C.I.E. – Carta di Identità Elettronica

Functional Specification

Version 2.1

1. Revision History

Version	Description	Issuing Date
2.0	**First Issue**	**2008/02/04**
2.1	**Updated #8. - Answer To Reset (ATR)** **Typos on page 23, 54, 61, 73.**	**2011/07/26**

Table of Contents

2. Introduction

This document is the Functional Specification of the Italian Electronic Identification Card "Carta di Identità Elettronica" (CIE). It describes how the card works, seen from its external interface, without dealing with the implementation details.

The behaviour described hereby is intended as mandatory. A CIE compliant card has to comply entirely with the indication contained in this document. The document assumes then the value of a reference.

Anyway, this does not forbid a card implementing this specification to have additional features, provided that they do not modify in any way the protocols described. In this sense, the specification has to be meant as the description of the minimum functionalities demanded to the card.

The ISO 7816 norms are the basis of this specification, and have to be taken into account when reading this document. Data structures, commands and codings described here proceed from the standards. This specification aims to clear any possible point left open in the standards.

The reader will not find then a copy of the standard themselves, but a consistent description of the elements taken from the standards (data structures, command, security concepts).

If any of the information in this specification differs from the quoted standards, the rules hereby contained shall take precedence. This applies also to possible future standard issued on the subject.

3. Document Scope

The document describes:

- ATR and protocols
- Data Types and Structures
- Security Concepts
- APDU commands

There are no implementation specific elements, or elements that demand a specific implementation of a feature, exception taken when the implementation may have a security impact.

The document is intended as technology neutral, does not impose a specific technology, and is open to the use of proprietary platforms, JavaCard platforms, or Multos platforms.

Each card manufacturer has been left free to implement his own process up to the pre-personalization phase of the card. Subsequent phases are disciplined by this specification.

4. Reference Documents

1. ISO/IEC 7816-3 second edition: Signal and transmission protocols
2. ISO/IEC 7816-4 Interindustry commands for interchange
3. ISO/IEC 7816-5 Numbering System and registration procedure for application identifiers
4. ISO/IEC 7816-8 Security related interindustry commands
5. ISO/IEC 7816-9 Additional interindustry commands and security attributes
6. Carta d' Identità Elettronica, Specifiche dei Comandi del Sistema Operativo (APDU), version 1.0 (January 11, 2000)
7. AIPA CNS Working Group May 23, 2002 Meeting Report
8. D.M. “Regole Tecniche sulla C.I.E.” pubblicato sulla Gazzetta Ufficiale n°261 del 9/11/2007

5. Definitions and acronyms

5.1 Definitions

Asymmetric Cryptographic Technique - A cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformations have the property that, given the public transformation, it is computationally infeasible to derive the private transformation.

Authentication data is information used to verify the claimed identity of a user.

Authorized user means a user who may perform an operation.

Card session - the interval of time between two card reset

Certificate means an electronic attestation, which links the SVD to a person and confirms the identity of that person.

Certification Authority - Trusted third party that establishes a proof that links a public key and other relevant information to its owner.

Data Integrity - The property that data has not been altered or destroyed in an unauthorized manner

Decipherment - The reversal of a corresponding encipherment

Digital Signature - An asymmetric cryptographic transformation of data that allows the recipient of the data to prove the origin and integrity of the data, and protect the sender and the recipient of the data against forgery by third parties, and the sender against forgery by the recipient.

Hash Function - A function that maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

- It is computationally infeasible to find for a given output an input, which maps to this output.
- It is computationally infeasible to find for a given input a second input that maps to the same output.

Additionally, if the hash function is required to be collision-resistant, it must also satisfy the following property:

- It is computationally infeasible to find any two distinct inputs that map to the same output.

Integrated Circuit(s) Card - A card into which one or more integrated circuits are inserted to perform processing and memory functions.

Key - A sequence of symbols that controls the operation of a cryptographic transformation.

Private Key - That key of an entity's asymmetric key pair that should only be used by that entity. In the case of a digital signature scheme, the private key defines the signature function.

Public Key - That key of an entity's asymmetric key pair that can be made public. In the case of a digital signature scheme, the public key defines the verification function.

Public Key Certificate - The public key information of an entity signed by the certification authority and thereby rendered unforgeable.

Symmetric Cryptographic Technique - A cryptographic technique that uses the same secret key for both the originators and recipient transformation. Without knowledge of the secret key, it is computationally infeasible to compute either the originator's or the recipient's transformation.

Terminal - The device used in conjunction with the ICC at the point of transaction to perform a financial transaction. It incorporates the interface device and may also include other components and interfaces such as host communications.

Warm Reset - The reset that occurs when the reset (RST) signal is applied to the ICC while the clock (CLK) and supply voltage (VCC) lines are maintained in their active state.

5.2 Acronyms

AC	Access Conditions
AID	Application Identifier
APDU	Application Protocol Data Unit
ASN	Abstract Syntax Notation
ATR	Answer-To-Reset
BER	Basic Encoding Rules
BS	Base Security
BSO	Base Security Object
CC	Common Criteria Version 2.1
CGA	Certification Generation Application
CIE	Carta d'Identità Elettronica
CNS	Carta Nazionale Servizi
CSE	Current Security Environment
DES	Data Encryption Standard
DF	Directory File
DIR	Directory
DS	Digital Signature
EAL	Evaluation Assurance Level
EF	Elementary File
ETU	Elementary Time Unit
FCI	File Control Information
FID	File ID
HI	Human Interface
HW	Hardware
ICC	Integrated Circuit Card
ID	Card Identity Card
I/O	Input/Output
IT	Information Technology

lsb	Last Significant Bit (b0)
LSB	Last Significant Byte
MAC	Message Authentication Code
MF	Master File
msb	Most Significant Bit (b7)
MSB	Most Significant Byte
MSE	Manage Security Environment (command)
MTSC	Manufacturer Transport Secure Code
OS	Operating System
OCI	Object Control Information
PIN	Personal Identification Number
PP	Protection Profile
PPSC	Pre Personalization Secure Code
PSO	Perform Security Operation (command)
RFU	Reserved for Future Use
SCD	Signature-Creation Data
SD	Signatory's data
SDO	Signed Data Object
SE	Security Environment
SECI	Security Environment Control Information
SEO	Security Environment Object
SF	Security Function
SFI	Short File Identifier
SHA	Secure Hash Algorithm
SFP	Security Function Policy
SM	Secure Messaging
SOF	Strength of Function
SSCD	Secure Signature-Creation Device
ST	Security Target
SVD	Signature-Verification Data
TLV	Tag Length Value

5.3 Document conventions

The default base is hexadecimal, with Big Endian convention. This means that in the string ‘AB CD’, the byte with value ABh is the MSB.

Individual bit identification in a byte ranges from 0 to 7, bit 7 being the msb.

x || y Strings concatenation: the string x is concatenated on the right with the string y.

x (lower case or upper case x): any bit value, any group of bit value. Examples:

X

- 0000 000x means 0000 0001 or 0000 0000
- Axh can be any byte value having the most significant nibble equal to A
- xxh can be any byte value

h signals the hexadecimal notation

by Bit y. Example:

- b4 means 'bit 4'

x.y The bit y of the byte X. Example:

- P1.7 means bit 7 of the byte named P1

6. Electrical Parameters

For all electrical signal shapes and ranges the rules indicated in [1] apply.

The value for Vcc during the operational phase is up to 5 Volts.

The card SHALL NOT require a Vpp.

7. Communication Protocol

The card has to support the protocol T=1. It may support also T=0, but protocol T=1 has to be used during the use phase.

The standard ISO PPS procedure described in [1] applies, with T=1 as first proposed protocol.

Supported communication speeds are at least:

bps with CLK = 3.5712MHz	F	D	PPS1	ETU duration (CLK Cycles)
9600 (default)	372	1	01h or 11h	372
115200	372	12	08h or 18h	31

Table 1: supported communication speeds

The maximum communication speed (115200bps @ 3.5712MHz) SHOULD be used during the use phase to allow optimal performances.

8. Answer To Reset (ATR)

The Answer To Reset byte string returns information about the communication protocol, the card and application type, the life cycle status, etc. and is the first step in the PPS protocol.

While some of the bytes are defined by ISO, as they are used by the communication protocol, there is the possibility to define the value of other bytes, called Historical Bytes. Please refer to [1] for the meaning of the single ATR characters.

To avoid possible interoperability problems, only the historical bytes are mandated, while the value (and, if applicable, the presence) of all the interface characters is not fixed in this specification.

Nevertheless, Initial Character, Format Character and Interface Characters are to be interpreted as indicated in [1], to allow a correct parsing of the byte string.

The Historical Bytes are in proprietary format, and convey information about the application. All application parameters (eg. card profile, etc) are defined in this specification, and are therefore implicitly known, and do not need to be given in the ATR.

H8-H13 of historical bytes are used to represent the string "**ITID**" followed by the APDU specification version (1 byte) and the File System Specification version (1 byte).

Value	Symbol	M	Description
3Bh	TS	Y	Initial Character: Direct convention (Z=1 and lsb first)
1xxx1111	T0	Y	Format Byte. Indicates the presence of interface characters and the number of historical bytes: - TD1 has to be present - 15 Historical bytes
XXh	TA1 TB1 TC1	N	Global Interface Characters. If present, they must be interpreted according to [1]. If present, TB1 MUST be set to 00 (no Vpp is needed)
1xxx0001	TD1	Y	TD2 present. T=1 is the first offered protocol
XXh	TA2 TB2 TC2	N	Global And Specific Interface Characters. If present, they must be interpreted according to [1]
31h	TD2	Y	TA3 present TB3 present T=1 is the first offered protocol
10h to FEh	TA3	Y	IFSI: Initial IFSC
Hi nibble: 0h to 9h Low nibble: 0h to Fh	TB3	Y	BWI, CWI
00h	H1	Y	Status information shall be present at the end of the historical bytes not as TLV object.
6Bh	H2	Y	11 bytes of Pre-Issuing Data
XXh	H3	Y	Pre-Issuing data byte 1: ICM – IC manufacturer, coded according to [1]
0xxx xxxx	H4	Y	Pre-Issuing data byte 2:

			ICT – mask manufacturer, coded according to Table 3
XXh XXh	H5/H6	Y	Pre-Issuing data byte 3/4: Operating System version, assigned by chip vendor.
02h	H7	Y	Pre-Issuing data byte 5: DD1: ATR coding version: Second version
49h	H8	Y	Pre-Issuing data byte 6: DD2: 'I'
54h	H9	Y	Pre-Issuing data byte 7: DD3: 'T'
49h	H10	Y	Pre-Issuing data byte 8: DD4: 'I'
44h	H11	Y	Pre-Issuing data byte 9: DD5: 'D'
20h	H12	Y	Pre-Issuing data byte 10: DD6: APDU specification version (2.0)
20h	H13	Y	Pre-Issuing data byte 11: DD7: C.I.E. File System version (2.0)
31h	H14	Y	1 byte of Card Services Data
80h	H15	Y	Direct Application Selection supported No Data Object Available
XXh	TCK	Y	Check character as specified in [1]

Table 2: CIE ATR specification

Value	ICT Mask Manufactuer
01	Kaitech
02	Gemplus
03	Ghirlanda
04	Giesecke & Devrient
05	Oberthur Card Systems
06	Orga
07	Axalto
08	Siemens
09	STIncard
10	GEP
11	EPS Corp
12	Athena
13	Gemalto

Table 3: ICT Mask Manufacturers coding

9. Application Selection

No explicit application selection is needed. After ATR the application is implicitly selected. The MF remains selected after the ATR.

