ISO9796-2 standard using SHA-1 or SHA-256 with message recovery.

Depending on the hashing algorithm, the lengths of some fields in CERTSIGN vary. In the following schemes Hlength depends on the hashing algorithm:

- SHA-1 → Hlength = 22 (i.e. 20 + 2)
- SHA-256 → Hlength = 34 (i.e. 32 + 2)

The signature applies on the concatenated value field of the DO described in the CPI.

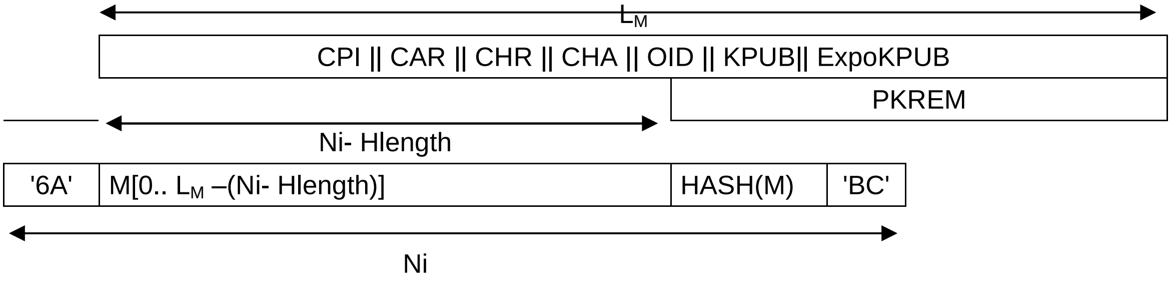
EX: BYTE STRING OF THE CERTIFICATE

M = CPI || CAR || CHR || CHA || OID || KPUB|| ExpoKPUB

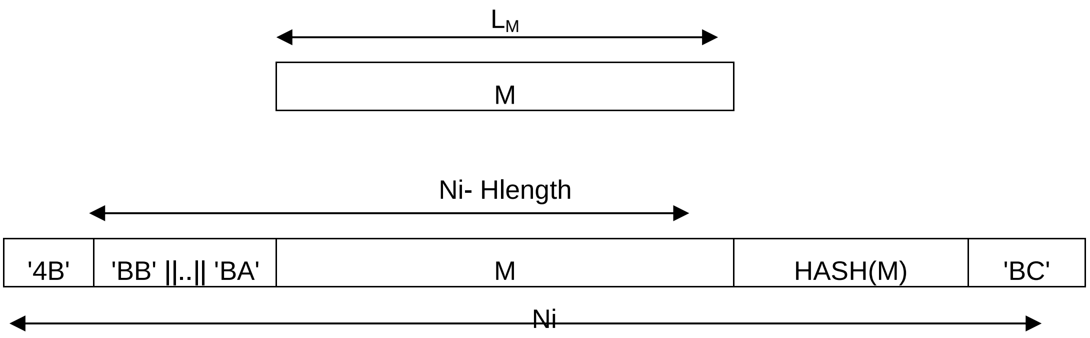
- Ni is the CA key length. According to ISO 9796-2 with SHA-1, the signature is compute on a Ni-long M0 byte string.

<u>Signature calculation is as follows:</u>

- Case 1: If L_M > Ni- Hlength

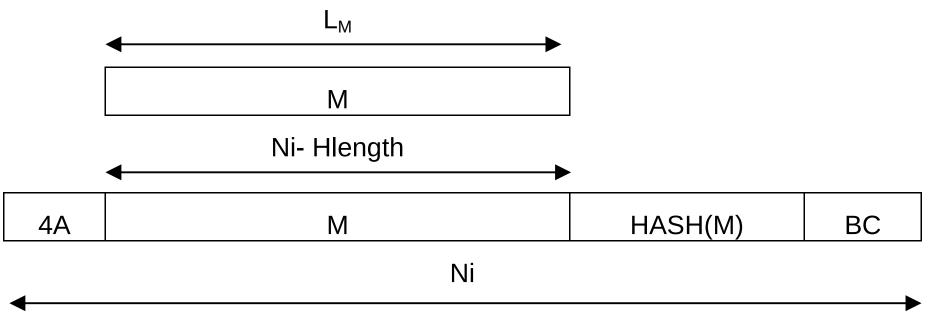

L_M
CPI || CAR || CHR || CHA || OID || KPUB|| ExpoKPUB
PKREM
Ni- Hlength
'6A'
M[0.. L_M –(Ni- Hlength)]
HASH(M)
'BC'
Ni

- Case 2: If L_M < Ni- Hlength



Note: In that case there is no remainder. The message is padded with 'BB' bytes to Ni-2 bytes.

- Case 3: If $L_{M\,=}$ Ni- Hlength



Note: No remainder for case 3.

7.2.5.8 PKREM

PKREM is the certificate remainder (might be absent) that is included in for the hash calculation, but excluded from signature calculation (see §7.2.5.7).

7.2.6 Certificate verification

Before using a certificate, the card verifies it.

A user certificate is always signed by an authority (Certification Authority - CA). This CA owns a certificate that can also be signed by another CA, and so on.

The top level CA (Root CA – RCA) owns a certificate signed with its own private key. The public key associated shall be loaded on every smart card during personalization.

When an IFD wants a user certificate to be certified by the card, it shall verified among the CA already known by the card (i.e; a trusted CA, by browsing the CIA application). If the user certificate was not signed by a trusted CA it shall present all the required certificates to the card, from the first one (signed by a trusted CA) down to the user certificate the IFD want the card to certify.

On the final stage, the End entity certificate owns a role and an algorithm associated to an authentication procedure that can be engaged after this verification steps.

The certificate verification enables the ICC to make sure the end entity that is going to send it its credential (public key for authentication) is genuine. The ICC can go through that chain using the CVC mechanism and thus verifying the delegate CA is known by the RCA. Afterwards, ICC could verify another delegate CA using the key previous delegate CA verified by the RCA. This operation can go on until the ICC receives the End Entity key used to authenticate the IFD.

Following this analysis, the management of certificates can be split into several steps:

- Use of the *Root Certification authority (RCA)* : the only trusted entity of the ICC used for the PKI management
- Use of a *Delegate Certification authority* : an intermediate trusted entity, that was (indirectly) presented by the RCA
- Use of the *End Entity Key*: the final trusted entity, which was (indirectly) presented by the RCA. This entity wants to get authenticated by the ICC.

7.2.6.1 Execution flow for the verification of a certificate chain

Terminal IFD		Card ICC
Construct the certificate chain from CertRoot to CertIFD		

Selects the Root Key in ICC
MSE
P1='81',
P2='B6', // CRT DST
Data = CRT DST with KeyRef identifier = Ref(SDO(RootAC))
	→
	←
		Verify that the key exist and select it

Presents for Verification the first CA Certificate of the chain after the root VERIFY CERTIFICATE P1='00' P2='AE' Data = Certificate as specified in §7.2.4	→ ←	Verify the certificate and save the key and its parameters in the current key context

Selection of the newly available MSE
P1='81'
P2='B6',
Data = CRT DST with KeyRef = CHR
	→
	←
		Save incoming KeyRef in the CRT DST of the current SE having check the key is known.

Terminal IFD		Card ICC
Presents for Verification the last Certificate (User) of the chain VERIFY CERTIFICATE P1='00' P2='AE' Data = Certificate in the format as specified in §7.2.4	→ ←	Verify the certificate and save the key and its parameters in the current key context

7.2.6.2 First step: Root Certification Authority selection

This key shall always be selected prior to engage certificate chain verification.

The Root CA Key shall be stored in a SDO (RSA public portion) in the card. Its attributes are:

- CHA: PKI application AID || Key role Id (typical values are '22' or '21').
- CHR Name of the RCA, that issued the subsequent certificate
- KEYPUB: Public key for certificates verification.

This key can only be used to process a certificate verification.

The certificates verification is processed in two steps:

- Selection of the RCA key by its internal Identifier (SDO identifier)
- Verification of the certificate

The selection and use of the RCA is depicted in §5.2.3.2

Any RSA key public portion SDO may be used as RCA key to initiate the chain certificate verification, provided it contains a CHR and a CHA. The security remains on the knowledge of the private portion that shall never be disclosed.

Once the RCA public key lead to a successful certificate verification, the key nested in the certificate verified is loaded in the current context:.

- CHR: Identifier of the current key. It is the identifier of the key used for the subsequent step (certificate operation or device authentication or role authentication)
- CHA: Containing the role associated to the current key (typical value are '22' or '21')
- KEYPUB: Current public key for verifying certificates
- OID: OID of the current key –specifies the use and algorithm of the current key.

They key extracted from the certificate and loaded in the current context might either be

- a delegate CA key (used for certificate verification)
- a end entity key (used for authentication)

7.2.6.3 Subsequent step: Delegate Certificate Authority verification

When the certificate chain was initiated by processing a first certificate (with a RCA or another delegate CA) on the ICC, the ICC can process subsequent certificate. That certificate might convey a delegate CA (intermediate entity used in the PKI chain - see §7.2.2) that enables only the ICC to verify a subsequent certificate conveying public key for certificate verification or authentication.

From the previous certificate verification, the ICC has the following current cryptographic context:

- CHR: Identifier of the current key. It is the identifier of the key used for the subsequent step (certificate operation or device authentication or role authentication)
- CHA: Containing the role associated to the current key (typical value are '22' or '21')
- KEYPUB: Current public key for verifying certificates
- OID: OID of the current key –specifies the use and algorithm of the current key

The certificates verification is processed in two steps:

- Selection of the delegate CA key (that should match the current context's CHR))
- Processing of the certificate

The selection and use of the delegate CA is depicted in §5.2.3.2

Once the delegate CA public key leads to successful certificate verification, the key nested in the certificate verified is loaded in the current context:

- CHR: Identifier of the current key. It is the identifier of the key used for the subsequent step (certificate operation or device authentication or role authentication)
- CHA: Role associated to the current key (typical value are '22' or '21')
- KEYPUB: Current public key for verifying certificates
- OID: OID of the current key –specifies the use and algorithm of the current key.

They key extracted from the certificate and loaded in the current context might either be

- a delegate CA key (used for certificate verification)
- a end entity key (used for authentication)

7.2.6.4 Last step: End Entity key

The end entity key is the last part of a certificate chain. In that case, the key loaded in the current cryptographic context and extracted from the last certificate verification) is not dedicated to perform a certificate verification but to perform an asymmetric external authentication such as the:

- External authentication in the "device authentication with privacy protection "
- External role authentication

From the previous certificate verification, the ICC has the following current cryptographic context:

- CHR: Identifier of the current key. It is the identifier of the key used for the authentication step (device authentication or role authentication)
- CHA: Containing the role associated to the current key ('01' in case of device authentication with privacy protection or proprietary IAS format in case of role authentication)
- KEYPUB: Current public key for verifying certificates
- OID: OID of the current key –specifies the use and algorithm of the current key

The key is used to process the authentication following these three steps:

- Selection of the authentication key (that should match the current context's CHR)

- Generation of a challenge
- Verification of a Signature with the authentication key loaded in the current context (computed partly over the challenge the ICC generated).

The selection and use of the authentication key is depicted in:

- §5.2.3.2 for the external authentication step of the device authentication with privacy protection
- §5.3.2 for asymmetric external role authentication

Once the external authentication was successfully performed, the privileges associated to the public key is validated, i.e, the role ID (nested in the CHA field) associated to the key used for the external authentication becomes valid and might be used to fulfill access conditions.

The way each access conditions is interpreted is described in

- §5.2.3.6 for device authentication with privacy protection
- §5.3.2.4 for asymmetric external role authentication

7.2.6.5 Verification of certificates: description of internal processing

This chapter aims to describe the way each certificate shall be verified once it is received by the ICC. In particular, operations described in §7.2.6.2 and §7.2.6.3 should behave that way.

This chapter considers the key used for certificate verification has already been selected through the MSE Set commands.

- Cert: Incoming certificate to verify
- Context: current context previously selected through a MSE SET
    - CHR: Current key reference
    - CHA: Current key CHA to verify the intended use of the public key
    - KEYPUB: Public portion of the current key
    - OID: Current key algorithm identifier associated to the current key

Step	Action	Potential error
0	Verify the Context.CHA is compatible with the certificate verification (Value '21' or '22')	Current key not useable for certificate verification
1	Read CAR in Cert and check it match with the current Context.CHR (Link between the certificate AKI on the current key context SKI)	The Cert is not certified by the current key context
2	Verify Digital signature according to ISO 9796-2 algorithm	Incorrect signature
3	Rebuilt the full certificate Cert	
4	Retrieve the CPI of Cert	
5	Retrieve the associated profile in the ICC	CPI unknown
6	Verify that OID in Cert matches the one in the associated profile.	OID not matching the CPI
7	Update the current context with the one stored in Cert. • Context.CHR = Cert.CHR • Context.CHA = Cert.CHA • Context.KEYPUB = Cert.KEYPUB • Context.OID = Cert.OID	
8	End of Certificate Verification	

Note

- Current key context is updated following each certificate verification. Therefore the last one replaces the previous.
- Current key context is erased upon application selection (i.e. ADF).

7.2.7 Profiles of certificates supported

An [IAS ECC] Application shall support the following set of CPI:

- Root certification authority used to initiate the chain certificates verification[17] :

CHA	PuK	OID	Algorithm to use with the nested Public Key	CPI
AID || '21'	Modulus (128 B) || Exponent (4 B)	'2B 24 03 04 02 02 0X'	C.CA: RSA, 1024 bits, SHA according to x, 9796-2	'03'
	Modulus (192 B) || Exponent (4 B)	'2B 24 03 04 02 02 0X'	C.CA: RSA, 1536 bits, SHA according to x, 9796-2	'0F'
	Modulus (256 B) || Exponent (4 B)	'2B 24 03 04 02 02 0X'	C.CA: RSA, 2048 bits, SHA according to x, 9796-2	'13'

- Delegate certification authority. Key used to verify the next certificate of the chain:

CHA	PuK	OID	Algorithm to use with the nested Public Key	CPI

[17] - This table is provided as an example. The card never verifies such a certificate as it owns the public key portion in a SDO.

