DDU – Documento Unificato
Functional Specification

V1.0.0

Document History

Version	Dates	Nature of Modification
1.0.0	20/04/2014	draft

Table of Contents

TABLES

Change History

This is a draft of the first version.

1. Introduction

This document is the Functional Specification of the Italian "Documento Unificato" (DDU). It describes how the card works, seen from its external interface, without dealing with the implementation details.

The behavior described hereby is intended as mandatory. A DDU 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.

2. Audience

This specification is addressed to:

- Card Manufacturers, and mask manufacturers in general, who need to implement the DDU
- Middleware providers, implementing the first layer of software talking to the DDU, needing to know the APDU interface during the use phase
- Services Centers, in charge of the personalization of the cards, who will need the commands to personalize and test the cards before the use phase

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 prepersonalization phase of the card. Subsequent phases are disciplined by this specification.

4. Reference Documents

1	ISO/IEC	1997	ISO/IEC 7816-3 second edition: Signal and transmission protocols
2	ISO/IEC	1995	ISO/IEC 7816-4 Interindustry commands for interchange
3	ISO/IEC	1994	ISO/IEC 7816-5 Numbering System and registration procedure for application identifiers
4	ISO/IEC	1998	ISO/IEC 7816-8 Security related interindustry commands
5	ISO/IEC	2000	ISO/IEC 7816-9 Additional interindustry commands and security attributes
6	ANCI	January 11, 2000	Carta d' Identità Elettronica, Specifiche dei Comandi del Sistema Operativo (APDU), version 1

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
DDU	Documento Unificato
DF	Directory File
DER	Distinguished Encoding Rules
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
PDA	Personal Digital Assistant
PDC	Patient Data Card
PIN	Personal Identification Number
PP	Protection Profile
PPSC	Pre Personalization Secure Code
PSO	Perform Security Operation
RFU	Reserved for Future Use
RSA2	RSA key with 2048 bitlenght
SCD	Signature-Creation Data
SD	Signatory's data
SDO	Signed Data Object
SE	Security Environment
SECI	Security Environment Control
SEO	Security Environment Object
SF	Security Function SFI Short File
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 X	(lower case or upper case x): any bit value, any group of bit value. Examples: 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 5 Volts.
The card shall not require a Vpp.

7. Communication Protocols

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 in CLK cycles
9600 (default)	372	1	01h or 11h	372
115200	372	12	08h or 18h	31

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.

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 shall be 00h (no Vpp 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
High 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.
6B	H2	Y	11 bytes of Pre-Issuing Data
XXh	H3	Y	Pre-Issuing data byte 1: ICM – IC manufacturer, coded as in ISO 7816
0xxx xxxx	H4	Y	Pre-Issuing data byte 2: ICT – Mask manufacturer, coding assigned by the owner of this specification
XXXXh	H5/H6	Y	Pre-Issuing data byte 3/4: Operating system version, assigned by the OS manufactorer
01h	H7	Y	Pre-Issuing data byte 5: DD1: ATR coding version: first version
XXh	H8	Y	Pre-Issuing data byte 6: DD2: Netlink card type, as defined by referenced Netlink specifications.
0Xh	H9	Y	Pre-Issuing data byte 7: DD3: Security Certification tag: 00 – card not certified 01 – card certified

44h	H10	Y	Pre-Issuing data byte 8: DD4: 'D'
4Dh	H11	Y	Pre-Issuing data byte 9: DD5: 'D'
55h	H12	Y	Pre-Issuing data byte 10: DD6: 'U'
10h	H13	Y	Pre-Issuing data byte 11 DD7: Application version: DDU version 1.0
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].

The historical byte H4 (ICT) is coded as follows:

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

H4 – ICT Coding

9. Card Capabilities

In order to fully describe the capabilities of the card, such functionality are written to a Trasparent File under the MF with FID = XXXX with the following structure:

TAG	LENGHT	Type	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	'81' '80'	b8=1: Application selection by AID b7..b2= 000000 (RFU) 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'	DDU 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 V1.0.0 = '10'
'47'	**'04'**	**Card capabilities tag**		**Tag for next 4 bytes**
		Card capabilities data byte 1 *Selection method*	'90'	**DF selection** b8=1: DF selection by path b5=1: DF selection using file identifier **EF selection** b3=0: file selection using short file identifier is NOT supported
		Card capabilities data byte 2 *Data coding byte*	'01'	b4...b1 = 0001: data unit size is 1 byte
		Card capabilities data byte 3 Miscellaneous	'C0', '80', 'D0', '90', '40', '50', '10', '00'	b8=1: command chaining is supported b8=0: command chaining is NOT supported b7=0: Extended Lc and Le fields NOT supported b7=1: Extended Lc and Le

				fields supported b5, b4=00 : no logical channel is supported b5, b4=10 : channel number assignment by the card Maximum number of channels supported :4
		Card capabilities data byte 4 Secure messaging TLV coding	‘C0’ ‘D0’ ‘E0’ ‘F0’ ‘40’ ‘50’ ‘60’ ‘70’	b8 = 1 Secure Messaging TLV coding is BER-TLV TLV for short format APDU b7 = 1 Secure Messaging TLV coding is Simple-TLV for short format APDU b6 = 1 Secure Messaging TLV coding is BER-TLV for extended lenght format APDU b5 = 1 Secure Messaging TLV coding is Simple-TLV for extended lenght format APDU b4...b1 = 0000 The value b7 = 1 is mandatory.
‘E0’	’36’	Miscellaneous	‘XX…XX’	12 concatenated data objects with tag ‘02’ (Universal class). ‘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. '02' L .xx .xx. = FID of Elementary file used for reading/writing data, used by APDU command, that exceeds 256 bytes **without** secure messaging. Such Elementary File is used only by cards without extended length or command chaining capabilities. If such feature is not implemented then FID is 0000. '02' L .xx .xx. = FID of Elementary file used for reading/writing data, used by APDU command, that exceeds 256 bytes **with** secure messaging. Such Elementary File is used only by cards without extended length or command chaining capabilities. If such feature is not implemented then FID is 0000. '02' L .xx .xx.xx. = Class, Option and Algorithm byte for RSA2 EXP-ENCRYPT/DECRYPT 02' L .xx .xx.xx. = Class, Option and Algorithm byte for RSA2 MOD-ENCRYPT/DECRYPT 02' L .xx .xx.xx. = Class, Option and Algorithm byte for RSA2 EXP-SIGN 02' L .xx .xx.xx. = Class, Option and Algorithm byte for RSA2 MOD-SIGN 02' L .xx .xx.xx. = Class, Option and Algorithm byte for RSA2 EXP-EXTERNAL_AUTH 02' L .xx .xx.xx. = Class, Option and Algorithm byte for RSA2 MOD-EXTERNAL_AUTH

'82'	**'02'**	**Status indicator**		**Tag for next 2 bytes**
	Status Word	'9000'	SW1+ SW2	

10. Application Selection

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

11. Data Structures

The DDU 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.

11.1 File Categories

The DDU file system of 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.

11.2 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 TVL structured records.
- **EF Cyclic:** Cyclic file with fixed record length.

11.2.1 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.

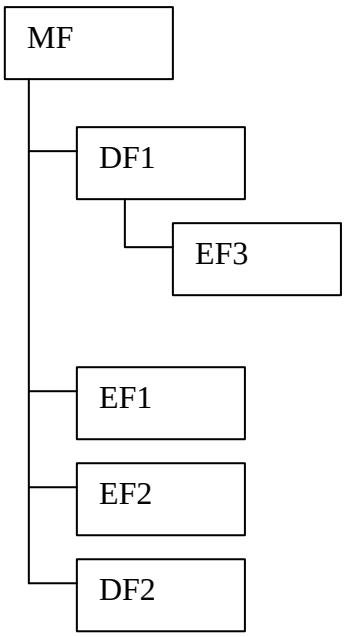

Figure 1: Example of a tree representation of a file structure

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**.**

A minimum nesting level of 8 DF has to be supported.

11.2.2 File ID

A file identifier coded on 2 bytes references each file. In order to select unambiguously any file by its identifier, all files immediately under a given DF shall have different file identifiers.

Some ID are reserved by ISO:

File ID	File Name	Description
3F00h	Master File (MF)	File system root
3FFFh	Not used	Reserved
FFFFh	Not used	Reserved

Table 1: Reserved File ID

11.2.3 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*.