10. Data Structures

The CIE holds data either in files, or in objects. The difference between them lies in the logical organization and in the access mode.

Files are organized in a file system, organized in a hierarchical tree structure.

To access a file, it has to be selected (either implicitly or explicitly), and has then to become the current file.

Objects are referenced implicitly or explicitly by dedicated commands, do not need to be selected (actually they cannot be selected with a Select command).

Some implementation may actually map objects onto internal files, that will be allocated in the file system, but nevertheless the access rules for those objects remain the same.

File system organization, coding and access are regulated by [2], as described in the following chapters.

10.1 File Categories

The CIE file system is based on three categories of basic file components:

- The root of the file system, the Master File (MF),
- Directory files, denoted as dedicated files (DF),
- Generic data files, denoted as Elementary files (EF).

The **Master File (MF)** is the root of the file system and is always the initial entry point to the file system. After a reset of the card, the MF is selected.

The **Dedicated Files (DFs)** are similar to Directories in traditional file systems.

DFs can contain Elementary Files (EFs), and/or other DFs. The MF can be considered to be a special DF that contains all the files.

The **Elementary File (EF)** is used for data storage. For this reason EFs are also referred to as data files. File access is similar to traditional file systems. To access a file (for reading, writing, or any other operation), it has to be selected.

10.2 Data File Types

The following structures of EF are defined:

Transparent structure:

The EF is seen at the interface as a sequence of bytes.

Record structure:

The EF is seen at the interface as a sequence of individually identifiable records.

The following attributes are defined for record structured EFs.

- **Size of the records:** fixed or variable
- **Organization of the records:** linear or cyclic structure

The following EF types are then used:

- **EF Transparent**: (also called binary files)
- **EF Linear Fixed**: linear EF with all records of a preset fixed size.
- **EF Linear Variable TLV**: Linear file with TLV structured records.
- **EF Cyclic**: Cyclic file with fixed record length.

10.3 File System Structure

The file structure is usually shown using the tree (or hierarchical) representation. Here, the logical layout of the file structure is shown so that one can easily find the path to a particular file. The following diagram provides an example of this type of layout.

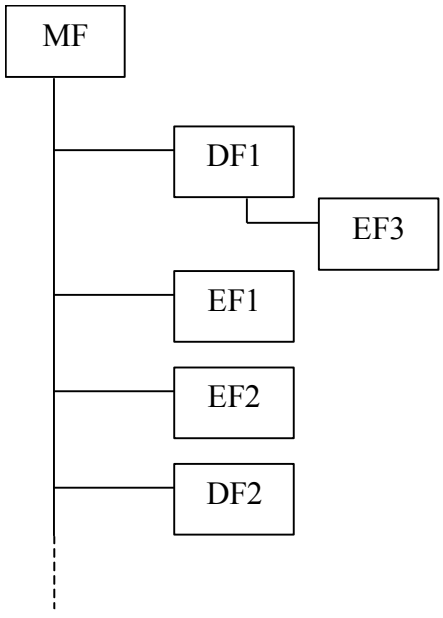

Figure 1: ISO 7816 file system structure

Elementary Files can be referenced by their 2 byte File identifier (FID), while Dedicated Files can be referenced by their File identifiers (FID) or by the application name (AID) up to 16 bytes long.

CIE allows at least a nesting level of 8 DF.

10.4 File ID

Each file is referenced by a 2 byte identifier. In order to select unambiguously any file by its identifier, all files immediately under a given DF shall have different file identifiers.

IDs **3F00h** (Master file), **3FFFh** and **FFFFh** are reserved by ISO.

10.5 File Selection

All file oriented commands (e.g. Read Record, Update Binary) act on the current file.

After reset the MF is implicitly selected (does not need an explicit Select File command). When a DF is selected, it becomes the current DF. There is always a current DF, at all times.

If an Elementary File is selected, it becomes the current EF. Not always the current EF is defined. After the selection of a DF, the current EF is undefined.

If an EF with record structure is selected, then the record oriented commands (Append Record, Read Record, Update Record) may affect the current record pointer.

After the EF selection, the current record pointer is undefined for linear record EF, while it points to the newest record for cyclic record EF. Some record access mode will then affect the value of the record pointer.

Current DF and current EF are sometimes referred to as the current framework.

10.6 Files usage

Dedicated Files are used to group related data. EF are instead used as data containers.

These are the attributes of a Dedicated File:

Attribute	Description
ID	2-byte File Identifier
DF AID	Application IDentifier, up to 16 bytes

Table 4: DF attributes

These are the attributes of an Elementary File:

Attribute	Description
ID	2-byte File Identifier
File Type	Transparent, Linear Fixed,

	Linear Variable TLV or Cyclic EF
Size	Net file size, without system overhead (2 bytes)
Rec number	number of records in a record file (max 254)

Table 5: EF attributes

In addition to the listed attributes, each file has security related attributes, described later.

11. Security Architecture

The security architecture of the card is based on the following components:

- Current Security Status
- Access Conditions (AC)
- Base Security Objects (BSO)
- Current Security Environment (CSE)
- Security Environment Object (SEO)

Together, they set the rules for using the resources of the card.

11.1 Access Conditions and Security Status

Every command has specific security conditions to meet for its execution. The specific conditions depend on the current framework, and on the operation to perform.

The information that links the object, the operation to control and the rules to apply is the Access Condition (AC).

An Access Condition can be attached to a file (EF, DF or MF), or to other card objects, as described later in the document. It tells in which status the card has to be in order to allow a specific operation on a specific object.

The Access Conditions relative to an object are grouped in a byte string logically attached to the object, where each byte is associated to one operation or to a group of operations, and can assume one of the values in the next table:

Byte Value	Condition
00h	ALWAYS
01h..1Fh	ID of a BSO
FFh	NEVER

Table 6: AC values

The values ALWAYS and NEVER mean that the operation is respectively always or never allowed. In the other cases, the value of the AC is the ID of a TEST BSO (see Section 11.2) that contains the rules for the access.

The card maintains an internal Security Status (Current Security Status) that records which access conditions are granted at a given time. “The Current Security Status may be regarded as a set of boolean conditions one for each TEST BSO to use”.

By default, at card reset, all access rights are in the status ‘not granted’. The commands VERIFY, EXTERNAL AUTHENTICATE, RESET RETRY COUNTER may change the Current Security Status, granting some rights, changing the value of a given boolean condition. In this case, we say that the “access has been granted”, or that the “access condition has been verified”.

Access Conditions related to files (MF, DF, EF) are represented by a 9-byte string, whose meaning is shown in the following tables:

Byte Nr.	AC	Protected Commands
1	RFU	Reserved For Future Use
2	AC_UPDATE	PUT DATA (DATA_OCI) PUT DATA (DATA SECI)
3	AC_APPEND	PUT DATA (DATA_OCI) PUT DATA (DATA SECI)
4	RFU	Set to FFh
5	RFU	Set to FFh
6	RFU	Set to FFh
7	AC_ADMIN	PUT DATA (DATA FCI)
8	AC_CREATE	CREATE FILE
9	RFU	Reserved For Future Use (Set to FFh)

Table 7: MF/DF AC coding

Byte Nr.	AC	Protected Commands
1	AC_READ	READ BINARY READ RECORD
2	AC_UPDATE	UPDATE BINARY UPDATE RECORD
3	AC_APPEND	APPEND RECORD

4	RFU	Set to FFh
5	RFU	Set to FFh
6	RFU	Set to FFh
7	AC_ADMIN	PUT DATA (DATA FCI)
8	RFU	Reserved For Future Use (Set to FFh)
9	RFU	Reserved For Future Use (Set to FFh)

Table 8: EF AC coding

11.2 Base Security Objects (BSO)

A Base Security Object (BSO) is a container for secret/sensitive data. BSO are used in cryptographic operation and for the verification of the access conditions to the resources of the smart card. They are referenced using an object Identifier (BSO ID) that must be unique at DF level.

There are several types and sub-types of BSO, namely there are:

- TEST BSO:
    - PIN (Personal Identification Number): is a byte string that card user has to know in order to grant certain levels of authentication
    - C/R TEST: used in challenge / response operations, contains a key
    - LOGICAL: is used to indicate a logical operation to perform with the access condition
- SM BSO: They contain keys for Secure Messaging
- PSO BSO: They contain keys used by the Perform Security Operation command

The TEST BSO are associated to binary security conditions in the card, that can be either VERIFIED (True) or UNVERIFIED (False).

When the test associated to the BSO is successfully performed (eg, a successful Verify command for a PIN object), then the associated security condition is considered VERIFIED.

The attributes of a BSO are detailed in the tables below.

Field Name	Length (Bytes)	Value Description
BSO CLASS	1	See Table 10

BSO ID	1	01h to 1Fh
OPTION	1	See Table11
FLAGS	1	• For TEST Objects represents the maximum number of retries for verify or C/R test (<= 0Fh) • Not used (set to 00h) for other types of objects
ALGORITHM	1	See Table12
ERROR COUNT	1	(<= 0Fh) number of wrong retries left for PIN verification and external authentication. No meaning for LOGICAL objects (MUST be set to 0Fh)
USE COUNT	1	Set to FFh
RFU	1	Set to FFh
VALIDITY COUNTER	1	Limits the use of an access right and it is meaningful for TEST objects only. Ranges from 00h to FEh, with 00h and FFh stands for 'no limitation'. Typically set to 00h to a general purpose PIN, and to 01h for PIN protecting a signature key.
MINIMUM LENGTH	1	This indicates the minimum PIN length, for PIN objects, or the minimum length of the challenge for C/R objects. For other BSO types this byte MUST be set to 00h
BSO AC	7	See Table13
BSO SM	16	OPTIONAL See Table17
BSO BODY LENGTH	1	The length (N) of the BSO Body field
BSO BODY	N	Key, PIN or Logical Object value.