CHA	PuK	OID	Algorithm to use with the nested Public Key	CPI
AID || '22'	Modulus (128 B) || Exponent (4 B)	'2B 24 03 04 02 02 0X'	C.ICC: RSA, 1024 bits, SHA according to x, 9796-2	'03'
	Modulus (192 B) || Exponent (4 B)	'2B 24 03 04 02 02 0X'	C.ICC: RSA, 1536 bits, SHA according to x,9796-2	'0F'
	Modulus (256 B) || Exponent (4 B)	'2B 24 03 04 02 02 0X'	C.ICC: RSA, 2048 bits, SHA according to x,9796-2	'13'

- Key used for the IFD authentication in the device authentication with privacy protection:

CHA	PuK	OID	Algorithm to use with the nested Public Key	CPI
AID || '01	Modulus (128 B) || Exponent (4 B)	'2A 81 22 F4 2A 02 04 01 0X'	C.AUTH: RSA, 1024 bits, SHA according to x, 9796-2	'88'
	Modulus (192 B) || Exponent (4 B)	'2A 81 22 F4 2A 02 04 01 0X'	C.AUTH: RSA, 1536 bits, SHA according to x, 9796-2	'89'
	Modulus (256 B) || Exponent (4 B)	'2A 81 22 F4 2A 02 04 01 0X'	C.AUTH: RSA, 2048 bits, SHA according to x, 9796-2	'8A'

- Key used for the Asymmetric role authentication:

CHA	PuK	OID	Certificate computation algorithm	CPI
AID || '80 xx xx xx	Modulus (128 B) || Exponent (4 B)	'2A 81 7A 01 81 2F 02 01 0X'	C.RA: RSA, 1024 bits, SHA according to x, 9796-2	'84'
	Modulus (192 B) || Exponent (4 B)	'2A 81 7A 01 81 2F 02 01 0X'	C.RA: RSA, 1536 bits, SHA according to x, 9796-2	'85'
	Modulus (256 B) || Exponent (4 B)	'2A 81 7A 01 81 2F 02 01 0X'	C.RA: RSA, 2048 bits, SHA according to x, 9796-2	'86'

With:

- x='1' for SHA-1
- x='4' for SHA-256

8 PROTOCOL MANAGEMENT

8.1 Communication interface and supported protocols

According to [R18], the **[IAS ECC]** shall use a contact and/or a contactless interfaces. It shall support at least one protocol among the following:

- T=0
- T=1
- T=CL

However, this list is not exhaustive. The support of other protocol is out of the scope of the specification.

8.2 Extended length & command chaining

Extended length feature is optional. The historical bytes of the ATR, stored in the EF.ATR (see §2.3) indicates if the **[IAS ECC]** platform can handle extended length. If ATR/ATS specifies extended length is supported, then data object 'E0' in EF.ATR shall be interpreted. It encodes the following information:

- Maximum length for the incoming command without secure messaging
- Maximum length for the incoming command with secure messaging
- Maximum length for the response data without secure messaging
- Maximum length for the response data with secure messaging

As the **[IAS ECC]** specification

- mandates to use command chaining on the incoming command
- the extended length feature may be used for the data retrieval (if supported) in T=1 & T=CL protocol.

If the extended length is supported, the two significant information are therefore:

  - Maximum length for the response data without secure messaging
  - Maximum length for the response data with secure messaging

These lengths are the maximum capabilities the ICC provides. These are upper limits that the IFD shall not bypass.

8.3 Secure messaging on incoming command

Each command wrapped through the secure messaging layer becomes a case 4 command. Therefore, this feature has an impact on the protocol layer.

8.3.1 T=0 protocol

As each command becomes a case 4 command, in T= 0 protocol, the command shall be handled that way:

Incoming data	Outgoing data	Status Word returned
Incoming command CLA INS P1 P2 Lc Data	N/A	SW = ‘61 L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data} ’ = ‘00’ means 256 bytes are available
Incoming command GET RESPONSE L_{Data}	Return L_{Data} bytes	If all the data were retrieved SW = ‘9000’

Each incoming command sent through the secure messaging layer will be processed in two steps.:

- Incoming command
- GET RESPONSE to retrieve the response data

8.3.2 T=1 & T=CL protocol

In T=1 and T=CL, the Le field should always be set to ‘00’.

8.4 Sending less than 255 bytes to the ICC

These are commands that convey data field that fits within 255 bytes, whether or not secure messaging is applied. They do not need any particular handling. This is fully addressed by [R1] and 14443, whether the protocol is T=0, T=1 or T=CL.

8.5 Sending more than 255 bytes to the ICC : command chaining

The **[IAS ECC]** handles several incoming command that may contain data whose length is greater that ‘FF’ = 255 bytes. This chapter clarifies the behavior of the application in such cases

8.5.1 Need for command chaining

Several commands require sending data greater than 255 bytes. :

- PUT DATA SDO
- EXTERNAL AUTHENTICATE for asymmetric role authentication or device authentication
- PSO Verify certificate
- PSO Decipher
- MSE Set KAT

On the other hand, some other commands should not be used with command chaining, as they may be used in another way

- UPDATE BINARY

8.5.2 Description of the command chaining

In such cases, the command chaining shall be used.

The command chaining is used as follows:

Let's consider the following data field to transmit: 'Data'

The data field 'Data' to send is chopped into several elementary blocks

'Data' = 'D_1' | 'D_2'|…|'D_n'

Each data block 'D_1'… 'D_{n-1}', has a given length L_{block}, and the final one 'D_n' has at most the length L_{block}.

Each elementary block is conveyed using a chain of commands. The bit 5 of the CLA byte of the command is used to indicate if the APDU conveys a part of the chain or the last part of the chain.

If the command, is a part of the chain, the bit 5 of the CLA byte shall be set to 1: CLA.chaining

If the command, is the last of the chain, the bit 5 of the CLA byte shall be set to 1: CLA.last

Therefore, in the example we considered, the Data field 'Data' may be conveyed that way :

CLA.chaining INS P1 P2 L_{block} 'D_1'

CLA.chaining INS P1 P2 L_{block} 'D_2'

- - - - - - - -

CLA.last INS P1 P2 L 'D_n'

Each command of the chain shall be sent in the correct order and shall be consecutive.

There could be as many command chained as needed to convey the all incoming data.

Each correct command of the chain shall return the SW 0x9000 upon correct reception

If the chain is broken, because an unexpected command has been sent, all the intermediate data stored in the internal context of the ICC is lost.

8.5.3 Case without secure messaging

In the case the commands are to be sent without secure messaging, the length of block shall be at most set to 0xFF = 255 bytes

8.5.4 Case with secure messaging

Each command of the chain on which the secure messaging is applied by the IFD, when sent in short length, shall fit within 255 bytes

Each command of the chain, shall be wrapped using the secure messaging described in §7.1.

Each command of the chain conveying the data 'D$_i$' is protected in integrity and confidentiality. In particular, each command of the chain is protected by a cryptographic checksum (MAC).

The secure messaging is applied to each command of the chain as described in §7.1 and if active the incoming data in short length shall fit within 255 bytes.

8.6 Data retrieval

The **[IAS ECC]** handles several commands returning data to the IFD. It is protocol dependant. This chapter clarifies the behavior of the application, which depends on

- the commands
- the protocol
- the number of data to retrieve

8.6.1 Commands to consider

The case 2 & Case 4 commands are the followings:

- GET CHALLENGE
- READ BINARY Odd
- READ BINARY Even
- INTERNAL AUTHENTICATE for C/S authentication or device authentication
- MUTUAL AUTHENTICATE for symmetric device authentication
- GET DATA SDO
- PSO Compute Digital signature
- PSO decipher
- SELECT

Among all theses commands, some of them might return a large amount of data, which might not fit within 256 bytes. Therefore, we can sort out these commands within two categories:

8.6.2 Command returning less than 256 bytes

These are commands that return data that fit within 256 bytes, whether or not secure messaging is applied. They do not need any particular handling to retrieve the data. This is fully addressed by [R1] and 14443, whether the protocol is T=0, T=1 or T=CL.

These commands are :

- GET CHALLENGE

- MUTUAL AUTHENTICATE for symmetric device authentication
- PSO decipher
- SELECT

Note:

If the protocol is T=0, the dedicated command GET RESPONSE (see [R1] & [R2]) will be used for Case 4 commands (MUTUAL AUTHENTICATE for symmetric device authentication, PSO decipher, and SELECT). This command is totally transparent for the **[IAS ECC]** application, as it is not handled by the application, but rather by the protocol layer. Therefore it does not change at all the internal state of the application.

8.6.3 Particular issue for the READ BINARY command

Two commands may be use to read data from a file :

- READ BINARY Odd
- READ BINARY Even

They are used to read Le bytes within a file. When 'Le' > 'E7' (or '00'), the ICC interprets it as 'E7' = 231 bytes. Therefore, these commands will never return more than 256 bytes. The size of data returned by the READ BINARY command (EVEN and ODD) is limited to 'E7' = 231 bytes (included).

Note :

- In case of READ BINARY Odd, the block returned includes the TLV structure, i.e. the length of the block includes the length of the tag '53' and the length field L
- The case of the command READ BINARY used with extended length is out of the scope of this chapter.

8.6.3.1 READ BINARY in T=0

The READ BINARY EVEN command is case 2 command. It returns Le bytes (at most) where Le is the value set in the incoming command. The maximum size of the data block returned is limited to 'E7' = 231 bytes

- If Le (or more) bytes are available, Le bytes are returned upon reception of READ BINARY EVEN command. The size of the data block returned is limited to 'E7' = 231 bytes
- If less than 'Le' bytes are available, the Status Word '6CL$_{\text{Data}}$' is returned upon reception of the READ BINARY EVEN command. The data are retrieved by sending again the command READ BINARY EVEN with the length indicated by L$_{\text{Data}}$. The second READ BINARY command is transparent to the application .It is handled by the transport layer and does not modify the internal state of the application.

The READ BINARY ODD command is a case 4 command. Depending on the size of the available data in the file, it returns up to 'E4' = 228 bytes (if available), or the remaining bytes (if smaller than 'E4' = 228 bytes) embedded within the TL structure.

- The Status Word ‘61L_{Data}’ is returned upon reception of the READ BINARY ODD command. The data are retrieved using GET RESPONSE command.

Here is depicted the command flow for the command READ BINARY Odd & Event in T= 0

- READ BINARY EVEN : Le bytes required & more than Le byte available

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY EVEN Le	Le bytes (the size is limited to ‘E7’)	SW = ‘9000’

- READ BINARY EVEN : Le bytes required & less than Le byte available

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY EVEN Le	N/A	SW = ‘6C L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data}’ is limited to ‘E7’
Incoming command READ BINARY EVEN L_{Data}	Return L_{Data} bytes	SW = ‘9000’

- READ BINARY ODD

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY ODD	N/A	SW = ‘61 L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data} ’ is limited to ‘E7’
Incoming command GET RESPONSE L_{Data} (see §8.6.5)	Return L_{Data} bytes	SW = ‘9000’

8.6.3.2 READ BINARY in T=1 or T=CL

The READ BINARY EVEN command is case 2 command. It returns Le bytes (at most) where Le is the value set in the incoming command. The maximum size of the data block returned is limited to ‘E7’ = 231 bytes

- If Le (or more) bytes are available, Le bytes are returned upon reception of READ BINARY EVEN command. The size of the data block returned is limited to ‘E7’ = 231 bytes
- If less than ‘Le’ bytes are available, the remaining bytes are returned and the Status Word ‘6282’ is returned.

The READ BINARY ODD command is a case 4 command. Depending on the size of the available data in the file, it returns up to ‘E4’ = 228 bytes (if available), or the remaining bytes (if smaller than ‘E4’ = 228 bytes) embedded within the TL structure.

- If Le (or more) bytes are available, Le bytes are returned upon reception of READ BINARY EVEN command. The size of the data block returned is limited to ‘E7’ = 231 bytes
- If less than ‘Le’ bytes are available, the remaining bytes are returned and the Status Word ‘6282’ is returned.

Here is depicted the command flow for the command READ BINARY Odd & Even in T= 1 or T=CL

- READ BINARY EVEN : Le bytes required & more than Le byte available

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY EVEN	Le bytes (the size is limited to ‘E7’)	SW = ‘9000’

- READ BINARY ODD : Le bytes required & less than Le byte available

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY ODD	Return L_{Data} bytes (‘L_{Data}’ is limited to ‘E7’)	SW = ‘6282’

- READ BINARY ODD : Le bytes required & more than Le byte available

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY ODD	Le bytes (the size is limited to ‘E7’)	SW = ‘9000’

- READ BINARY EVEN : Le bytes required & less than Le byte available

Incoming data	Outgoing data	Status Word returned
Incoming command READ BINARY ODD	Return L_{Data} bytes (‘L_{Data}’ is limited to ‘E7’)	SW = ‘6282’

8.6.3.3 READ BINARY with Secure messaging

If the secure messaging is used, the length of data to return is indicated within the D.O. ‘97’ (see §7.1.6).

Again, when ‘Le’ > ‘E7’ (or ‘00’), the ICC interprets it as ‘E7’ = 231 bytes. The size of plain text data returned by the READ BINARY command (EVEN and ODD) is limited to ‘E7’ = 231 bytes (included), so that the data protected in integrity & confidentiality with the secure messaging does not exceed 256 bytes.

When the ICC receives the READ BINARY command requesting ‘Le’ bytes, If ‘Le’ bytes are available in the file, the ICC takes these ‘Le’ bytes and wrap it thanks to the secure messaging layer. If less than ‘Le’ byte are available, the remaining bytes are taken and wrapped through the secure messaging layer.