11.2.4 Use of the files

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

These are the attributes of an Elementary File:

Attribute	Description
ID	2-byte File IDentifier
File Type	Transparent, Fixed Len or Variable Len or Cyclic Record EF
Size	Net file size, without system overhead
Rec number	number of records in a file, max 254

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

These are the operations possible on the current DF, and the related commands:

Creation of a DF	Create File
Creation of a EF	Create File
Attribute Modification	Put Data
File selection	Select

These are the operations possible on the current EF:

Add a record (record EF only)	Append Record
Read a record (record EF only)	Read Record

Write/Update a record	Update Record
Read binary data	Read Binary
Write/Update binary data	Update Binary
File selection	Select

Note: in this specification the ISO 7816-4 commands Write Record and Write Binary are not supported. For the sake of simplicity, the terms 'write and 'update' are equivalent in this specification and both refer to the operation defined by ISO 7816-4 as 'update'

All the commands listed above, together with the Select File command are globally referred to as "File Related Commands"

All operations acting on files are conditioned to the verification of specific security conditions, as detailed in the chapter 11.

12. 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.

12.1 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
FF **"(or 66h)"**	NEVER

Table 2: AC values

The values ALWAYS and NEVER mean that the operation is respectively always or never allowed.

*In order to identify the AC Never the Byte Value "66h" can be **optionally** used.*

*This additional Access Condition can be **optionally** used on a temporary BSO to speed up the personalization procedure.*

The auxiliary temporary "66h" object can inherit access privileges (ie. write on read only files) which are needed during the personalization process but are immediately disabled when the session closes.

Therefore the auxiliary object will be invalidated irreversibly so that all the AC configured on the 0x 66 value are inaccessible and correspond to the value 0 x FF = Never.

In the other case, the value of the AC is the ID of a TEST BSO (see chapter 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 No.	Access Condition	Protected Command
1	RFU	-
2	AC UPDATE	PUT DATA DATA_OCI PUT DATA DATA_SECI
3	AC APPEND	PUT DATA DATA_OCI PUT DATA DATA_SECI
4,5,6	RFU	-
7	AC ADMIN	PUT DATA DATA_FCI
8	AC CREATE	CREATE FILE
9	RFU	-

Table 3: MF/DF AC coding

Byte No.	Access Condition	Protected Command
1	AC READ	READ BINARY, READ RECORD
2	AC UPDATE	UPDATE BINARY, UPDATE RECORD
3	AC APPEND	APPEND RECORD
4,5,6	RFU	-
7	AC ADMIN	PUT DATA DATA_FCI
8,9	RFU	-

Table 4: EF AC coding

12.2 Base Security Object (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 Identifier) that must be unique at DF level.

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

TEST BSO

PIN objects - used in PIN verification, contains a byte string to be used as PIN

C/R TEST - used in Challenge Response protocols, contains a key

LOGICAL - 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 Title	Length	Value and Description
BSO Identifier	2	See description below
Option Byte	1	See description below
Flags Byte	1	See description below
Algorithm Byte	1	See description below
Error count Byte	1	See description below
Use Count Byte	1	See description below
RFU	1	Set to FFh
Validity Counter	1	See description below
Minimum length Byte	1	See description below
BSO AC	7	Access Conditions bytes as received in PUT DATA DATA_OCI
BSO SM (optional)	16	Secure messaging bytes as received in PUT DATA DATA_OCI (Optional)
BSO Body Length	1	The length of the BSO Body field
BSO body	N	PIN, Key, etc.

Table 5: BSO structure

Byte	Description	Value
1	BSO Class: Used to code the type of the object:	
	First or only key component for Authentication (TEST object)	00h
	Second key component for Authentication (TEST object)	01h
	Secure Messaging key (SM object)	10h

	First or only key component for Encryption, Decryption and Digital Signature (PSO object)	20h
	Second key component for Encryption, Decryption and Digital Signature (PSO object)	21h
2	BSO ID	01h..1Fh

Table 6: BSO Identifier

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

The BSO ID byte has to be unique under a given DF and within each of the following classes:

- TEST BSO
- SM BSO
- PSO BSO

Example: there can be only one TEST BSO with ID 01h under the MF.

12.2.1 Option Byte

A number of processing options are fixed in this byte. All values are mandated.