Table 9: BSO attributes

BSO Body contains the value of the BSO:

For PIN objects: the PIN value, in plain text. The CIE supports numeric PIN only, the PIN is coded in ASCII, with an optional trailing padding set to FFh.

Example:

PIN value: "1234"

ByteString: "31h32h33h34hFFhFFhFFhFFh" (N=8)

For Key objects: the value of the key component, or the value of the key itself.

For LOGICAL objects: A 2 operand logical operation in RPN, with the format:

<Operand 1> || <Operand 2> || <Operation>

where

<Operand n> is a valid BSO ID (from 01h to 1Fh)

<Operation> = 00h for AND or FFh for OR

For the security condition associated to a LOGICAL object to be VERIFIED, the result of the logical condition must hold true. This allows to have complex security checks, such as "External Authenticate AND PIN"

Value	Description
00h	First or only key component for Authentication (TEST object)
01h	Second key component for Authentication (TEST object)
10h	Secure Messaging key (SM object)
20h	First or only key component for Encryption, Decryption and Digital Signature (PSO object)
21h	Second key component for Encryption, Decryption and Digital Signature (PSO object)

Table 10: BSO CLASS byte coding

In the table above the first component of a RSA key is always the modulus, while the second one is always the exponent.

In each DF, CLASS+ID univocally identifies a BSO. For example: there can be only one TEST BSO (CLASS 00h) with ID 01h under the MF.

BSO TYPE	OPTION Byte Value
RSA private key Exponent for ENC/DEC and DS PIN LOGICAL	02h
RSA private key Modulus for ENC/DEC and DS	22h

RSA public key Exponent for Authentication	01h
RSA public key Modulus for Ext. Authentication	21h
3DES key for Ext. Authentication, SM and ENC/DEC	83h

Table 11: OPTION byte coding

BSO Type	Algorithm	Algorithm byte value
RSA private key for ENC/DEC	RSA_PURE (padded leading 0)	0Ch
RSA private key for DS RSA public key for EXT. AUTH.	RSA	88h
3DES key for ENC/DEC 3DES key for SM ENC	3DES CBC	03h
3DES key for EXT. AUTH. 3DES key for SM SIG	3DES (auth) MAC3 (SM)	82h
PIN	Secure Verify	87h
LOGICAL	Logical AND/OR	7Fh

Table 12: Algorithm byte coding

Byte Nr.	Access Condition	Protected Command
1	AC_USE	VERIFY PSO_CDS PSO_DEC PSO_ENC EXTERNAL AUTHENTICATION
2	AC_CHANGE	CHANGE REFERENCE DATA CHANGE KEY DATA
3	AC_UNBLOCK	RESET RETRY COUNTER

4	RFU	Set to FFh
5	RFU	Set to FFh
6	RFU	Set to FFh
7	AC_GENKEYPAIR	GENERATE KEY PAIR

Table 13: BSO AC

11.3 CSE / SEO

The Current Security Environment (CSE) is a set of references to the security objects that will be used by Perform Security Operation (PSO) commands, and may be used by EXTERNAL AUTH.

It has three components: Confidentiality (CON), Digital Signature (DS) and TEST. Each of these components is the ID of the BSO to be used.

The CSE is not defined after a reset (CSE in "not initialized" state), and goes in the state "initialized" only after a Manage Security Environment (MSE) RESTORE command, that actually copies into it the values contained in the Security Environment Object referenced by the command.

A Security Environment Object (SEO) has the following structure:

Title	Length	Value And Description
SEO Identifier	1	00h to FEh (EFh reserved by ISO and SHOULD NOT be used) it must be unique at DF level
SEO AC_RESTORE	1	Access Condition for MSE RESTORE command
RFU	1	Set to FFh
SEO Component CON	1	Valid PSO BSO ID byte (BSO class byte 20h)
SEO Component DS	1	Valid PSO BSO ID byte (BSO class byte 20h)
SEO Component TEST	1	Valid TEST BSO ID byte (BSO class byte 00h)

Table 14: SEO structure

11.4 Secure Messaging

The Secure Messaging (SM) is used to protect the communication between the interface device (IFD) and the smartcard. There are two ways to communicate data in SM format:

- SM_ENC: APDU commands with enciphered data (data confidentiality)
- SM_SIG: APDU commands with cryptographic checksum (data authentication and integrity)

It is possible to use a combination of the two (ENC followed by SIG). It is recommended for security reasons to use SM_ENC only in conjunction with SM_SIG.

The presence and the type of SM used is indicated by the APDU Class byte.

Only the logical channel 0 is used, therefore the supported APDU Class byte values are:

- X0h – No Secure Messaging
- XCh - 3DES_CBC or MAC3 are performed on the transmitted data using symmetric keys. The command header is authenticated if MAC is used.

The symmetric keys used by the smartcard to decrypt and/or verify the APDU commands are obtained from the SM condition related to the object or file on which the APDU command should operate.

The SM conditions for BSOs and files are defined at object/file creation time by the commands CREATE FILE and PUT DATA (DATA_OCI).

The SM conditions for BSOs and files are managed using the commands PUT DATA (DATA_OCI) and PUT DATA (DATA_FCI).

The algorithms used in SM are

- double length key 3DES for SM_ENC
- MAC3 for SM_SIG

the cryptographic algorithms are detailed in Chapter 13.

Before any SM command involving a signature (SM_SIG and SM_ENC_SIG), an n*8-byte long random has to be sent from the card to the terminal, with a GetChallenge command.

The random generated by Get Challenge is valid for only one SM command.

11.4.1 Secure Messaging Condition

The possible values for the SM conditions are:

- 01h… 1Fh: object ID of the SM BSO to use to compute ENC/SIG
- FFh: no Secure Messaging condition defined for the operation.

Byte Nr.	SM Condition	Involved Commands
1..2	RFU	Set to FFh
3	ENC UPDATE/APPEND (objects)	PUT DATA DATA_OCI
4	SIG UPDATE/APPEND (objects)	PUT DATA DATA_SECI
5..12	RFU	Set to FFh
13	ENC ADMIN	PUT DATA DATA_FCI
14	SIG ADMIN	
15	ENC CREATE	CREATE FILE
16	SIG CREATE	CREATE FILE
17..24	RFU	Set to FFh

Table 15: MF/DF SM Conditions coding

Byte Nr.	SM Condition	Involved Commands
1	ENC READ OUT	READ BINARY
2	SIG READ OUT	READ RECORD
3	ENC UPDATE	UPDATE BINARY
4	SIG UPDATE	UPDATE RECORD
5	ENC APPEND	APPEND RECORD
6	SIG APPEND	
7..12	RFU	Set to FFh
13	ENC ADMIN	PUT DATA DATA_FCI
14	SIG ADMIN	
15..22	RFU	Set to FFh
23	ENC READ IN	READ BINARY
24	SIG READ IN	READ RECORD

Table 16: Elementary Files SM Conditions coding

Byte Nr.	SM Condition	Involved Commands
1	ENC USE IN	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY, EXTERNAL AUTH
2	SIG USE IN	
3	ENC CHANGE	CHANGE REFERENCE DATA, CHANGE KEY DATA
4	SIG CHANGE	
5	ENC UNBLOCK	RESET RETRY COUNTER
6	SIG UNBLOCK	
7..14	RFU	Set to FFh
15	ENC USE OUT	PSO_DEC, PSO_ENC, PSO_CDS, PSO_DEC, PSO_ENC, PSO_CDS
16	SIG USE OUT	

Table 17: BSO SM Conditions coding

11.4.2 APDU command in SM_SIG mode

When an APDU command is transmitted in SM_SIG mode (CLA = XCh) a MAC (signature) is computed on the command header and on the data field. The MAC is added to the command data field.

The final APDU set up can be split in 4 steps:

- **Step 1: set up the HEADER BLOCK**

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	XX	XX	XX	U	Data (U bytes)	

Header Block Structure:

Random Number	CLA	INS	P1	P2	Padding

N_Rand (n*8 random bytes)	XCh	XX	XX	XX	ISO / IEC 9797 method 1 (4 bytes)[1]

N_Rand is a random number generated by a GET CHALLENGE command.

- **Step 2: set up the Plain Text Object (P_T_O) and the Plain Text Block (P_T_B)**

Plain Text Object (P_T_O):

T	L	V
81h	U	Data (U bytes)

Note: in case of U=0, two formats for P_T_O are possible:

1. Plain Text Object = '81h' '00h' or

2. Plain Text Object = empty, the tag 81h is absent.

If the card accepts just one of these two formats, it must return an error when it receives the other format.

Plain Text Block (P_T_B):

P_T_O	Padding
U+2 bytes	ISO / IEC 9797 method 1 (S=0..7 bytes)

- **Step 3: MAC computation and MAC Object (MAC_OBJ) set up**

Compute the MAC over the bytes string obtained as follows:

Header Block	P_T_B
(n+1)*8 bytes	U+S+2 bytes

M=MAC(Header Block || P_T_B) is a 8 bytes long string of bytes.

M is used to build the **MAC_OBJ** (10 bytes long):

T	L	V
8Eh	08	M

[1] Padding ISO/IEC 9797 method 1: The data shall be appended with as few (possibly none) '00h' bytes as necessary to obtain a data string whose length (in bytes) is an integer multiple of 8.In this case, we have (n*8+4) bytes so padding will consist of 4 '00h' bytes.

- **Step 4: APDU command set up in SM_SIG format**

The resulting APDU in SM_SIG format is:

CLA	INS	P1	P2	Lc	Data Field		Le
XCh	XX	XX	XX	U+12	P_T_O	MAC_OBJ	XX

11.4.3 APDU in SM_ENC mode

When an APDU command is transmitted in SM_ENC mode (CLA = XCh) a cipher text is computed on the command data field and transmitted in the SM APDU command.

- **Step 1: set up the Cipher Text Object (C_T_O)**

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	XX	XX	XX	U	Data (U bytes)	

Computing the **Cipher Text**:

Data Field	Padding
Data (U bytes)	ISO / IEC 9797 method 2[2] (P=1..8 bytes)

The **Cypher Text Object** (C_T_O) is built as follow:

T	L	Value	
87h	U+P+1	Padding Indicator (01h)	Cipher Text (U+P bytes)

The C_T_O obtained is then composed by W=U+P+3 bytes.

- **Step 2: APDU command set up in SM_ENC format**

CLA	INS	P1	P2	Lc	Data Field	Le
XCh	XX	XX	XX	W	C_T_O	XX

[2] Padding ISO/IEC 9797 method 2: the data shall be appended with a single '80h' byte. The resulting data shall then be appended with as few (possibly none) '00h' bytes as necessary to obtain a data string whose length (in bytes) is an integer multiple of 8.

11.4.4 APDU Command in SM_SIG and SM_ENC

When an APDU command is transmitted in SM_SIG and SM_ENC mode (CLA = XCh) a MAC is computed on the command header and on the command data field cipher text and transmitted in the SM APDU command with the cipher text.