Finally, these data can be retrieved by the IFD according to the protocol rules for case 4 commands. As the data to return will never be greater than 256 bytes, this case is fully addressed by [R1]

8.6.4 Command returning more than 256 bytes

Several commands may return data greater than 256 bytes (with or without secure messaging). :

- INTERNAL AUTHENTICATE for C/S authentication or device authentication
- GET DATA SDO
- PSO Compute Digital signature
- PSO Decipher

When these commands are correctly processed:

- the outgoing data are prepared (signature computed,…)
- if needed, depending on the security level required, the data are wrapped or not through the secure messaging layer.

Then the data are made available to the IFD

This amount of data has to be retrieved from the ICC. This chapter describes the way to do so, depending on the protocol used for the communication (T=0, T=1 or T=CL)

8.6.4.1 Data retrieval in T=0 protocol

The IFD should set the ‘Le’ field to ‘00’.

In such case, the data retrieval is performed by using the command GET RESPONSE. The IFD shall recover the data that way:

Incoming data	Outgoing data	Status Word returned
Incoming command CLA INS P1 P2 Lc Data (case 4) Or CLA INS P1 P2 Le (case 2)	N/A	SW = ‘61 L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data} ’ = ‘00’ means 256 bytes are available
Incoming command GET RESPONSE L_{Data} (see §8.6.5)	Return L_{Data} bytes	If there are still remaining data SW = ‘61 L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data} ’ = ‘00’ means 256 bytes are available If all the data were retrieved SW = ‘9000’
Carry on the sequence of GET RESPONSE command until the SW ‘9000’ is returned		

Once the incoming command is received, the outgoing data can only be retrieved by using the command GET RESPONSE (see §8.6.5) as described above.

This command GET RESPONSE does not modify at all the internal status of the IAS application (authentication status;….). This command is just handled by the transport layer.

Once the sequence of data retrieval using GET RESPONSE command has started, it shall be carried on until the entire completion of the sequence. If a command different from the GET RESPONSE command is sent during the data retrieval sequence, all the remaining data to recover are erased, and are not retrievable at all.

8.6.4.2 Data retrieval in T=1 & T=CL protocol

Three different scenarii may be considered :

- The ICC supports extended length for data retrieval (outgoing data) and the data to return fit within the maximum length for the response data (set in the ATR see §2.3)
- The ICC does not support extended length for data retrieval (outgoing data)
- The data to return do not fit within the maximum length for the response data (set in the ATR)

Note: In case the extended length feature is used, the maximum length for the response data to consider is the Maximum length for the response data with secure messaging.

8.6.4.2.1 The ICC supports extended length for data retrieval (outgoing data) and the data to return fit within the maximum length for the response data

If the **[IAS ECC]** application supports extended length, the data are returned using this feature.

To do so, the incoming command, shall be sent using the extended length (see [R1]), i.e.

- the Lc field shall be encoded over three bytes : 00 XX YY (for Case 4 command)
- the Le field shall be encoded over two bytes set to zero : ‘0000’

The ICC returns all the available data to the IFD, using extended APDU (see [R1]).

8.6.4.2.2 The ICC does not support extended length

If the ICC does not support extended length, the outgoing data shall be retrieved using short length.

The data retrieval is performed by using the command GET RESPONSE. The IFD shall recover the data that way:

Incoming data	Outgoing data	Status Word returned
Incoming command CLA INS P1 P2 Lc Data Le	Return 256 bytes	SW = ‘61 L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data} ’ = ‘00’ means 256 bytes are available
Incoming command GET RESPONSE L_{Data} (see §8.6.5)	Return L_{Data} bytes	If there are still remaining data SW = ‘61 L_{Data}’ ‘L_{Data} ‘ indicates the length of data to retrieve ‘L_{Data} ’ = ‘00’ means 256 bytes are available If all the data were retrieved SW = ‘9000’
Carry on the sequence of GET RESPONSE command until the SW ‘9000’ is returned		

Once the incoming command is received, the outgoing data can only be retrieved by using the command GET RESPONSE (see §8.6.5) as described above.

This command GET RESPONSE does not modify at all the internal status of the IAS application (authentication status;….). This command is just handled by the transport layer.

Once the sequence of data retrieval using GET RESPONSE command has started, it shall be carried on until the entire completion of the sequence. If a command different from the GET RESPONSE command is sent during the data retrieval sequence, all the remaining data to recover are erased, and are not retrievable at all.

8.6.4.2.3 The data to return do not fit within the maximum length for the response data

If the **[IAS ECC]** application supports extended length,the data are returned using this feature.

To do so, the incoming command, shall be sent using the extended length (see [R1]), i.e.

- the Lc field shall be encoded over three bytes : 00 XX YY (for Case 4 command)
- the Le field shall be encoded over two bytes set to zero : ‘0000’

However, the whole data to return might not fit within the maximum length for the response data field.

The data retrieval is performed by using the command GET RESPONSE. The IFD shall recover the data that way:

Let L_{max} be the maximum length for the response data.

Incoming data	Outgoing data	Status Word returned
Incoming command CLA INS P1 P2 Lc Data Le	Return L_{max} bytes	SW = ‘6100’ It means at most L_{max} bytes are still available
Incoming command GET RESPONSE 00 L_{max} Or GET RESPONSE 00 00 00 (see §8.6.5)	Return at most L_{max} bytes	If there are still remaining data SW = ‘6100’ It means at most L_{max} bytes are still available If all the data were retrieved SW = ‘9000’
Carry on the sequence of GET RESPONSE command until the SW ‘9000’ is returned		

Once the incoming command is received, the outgoing data can only be retrieved by using the command GET RESPONSE (see §8.6.5) as described above.

This command GET RESPONSE does not modify at all the internal status of the IAS application (authentication status;….). This command is just handled by the transport layer.

Once the sequence of data retrieval using GET RESPONSE command has started, it shall be carried on until the entire completion of the sequence. If a command different from the GET RESPONSE command is sent during the data retrieval sequence, all the remaining data to recover are erased, and are not retrievable at all.

8.6.5 GET RESPONSE

CLA	‘00’
INS	‘C0’
P1	‘00’
P2	‘00’
L_e field	‘LL’

DATA FIELD	Data to retrieve from the ICC
SW1-SW2	‘6985’ - There are no data to retrieve ‘61xx’ - There are still ‘xx’ bytes to retrieve from the ICC ‘9000’ - Correct processing – The data retrieval sequence is completed

Depending on the use case, the Le field (‘LL’’) may be a short length (encoded over one byte) or an extended length (encoded over three bytes 00 xx yy).

9 APDU references

9.1 Command ⇔response pairs

APDU command-response pairs are handled as indicated in [R1].

9.2 CLASS byte coding

CLA indicates the class of the command. According to [R2], 'FF' is an invalid value.

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

The values '000x XXXx' 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
			-	-	-	×	×	**Logical channel number from zero to three**	NO

Table 24 - CLASS byte interindustry values

9.3 The status bytes SW1 & SW2

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

As indicated to ISO/IEC 7816-3 specifications, the following values are **invalid**:

- '6XXX'
- '9XXX'
- '6'X''

The following values are **interindustry**:

- '61XX'
- '62XX'
- '63XX'
- '64XX
- '65XX'
- '66XX'
- '68XX'

As indicated to ISO/IEC 7816-3 specifications, the following values are **proprietary**:

- '67XX'
- '6BXX'
- '6DXX'
- '6EXX'
- '6FXX'
- '9XXX',

Except the following values that are **interindustry**:

- '6700'
- '6B00'
- '6D00'

- '69XX'
- '6AXX'
- '6CXX'.

- '6E00'
- '6F00'
- '9000'

	SW1-SW2	Meaning
Normal processing	'9000'	No further qualification
	'61XX'	SW2 encodes the number of data bytes still available
Warning processing	'62XX'	State of non-volatile memory is unchanged (further qualification in SW2)
	'63XX'	State of non-volatile memory has changed (further qualification in SW2)
Execution error	'64XX'	State of non-volatile memory is unchanged (further qualification in SW2)
	'65XX'	State of non-volatile memory has changed (further qualification in SW2)
	'66XX'	Security-related issues
Checking error	'6700'	Wrong length; no further indication
	'68XX'	Functions in CLA not supported (further qualification in SW2)
	'69XX'	Command not allowed (further qualification in SW2)
	'6AXX'	Wrong parameters P1-P2 (further qualification in SW2)
	'6B00'	Wrong parameters P1-P2
	'6CXX'	Wrong L_e field; SW2 encodes the exact number of available data bytes
	'6D00'	Instruction code not supported or invalid
	'6E00'	Class not supported
	'6F00'	No precise diagnosis

Table 25 - 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

SW1	SW2	Meaning
'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' (error)	'00'	No information given
	'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'	Lc inconsistent with TLV structure
	'86'	Incorrect parameters P1-P2
	'87'	Lc 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 26 - Specific interindustry warning and error conditions

9.3.1 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).

9.4 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)
-	-	-	×	×	×	×	×	User authentication DO reference
0	0	0	0	0	0	0	0	Forbidden

Table 27 - coding of P2: qualifiers of the reference data

9.4.1 VERIFY

CLA INS P1 P2	ISO ‘20’ ‘00’ See Table 27
L_c field	Variable
Data field	Verification data

SW1-SW2	‘6A86’ - P1 ≠ ‘00’ ‘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.

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

If the referenced PIN is successfully verified, the remaining tries counter returns to its maximum and the validation flag is set to "true".

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

9.4.2 CHANGE REFERENCE DATA

CLA INS P1 P2	ISO '24' '00' See Table 27
L_c field	Length of command data field
Data field	Current Password || New reference data (i.e. new PIN or password)

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
	'6983'	- Referenced PIN not successfully verified AND no subsequent tries are allowed (remaining tries counter reached 0)
	'6984'	- Referenced PIN usage counter reached 0

A successful execution of the command reset the PIN tries counter to the maximum.

The PIN usage counter is decremented by this command even if verification fails.

Upon successful execution of Change Reference Data:

- The PIN tries counter is reset to its maximum as defined during the SDO personalization.
- The PIN presentation flag is reset (i.e. the PIN shall be presented again to access data it protects).

When this command is received

- The application retrieves the current length L (stored within the SDO body length of the selected PIN P)
- The current password is considered to be the first L bytes of the data field.
- Performs the PIN verification by checking P value and the current password.
- Update P with the new reference data
- If needed, update L with the new PIN length. The length of the new password is :

$L_{new} = L_c — L_{old}$

This command can only be executed on a PIN not in a blocked state (usage counter or error counter reached '0')

9.4.3 RESET RETRY COUNTER

CLA INS P1 P2	ISO '2C' '02' (New reference data) or '03' (Data field is absent) See Table 27
L_c field	Variable

DATA FIELD	Absent (P1 = '03') or New reference data (P1 = '02')

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.

A successful Reset Retry Counter command execution unblocks a user authentication SDO (i.e. PIN or password).

For compliancy with [R17] in the case of a qualified signature only, this command shall be consecutive to a VERIFY providing the corresponding resetting code. This means the only unblocking condition that can be provided for a PIN is another PIN commonly called PUK.

The successful process of the command performs the following:

- The unblocking PIN validation flag remains unchanged;
- The referenced PIN remaining tries counter is reset to the maximum error counter;
- The referenced PIN usage counter remains unchanged (if it exists);

If P1 = '02' the referenced PIN value is replaced with the new reference data;

The referenced PIN validation flag is reset (to false).

9.5 Authentication commands

9.5.1 GET CHALLENGE

CLA INS P1 P2	ISO ‘84’ ‘00' ‘00’
L_e field	‘08’

DATA FIELD	Challenge
SW1-SW2	‘6A86’ - P1 ≠ ‘00’ or P2 ≠ ‘00’ ‘6700’ - Lc ≠ ‘08’

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

9.5.2 INTERNAL AUTHENTICATE. PK-DH scheme

CLA INS P1 P2	‘0C’ ‘88’ ‘00’ Algorithm reference → no further information (information available in current SE) ‘00’ Secret reference → no further information (information available in the current SE)
L_c field	‘08’
Data field	RND.IFD
L_e field	Variable

DATA FIELD	SN.ICC || SIG Having: SIG = RSA $_{[SK.ICC.AUT]}$ (‘6A’ || PRND (Padding_bytes_randomly_generated[18]) || C || ‘BC’) where C = HASH[19] (PRND || K_{ICC} || SN.ICC || RND.IFD || K_{IFD} || DH.P) Where SN.ICC is the 8 least significant bytes of the ICC serial number

[18] - Padding bytes are added such as the length of the data to encrypt is equal to the length of the key modulus.

[19] - HASH ≡ SHA-1 or SHA-256

SW1-SW2	'6A86' - P1P2 ≠ '0000' '6700' - Lc ≠ '08' '6E00' - Wrong CLA (secure messaging not present) '6985' - Authentication process not respected (Bad order of operations), or SE content doesn't allow processing the command. '6A88' - Data needed for internal authenticate not found.

The IFD calls INTERNAL AUTHENTICATE. The incoming challenge is RND.IFD. The ICC computes the signature over the challenge and the key token K_{IFD}.K_{ICC} and returns the result.

The command is protected under secure messaging.

The IFD verifies the card answer using the trusted key PK.ICC.AUT. It gets evidence that the signer (certificate holder) is identical to the one that made the key negotiation.

9.5.3 EXTERNAL AUTHENTICATE. PK-DH scheme

CLA	'0C'
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)
L_c field	Variable
DATA FIELD	Having: SN.IFD || SIG = RSA $_{[SK.IFD.AUT]}$ ('6A' || PRND (Padding_bytes_randomly_generated[18)] || C || 'BC') where C = HASH[20] (PRND || K_{IFD} || SN.IFD || RND.ICC || K_{ICC} || DH.P)

SW1-SW2	'6A86' - P1P2 ≠ '0000' '6700' - Wrong length. '6E00' - Wrong CLA (secure messaging not present) '6985' - Authentication process not respected (Bad order of operations), or SE content doesn't allow processing the command. '6A88' - Data needed for external authentication not found. '6300' - external authentication failed. '9000' - external authentication successful.