- **Step 1: set up the HEADER BLOCK**

Header Block is set up as described in step 1 of SM_SIG mode description.

- **Step 2: set up the Cipher Text Block (C_T_B)**

The Cipher Text Block (**C_T_B**) is composed as follow:

C_T_O	Padding
U+P+3 bytes	ISO / IEC 9797 method 1 (S=0..7 bytes)

Cipher Text Object is set up as described in step 2 of SM_ENC mode.

- **Step 3: MAC computation and MAC Object (MAC_OBJ) set up**

MAC is computed on the string:

Header Block	C_T_B
(n+1)*8 bytes	U+P+3+S bytes

M=MAC(Header Block || C_T_B) is an 8 bytes long string.

The MAC_OBJ (10 bytes) is composed as follow:

T	L	Value
8Eh	8	M

- **Step 4: APDU command set up in SM_SIG and SM_ENC mode**

CLA	INS	P1	P2	Lc	Data Field		Le
XCh	XX	XX	XX	W=U+P+13	U+P+3	MAC_OBJ	XX

11.4.5 SM Response in SM_ENC

Original APDU response:

Response Data Field (R bytes)	Status Word (SW1-SW2)

- **Step 1: Response Data Field is padded and encrypted**

Data	Padding
Response Data Field (R bytes)	ISO 9797 mode 2 (P=1..8 bytes)

ENC is the string of (R+P) bytes obtained enciphering the padded data.

- **Step 2: set up the Cipher Text Object (C_T_O)**

T	L	Value	
87h	R+P+1	Padding Indicator (01h)	ENC (R+P bytes)

- **Step 3: APDU Response in SM_ENC**

Enciphered Response	SW
C_T_O (R+P+3 bytes)	SW1||SW2

Example:

The terminal wants to read 5 bytes from the card, in SM_ENC. The answer is at least 3 bytes plus the padding plus the net length (Le) of the data, the command sent is: 0Ch B0h 00h 00h 0Bh

11.4.6 SM Response in SM_SIG

Original APDU response:

Response Data Field (R bytes)	Status Word (SW1-SW2)

- **Step 1: set up the HEADER BLOCK**

Header Block is set up as described in step 1 of SM_SIG mode description.

- **Step 2: set up the P_T_O and P_T_B**

P_T_O:

T	L	V
81h	R	Response Data Field

P_T_B:

P_T_O	Padding
R+2 bytes	ISO / IEC 9797 method 1 (S=0..7 bytes)

- **Step 3: MAC computation and MAC Object (MAC_OBJ) set up**

MAC is computed on the string:

Header Block	P_T_B
(n+1)*8 bytes	R+S+2 bytes

The resulting M=MAC(Header Block || P_T_B) is an 8 bytes long string.

The MAC_OBJ (10 bytes) is composed as follow:

T	L	Value
8Eh	8	M

- **Step 3: APDU Response in SM_SIG mode**

Enciphered Response		SW
P_T_O (R+2 bytes)	MAC_OBJ	SW1||SW2

Note: the response in SM_SIG mode is 12 bytes wider than the plain one, so the maximum net length (Le) allowed in this mode is 256-12=244 bytes.

Example:

The terminal wants to read 5 bytes from the card, in SM_SIG. Since the answer is 12 bytes longer than the net length of the data, the command sent is: 0Ch B0h 00h 00h 11h

11.4.7 SM Response in SM_ENC_SIG

Original APDU response:

Response Data Field (R bytes)	Status Word (SW1-SW2)

- **Step 1: set up the HEADER BLOCK**

Header Block is set up as described in step 1 of SM_SIG mode description.

- **Step 2: set up the Cipher Text Object (C_T_O)**

C_T_O is built as described in Step 1 and 2 of SM Response in SM_Enc section

- **Step 3: set up the Cipher Text Block (C_T_B)**

C_T_B:

C_T_O	Padding
R+P+3 bytes	ISO / IEC 9797 method 1 (S=0..7 bytes)

- **Step 4: MAC computation and MAC Object (MAC_OBJ) set up**

MAC is computed on the string:

Header Block	C_T_B
(n+1)*8 bytes	R+P+S+3 bytes

The resulting M=MAC(Header Block || C_T_B) is an 8 bytes long string.

The MAC_OBJ (10 bytes) is composed as follow:

T	L	Value
8Eh	8	M

- **Step 5: APDU Response in SM_SIG and SM_ENC mode**

Enciphered Response	SW

C_T_O (R+P+3 bytes)	MAC_OBJ	SW1||SW2

12. APDU Reference

The CIE supports a subset of the APDU commands specified by the ISO norms. The present chapter describes each of the supported APDUs and operating modes.

The only data element supported is byte. Therefore, all offsets, and length of strings are expressed as bytes.

APDU Command	CLA (Hex)	INS (Hex)
APPEND RECORD	0X	E2
CHANGE KEY DATA	9X	24
CHANGE REFERENCE DATA	0X	24
CREATE FILE	0X	E0
EXTERNAL AUTHENTICATE 0Xh 82h	0X	82
GENERATE KEY PAIR	00	46
GET CHALLENGE	00	84
GIVE RANDOM	80	86
MSE	00	22
PSO_CDS	0X	2A
PSO_DEC	0X	2A
PSO_ENC	0X	2A
PUT DATA	0X	DA
READ BINARY	0X	B0
READ RECORD	0X	B2
RESET RETRY COUNTER	0X	2C
SELECT FILE	00	A4
UPDATE BINARY	0X	D6
UPDATE RECORD	0X	DC
VERIFY	0X	20

Table 18: APDU commands supported by CIE operating system

Each APDU command will produce an output having such a format:

Response data	Status Condition or Status Word (SW)

- Response Data is the string of bytes returned by command evaluation. Many commands have empty Response Data field.
- SW is a 2 bytes long coded description of command evaluation result. See

SW1	SW2	Description
63	00	Authentication Failed
65	81	Memory Error
67	00	Invalid Lc value
68	81	Logical Channel not supported
68	83	Chaining Error
69	81	File type inconsistent with command
69	82	Security status not satisfied
69	83	Authentication method blocked – BSO blocked
69	84	Referenced BSO is invalid
69	85	Condition of use not satisfied
69	86	No current EF selected or Command not allowed
69	87	Expected SM data object missing
69	88	Invalid SM data object
69	99	Logical Channel not supported
6A	80	Incorrect parameters in data field
6A	81	Function not supported
6A	82	File not found
6A	83	Record not found
6A	84	Not enough memory
6A	85	Lc inconsistent with TLV structure

6A	86	Incorrect P1-P2
6A	87	Lc inconsistent with P1-P2
6C	00	Le inconsistent with expected data
6D	00	INS is invalid
6E	00	CLA is invalid
6F	00	General Error
6F	86	Key object not found
6F	87	Chaining Error
6F	FF	Internal Error
90	00	Command successful

Table 19: Status Word list

12.1 PUT DATA

CLA	INS	P1	P2	P3	Data Field
0X	DA	See Table21	See Table21	According to P1 and P2	DATA_OCI DATA_FCI DATA_SECI

Table 20: PUT DATA command

P1	P2	Description
01	6E	Creation and administration of the following types of BSO: • RSA KPRI EXP-CRYPT/DECRYPT • RSA KPRI MOD-CRYPT/DECRYPT • RSA KPRI EXP-SIGN • RSA KPRI MOD-SIGN

		• RSA KPUB EXP-EXT AUTH • RSA KPUB MOD-EXT AUTH • 3DES CRYPT/DECRYPT • 3DES-SM • 3DES-EXT AUTH • PIN • LOGICAL_OBJECT For the explanation on “Data Field” please refer to the DATA_OCI chapter.
01	6F	Creation and administration of the following FCI: • AC (Access condition) • AID (DF Name) • SM (Secure Message) For the explanation on “Data Field” please refer to the DATA_FCI paragraph.
01	6D	Creation of SE Objects. For the explanation on “Data Field” please refer to the DATA_SECI paragraph.

Table 21: Description of P1 and P2 in PUT DATA command

Description:

PUT DATA allows the creation and the administration of BS and SE objects.

Additionally, PUT DATA also allows the administration of some attributes of the files (EF/DF).

In the case of BS and SE objects, the command can create the object or it can change fields in it.

In case of files (EF/DF), the command allows to change attributes of the file, i.e. access conditions, secure messaging, etc.

The creation of files is done with the "CREATE FILE" APDU.

12.1.1 PUT DATA – DATA_OCI

The PUT DATA - DATA_OCI creates a BSO or updates the fields in a existing BSO.

The “Data Field” of PUT DATA - DATA_OCI may contain up to 5 OCI (Object Control Information) in TLV format. The OCI1 is the BSO ID. The OCI4 is optional.

The PUT DATA - DATA_OCI acts on the current DF. Two cases are possible:

- **Create BSO** - The current DF doesn't contain a BSO with ID equal to OCI1. In this case a new BSO is created in the DF.
- **Update BSO** - The current DF does contain a BSO with ID equal to OCI1. In this case all fields in the existing BSO are updated with the information passed in the command. In Update mode, the last field (BSO value) can be absent, so that the value is not changed. The use of this command is deprecated to update fields different from AC (OCI3) and SM (OCI4) and when the new condition is less restrictive than the old one, i.e. to change an access condition from Never to Always.

Security:

The access condition to satisfy is the AC_APPEND of the current DF for the Create BSO case and the AC_UPDATE of the current DF for the case update BSO:

OCI1			**OCI2**			**OCI3**			**OCI4**			**OCI5**		
Mandatory			**Mandatory**			**Mandatory**			**Optional**			**Mandatory (Creation mode)**		
T	L	V	T	L	V	T	L	V	T	L	V	T	L	V

Table 22: Data OCI template

OCI Nr.	T	L	V	Description
1	83h	2	BSO Address	BSO CLA (See Table24) || BSO ID
2	85h	8	BYTE 1: OPTIONS	See Table11
			BYTE 2: FLAGS	See Table9 for description
			BYTE 3: ALGORITHM	See Table12
			BYTE 4: ERROR COUNT	See Table9 for description
			BYTE 5: USE COUNT	Set to FFh (means unlimited use)
			BYTE 6: FFh	
			BYTE 7: VALIDITY COUNTER	See Table9 for description
			BYTE 8: MINIMUM LENGTH	See Table9 for description
3	86h	7	AC bytes	Bytes used for AC (See Table13)
4	CBh	16	SM bytes	Bytes used for SM (See Table17)
5	8Fh	N	N bytes	Value of the security object (PIN,

				Key, Logical). See Table26

Table 23: DATA_OCI description

Object Class	Value
RSA KPRI EXP – CRYPT / DECRYPT	21h
RSA KPRI MOD – CRYPT / DECRYPT	20h
RSA KPRI EXP – SIGN	21h
RSA KPRI MOD – SIGN	20h
RSA KPRI EXP – EXT. AUTH.	01h
RSA KPRI MOD – EXT. AUTH.	00h
3DES CRYPT / DECRYPT	20h
3DES SM	10h
3DES – EXT. AUTH.	00h
PIN	00h
LOGICAL OBJECT	00h

Table 24: BSO Class coding table

Object Description	Class	Option	Algorithm
RSA KPRI EXP – CRYPT / DECRYPT	21h	02h	0Ch
RSA KPRI MOD – CRYPT / DECRYPT	20h	22h	0Ch
RSA KPRI EXP – SIGN	21h	02h	88h
RSA KPRI MOD – SIGN	20h	22h	88h
RSA KPUB EXP-EXT AUTH	01h	01h	88h
RSA KPUB MOD-EXT AUTH	00h	21h	88h
3DES CRYPT/DECRYPT	20h	83h	03h
3DES – SM Auth	10h	83h	82h

3DES – SM Cipher	10h	83h	03h
3DES – EXT. AUTH.	00h	83h	82h
PIN	00h	02h	87h
LOGICAL OBJECT	00h	02h	7Fh

Table 25: BSO resuming table

Object Type	Value and format
PIN	PIN Value (in ASCII format)
RSA modulus	<Length of the following><00h><Modulus>
RSA exponent	<Length of the following><00h><Exponent>
DES / 3DES key	Key value

Table 26: BSO value (tag 8F)

12.1.2 PUT DATA – DATA_FCI

The PUT_DATA – DATA_FCI is used to update attributes of the current DF/EF. The format of the data field of PUT_DATA – DATA_FCI is shown in Table 27 and Table 28.

This APDU applies on the current selected file (DF or EF). When no current EF exists than the command applies on the current DF. When the current file is an EF, the FCI1 must not be present.

The update of AC (FCI2) and/or SM bytes (FCI3) is deprecated when the new condition is less restrictive than the old one, i.e. to change an access condition from Never to Always.

Security:

The access condition to satisfy is AC_ADMIN.

FCI1			FCI2			FCI3		
T	L	V	T	L	V	T	L	V

Table 27: DATA_FCI template

FCI Nr.	T	L	V	Description

1	84h	P<=10h	AID byte string	DF Name Optional
2	86h	Q	AC bytes	See Table7 and Table8
3	CBh	R	SM bytes	See Table15 and Table16

Table 28: DATA_FCI description

12.1.3 PUT DATA – DATA_SECI

The PUT DATA - DATA_SECI creates a Security Environment Object (SEO). The "Data Field" is composed of a sequence of 3 Security Environment Control Information (SECI) in TLV format.

The command also allows updating fields in an existing SEO.

The PUT DATA - DATA_ SECI acts on the current DF. Two cases are possible:

- **Create SEO** - The current DF doesn't contain a SEO with the ID equal to SECI1. In this case a new SEO is created in the current DF.
- **Update SEO** - The current DF does contain a SEO with the ID equal to SECI1. In this case it is possible to update the access conditions (SECI2).

Security:

The access condition to satisfy is the AC_APPEND of the current DF for the creation of objects or AC_UPDATE for the administration.

SECI1			**SECI2**			**SECI3**		
T	L	V	T	L	V	T	L	V

Table 29: DATA_SECI template

SECI Nr.	**T**	**L**	**V**	**Description**
1	83h	1	SEO ID	(00h..FEh)
2	86h	2	BYTE 1: AC_RESTORE	AC for MSE_RESTORE
			BYTE 2: set to FFh	RFU

3	8Fh	6	BYTE 1: Not used – MUST be set to 00h	
			BYTE 2: COMP_CDS	BSO ID for digital signature
			BYTE 3: Not used – MUST be set to 00h	
			BYTE 4: COMP_CON	BSO ID for Encrypt / Decrypt
			BYTE 5: COMP_Ext_Auth	BSO for Ext Auth ID
			BYTE 6: Not used – MUST be set to 00h	

Table 30: DATA_SECI description

12.2 CREATE FILE

CLA	INS	P1	P2	P3	Data Field
0X	E0	00	00	LC=Data Field Length	Data (See Table32 for description)

Table 31: CREATE FILE command

The CREATE FILE command creates an EF or a DF under the current DF. After the creation, the new file is the current file.

If the new file has a record structure, after the creation of the file the current record is undefined.

The data of the create file command is a TLV which includes 6 TLV coded FCI (see Table33). The sixth FCI is optional.

T	L	V					
62h	N	FCI1	FCI2	FCI3	FCI4	FCI5	FCI6 (Optional)

Table 32: CREATE FILE data template

FCI Nr.	T	L	V	Description
1	80h or 81h	02	If tag=80h (EF only) file size If tag=81h (DF only) DF size	File Size

2	82h	03	BYTE 1: File Type	01	EF Transparent
				02	Linear Fixed
				05	Linear Variable TLV
				06	Cyclic
				38h	DF
			BYTE 2: RFU		
			BYTE 3: Record Size	Meaningful for Linear Fixed and Cyclic files only	
3	83h	02	ID	File ID	
4	85h	01	01	MUST be set to 01	
5	86h	09	AC bytes	See Table8 for description	
6	CBh	18h	SM bytes	See Table16 for description	

Table 33: FCIs description

Notes:

- It is not allowed to create files with FID "3F00", "3FFF","FFFF".
- If the command creates a DF, the current DF is the new DF and the current EF is undefined.
- If the command creates an EF, the current EF is the new EF and the current DF is unchanged.
- If the command creates a linear EF, the current record is not defined.
- If the file ID of the file to be created already exists in current directory, the creation is not allowed.

Security:

The access condition to satisfy is AC_CREATE.

12.3 SELECT FILE

CLA	INS	P1	P2	P3	Data Field	LE
00	A4	See	00	LC=Data Field Length	Data (See Table35)	00[3]

[3] LE=00 means that all the available FCI data can be returned if available.

		Table35				

Table 34: SELECT FILE Command

SELECT FILE command allows the selection of a file (EF or DF). The following selection modes are supported:

Selection Mode	P1	Data
Selection of the EF or the DF with the given FID under the current DF	00	File ID (2 bytes)
Selection of the MF	00	Empty (0 bytes)
Selection of the MF	00	3Fh 00h (2 bytes)
Selection of the parent DF	03	Empty (0 bytes)
DF Selection by AID	04	DF AID (1..16 bytes)
Select EF or DF by absolute path selection	08	EF or DF path (m*2 bytes)
Select EF or DF by relative path selection	09	EF or DF path (m*2 bytes)

Table 35: SELECT FILE P1 and Data Field coding

The partial ID match selection is not required.

After a DF selection, the current EF is undefined. After a record structured EF selection, the current record pointer is undefined.

Security:

No security conditions

SELECT FILE Response:

SELECT FILE command, after the execution, may give, as an output, a sequence of FCI objects structured as described in

T	L	V			
6Fh (or 62h)	N	FCI1	FCI2	FCI3	...

Table 36: SELECT FILE command response description

Only the following FCI object is mandatory in case of DF or EF selection, while all other tags in the FCI can be proprietary implementations and should be ignored by C.I.E. applications.

T	L	V
80h or 81h	02	File Size

12.4 READ BINARY

CLA	INS	P1	P2	P3
0X	B0	Offset (Hi-byte)	Offset (Lo-byte)	LE=Number of bytes to read

Table 37: READ BINARY command

The Read Binary command reads part or all the data in a transparent EF. This command is processed on the currently selected EF.

The offset value in the parameters P1 and P2 sets the starting point of the byte string to read within the file. The offset is calculated from the start of the file (0000 is the first position, 0001 the second and so on).

The maximum number of bytes that can be read in one command can't exceed 256. If more bytes are required, then the amount must be spread up onto multiple read binary commands.

Selection by SFI is not required. The bit P1.7 MUST be set to zero by the terminal.

If LE=00 then all available bytes in the file shall be returned by the card up to the end of the file, and up to 256 bytes.

Note:

With old implementations of CIE one of the following situations may occur:

1) Le = 00h: if FileLength>256 then an error is returned, otherwise the card returns all available data, up to the end of the file

2) Le !=00h

- Le > FileLength-P1P2: the card can return all bytes until the end of the file or an error 6Cxx with xx = FileLength-P1P2.
- Le <= FileLength-P1P2: returns Le bytes

For compatibility reasons, CIE applications will have to deal with these possible situations.

Security:

The operation is possible if the access conditions for READ on the current EF are satisfied.

Note:

If SM condition ENC_READ_IN is specified on the file to be read, then the data to be encrypted are just the 8 padding bytes (80 00 00 00 00 00 00 00).

12.5 UPDATE BINARY

CLA	INS	P1	P2	P3	Data Field
0X	D6	Offset (Hi-byte)	Offset (Lo-byte)	LC=Data Length	Data to be written

Table 38: UPDATE BINARY command

Updates a transparent EF with a variable-length string. This command is used to replace data in a currently selected EF. P1 || P2 is the offset from begin of file, of the first byte to update.

The offset is calculated from the start of the file (0000 is the first position, 0001 the second and so on).

The maximum number of bytes that can be sent in one command can't exceed 255. If more bytes are required, then the amount must be spread up onto multiple update binary commands.

Selection by SFI is not required. The bit P1.7 MUST be set to zero by the terminal.

Security:

The operation is possible if the AC_UPDATE condition on the current EF is satisfied.

12.6 APPEND RECORD

CLA	INS	P1	P2	P3	Data Field
0X	E2	00	00	LC=Data Length	Data to be written in the record

Table 39: APPEND RECORD command

This command creates a new record in the current record structured EF.

At the end of the command, the record appended becomes the current record.

If the selected file has a linear structure, the command writes the new record at the end of the file, provided there is enough memory available in it. If there is not enough memory, an error is returned.

The records are numbered according to their order of creation. Therefore the record #1 is the oldest created record.

It is not possible to address the record number 0 and the record number FFh, so, within each record EF of linear structure it is not possible to create more than 254 records.

If the selected file has a cyclic structure, the command writes the new record at the end of the file if it is not full, otherwise it overwrites the oldest one. The record created last is numbered #1.

In the case of a linear fixed or cyclic structure, the length of the record to be written shall correspond with the one specified during the file creation. In the case of TLV format, the data have to respect the TLV format. It is assumed that the TAG field and the LEN field are one byte long each.

Tag values are not checked by the commands, anyway tags with value 00h and FFh are not allowed.

Security:

The command can be performed only if the access condition for APPEND for the current EF is satisfied.