The incoming signature verification is made using PK.IFD.AUT received in C_CV.IFD.AUT that was provided as incoming parameter in the previous PSO VERIFY CERTIFICATE. PK.IFD.AUT reference is the CHR presents in the same C_CV.IFD.AUT.

[20] - HASH ≡ SHA-1 or SHA-256

9.5.4 MUTUAL AUTHENTICATE. SK scheme

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)
L_c field	'48'
DATA FIELD	IncomingEncryptedText || MAC Having: IncomingEncryptedText = 3DESCBC[KENC](RND.IFD || SN.IFD || RND.ICC || SN.ICC || KIFD) and MAC = MAC_Algo [KMAC](IncomingEncryptedText)
L_e field	'48'

DATA FIELD	OutgoingEncryptedText || MAC Having: OutgoingEncryptedText = 3DESCBC[KENC](RND.ICC || SN.ICC || RND.IFD || SN.IFD || KICC) and MAC = MAC_Algo [KMAC](OutgoingEncryptedText)
SW1-SW2	'6A86' - P1P2 ≠ '0000' '6700' - Lc ≠ '48' '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.

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.

9.5.5 EXTERNAL AUTHENTICATE for Role authentication

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)
L_c field	Variable
DATA FIELD	Asymmetric Role authentication: SIG = DS $_{[SK.IFD.ROLE\_AUT]}$ (RND.ICC||SN.ICC) Symmetric Role authentication: ENCRYPT || MAC Having: ENCRYPT = 3DESCBC[KROLE_ENC](RND.ICC||SN.ICC) and MAC = MAC_Algo [KROLE_MAC](ENCRYPT) Where SN.ICC is the 8 least significant bytes of the ICC serial number

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 OK.

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

EXTERNAL AUTHENTICATE should be preceded by the GET CHALLENGE command.

9.5.6 INTERNAL AUTHENTICATE for Client/Server authentication

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)

L_c field	Lc ≤ 40% of the key length in Bytes (i.e. '33' for 1024 bits, '4C' for 1536 bits and '66' for 2048)
Data field	Incoming message
L_e field	Variable

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.

9.6 Perform security operation commands

9.6.1 PSO – HASH

CLA INS P1 P2	ISO ‘2A’ ‘90' ‘A0’
Lc field	Data field length shall be: In SHA1 ≤ ‘60’ In SHA256 ≤ ‘6C’
Data field	T1 || L1 || V1 || T2 || L2 || V2 T1 = ‘90’ Intermediate hash code L1 = ‘00’ or 28 bytes for SHA-1 or 40 bytes for SHA-256 V1 = ’intermediate hash value’ || ‘8 bytes (nb of bits already hashed)’ T2 = ‘80’ last block ‘0’ < L2 ≤ '40' V2 = Text to hash without padding

SW1-SW2	‘6985’ - If the AlgID of the CRT HT of the current SE is incorrect. ‘6A80’ - TLV structure error. ‘6700’ - Bad length

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.

9.6.2 PSO – Compute Digital Signature

CLA INS P1 P2	ISO ‘2A’ ‘9E' ‘9A’ Data to sign
Le field	Signature length = ‘00’

Data field	Digital signature - unformatted
SW1-SW2	‘6985’ - No hash available '6A88' - Current SE problem ‘6A86’ - Incorrect P1-P2 ‘6982’ - Signature key access conditions not fulfilled

9.6.3 PSO – Verify Certificate

CLA INS P1 P2	ISO ‘2A’ ‘00' ‘AE’
Lc field	Variable
Data field	Incoming certificate to verify CVC

SW1-SW2	‘6A86’ - Incorrect P1-P2 ‘6A80’ - The received certificate is not a CVC formatted structure. Certificate profile not supported (unknown CPI) Current SE problem Certificate verification failed. '6985' - Certificate profile not supported (unknown CPI) ‘6A88’ - Current SE problem ‘6982’ - Certificate verification failed.

9.6.4 PSO – Decipher

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

Data field	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

PSO-Decipher command is used to decipher an encrypted message using a key stored in the card.

9.7 File Management System

This chapter groups commands related to explicit file selection and / or file life cycle. In this context, one shall notice that P1-P2 coding described in the two following tables are used by all the command of the underlying chapters.

The following tables (Table 28 and Table 29) show the coding of P1 and P2 for the READ BINARY UPDATE BINARY commands:

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 28 - P1-P2 coding when B1 of INS = 0b0

P1								P2							
B8	B7	B6	B5	B4	B3	B2	B1	B8	B7	B6	B5	B4	B3	B2	B1
0	0	0	0	0	0	0	0	0	0	0	Short File Identifier 1 ≤ SFI ≤ 30				
Identifier of the file where to apply the command. P1-P2 = '0000' means current file															

Table 29 - P1-P2 coding when B1 of INS = 0b1

When SFI is used no current EF is required. When the command is completed, the current EF becomes the one identified with the SFI. The SFI is searched only in the current DF.

9.7.1 SELECT

COMMAND PARAMETER	MEANING
CLA	ISO
INS	'A4'
P1	See Table 30
P2	See Table 31
L_c 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, DF name or relative path
L_e field	Variable

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' - Nc inconsistent with parameters P1-P2

This command is used to select a file (EF, DF) or an application (Root or ADF). After a successful selection the file selected becomes the current file. After the reset the current DF is the Root and no EF is selected.

After a successful selection following rules shall 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.

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	- Select Root	Root identifier (0x3F00)
						0	1	- Select child DF	DF identifier
						1	0	- Select EF under the current DF	EF identifier

						1	1	- Select parent DF of the current DF. Upper limit = ADF or Root	None
				0	1	X	x	**Selection by name**	
						0	0	- Select by DF name (ADF or root)	AID
				1	0	X	x	**Selection by path**	
						0	1	- Select from the current DF	Path without the current DF identifier

Table 30 - Selection, file and life cycle commands P1 possible values

B8	B7	B6	B5	B4	B3	B2	B1	MEANING
0	0	0	0	-	-	x	x	**File occurrence**
				-	-	0	0	- First or only occurrence
				x	X	-	-	**File control Parameters**
				0	0	-	-	Not supported
				0	1	-	-	- Return FCP template, mandatory use of FCP tag and length
				1	1	-	-	- No data in response field

Table 31 - Selection, file and life cycle commands P2 possible values

9.7.2 READ BINARY

This command is for reading the content of a transparent EF.

9.7.2.1 Even INS

Classical READ BINARY:

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'B0' As indicated in Table 28
L_e field	Number of bytes to read

RESPONSE PARAMETER	MEANING
Data field	Data read
SW1-SW2	See Table 34

Table 32: READ BINARY command-response pair (even instruction code)

9.7.2.2 Odd INS

Enhanced READ BINARY:

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'B1' As indicated Table 29
L_c field	'04'
Data field	'54' L_{54} = '02' {Offset in file encoded in 2 bytes}
L_e field	Number of bytes to return (including tag and length of the discretionary data object – therefore it shall be different from ‘01’)

The offset in the data field of the command APDU is a value relative to the beginning of the file.

RESPONSE PARAMETER	MEANING
Data field	Discretionary data object with tag ‘53’
SW1-SW2	See Table 34

Table 33: READ BINARY command-response pair (odd instruction code)

	MEANING
SW1-SW2	'6282' - End of file reached before reading 'Ne' 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 34: SW1-SW2 for READ BINARY command

As the ICC does not process the response data, the discretionary data object with tag '53' shall always be used in the response data field.

If the 'Le' byte equals '82', the command cannot be processed properly. In this case the discretionary data object '53' - '7F' - {data read from the file} or '53' - '81 7F' - {data read from the card} shall be returned

For more details about the behaviour of the READ BINARY command ,see §8.6.3

9.7.3 UPDATE BINARY

This command is for updating the content of a transparent EF.

9.7.3.1 Even INS

Classical UPDATE BINARY:

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'D6' As indicated in Table 28
L_c field	Length of data field
Data field	Data to write
L_e field	Absent

RESPONSE PARAMETER	MEANING
Data field	Absent
SW1-SW2	See Table 37

Table 35: UPDATE BINARY command-response pair (even instruction code)

9.7.3.2 Odd INS

Enhanced UPDATE BINARY:

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'D7' As indicated Table 29
L_c field	Length of data field
Data field	'54' '02' {Offset data object in file} || '53' L_{53} {Discretionary data object}
L_e field	Absent

RESPONSE PARAMETER	MEANING
Data field	Absent
SW1-SW2	See Table 37

Table 36: UPDATE BINARY command-response pair (odd instruction code)

	MEANING
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)

Table 37: SW1-SW2 for UPDATE BINARY command

9.7.4 CREATE FILE

This command enables to create an EF.

The EF is created in the current DF. Once creation is successful, EF becomes the current EF.

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'E0' '0000'
L_c field	Length of data field
Data field	File Control parameter of the EF to create
L_e field	Absent

RESPONSE PARAMETER	MEANING
Data field	Absent
SW1-SW2	'6982' - Security status not satisfied '6985' - Conditions of use not satisfied ‘6A89’- File already exist - EFID/SFI already exist ‘6A84’- Memory failure – no more memory to create the file '6A86' - Incorrect parameters P1-P2 ‘6A80’ – Wrong Data field - The FCP has a wrong format

Note:

- The Life cycle status byte D.O. if present, is not interpreted
- The interpretation of the SFI is described in §3.3.4.1.5

9.7.5 DELETE FILE

This command deletes the current EF or DF. The command applies only on the file currently selected. SDO present in a DF are deleted too.

There is no current EF following an EF deletion. Having deleted a DF, the card set its parent DF as the current DF; there is no current EF.

Security attributes relevant to a DF deletion are present in the DF FCP.

Security attributes relevant to an EF deletion are present in the EF FCP.

COMMAND PARAMETER	**MEANING**
CLA INS P1-P2	ISO 'E4' '0000' – Delete the currently selected file (DF or EF)
L_c field	‘00’
Data field	Absent
L_e field	Absent

RESPONSE PARAMETER	**MEANING**
Data field	Absent
SW1-SW2	'6982' - Security status not satisfied '6985' - Conditions of use not satisfied – File is an ADF or Root '6A86' - Incorrect parameters P1-P2 '6700' - Wrong length

9.7.6 ACTIVATE FILE

This command is used to turn a DF or EF to the activated state.

No error occurs if the file was already activated

The command applies on the file currently selected.

COMMAND PARAMETER	**MEANING**
CLA INS P1-P2	ISO '44' '0000' – Activate the currently selected file (DF or EF)
L_c field	‘00’
Data field	Absent
L_e field	Absent

RESPONSE PARAMETER	**MEANING**
Data field	Absent
SW1-SW2	'6982' - Security status not satisfied '6985' - Conditions of use not satisfied '6A86' - Incorrect parameters P1-P2 ‘6700’ – Wrong Length

9.7.7 DEACTIVATE FILE

This command is used to put in the deactivated state a DF or EF. If the file was already deactivated no error occurs.

The command applies to the file (DF or EF) currently selected.

COMMAND PARAMETER	**MEANING**
CLA INS P1-P2	ISO '04' '0000' – Deactivate the currently selected file (DF or EF).
L_c field	‘00’
Data field	Absent
L_e field	Absent

RESPONSE PARAMETER	**MEANING**
Data field	Absent
SW1-SW2	'6982' - Security status not satisfied '6985' - Conditions of use not satisfied – File is an ADF or the ROOT '6283' - Selected file is already deactivated '6A86' - Incorrect parameters P1-P2 ‘6700’ – Wrong Length

9.7.8 TERMINATE DF

This terminates a DF. If the DF was already terminated no error occurs.

The command applies to the DF currently selected.

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'E6' '0000' – Terminate the currently selected DF.
L_c field	‘00’
Data field	Absent
L_e field	Absent

RESPONSE PARAMETER	MEANING
Data field	Absent
SW1-SW2	'6982' - Security status not satisfied '6985' - Conditions of use not satisfied '6285' - Selected file is already terminated '6A86' - Incorrect parameters P1-P2 ‘6700’ – Wrong Length

9.7.9 TERMINATE EF

This command terminates an EF. If the EF was already terminated no error occurs.

The command applies on the current selected EF.

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'E8' '0000' – Terminate the currently selected EF.
L_c field	‘00’
Data field	Absent
L_e field	Absent

RESPONSE PARAMETER	MEANING
Data field	Absent
SW1-SW2	'6982' - Security status not satisfied '6985' - Conditions of use not satisfied – No current EF '6285' - Selected file is already terminated '6A86' - Incorrect parameters P1-P2 ‘6700’ – Wrong Length

9.8 Data object command

9.8.1 GET DATA

9.8.1.1 GET DATA for SDO

This command allows retrieving the whole DOUP or DOCP (except for DH keys) of a SDO. Incoming data for this command is an extended header list, being a constructed TAG only. Some SDO parameters can be retrieved some other cannot, depending on the type of the object, refer to § 3.4.5 and § 3.4.6 for more information.

This implies that parameters cannot be retrieved independently. Nevertheless, for Domain Parameters Keys (DH keys) each parameter of the DOCP shall be retrieved independently. Thus using an extended header list being the primitive TAG associated to the required parameter (see §3.4.6.5).

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

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

<u>Using the extended header list for SDO retrieval:</u>

The extended header list requires the retrieval of a constructed data object only (e.g. a complete SDO, or only its header).