12.7 READ RECORD

CLA	INS	P1	P2	P3
0X	B2	Record Number or Record Identifier	Record Access Method	LE=Number of bytes to read

Table 40: READ RECORD command

This command reads the contents of one record from current EF.

If the current EF has not a record structure an error will be generated.

Selection by SFI is not required. Therefore the 5 most significant bits of P2 MUST be forced to 0 by the terminal.

It's possible to read a record by record identifier only if the record is a simple TLV.

The parameters bytes P1 and P2 tell the way to access the record.

P1 contains either a record number, to access a record by its logical position, or a tag to be searched in the file.

Access by record position (P1=00 or P2 =04)

This type of access is possible for all kind of record EF.

P2=04: read current/absolute

- P1=0 : Read the current record; the record pointer does not change
- P1=n : Read the record number n; the record pointer does not change

n has to be different from FFh.

P1=00:

- P2=00h: read first: read the record number 1: set the record pointer to 1
- P2=01h: read last: read the record with the highest record number; set the record pointer to the maximum
- P2=02h: read next: read the record whose record number is one more than the current one and set it as the current selected record
- P2=03h: read previous: read the record whose record number is one less than the current one and set it as the current selected record

Access by tag (P1!=00 and P2 !=04)

This access method is only possible for linear TLV record EF. The Access by tag is deprecated for backward compatibility reasons.

- **P2=00h**: read first occurrence - read the record with the tag given in P1 with the smallest record number; set the record pointer the found record (if any)
- **P2=01h**: read last occurrence - read the record with the tag given in P1 with the highest record number; set the record pointer to the found record (if any)
- **P2=02h**: read next occurrence - read the record with the tag given in P1 searching the file from the current record in the direction of increasing record numbers; set the record pointer to the record found (if any)
- **P2=03h**: read previous occurrence - read the record with the tag given in P1 searching the file from the current record in the direction of decreasing record numbers; set the record pointer to the found record (if any)

If the current record is not defined (EF just selected), then there is equivalence between the modes:

First and Next

Last and Previous

Security:

The command can be performed only if the access condition for the READ function of the EF is satisfied.

Note:

If SM condition ENC_READ_IN is specified on the file to be read, then the data to be encrypted are just the 8 padding bytes (80 00 00 00 00 00 00 00).

12.8 UPDATE RECORD

CLA	INS	P1	P2	P3	Data Field
0X	DC	Record Number or Record Identifier	Record Access Method	LC=Number of bytes to write	Record Data

Table 41: UPDATE RECORD command

This command replaces the content of a record in the current EF with the string bytes contained in the Data Field.

If the current EF has not a record structure, an error will occur.

Selection by SFI is not required. Therefore the 5 most significant bits of P2 MUST be forced to 0 by the terminal.

The parameter bytes P1 and P2 tell the way to access the record.

In the UPDATE RECORD command, the only way to access a record is by its logical position: P1 then contains a record number, or 00h to indicate the current record.

Access by record position (P1=00 or P2 =04)

This type of access is possible for all kind of record EF.

P2=04: update current/absolute

- P1=0 : update the current record;
- P1=n : update the record number n;

n has to be different from FFh.

P1=00:

- P2=00: update the record number 1; set the record pointer to 1
- P2=01: update last: update the record with the highest record number; set the record pointer to the highest record number;
- P2=02: update next: update the record whose record number is one more than the current one; set the record pointer to the updated record;
- P2=03: update previous: update the record whose record number is one less than the current one; set the record pointer to the updated record;

The use of this command on cyclic files with P2 different from 03 is deprecated for backward compatibility.

If the current record is not defined (EF just selected), then there is equivalence between the modes:

First and Next

Last and Previous

Security:

The command can be performed only if the AC_UPDATE of the EF is satisfied.

12.9 VERIFY

CLA	INS	P1	P2	P3	Data Field
0X	20	00	b7 = 0 PIN under the MF b7 = 1 PIN search with backtracking b6 = 0 b5 = 0 b4 --- b0 PIN Identifier (BSO ID)	LC=PIN Length	PIN value

Table 42: VERIFY command

This APDU compares the data sent from the interface device with the reference data stored in the card, and sets the security status according to the comparison result.

The card maintains an internal retry counter for each BSO. The comparison is initiated only if the retry counter is greater than zero.

When the comparison fails, an error code is returned and the number of retries stored in the BSO is decremented. When this number reaches the value of 0, the authentication mechanism is blocked.

When the comparison succeeds, the security status of the card changes. The number of retries of the BSO is reset to the “maximum number of consecutive wrong attempts”, and a flag is set in the card to signal the correct verification of the relevant BSO.

The BSO to be verified is indicated by the parameter P2. It can be searched under the MF or with a backtracking mechanism, starting from the current DF.

Security:

The access condition to satisfy is AC_USE of relevant BSO.

12.10 CHANGE REFERENCE DATA

This command is used to change the data field of a PIN type base security object (BSO).

CLA	INS	P1	P2	P3	Data Field
0X	24	00 = implicit test 01 = explicit test	b7 = 0 PIN under the MF b7 = 1 PIN search with backtracking b6 = 0 b5 = 0 b4 --- b0 PIN Identifier (BSO ID)	LC = m (old) + n (new) where m is the length of the old PIN (m=0 if the old PIN has already been verified) and n is the length of the new PIN value	Verify data (P1=0) or empty (P1=1) || New reference data

Table 43: CHANGE REFERENCE DATA command

If **P1 = 00h**, the data of length m of the PIN object referenced in the AC_CHANGE of the BSO referenced by parameter P2 are compared with the first m bytes of the Input Data Field.

The data have to be equal in number and value. Partial string match is not considered as a valid match.

If the comparison fails, the retry counter of the BSO is decremented, and if it reaches zero, the BSO is blocked.

If the comparison succeeds, the data field of the referenced BSO is updated with the next n bytes in the Data Field of the command. The retry counter is reset to its preset maximum.

If P1 = 01h, the right of the AC CHANGE of the BSO referenced by parameter P2 must have been granted before (i.e. by a VERIFY command). No Verify data are sent and the BSO PIN data are overwritten with the new reference data of the Input Data Field.

If the belonging AC_CHANGE right has not been granted, the command will be immediately rejected.

Change Reference Data has not to be used on Logical objects. No special error condition has to be issued for this case.

Security:

The access condition to satisfy is AC_CHANGE of relevant BSO.

12.11 CHANGE KEY DATA

CLA	INS	P1	P2	P3	Data Field
9X	24	Key Class (See Table24) PIN and LOGICAL OBJECTS are not allowed.	b7 = 0 BSO under the MF b7 = 1 BSO search with backtracking	LC = N	New key data (see Note below)

			b6 = 0 b5 = 0 b4 --- b0 BSO ID		

Table 44: CHANGE KEY DATA command

This command is used to change the data field of a key referenced by P1/P2 to the value given in the Data Field and set the error counter to the maximum.

The length of the new key data must be equal to the length of the old key data.

Note:

In case of a DES key, the new key data is exactly the key value, while in case of a RSA key, data can be presented to the card in two formats:

- New key data is the effective value of the key;
- New key data is coded as in case of Put Data OCI command:
  - <Len of the following><00h><Modulus>, or
  - <Len of the following><00h><Exponent>

If the card accepts just one of these two formats, it must return an error when it receives the other format and it must not update the value of the key.

CIE applications have to take in account the different behaviour that different card models may have due to backward compatibility reasons.

Security:

The access condition to satisfy is AC_ CHANGE of relevant BSO.

12.12 EXTERNAL AUTHENTICATE

CLA	INS	P1	P2	P3	Data Field
0X	82	00	b7 = 0 BSO under the MF b7 = 1 BSO search with backtracking b6 = 0 b5 = 0 b4 --- b0 BSO ID	LC = N	C/R response

Table 45: External Authenticate command

This command allows the card to authenticate an external entity by means of a challenge-response protocol.

It is possible to perform an External Authenticate only after a Get Challenge command. The Get Challenge makes the card generate internally a random, that is stored internally and sent to the terminal.

It is possible to have other commands between a Get Challenge and its External Authenticate, as long as they are issued during the same card session (between 2 resets).

The random generated during the Get Challenge command is valid for only one External Authentication.

The Data Field of the command contains the result of the cryptographic operation made on the challenge.

The card compares this value with the value computed internally, and if they match, the security status of the card changes accordingly.

The algorithm used for the cryptogram computation is set in the BSO, so P1=0.

BSOs that can be used with External Authenticate are:

- RSA KPUB – EXT AUTH
- 3DES – EXT AUTH

The parameter P2 contains the scope and the ID of the BSO. If the BSO ID is equal to 0 then the BSO is searched in component TEST of the Current Security Environment (CSE).

The length of the data field has to match the length of the challenge, and has to be 8 bytes if 3DES is used, or exactly the length of the key modulus in the other case (RSA). If it is not the case, an error is returned.

When the command is executed with success, the access right is granted and the error counter related to the relevant object (BSO) is set to its max value.

If the command is not executed with success, then access right is not granted and the error counter is decreased by one.

Security:

The access condition to satisfy is AC_USE. The BSO object must not be in a blocked status.

Recommendation:

The use of External Authenticate in SM_SIG mode is deprecated.

12.13 RESET RETRY COUNTER

CLA	INS	P1	P2	P3	Data Field
0X	2C	00 for PIN XX for other BSO type	b7 = 0 BSO under the MF b7 = 1 BSO search with backtracking b6 = 0 b5 = 0 b4 --- b0 (BSO ID)	LC = m (verify) / 00 + n (new) / 00	Verify data or empty + New reference data or empty

Table 46: RESET RETRY COUNTER command

This command sets the error counter of a security base object (BSO) to its maximum preset value.

P1 coding has been extended with respect to what is indicated in ISO 7816-8.

The RFU bits in P1 have been used to signal the object description.

Only test object can be referenced.

P1 Value	Description
0000 0000	The data field contains “Verification Data” and “New Reference Data”. This mode is valid only when the object referred to by P2 has as access condition for AC_UNBLOCK a reference to a PIN object.
Xxxx x001	The data field contains only “Verify Data”. This mode is only valid when the object referred to by P2 has as access condition for AC_UNBLOCK a reference to a PIN object.
Xxxx x011	The data field is empty

Table 47: P1 coding for RESET RETRY COUNTER command

Note 1: the mode with P1=00h is allowed for PIN objects only.

Note 2: the modes with P1= xxxx x001 and xxxx011 are allowed for any object, including keys. It is not possible to change the value of the keys with this command.

The parameter P2 identifies the BSO to use.

Three cases are possible:

P1=xxxx x000:

- P2 has to be the reference to a PIN object, whose data length is n.
- the object referenced by P2 has an access condition for AC_UNBLOCK that refrences another PIN object whose data length is m.

the first m bytes of the command Data Field are compared with the object data of the PIN object which is referenced by the AC_UNBLOCK of the PIN object referenced in P2. If the comparison succeeds the error counter is set to its maximum value, and the object data are replaced by the next n bytes in the command Data Field.