Example, read the DOCP of the PIN 1:

COMMAND PARAMETER	**MEANING**
CLA INS P1-P2	ISO 'CB' '3FFF'

<table>
<tr><td>L_c field</td><td>‘0A’</td></tr>
<tr><td>Data field</td><td><table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'4D' {...}</td><td>‘08’</td><td>Extended Header List</td></tr>
<tr><td>'70' {...}</td><td>‘06’</td><td>Interindustry template for further objects.</td></tr>
<tr><td>’BF8101’ {...}</td><td>‘02’</td><td>PIN 1</td></tr>
<tr><td>‘A0’ {...}</td><td>‘80’</td><td>DOCP</td></tr>
</table></td></tr>
<tr><td>L_e field</td><td>Absent</td></tr>
</table>

Example, read the DOUP of the PIN 1:

<table>
<tr><th>COMMAND PARAMETER</th><th>MEANING</th></tr>
<tr><td>CLA
INS
P1-P2</td><td>ISO
'CB'<br>'3FFF'</td></tr>
<tr><td>L_c field</td><td>‘0A’</td></tr>
<tr><td>Data field</td><td><table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'4D' {...}</td><td>‘08’</td><td>Extended Header List</td></tr>
<tr><td>'70' {...}</td><td>‘06’</td><td>Interindustry template for further objects.</td></tr>
<tr><td>’BF8101’ {...}</td><td>‘03’</td><td>PIN 1</td></tr>
<tr><td>‘7F41’ {...}</td><td>‘80’</td><td>DOUP</td></tr>
</table></td></tr>
<tr><td>L_e field</td><td>Absent</td></tr>
</table>

Example, read the DOUP the SE1:

<table>
<tr><th>COMMAND PARAMETER</th><th>MEANING</th></tr>
<tr><td>CLA
INS
P1-P2</td><td>ISO
'CB'<br>'3FFF'</td></tr>
<tr><td>L_c field</td><td>‘0A’</td></tr>
<tr><td>Data field</td><td><table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'4D' {...}</td><td>‘08’</td><td>Extended Header List</td></tr>
<tr><td>'70' {...}</td><td>‘06’</td><td>Interindustry template for further objects.</td></tr>
<tr><td>’BFFB01’ {...}</td><td>‘02’</td><td>SE 1</td></tr>
<tr><td>‘7B’ {...}</td><td>‘80’</td><td>DOUP</td></tr>
</table></td></tr>
<tr><td>L_e field</td><td>Absent</td></tr>
</table>

9.8.1.2 GET DATA for K.ICC retrieval

This command takes place during the ICC/IFD mutual authentication using PK-DH asymmetric scheme. It shall be called following a MSE SET KAT command; thus, the IFD retrieves the K.ICC generated by the card.

COMMAND PARAMETER	**MEANING**
CLA INS P1-P2	‘00’ 'CB' '3FFF'
L_c field	‘06’
Data field	‘4D04’ Extended header list: ‘A6’ template KAT ‘02’ length of subsequent TL pair ‘9100’ TL pair to indicate retrieval of K.ICC
L_e field	Variable

RESPONSE PARAMETER	**MEANING**
Data field	DH Public session key K.ICC formatted as follows: ‘A6’ L_{A6} { ‘91’ L_{91} { K.ICC value} } According to the length of the DH SDO previously chosen: • 1024 bit-long key: L_{A6} = ‘8183’ (131), L_{91} = ‘8180’ (128) • 1536 bit-long key: L_{A6} = ‘81C3’ (195), L_{91} = ‘81C0’ (192) • 2048 bit-long key: L_{A6} = ‘820104’ (260), L_{91} = ‘820100’ (256)
SW1-SW2	'6700' - Wrong length; no further indication '6A86' - Incorrect parameters P1-P2 '6A80' - Incorrect parameters in the command data field '6985' - Wrong state

9.8.2 PUT DATA

PUT DATA command is applicable only for SDO.

It can only be used for updating an existing SDO.

COMMAND PARAMETER	**MEANING**
CLA INS P1-P2	ISO 'DB' '3FFF' – Current DF
L_c field	Variable
Data field	Data to put BER-TLV formatted

L_e field	Absent

RESPONSE PARAMETER	MEANING
Data field	Absent
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 can not be put

Example, update RSA Public Key:

<table>
<tr><th>COMMAND PARAMETER</th><th>MEANING</th></tr>
<tr><td>CLA
INS
P1-P2</td><td>ISO
'DB'<br>'3FFF'</td></tr>
<tr><td>L_c field</td><td>‘L_C’</td></tr>
<tr><td>Data field</td><td>
<table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'70' {...}</td><td>‘L_{70}’</td><td>Interindustry template for further objects.</td></tr>
<tr><td>’BFA001’ {...}</td><td>‘L_{BFA001}’</td><td>RSA Public Key</td></tr>
<tr><td>‘7F49’ {...}</td><td>‘L_{7F49}’</td><td>DOUP</td></tr>
<tr><td>'81' {...}</td><td>‘L_N’</td><td>Modulus</td></tr>
<tr><td>'82' {...}</td><td>‘L_E’</td><td>Public Exponent</td></tr>
<tr><td>'5F20' {...}</td><td>‘L_{CHR}’</td><td>Certificate Holder Reference</td></tr>
</table>
</td></tr>
<tr><td>L_e field</td><td>Absent</td></tr>
</table>

Example, update RSA Private Key:

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO 'DB' '3FFF'
L_c field	‘L_C’

<table>
<tr><td>Data field</td><td>
<table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'70' {...}</td><td>‘L_{70}’</td><td>Interindustry template for further objects.</td></tr>
<tr><td>'BF9001' {...}</td><td>‘L_{BF9001}’</td><td>RSA Private Key</td></tr>
<tr><td>‘7F48’ {...}</td><td>‘L_{7F48}’</td><td>DOUP</td></tr>
<tr><td>'92' {...}</td><td>‘L_{92}’</td><td>P</td></tr>
<tr><td>'93' {...}</td><td>‘L_{93}’</td><td>Q</td></tr>
<tr><td>'94' {...}</td><td>‘L_{94}’</td><td>Q-1 mod P</td></tr>
<tr><td>'95' {...}</td><td>‘L_{95}’</td><td>dP</td></tr>
<tr><td>'96' {...}</td><td>‘L_{96}’</td><td>dQ</td></tr>
</table>
</td></tr>
<tr><td>L_e field</td><td>Absent</td></tr>
</table>

9.8.3 GENERATE ASSYMETRIC KEY PAIR

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

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO '47' '0000'
L_c field	Length of command data field = ‘05’
Data field	Inter industry template: ‘70’ L_{70} {Tag of the SDO private portion referencing the key pair to generate}
L_e field	Variable

RESPONSE PARAMETER	MEANING
Data field	Absent
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

Example, generation of a RSA key pair:

COMMAND PARAMETER	MEANING
CLA INS P1-P2	ISO '47' '0000'
L_c field	‘05’

<table>
<tr><td>Data field</td><td>
<table>
<tr><th>Tag</th><th>Length</th><th>Value</th></tr>
<tr><td>'70'</td><td>‘03’</td><td>‘BFXXYY’
Private portion with
• XX = Type RSA private key ‘90’
• YY = Identifier ‘01’</td></tr>
</table>
</td></tr>
<tr><td>L_e field</td><td>Absent</td></tr>
</table>

Note:

- The type of the key pair to generate is indicated by the byte XX, which is the class of the SDO of the private portion.
- The identifier of the key pair to generate is indicated by YY. It is the identifier of both portions (private & public)

9.9 Manage security environment commands

9.9.1 MSE RESTORE

This command restores a SSE from the non-volatile memory of the card.

The requested SSE (SEID) is copied into the CSE content.

COMMAND PARAMETER	MEANING
CLA INS P1 P2	ISO ‘22’ ‘F3' Identifier of the SSE (SEID)
L_c field	00
Data field	Absent
L_e field	Absent

COMMAND PARAMETER	MEANING
Data field	No data returned
SW1-SW2	'6A86' - Incorrect P1 (P1 <> ‘F3’) '6A88' - Referenced data not found – No SE found ‘6982’- Security Status not satisfied '6700' - Wrong length

9.9.2 MSE SET

These commands set or replace a CRT in the CSE.

No verification regarding the SDO references is realized. Others parameters are checked against supported values. The order of the data objects in the command data field is define by the CRT templates and every data object shall be present only once.

COMMAND PARAMETER	MEANING
CLA INS P1 P2	ISO ‘22’ refer to CRT template indicated in 10.6.2 refer to CRT template indicated in 10.6.2
L_c field	Input data length
Data field	Control reference template to use
L_e field	Absent

Status words for MSE SET:

SW1-SW2	'6A86' - Incorrect P1 (P1 ≠ ‘x1’) '6A81' - Function not supported (wrong P2 – wrong template) '6A80' - Incorrect parameters in the command data field – template not recognized

Table 38: SW1-SW2 for MSE SET command

10 ANNEX

10.1 Cryptographic Checksum Algorithms (symmetric keys)

The card supports a single MAC algorithm that is:

8 bytes MAC using retail MAC algorithm according to ISO/IEC 9797-1 padding 2 algorithm 3.

To compute a cryptographic checksum according to this algorithm, the incoming data is first padded as shown chapter 10.1.1. The resulting data byte string becomes the input of the triple DES CBC algorithm.

10.1.1 ISO/IEC 9797-1 padding 2

ISO/IEC 9797 mode 2 padding is performed as follows:

1. A mandatory extra byte with the value '80' is appended to the incoming data.
2. The bytes string resulting from the previous step is completed with one or more '00' bytes.

   The number of '00' added bytes should be such that the final string length can be divided by 8 (String Length mod 8 = 0).

Data	Padding	
D	'80'	'00' ….. '00'
◄──── String.length mod 8 = 0 ────►		

Table 39: ISO 9797 padding 2 scheme

10.1.2 ISO 9797-1 algorithm 3 scheme

The algorithm is explained in the following scheme :

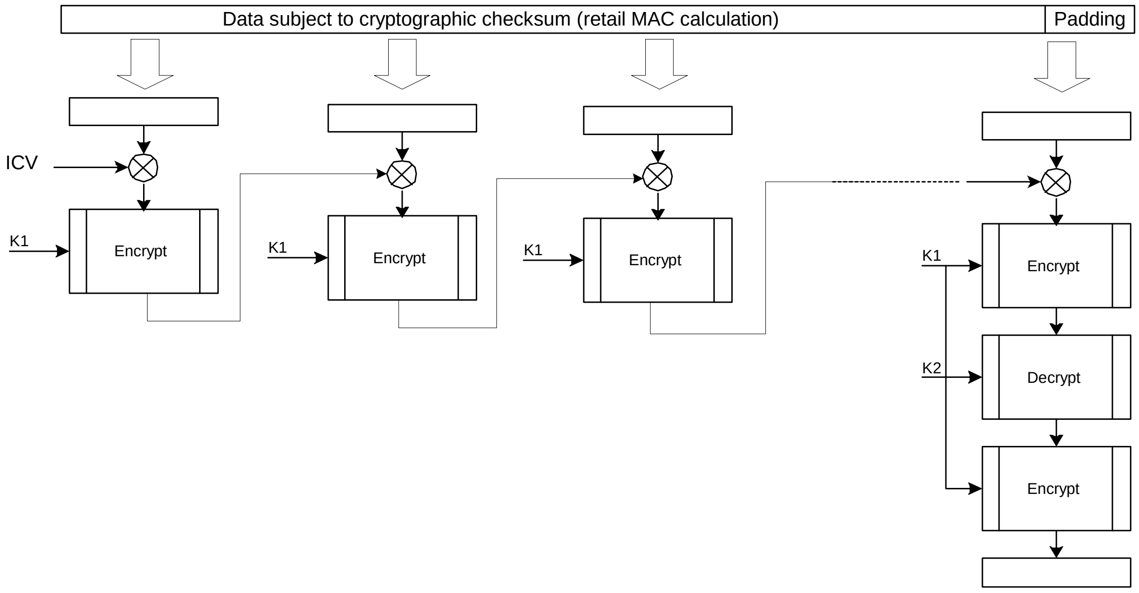

Data subject to cryptographic checksum (retail MAC calculation)
Padding
ICV
K1
Encrypt
K1
Encrypt
K1
Encrypt
K1
Encrypt
K2
Decrypt
Encrypt

Figure 17 - ISO ISO/IEC 9797-1 padding 2 algorithm 3

Each Box “Encrypt” (resp. “Decrypt”) performs a single DES operation in mode encrypt (resp. decrypt)

That Initial Chaining Vector ICV can optionally be specified in the corresponding cryptographic checksum CtrlRT.

10.2 Extended Header List

This mechanism indirectly references data objects to retrieve some of them. It is the concatenation of tag + length pairs with no delimitation that are associated with a byte string made of the concatenated value fields associated with the pairs in the same order.

Incoming data elements indicate the card how to construct the response data field:

- SDO are addressed using an extended header list. The tag '004D' in the GET DATA command identifies its use.
- Extended header lists are used to define a certificate profile during card personalization

An extended header list is made of two parts:

1. The tag-Length pairs concatenation that is as follows:
    - If the tag is primitive (bit 6 = 0)
        - A null length indicates the complete data is required.
    - If the tag is constructed (bit 6 = 1)
        - Length = '80' indicates the full constructed data object is requested.
2. The associated bytes string that can ever be a concatenation of data objects value fields or a concatenation of the full data object TLV structured.

- When returning SDO, the card uses the DO TLV structure.

When calling GET DATA, extended header is used to format incoming data, while outgoing data is the byte string resulting from the concatenation of the requested value fields.

10.3 Data object coding using Simple-TLV and BER-TLV

10.3.1 Introduction

If encoded in TLV, any data field, or concatenation of data fields, is a sequence of data objects. This clause specifies two categories of data objects: SIMPLE-TLV data objects and BER-TLV data objects.

The purpose of this chapter is not to provide training on BER data coding, but to focus on the BER requirements for this applet.

10.3.2 Simple-TLV coding

Each SIMPLE-TLV data object shall consist of two or three consecutive fields:

- A mandatory tag field,
- A mandatory length field,
- A conditional value field.

The coding of these fields is as follows:

- The tag field consists of a single byte encoding a tag number from 1 to 254. The values '00' and 'FF' are invalid for tag fields. If a record is a SIMPLE-TLV data object, then the tag may be used as record identifier.
- The length field consists of one or three consecutive bytes.
  - If the first byte is not set to 'FF', then the length field consists of a single byte encoding a number from zero to 254 and denoted N.
  - If the first byte is set to 'FF', then the length field continues on the subsequent two bytes with any value encoding a number from zero to 65 535 and denoted N.
- If N is zero, there is no value field, i.e., the data object is empty. Otherwise (N > 0), the value field consists of N consecutive bytes.

The following cases may happen when coding and object using Simple-TLV means:

Byte 1	Byte 2	Data
TAG	1 ≤ Lg ≤ 'FE'	-------- Lg bytes --------

Byte 1	Byte 2	Bytes 3 and 4	Data
TAG	FF	0 ≤ length ≤ 65535	-------- Lg bytes --------

Byte 1	Byte 2
TAG	00

10.3.3 BER-TLV coding

The Basic Encoding Rules, abbreviated BER, give one or more ways to represent any value as an octet string.

There are three methods to encode a value under BER, the choice of which depends on the type of value and whether the length of the value is known. The three methods are:

- Primitive, definite-length encoding.
- Constructed, definite-length encoding.
- Constructed, indefinite-length encoding.

Indefinite-length encoding method is not be implemented for this card. Consequently, it is excluded from the rest of the explanations.

In each method, the BER encoding has two or three parts:

- **Identifier bytes**. These identify the class and tag number of the ASN.1 value, and indicate whether the method is primitive or constructed.
- **Length bytes**. These give the number of contents octets.
- **Contents bytes**. For the primitive, definite-length method, these give a concrete representation of the value. For the constructed methods, these give the concatenation of the BER encoding of the components of the value.

10.3.3.1 The primitive and the constructed data representation methods

The primitive data representation method is used to represent single BER-coded information such as an integer, an offset, a length, etc. When several primitive objects are required to represent the data, a constructed BER-TLV data object is used.

- Any BER-TLV data object is denoted:

  {T-L-V}

  with a tag field followed by a length field encoding a number. Depending on whether the number is zero or not, the value field is absent (empty data object) or present.
- Any constructed BER-TLV data object is denoted:

  {T-L- {T1-L1-V1} ... - {Tn-Ln-Vn} }

  with a tag field followed by a length field encoding a number. If the number is not zero, then the value field of the constructed data object, i.e., the template, consists of one or more BER-TLV data objects, each one consisting of a tag field, a length field encoding a number and if the number is not zero, a value field.

10.3.3.2 Encoding methods

10.3.3.2.1 Primitive, definite-length method

The parts of the primitive, definite-length BER encoding method are as follows:

IDENTIFIER BYTES.

There are two forms: low tag number (for tag numbers between 0 and 31) and high tag number (for tag numbers 32 and greater).

- **Low-tag-number form**. One octet. Bits 8 and 7 specify the class (see Table 2), bit 6 has value "0," indicating that the encoding is primitive, and bits 5-1 give the tag number.

Class	b8	b7	b6	bits 5 to 1
Universal	0	0	0 (primitive encoding)	Tag number, 0 ≤ Tag ≤ 31
Application	0	1		
Context-specific	1	0		
Private	1	1		

Table 40 - BER-TLV TAG field low-tag-number format

- **High-tag-number form**. Two or more octets. First octet is as in low-tag-number form, except that bits 5-1 all have value "1". Second and following octets give the tag number, base 128, most significant digit first, with as few digits as possible, and with the bit 8 of each octet except the last set to "1."

High-tag-number form 1st byte coding								
Class	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**
Universal	0	0	0 (primitive encoding)	1	1	1	1	1
Application	0	1						
Context-specific	1	0						
Private	1	1						
High-tag-number form 2nd to nth bytes coding								
Class	**b8**	**Bits 7 to bit 1**						
Not the last byte	1	Tag number base 128 – most significant digit first As few digits as possible (max 2)						
Last byte	0							

Table 41 - BER-TLV TAG field high-tag-number format

LENGTH BYTES.

There are two forms: short (for lengths between 0 and 127), and long definite (for lengths between 0 and 21008-1). For the purpose of this applet, there might be up to three bytes for coding the length. This authorizes values from 0 to 65 535.

- **Short form.** One octet. Bit 8 has value "0" and bits 7-1 give the length.

Length octet short form coding	
Bit 8	**Bits 7 to 1**
0 (short form coding)	0 ≤ Length ≤ 127

- **Long form.** Two to 127 octets. Bit 8 of first octet has value "1" and bits 7-1 give the number of additional length octets. Second and following octets give the length, base 256, most significant digit first.

Length octet long form 1st byte coding	
Bit 8	**Bits 7 to 1**
1 (long form coding)	Number of additional length bytes (max 2)[21]

Length octets long form 2nd to nth bytes coding
Bits 8 to 1
Length base 256

- **ISO 7816 coding.** This is the format that is applicable for this applet.

Number of bytes	First byte	Second byte	**Third byte**	**Length**
1	'00' to '7F'	-	-	0 ≤ N ≤ 127
2	'81'	'00' to 'FF'	-	0 ≤ N ≤ 255
3	'82'	'0000' to 'FFFF'		0 ≤ N ≤ 65 535

Table 42 - ISO 7816 BER-TLV length field coding

CONTENTS BYTES.

These give a concrete representation of the value.

10.3.3.2.2 Constructed, definite-length method

The parts of the constructed, definite-length BER encoding method are as follows:

IDENTIFIER BYTES.

As described in Section 10.3.3.2.1, except that bit 6 has value "1", indicating that the encoding is constructed.

LENGTH BYTES.

As described in Section 10.3.3.2.1.

CONTENTS BYTES.

These give a concrete representation of the value.

10.3.3.2.3 Constructed, indefinite-length method

NOT SUPPORTED

[21] - The standard authorizes up to 127 bytes but the card accepts a maximum of 2 bytes.

10.4 Cryptographic Information Application (normative)

10.4.1 Description

The information in this chapter is for card personalization purpose.

From the applet standpoint, the PKCS#15 DF CIA is a matter of file system. The applet behavior does not rely on the DF CIA content. The DF.CIA attributes shall reflect the rules governing the cryptographic objects of the applet.

This chapter is informative. In case the interoperability requires the cryptographic information application, the implementation of DF.CIA shall conform to the CEN/TS 15480-2 normative clause Annex C, and may enhance it whenever necessary with further attributes according the resolutions of ISO/IEC FPDAM 7816-15 AM2:2007.

For instance by extending the former path definition (ISO/IEC 7816-15:2004 clause A.2.5), the former Access Mode definition (ISO/IEC 7816-15:2004 clause A.2.8), and the former CVC attributes definition (ISO/IEC 7816-15:2004 clause A.7.8)

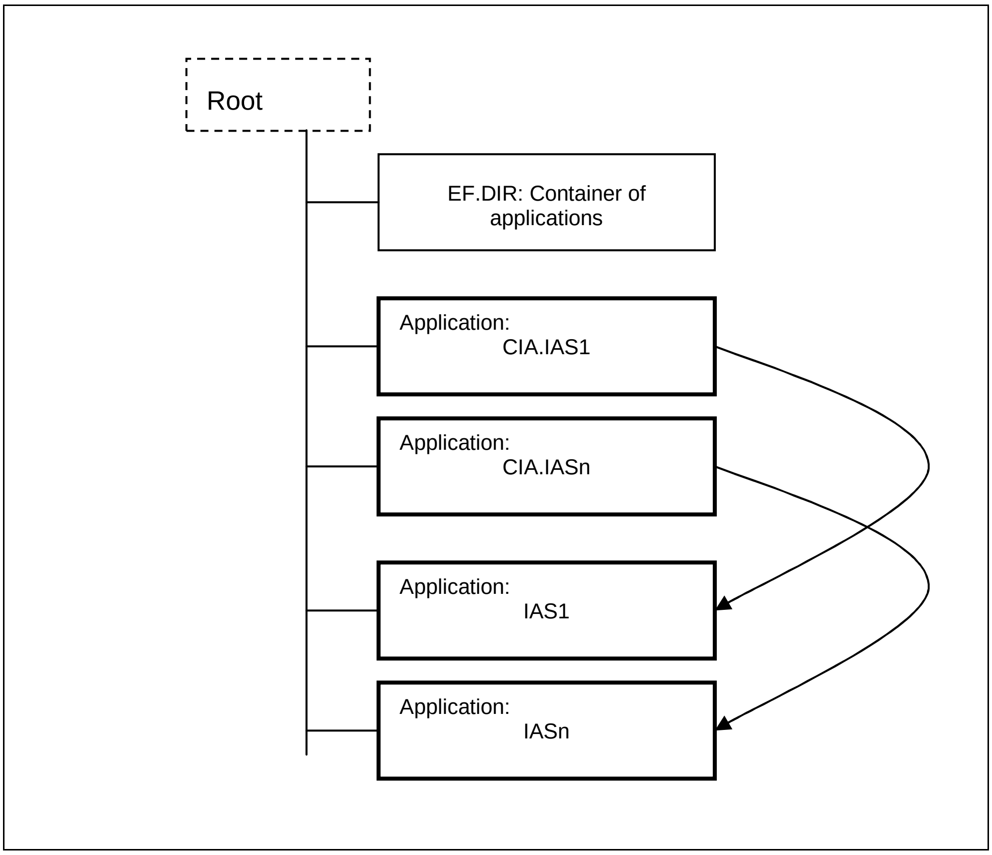
These enhancements apply to CIO within EF.OD, EF.PrKD, EF.PuKD, EF.SKD, EF.AOD and EF.CD.

10.4.1.1 Overview

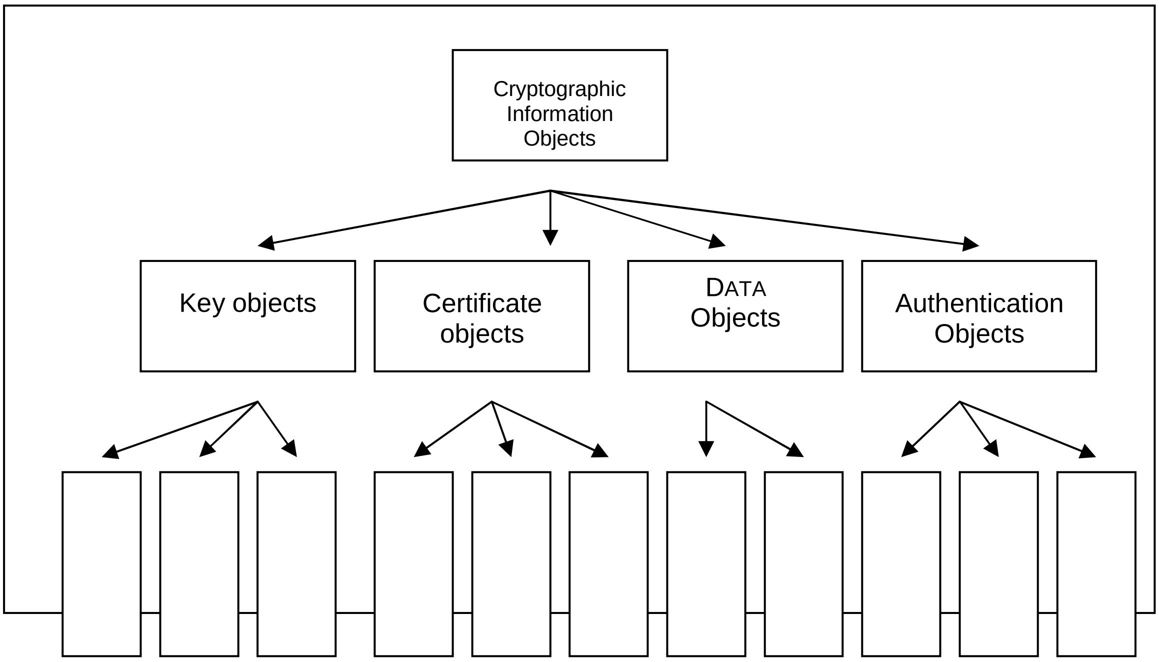
The Cryptographic Information application shall be designed according to [R6], this chapter specifies the use of CIA for the IAS base application. The purpose of such an application is to normalize the description of the cryptographic architecture of an associated application in order to improve interoperability, it describes:

- Cryptographic algorithm of the IAS associated application.
- The authentication objects (e.g. Password).
- The private and public keys present in the IAS associated application.
- The certificates associated to the keys of the IAS associated application.
- The trusted certificates of CA for verification of certificate chain.

For each IAS application presents in the card a CIA exists. The link is insured by the description of the CIA application in the EF.DIR file.



Such a class diagram summarizes the CIA description:



The goal of the CIA is to provide to the terminal all the descriptive files to gain a complete knowledge of the cryptographic objects and their use.

The CIA is simple file architecture whose access rights SHALL be correctly set:

- Always readable for all users.
- The writing authorization in the CIA file architecture shall be granted to special role in the security policy.

10.4.1.2 CIA application identification

The CIA is an ADF that is identified by an AID constructed as follows:

AID.CIA = E8 28 BD 08 0F || AID of the associated IAS application

When starting communicating with the card, the terminal selects the EF.DIR file that lists all card applications. The CIA applications can be identified by reading this file and by identifying the normalized RID prefix for CIA applications as defined in [R6]. After this step the terminal can select each CIA application and read its descriptive structure.