P1=xxxx x001:

- The PIN object referenced by P2 has an access condition for AC_UNBLOCK which references a PIN object whose data length is m.

the command Data Field is compared with the object data field of the PIN object referenced by the AC_UNBLOCK of the object referenced in P2. If the comparison succeeds the reset counter is set to its maximum value.

P1=xxxx x011:

The data field is empty. The AC for unblocking the referenced BSO has to be verified before the command is issued.

12.14 GET CHALLENGE

CLA	INS	P1	P2	P3
00	84	00	00	LE

Table 48: GET CHALLENGE command

This command makes the card generate and send a random number. The generation of random number is used for the next External Authenticate command or for the SM computations.

The generated random is valid only for the next External Authenticate. After that, a new Get Challenge has to be issued for another External Authenticate.

It is not needed to have External Authenticate follow immediately the Get Challenge.

A GET CHALLENGE command will give as an output LE bytes (the required random).

Security:

no access condition has to be granted.

12.15 GIVE RANDOM

CLA	INS	P1	P2	P3	Data Field
80	86	00	00	LC = n (length of data field)	Data from the terminal

Table 49: GIVE RANDOM command

This command allows the terminal to send a n byte random number. This random will be used for the next response-SM (SIG or ENC-SIG) calculation.

The random can be used only once. A new Give Random will overwrite the previous one.

Security:

no access condition has to be granted.

12.16 MANAGE SECURITY ENVIRONMENT (MSE)

CLA	INS	P1	P2	P3	Data Field
00	22	See Table51	See Table51	LC = Data Field Length	Data to be used in current security environment (CSE) in TLV format (See Table 51 and Table52 for more details)

Table 50: MSE command

The MSE command is used to load (mode RESTORE) or to set up (mode SET) the CSE (Current Security Environment). The MSE supported command modes are listed here:

Mode	P1	P2	Data Field
RESTORE	F3	SEO ID	Empty
SET	F1	See Table52	See Table52

Table 51: MSE mode coding

CSE Component	MSE SET command				Related commands
	P2	Data Field			
		T	L	V	

Confidentiality (CON)	B8	83h	1	Object ID	PSO_DEC
		84h			PSO_ENC
Authentication component (TEST)	A4	83h	1	Object ID	EXT AUTH
		84h			
Digital Signature Component (CDS)	B6	83h	1	Object ID	PSO_CDS
		84h			

Table 52: P2 and Data Field coding for MSE SET command

The CSE is the card security status, stored in volatile memory, and reset at every card session (i.e. after each reset of the card). For details on the CSE structure, see the dedicated chapter.

To illustrate the use of the CSE, consider the PSO_ENC and PSO_DEC commands. These commands use an explicit secure object where the key (public or private) is stored.

Thus, before the execution of a PSO command, the CSE component CON is set (via RESTORE or SET): this component refers to the BS object, which has to be used for the 3DES/RSA algorithms.

The CSE contains 3 components:

Component	Type of Objects Used	Related Commands
CON	RSA KPRI CRYPT/DECRYPT 3DES CRYPT/DECRYPT	PSO_DEC PSO_ENC
DS	RSA KPRI SIGN	PSO_CDS
TEST	3DES EXT AUTH RSA KPUB EXT AUTH	EXT AUTH

Table 53: CSE components

12.16.1 MSE mode RESTORE

The functionality of the MSE command in the RESTORE mode is the following:

- Use the backtracking mechanism to search the SE object whose ID is in P2
- If the wanted SE object is found, it becomes the CSE. Afterward the CSE can be used for execute the commands PSO_DEC, PSO_ENC, PSO_CDS, EXT AUTH.
- To execute the MSE RESTORE, the specified SE object has to be created in advance by the command PUT DATA – SECI.

Security:

The access condition to satisfy is AC_RESTORE.

12.16.2 MSE mode SET

This mode is valid only when a CSE is loaded in RAM by a previous MSE RESTORE command. It is used to set up the specific CSE component referenced by P2. The data field contains TLV data where the value field is the ID of the object to be used.

Security:

no access condition to satisfy.

12.17 GENERATE KEY PAIR

CLA	INS	P1	P2	P3	Data Field
00	46	00	00	LC=length of data field	Pvk (2 bytes) Pbk (2 bytes) ARMT (1 bytes) Dif_pq (1 bytes) Pub_Exp (2 bytes)

Table 54: GENERATE KEY PAIR command

This command is used to generate a key pair for RSA computations.

Proprietary data meaning:

- Pvk =The ID of the RSA KPRI object. The ID is 2 bytes long. The first byte is the object class and the second is the object ID. The RSA KPRI object has to be in the current DF.
- Pbk =The ID of the file that will contain the public key (LINEAR TLV type). The file has to be created empty in the current DF. The GENERATE KEY PAIR will create two TLV records with tags:
    - 10h for the key module
    - 11h for the key exponent

  and will put the key components in these records.

  The format for the key modulus is:

  <Len><00h><Modulus>

The record content will then be:

<10h><Len of the following><Len of the following><00h><Modulus>

The format for the key exponent is:

<Len><00h><Exp>

The record content will then be:

<11h><Len of the following><Len of the following><00h><Exp>

- ARMT = RFU, MUST be set to 00h
- Dif_pq = RFU, MUST be set to 00h
- Pub_Exp = is the length in bits of the key exponent. Pub_Exp shall be in the range 16…64 bits.

Security:

The access condition to satisfy is AC_GENKEYPAIR for the Private key BSO.

The AC_APPEND of the file that will contain the public key has to be verified.

12.18 PERFORM SECURITY OPERATION

12.18.1 PSO_DEC

CLA	INS	P1	P2	P3	Data Field
0X	2A	80	86	LC=length of data to be deciphered + 1 byte for padding indicator	00 (padding indicator) || enciphered data

Table 55: PSO_DEC command

This command deciphers the input data with a symmetric or an asymmetric key. The first byte in the input data is the indicator of the padding used.

The deciphered data is returned in the command response.

To use this command it is necessary to load in memory a current security environment (CSE) using the MSE command. The CSE CON component has to refer to an object of type:

- RSA KPRI CRYPT/DECRYPT
- 3DES CRYPT/DECRYPT

Security:

The access condition to satisfy is AC_USE of relevant BSO

Note:

When PSO_DEC is performed with a BSO with Algorithm byte set to 0x0C (RSA_PURE), the command doesn't perform any extra unpadding operation (padding indicator byte is ignored).

12.18.2 PSO_ENC

CLA	INS	P1	P2	P3	Data Field	LE
0X	2A	86h	80h	LC = Length of data to be enciphered (plain text)	Input data to be enciphered	Length of enciphered data

Table 56: PSO_ENC command

This command enciphers the input data with a key.

The enciphered data is returned in the response where the first byte is the used padding indicator:

00h (padding indicator) || The enciphered data

To use this command is necessary to load in memory a current security environment (CSE) by using a MSE command. The CSE CON component has to refer to an object of type:

- RSA KPRI CRYPT/DECRYPT
- 3DES CRYPT/DECRYPT

Security:

The access condition to satisfy is AC_USE.

Note:

When PSO_ENC is performed with a BSO with Algorithm byte set to 0x0C (RSA_PURE), the command doesn't perform any extra padding operation (padding indicator byte is ignored).

12.18.3 PSO_CDS

CLA	INS	P1	P2	P3	Data Field	LE
0X	2A	9E	9A	LC = Length of data to be signed	Input data to be signed	Length of signed data

Table 57: PSO_CDS

This command computes the digital signature (DS) of the input data.

The DS is given in the response data field.

To use this command it is necessary to load in memory a current security environment (CSE) by using a MSE command. The CSE has to refer to a DS component and to an object type:

- RSA KPRI SIGN

The input data are the digest of the hashed data.

The card performs a PKCS#1 BT1 padding on the input data before computing the signature.

Security:

The access condition to satisfy is AC_USE of relevant BSO.

12.19 Optional commands

Main Digital Signature schemes need files (EF and DF) to be deactivable / activable after the signature has reached a certain level of authentication within the card.

In the next sections two optional ISO standard APDU commands are described as a suggestion for implementing such requirement.

The implementation of these commands is absolutely optional. Cards manufacturers are free to fulfil Digital Signature requirements with proprietary commands.

12.19.1 AC bytes extension

Bytes 4 and 5 in MF/DF and EF ACs (See Table7 and Table8 for coding) are used in the following way:

Byte Nr.	AC	Protected Commands
...	...	...
4	AC_ DEACTIVATE	DEACTIVATE FILE
5	AC_ ACTIVATE	ACTIVATE FILE
...	...	..

Table 58: MF/DF and EF AC bytes modification

12.19.2 DEACTIVATE FILE

CLA	INS	P1	P2	P3	Data Field

0X	04	00	00	LC = 00	Empty

Table 59: DEACTIVATE FILE command

This command acts on the currently selected file.

It changes the status of the current file to "Deactivated".

A deactivated file can only be selected or activated. All other operations on this file will result in an error message.

If a DF has been deactivated, its content is deactivated as well.

Security:

AC_DEACTIVATE has to be fulfilled.

12.19.3 ACTIVATE FILE

CLA	INS	P1	P2	P3	Data Field
0X	44	00	00	LC = 00	Empty

Table 60: ACTIVATE FILE command

This command acts on the currently selected file.

It changes the status of the current file to "Activated".

Security:

AC_ACTIVATE has to be fulfilled.

13. Cryptographic algoritms

This annex describes the crypto used in CIE and it's split in two parts:

- Asymmetric Algorithms: RSA
- Symmetric Algorithms: DES, 3DES, MAC3

13.1 RSA (Rimvest-Shamir-Adleman)

In this section, the following notation will be followed:

Symbol	Description
p	First secret prime
q	Second secret prime
N	Modulus (N=p*q)
e	Public exponent
d	Private exponent
M	Plain data
C	Ciphered data

Table 61: Symbols legend

The public transformation in RSA (used for encipher a plaintext or verify a digital signature) is defined from the mathematical operation:

$$\mathbf{C = M^e \bmod N}$$

The private transformation in RSA (used for decipher a cipher text or to sign digitally) is defined from the mathematical operation:

$$\mathbf{M = C^d \bmod N}$$

The command GENERATE KEY PAIR in the CIE generates a key pair having the following characteristics:

- The modulus N is an 1024 bits long integer.
- The length in bits of the public exponent **e** is in the range from 16 to 64.

- The length in bits of the private exponent **d** is the same of the modulus.

The plain text M may be a 1024 bits integer smaller than the modulus (usually the most significant byte is zero).