10.4.1.3 EF.DIR

The list of supported applications in the card can be read from this file whose identifier is fixed in [R2] (EFID = '2F00' and SFI = '1E'). It must be immediately placed under the Root level and readable after the Reset of the card. It structure can be described as follows:

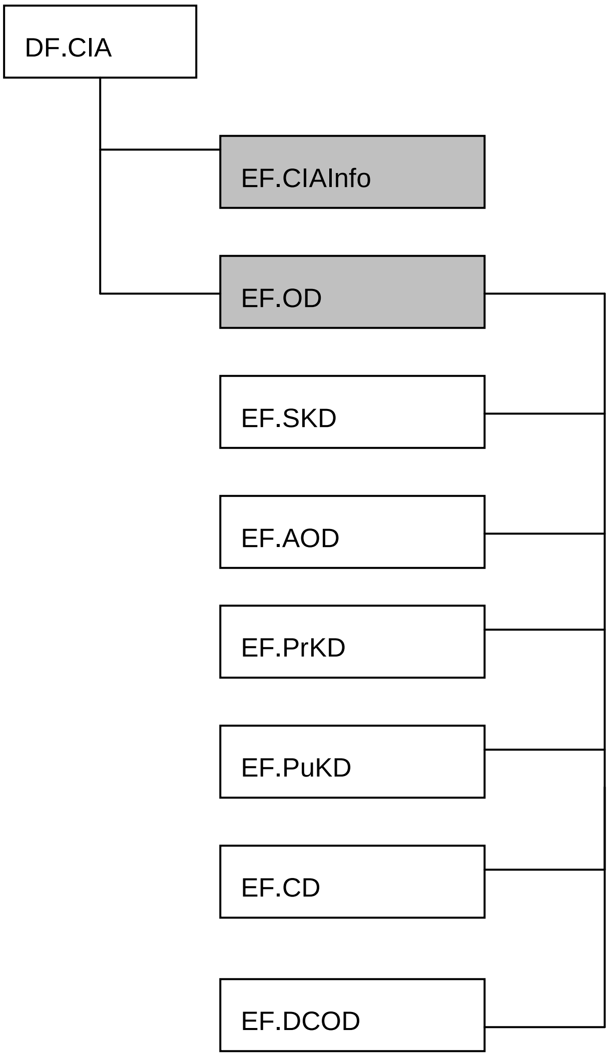
- Set of Application template, each one is composed of:
  - o An application template defined by the tag ‘61’ made of a data set:
    - Tag ‘4F’ Application identifier (AID) Mandatory.
    - Tag ‘50’ A descriptive string of the application (Recommended).
    - Tag ‘73’ Contains references on CIA application dynamic elements.
    - Others Tags can be supported, they are fully described in [R3] or [R2].

10.4.2 Data organization

The CIA application is a simple file structure. The access conditions on files should be:

- READ (ALWAYS)
- UPDATE (PROTECTED by a security mechanism PIN or external authentication depending on the personalization)

The file architecture is the following:



Note: gray files are mandatory, others files are optional

10.4.2.1 EF.CIAInfo

This file is mandatory (EFID = ‘5032’, SFI = ‘12’), it describes:

- VERSION: match the version defined in [R6], current version is 2
- SERIAL NUMBER: correspond to the serial number of the ICC. OPTIONAL
- MANUFACTURERID: give information on the card manufacturer
- LABEL: provides identifying information about the application.
- CARDFLAGS: provides information about the card itself, flags include:
  - If the card is read-only.

  - If there are cryptographic functions that require a user to be authenticated.
  - If the card supports pseudo-random number generation.

- SEINFO: enumerates the Security Environments specified in the associated IAS application.
- SUPPORTED ALGORITHMS: lists all algorithms supported by the card. Every algorithm are described with the following fields:
  - REFERENCE: unique identifier in the CIA application
  - PARAMETERS: parameters for using the algorithm if needed
  - OBJID: represent object identifier of the algorithm (OID)
  - SUPPORTEDOPERATIONS: list the operation that can be performed with this algorithm:

    compute-checksum, compute-signature, verify-checksum, verify-signature, encipher, decipher, hash and derive-key)
  - ALGREF: indicate the associated algorithm identifier

10.4.2.2 EF.OD

The way to build the relative path is defined as follows::

- If the file is in the same DF, EFID of the file is used to locate it
- The file may be located using absolute path from an ADF (indicated by its AID)

This file is mandatory (EFID = ‘5031, SFI = ‘11’), (Refer to [R17] §C.2.2) it describes:

- Files associated to each class of objects supported by the associated application:
  - Reference path to the private keys descriptor file EF.PrKD;
  - Reference path to the secret keys descriptor file EF.SKD.
  - Reference path to the public keys descriptor file EF.PuKD.
  - Reference path to the certificates descriptor file EF.CD.
  - Reference path to the data container descriptor file EF.DCOD.
  - Reference path to the authentications objects descriptor file EF.AOD.

For EF.AOD, EF.PrKD, EF.SKD, and EF.CD, the AccessControlRules within the CommonObjectAttributes may be extended as per ISO/IEC FPDAM 7816-15:2007 AM2 in order to cover the features of SDO access mode bytes:

```
AccessControlRule ::= SEQUENCE {
        accessMode              AccessMode,
        securityCondition       SecurityCondition,
... -- For future extensions
}

AccessMode ::= BIT STRING {
        read (0),
        update (1),
        execute (2),
        delete (3),
        attribute (4),
        pso_cds (5),
        pso_verif (6),
        pso_dec (7),
        pso_enc (8),
        int_auth (9),
        ext_auth (10)
}
```

10.4.2.3 EF.AOD

This file describes the entire authentication objects; an authentication object is characterized by its attributes.

- PIN authentication object is defined through the following fields (Refer to [R17] §C.2.3) :
    - Label: a descriptive string to identify the PIN
    - Access control rules
    - PUK ID: identifier of the unblocking code(s)
    - PIN identifier: provides a unique identifier in the CIA descriptive structure
    - Flags: describes whether the Pin is global or local
    - Type of password: describes the type of the password
    - The minimum length of the password if needed
    - The reference of the password used to identify it in authentication commands (e.g. VERIFY)
    - The path where this Pin is applicable (this path is relative to the IAS application and may be extended according the Path definition below)
- Card Verifiable Certificate – **PK scheme** – (used in device authentication or in role authentication) is defined through the following fields (Refer to [R17] §C.2.3.4):
    - Label: a descriptive string to identify the certificate authentication
    - AccessControlRules
    - CVC identifier: provides a unique identifier in the CIA descriptive structure in order to reference it in other CIA object
    - Certificate attribute which is only composed of the CHA field that must be contained in the CVC to fulfill the authentication (may be extended according CVCertificateAttributes definition below)

CVCertificateAttributes as per ISO/IEC FPDAM 7816-15:2007 AM2:

```
CVCertificateAttributes ::= SEQUENCE{
        value                           ObjectValue {CIO-OPAQUE.&Type},
        certificationAuthorityReference OCTET STRING OPTIONAL
... – For future extensions,
}
```

- External Authentication – **SK scheme** – (used in device authentication or in role authentication) is defined through the following fields (Refer to [R17] §C.2.3.5):
  - Label: a descriptive string to identify the certificate authentication
  - AccessControlRules
  - Identifier: provides a unique identifier in the CIA descriptive structure witch allows making the link to the private key description in the EF.SKD file

Extended Path as per ISO/IEC FPDAM 7816-15:2007 AM2:

```
Path ::= SEQUENCE {
        CHOICE {
                efidOrPath OCTET STRING,
                TagRef [0] SEQUENCE {
                        tag OCTET STRING,
                        efidOrPath OCTET STRING OPTIONAL
                 }
                AppFileRef [1] SEQUENCE {
                        aid [APPLICATION 15] OCTET STRING,
                        efidOrpath OCTET STRING
                }
                AppTagRef [2] SEQUENCE {
                        aid [APPLICATION 15] OCTET STRING,
                        tag OCTET STRING,
                        efidOrPath OCTET STRING OPTIONAL
                }
}
Index INTEGER (0 .. cia-ub-index) OPTIONAL,
Length [0] INTEGER (0 .. cia-ub-index) OPTIONAL
} ( WITH COMPONENTS {..., index PRESENT, length PRESENT}|
  WITH COMPONENTS {..., index ABSENT, length ABSENT} )
```

10.4.2.4 EF.PrKD

This file is optional and describes all the private key objects; a private key object is characterized by the following attributes (Refer to [R17] §C.2.4):

- LABEL: a descriptive string of the private key.

- FLAGS: indicate that the key is private and whether an authentication is required in order to make use of the key.
- USER CONSENT: present only on signature keys indicates the number of authorized use (this number is limited to 1 in the present IAS specification) of the private key before that the authentication object protecting this private key is reset.
- IDENTIFIER: a unique identifier in the CIA info in order to associate with the same ID the public key in the EF.PuK file.

- ACCESSCONTROLRULES: if the key is marked private indicates the access mode and associated security features (e.g. identifier in the CIA of the protection PIN), each rules is composed of an access mode followed by a security condition
  - AUTHORIZED ACCESS MODE: execute, read, update
  - SECURITY CONDITION: specify the identifier of the authentication object
- USAGE: authorized values in IAS application are
  - Sign: signature
  - KeyDecipher: key decryption
  - NonRepudiation: signature
- NATIVE: indicates if the associated algorithms to the key is implemented in the hardware (always TRUE in IAS application)
- ACCESS Flags: indicates that the key is sensitive and wether the key has been generated on board or not.
- KEY REFErence: external identifier to use in cyrpotgraphic commands
- ALGORITHM REFERENCE: identify the algorithm that can be used with the key, this reference is the same value than in the SupportedAlgorithms.Reference field.
- PATH: identify the path to the private key (relative to the IAS application).
- THE MODULUS LENGTH: as the IAS application support only RSA private keys this field specify the length in bits of the associated modulus key (n)

10.4.2.5 EF.SKD

This file is optional. It describes all the secret key objects; a secret key object is characterized by the following attributes (Refer to [R17] §C.2.5):

- LABEL: a descriptive string of the secret key.
- AccessControlRules
- FLAGS: indicate that the key is private and is never extractable
- IDENTIFIER: a unique identifier in the CIA info structure in order to reference it in other CIA object
- KEY REFERENCE: external identifier to use in cryptographic commands
- ALGORITHM REFERENCE: identify the algorithm that can be used with the key, this reference is the same value than in the SupportedAlgorithms.Reference field.
- PATH: identify the path to the private key (relative to the IAS application). May be an extended path

10.4.2.6 EF.PuKD

This file is optional and describes all the public key objects (public key of a card key pair, or public key of a CA to verify CVC) , a public key object is characterized by attributes (Refer to [R17] §C.2.6):

- LABEL: a descriptive string of the public key.

- AccessControlRules
- IDENTIFIER: identify uniquely the public key in the CIA structure it must correspond to the identifier of the private key if present in the card and to the certificate if it is present
- USAGE: authorized values in IAS application are
  - verify
  - verifyRecover
- NATIVE: indicates if the associated algorithms to the key is implemented in the hardware (always TRUE in IAS application)
- Key Reference: external identifier to use in cryptographic commands
- ALGORITHM REFERENCE: identify the algorithm that can be used with the key, this reference is the same value than in the SupportedAlgorithms.Reference field.
- PATH: identify the path to the public key (relative to the IAS application).may be an extended path.
- THE MODULUS LENGTH: as the IAS application support only RSA private keys this field specify the length in bits of the associated modulus key (n)

10.4.2.7 EF.CD

This file is optional and describes all the certificate object, a certificate key object is characterized by attributes (Refer to [R17] §C.2.7):

- LABEL: a descriptive string of the certificate.
- AccessControlRules
- IDENTIFIER: unique identifier in the CIA structure, must correspond to the identifier of the associated private key and to the public keys if they are present in the card
- AUTHORITY: indicate whether it is a certificate of a CA or not
- PATH: identify the path to certificate (relative to the IAS application). May be an extended path.

10.4.2.8 EF.DCOD

This file is optional and describes all the data container object (e.g. used to specify the way to retrieve the serial number), data container object is characterized by attributes:

- Serial Number description:
- LABEL: a descriptive string of the serial number.
- AccessControlRules
- PATH: indicates the value of the Data Object used in the IAS context to retrieve the SerialNumber (may be an extended path)

Others data objects can be added to this file and allow describing the way to retrieve those objects.

Note: Although the [IAS ECC] application doesn't specify the use of display message this file support the way to retrieve and to protect it (see ISO7816-15 for others details).

10.4.3 Optional, Conditional and Mandatory files

Few applications files are mandatory, conditional or optional for the **[IAS ECC]** and for interoperability purposes. The list of the files is the following:

- EF.DIR, indicates a list of applications supported by the card (EFID='2F00', SFI='1E')
- EF.ATR, indicates operating characteristics of the card (EFID='2F01', SFI='1D')
- EF.SN.ICC, contains the serial number SN.ICC of the ICC (EFID='D003', SFI='1C')
- EF.DH, contains the public key parameters for the Diffie-Hellman key exchange protocol (EFID='D004', SFI='1B')

More explanations for theses files, concerning the content or presence, could be found in [R15] (see §15.1).

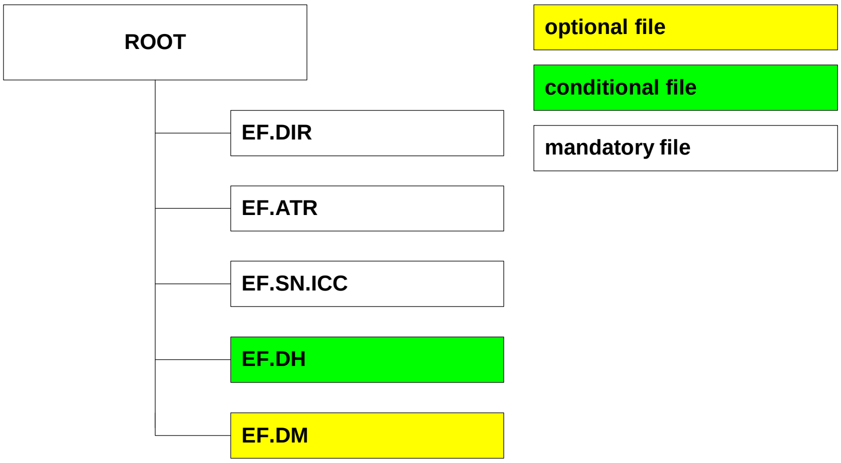


Figure 18: Example of file structure of an [IAS ECC] application

10.4.3.1 EF.SN.ICC (Smart card serial number)

[IAS ECC] card serial number is managed like PAN banking card is (see [R3], tag '5A' for PAN references).