13.2 Symmetric algorithms

Differently from RSA (and other asymmetric algorithms) where different keys (public and private) are used for direct and inverse transformation, symmetric algorithms are characterized by the use of the same key for both operations.

Symbol	Description
K	DES Key
K_1	The first DES key for 3DES computation
K_2	The second DES key for 3DES computation
K_3	The third DES key for 3DES computation
D_K	DES with key K
D^{-1}_K	Inverse DES with key K
$3D_{K1K2}$	3DES with keys K_1 and K_2
ENC	Enciphering
DEC	Deciphering
M	Plain data
C	Ciphered data

Table 62: Symbols legend for asymmetric cryptography

13.2.1 DES (Data Encryption Standard)

The Data Encryption Standard (DES) was developed by an IBM team around 1974 and adopted as a national standard in 1977 in fact is one of the FIPS approved algorithms for encryption. DES was the first official U.S. government cipher intended for commercial use and it's the most widely used cryptosystem in the world.

The DES algorithm is specified in FIPS 46-3.

In the CIE the DES is used only as Triple DES

13.2.2 3DES

Following figures show the of Triple DES for encipher a plaintext **M** with 2 or 3 keys.

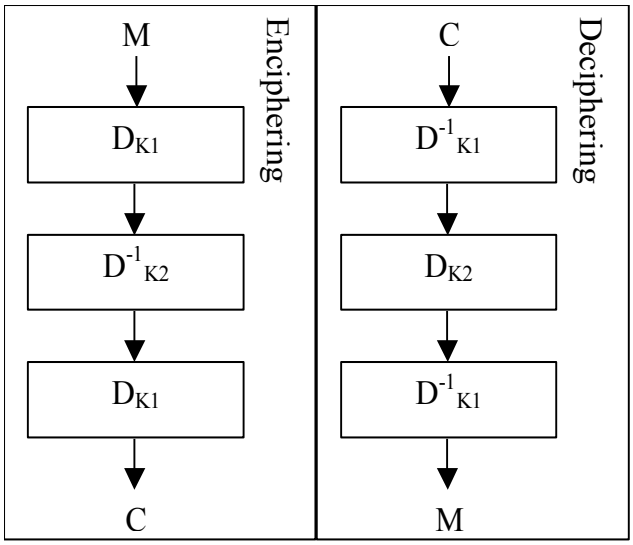

Figure 2: 3DES computation with 2 DES Keys

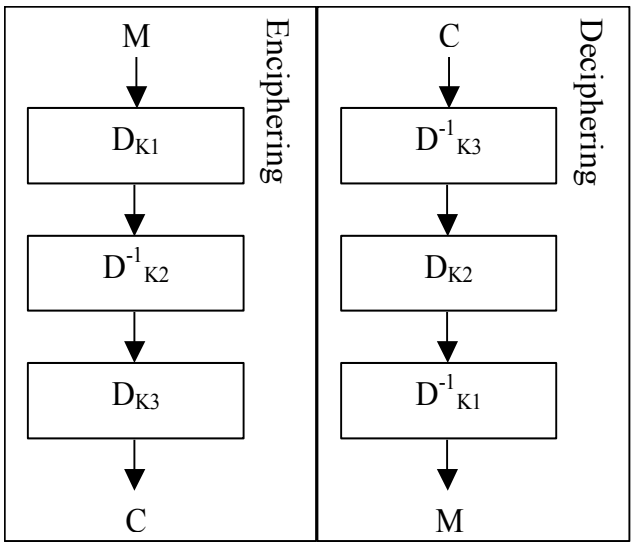

Figure 3: 3DES with 3 DES keys

13.2.3 Cipher Blocking Chain (CBC)

When the plaintext M length is more than 8-bytes it is possible to use the 3DES in CBC mode.

The CBC mode is a way to perform the DES algorithm on an n-bit plaintext M.

The symbols employed for the CBC mode are:

- a sequence of q plaintext blocks P_1, P_2, …, P_q each of 64 bits
- a key K
- a Starting Vector (SV) of 64 bits. For the CIE it is a string of 00h.

- a sequence of q cipher text blocks C_1, C_2, …, C_q each of 64 bits

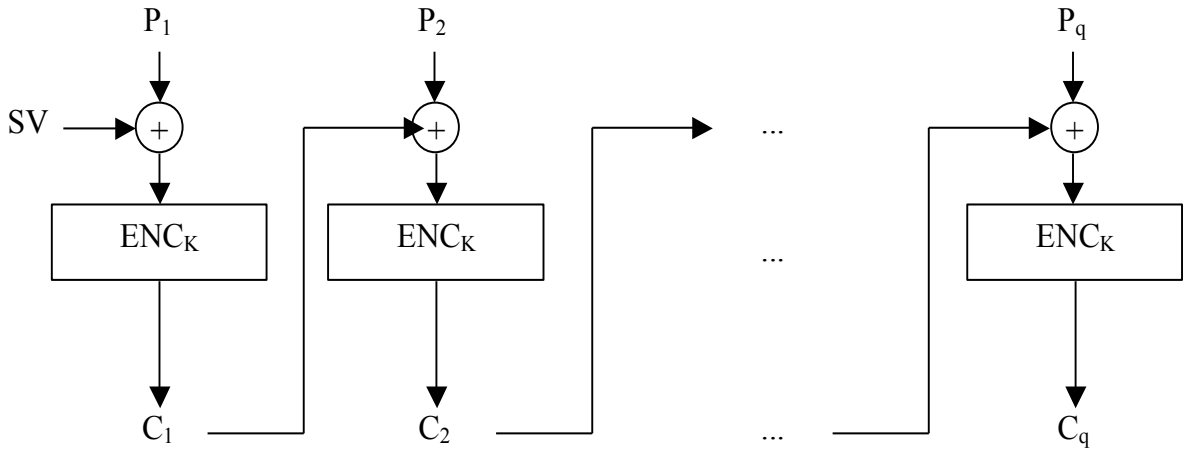

Figure 4: CBC enciphering scheme

In case of data **encryption**, the output is the string C obtained by chaining C_1,C_2,...,C_q.

In case of **MAC** computation, the output is the last block (C_q).

The consequent deciphering operation is:

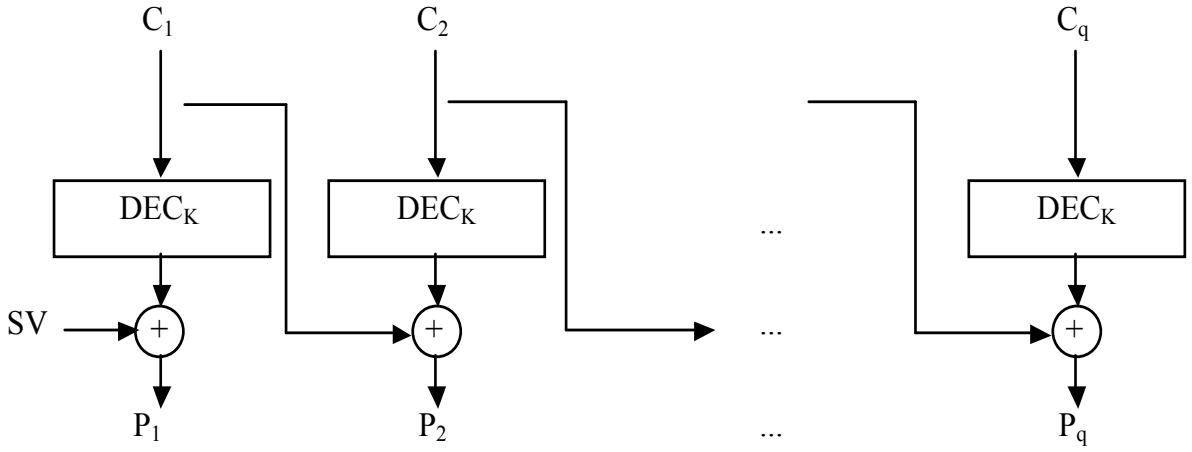

Figure 5: CBC deciphering scheme

13.2.4 MAC3

The MAC3 (Message Authentication Code) is a key-dependent one-way hash function. This mechanism is very useful to provide authenticity without secrecy. It uses a symmetric-key algorithm with a 16 or 24 bytes key.

The following figure shows the scheme of MAC 3. The SV is a string of 64-bit all set to 0.

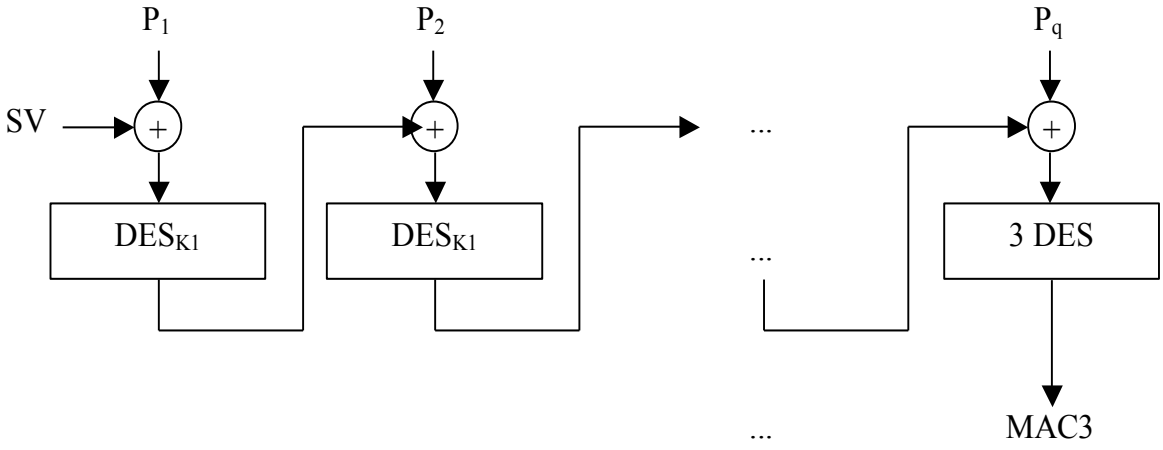

Figure 6: MAC3 computation scheme

13.3 Padding schemes

The following table resumes the padding used in the various card functionalities and in the APDUs.

FUNCTION		**DES/3DES**	**RSA**
PSO_DEC	Padding indicator	**ISO/IEC 9797 Mode 2**	**PKCS#1 V1.5 Block-type 2**
PSO_ENC	00	The padding is removed from the deciphered text	The padding is removed from the deciphered text
PSO_CDS		N/A	**PKCS#1 BT1**
SM_ENC		**ISO/IEC 9797 Mode 2**	N/A
SM_SIG		**ISO/IEC 9797 Mode 1**	N/A
EXT AUTH		**ISO/IEC 9797 Mode 1**	**PKCS#1 BT1**