[R17] describes the PAN structure:

	ISO/IEC 7812 PAN definition				
Length	**6 digits**			**12 digits**	**1 digits**
Data	IIN Issuer Identification Number			Smart card unique identifier	Luhn calculation
	MII	Country	Issuer identifier		

- IIN: 4 first digits of IIN, identify the issuer. It is structured as follows:

- MII (1digit BCD): industry identification. Authorized value are:

1 digit	Meaning
"0"	ISO / TC68 and over industries
"1"	Airline company
"2"	Airline company
"3"	Travel and leisure
"4"	Bank and finance
"5"	Bank and finance
"6"	Trade and banks
"7"	Oil
"8"	Telecommunications
"9"	National authority choice

As the current specification applies for national government, the chosen value is '9'

- Issuer identifier provided by the national authority. Structured as follows:
  - Country identifier according to ISO/IEC-3166 standard over 3 BCD digits (France = '250')
  - Issuer value coded over 2 BCD digits. Provided and managed by the national authority.

- Smart card unique identifier:
  - Serial number: allows uniquely identifying a smart card over all cards having the same IIN.
- Luhn calculation: CRC like calculation allows verifying the PAN integrity. Calculation result is 1 digit. Luhn calculation is described in ISO/IEC 7812-1 standard.

In order to get a correct length in byte for the serial number, padding bits set to ‘1111’ shall be added at the right side

Serial number = PAN || ‘F’

This serial number is stored in the file within a Data object ‘5A’:

Tag	L	Value
'5A'	Var	Serial number

10.4.3.2 EF.DH

This file is conditional and is present if the Device authentication with privacy protection is used. It contains the Domain parameters to use for the first step of the protocol.

This file contains a BER TLV data object as follows:

Tag	Length	Tag	Length	Value
'7F49'	L_{7F49}			
		'7F49'	L_{7F49}	Domain parameter 1
		--	--	-----
		'7F49'	L_{7F49}	Domain parameter n

This data object is a constructed BER TLV data object containing a data object for each domain parameter the [IAC ECC] card contains.

The data object for each domain parameter is built as follows:

Tag	Length	Tag	Length	Value
'7F49'	L_{7F49}			
		'81'	L_{81}	Prime p
		'82'	L_{82}	Order q
		'83'	L_{83}	Generator g

10.5 Summary of the available services

All the cryptographic services available in [IAS ECC] 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'
Asymmetric authentication for secure channel opening.	GetDataKicc up to Internal Authenticate PKDH	PKDH session key negotiation with privacy protection. Key derivation function using SHA-1	'1B'
		PKDH session key negotiation with privacy protection. Key derivation function using SHA-256	'9B'
Client/server authentication	Internal Authenticate Client / Server	PKCS#1 v1.5 RSA signature calculation – No internal hash computation.	'02'
Digital signature computation	PSO Hash followed by PSO Compute Digital Signature	RSA PKCS#1v1.5 – SHA-1.signature computation	'12'
		RSA PKCS#1 v1.5 – SHA-256.signature computation	'42'
Certification verification (CVC)	PSO Verify Certificate	RSA 9796-2 scheme 1 with SHA-1	'11'
		RSA 9796-2 scheme 1 with SHA-256	'41'

Ciphering key decryption	PSO Decipher	RSA PKCS#1 v1.5 message decryption	‘1A’
Role authentication (symmetric or asymmetric)	GET CHALLENGE / External Authenticate for role authentication	RSA 9796-2 scheme 1 with SHA-1	‘1E’
		RSA 9796-2 scheme 1 with SHA-256	‘9E’
		3DES CBC EDE 128 bits (encipherment) + Retail MAC	‘1C’
Hash calculation within the digital signature sequence	PSO Hash (prior to PSO Compute Digital Signature)	SHA-1	'10'
		SHA-256	‘40’

Table 47 - available services and algorithm identifiers

10.6 Supported CRT format

This chapter aims to present all the CRT templates supported by the [IAS ECC] card depending on whether it aims to be loaded in the Current Security Environment (CSE) or in a Static Security Environment (SSE).

Templates loaded in the CSE specify the card security context prior to any authentication or cryptographic operation, and is stored in the volatile memory of the card.

Templates loaded in the SSE specify the security policy for a SDO/File referencing this SSE (in SCB), and stored in the persistent memory of the card.

However, some of the SSE cannot be loaded in the Current Security Environment since they are used to specify the security policy only (for instance with multiple references, etc…). In such a case, the access right on the MSE RESTORE command is set to never preventing from restoring incoherent CRT template in the CSE.

10.6.1 CRT Templates dedicated to Security Policy (SSE)

The following templates are set inside a SE stored in persistent memory to protect a SDO or a File. The SE containing the template shall be referenced in the SCB of the SDO/File to protect. When accessing the SDO/File the verification is done upon the reference and the algorithm provided in the SSE.

Multiple referencing is applicable to SSE only and to all of them except for CT and CCT with implicit Key reference and algorithm value, since it is only relevant for security policy definition (OR condition between all references). As a matter of fact this has no sense in the card security context (CSE) because it leads to uncertainties between references. It is obvious that all the keys referenced in case of multiple referencing, shall have the same algorithm since there is only one algorithm for each SSE. For more information see §3.5.6.2.

10.6.1.1 Authentication template (CRT AT)

User authentication using Password or PIN:

Tag	Length	Value		
‘A4’	L			
		Tag	**L**	**Value**
		‘80’	‘01’	'00' ≡ algorithm identifier
		‘83’	‘01’	PIN reference
		‘95’	‘01’	'08' ≡ UQB for User Authentication

External asymmetric authentication with role validation to grant access rights:

Tag	Length	Value		
‘A4’	L			
		Tag	**L**	**Value**
		‘80’	‘01’	'1E' ≡ algorithm identifier for RSA PKCS#1 SHA-1 '9E' ≡ algorithm identifier for RSA PKCS#1 SHA-256

		'5F4C'	L	CHA indicating the privilege (their might be several CHA data object in this CRT)
		'8A'	'01'	CHA comparison algorithm (present only once)
		'95'	'01'	'80' ≡ UQB for external authentication

Internal asymmetric authentication during a PK-DH scheme for mutual authentication:

Tag	Length	Value		
'A4'	L			
		Tag	**L**	**Value**
		'80'	'01'	'1B' ≡ DH asymmetric authentication algorithm (privacy) with SHA-1 '9B' ≡ DH asymmetric authentication algorithm (privacy) with SHA-256
		'84'	'01'	Private key reference
		'95'	'01'	'40' ≡ UQB for internal authentication

IFD/ICC mutual authentication – SK scheme:

Tag	Length	Value		
'A4'	L			
		Tag	**L**	**Value**
		'80'	'01'	'0C' ≡ symmetric mutual authentication algorithm with SHA-1 '8C' ≡ symmetric mutual authentication algorithm with SHA-256
		'83'	'01'	Symmetric key set reference
		'95'	'01'	'C0' ≡ UQB for external mutual authentication

Symmetric role authentication:

Tag	Length	Value		
'A4'	L			
		Tag	**L**	**Value**
		'80'	'01'	'1C' ≡ Role authentication algorithm
		'83'	'01'	Symmetric key set reference
		'95'	'01'	'80' ≡ UQB for external authentication

10.6.1.2 Confidentiality (CRT CT) and Cryptographic Checksum templates (CRT CCT)

Template for confidentiality and Cryptographic Checksum convey the exact same data field content. However, one is identified from another using the TAG value:

- TAG = **0xB8** for Confidentiality (**CRT CT**)
- TAG = **0xB4** for Cryptographic Checksum (**CRT CCT**)

Template for secure messaging data in Confidentiality using implicit references (current SM session):

- No reference on mutual authentication mechanism.
- No reference on mutual authentication key used.

Tag	Length	Value		
‘B8’/‘B4’	L			
		Tag	**L**	**Value**
		‘80’	‘01’	'00' ≡ Secure messaging algorithm (implicit – issued from authentication algorithm)
		‘83’	‘01’	'00' ≡ Session key reference (implicit)
		‘95’	‘01’	'30' ≡ UQB for secure messaging on incoming and outgoing data

Template for secure messaging in Confidentiality using SK scheme:

Tag	Length	Value		
‘B8’/‘B4’	L			
		Tag	**L**	**Value**
		‘80’	‘01’	'0C' ≡ IFD/ICC mutual authentication algorithm reference – SK scheme – SHA-1. '8C' ≡ IFD/ICC mutual authentication algorithm reference – SK scheme – SHA-256.
		‘83’	‘01’	Mutual authentication symmetric key set reference
		‘95’	‘01’	'30' ≡ UQB for secure messaging on incoming and outgoing data

Template for secure messaging data Confidentiality using PK-DH scheme:

Tag	Length	Value		
‘B8’/‘B4’	L			
		Tag	**L**	**Value**
		‘80’	‘01’	'1B' ≡ IFD/ICC mutual authentication algorithm reference – PK-DH scheme – SHA-1. '9B' ≡ IFD/ICC mutual authentication algorithm reference – PK-DH scheme – SHA-256.
		‘84’	‘01’	Mutual authentication asymmetric key (private portion) reference
		‘95’	‘01’	'30' ≡ UQB for secure messaging on incoming and outgoing data

10.6.2 CRT Templates dedicated to Card Security Context (CSE)

10.6.2.1 Authentication template (CRT AT)

External asymmetric authentication link with a role to select a key in the current SE:

P1	P2	Length	Value		
‘81’	‘A4’	L			
			Tag	**L**	**Value**
			‘80’	‘01’	'1E' ≡ algorithm identifier for RSA 9796-2 SHA-1 '9E' ≡ algorithm identifier for RSA 9796-2 SHA-256
			‘83’	‘0C’	CHR ≡ public key reference

Internal asymmetric authentication during a PK-DH scheme for mutual authentication:

P1	P2	Length	Value		
‘41’	‘A4’	L			
			Tag	**L**	**Value**
			‘80’	‘01’	'1B' ≡ DH asymmetric authentication algorithm (privacy) with SHA-1 '9B' ≡ DH asymmetric authentication algorithm (privacy) with SHA-256
			‘84’	‘01’	Private key reference

External asymmetric authentication during a PK-DH scheme for mutual authentication:

P1	P2	Length	Value		
‘81’	‘A4’	L			
			Tag	**L**	**Value**
			‘83’	‘0C’	CHR ≡ public key reference

IFD/ICC mutual authentication – SK scheme:

P1	P2	Length	Value		
‘C1’	‘A4’	L			
			Tag	**L**	**Value**
			‘80’	‘01’	'0C' ≡ symmetric mutual authentication algorithm with SHA-1 '8C' ≡ symmetric mutual authentication algorithm with SHA-256
			‘83’	‘01’	Symmetric key set reference

Symmetric role authentication:

P1	P2	Length	Value
‘81’	‘A4’	L	

			Tag	**L**	**Value**
			‘80’	‘01’	'1C' ≡ Role authentication algorithm
			‘83’	‘01’	Symmetric key set reference

Client/Server authentication:

P1	**P2**	**Length**	**Value**		
‘41’	‘A4’	L			
			Tag	**L**	**Value**
			‘80’	‘01’	'02' ≡ algorithm identifier for IFC/ICC authentication RSA PKCS#1 -SHA-1 with not data formatting
			‘84’	‘01’	Asymmetric private key set reference

10.6.2.2 Confidentiality template (CRT CT)

Message decryption (e-mail ciphering key decryption):

P1	**P2**	**Length**	**Value**		
‘41’	‘B8’	L			
			Tag	**L**	**Value**
			‘80’	‘01’	'1A' ≡ Data decryption algorithm reference – RSA PKCS#1 V2.1.
			’84’	‘01’	Mutual authentication asymmetric key (private portion) reference

10.6.2.3 Digital signature template (CRT DST)

Data signature:

P1	**P2**	**Length**	**Value**		
‘41’	‘B6’	L			
			Tag	**L**	**Value**
			‘80’	‘01’	'12' ≡ algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-1 '42' ≡ algorithm identifier for signature using RSA PKCS#1 v1.5 - SHA-256
			‘84’	‘01’	Asymmetric key (private portion) reference

Certificate verification:

P1	**P2**	**Length**	**Value**		
‘81’	‘B6’	L			
			Tag	**L**	**Value**

			'80'	'01'	'11' ≡ algorithm identifier for signature using ISO 9796-2 scheme 1 – SHA-1 '41' ≡ algorithm identifier for signature using ISO 9796-2 scheme 1 – SHA-256
			'83'	'01'	Asymmetric key (public portion) reference

10.6.2.4 Hash template (CRT HT)

P1	**P2**	**Length**	**Value**		
'41'	'AA'	L			
			Tag	**L**	**Value**
			'80'	'01'	'10' ≡ algorithm identifier for SHA-1 '40' ≡ algorithm identifier for SHA-256

10.6.2.5 Key agreement template (CRT KAT)

P1	**P2**	**Length**	**Value**		
'41'	'A6'	L			
			Tag	**L**	**Value**
			'80'	'01'	'1B' ≡ algorithm identifier for key negociation with KDF based on SHA-1 '9B' ≡ algorithm identifier for key negociation with KDF based on SHA-256
			'83'	'01'	DH asymmetric key public portion reference
			'91'	Var.	Asymmetric session key public portion value