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
· RSA2 private key Exponent for ENC/DEC	See Card Capabilities table
· RSA2 private key Modulus for ENC/DEC	See Card Capabilities table
· RSA2 public key Exponent for Authentication	See Card Capabilities table
· RSA2 public key Modulus for Ext. Authentication	See Card Capabilities table

Table 7: BSO type and Option Byte

12.2.2 Flags Byte

The high nibble MUST be set to zero, while the low nibble is:

- For TEST and SM objects, the maximum number of retries for C/R or verification test.
- Not used, and set to 00h for other kinds of objects

12.2.3 Algorithm Byte

Each BSO may be used only with a defined algorithm and for a defined use. This information is coded in the Algorithm Byte.

BSO type	Algorithm	Algorithm Byte value
• RSA private key for ENC/DEC	RSA_PURE (padding leading 0)	0Ch
• RSA private key for DS • RSA public key for Ext. authentication	RSA	88h
• 3DES key for ENC/DEC • 3DES key for SM	3DES CBC	03h
• 3DES key for authentication • 3DES key for SM	3DES (auth) MAC3 (SM)	82h
• PIN	Secure Verify	87h
• LOGICAL	Logical AND/OR	7Fh

Table 8: BSO Algorithm byte

Please see the appendix for a description of the cryptographic algorithms used by the DDU.

12.2.4 Error Count Byte

For the referred BSO this integer value, ranging from 00h to 0Fh, is the number of wrong retries left for PIN verification and external authentication.

This field has no meaning for BSO type LOGICAL, and MUST be set to 0Fh.

12.2.5 Use Count Byte

For all BSO type this byte MUST be set to FFh.

12.2.6 Validity Counter

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.

12.2.7 Minimum length Byte

This indicates the minimum PIN length, for PIN objects, or the minimum length of the challenge for C/R objects.

For BSO type PIN this integer value is the PIN length.

For other BSO type this byte set to 00h, to signal that there is no preset minimum, and the challenge length depends on the algorithm.

12.2.8 BSO Access Condition

These are the access condition for the BSO, according to the following table:

Byte No.	Access Condition	Protected Command
1	AC USE	EXTERNAL AUTHENTICATION PSO_CDS PSO_DEC PSO_ENC VERIFY
2	AC CHANGE	CHANGE REFERENCE DATA CHANGE KEY DATA
3	AC UNBLOCK	RESET RETRY COUNTER
4,5,6	RFU	-
7	AC GENKEYPAIR	GENERATE KEY PAIR

Table 9: BSO AC coding

12.2.9 BSO Secure Messaging (optional)

This attribute is optional and if present codes the Secure Messaging options, as indicated by the following table:

Byte No.	SM Condition	Commands involved
1	ENC USE IN	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY, EXTERNAL AUTH.

2	SIG USE IN	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY, EXTERNAL AUTH.,
3	ENC CHANGE	CHANGE REFERENCE DATA/CHANGE KEY DATA
4	SIG CHANGE	CHANGE REFERENCE DATA/CHANGE KEY DATA
5	ENC UNBLOCK	RESET RETRY COUNTER
6	SIG UNBLOCK	RESET RETRY COUNTER
7..14	Rfu	-
15	ENC USE OUT	PSO_DEC, PSO_ENC, PSO_CDS
16	SIG USE OUT	PSO_DEC, PSO_ENC, PSO_CDS

Table 10: BSO AC coding

12.2.10 BSO body length

The BSO body length is the amount of memory reserved to the BSO file body. For example a typical length for a PIN or for a DES key is 8 bytes. It is needed for objects whose body is defined after the BSO creation itself (as keys).

The body length sets if the 3DES is to be used with a 16 or 24 byte key.

12.2.11 BSO body

Contains the value of the BSO, that is:

- for PIN objects:

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

Example:

PIN value: "1234"

ByteString: "31h32h33h34hFFhFFhFFhFFh"

- 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"

12.3 CSE/SEO

The Current Security Environment is a set of references to the security objects that will be used by Perform Security Operation 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 MSE RESTORE command, that actually copies into it the values contained in the Security Environment Object referenced by the command.

A SEO has the structure shown in Table 11.

Title	Length	Value and Description
SEO Identifier	1	00h..FEh, EFh should not be used (reserved by ISO)
SEO Access Condition	2	Access Condition bytes
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 11: SE Object structure

12.3.1 SEO Identifier

The SEO identifier is 1-byte long. It must be unique at DF level.

12.3.2 SEO Access condition

Access Conditions to satisfy when performing operations with the CSE.

Byte No.	Access Condition	Protected Command
1	AC RESTORE	MSE RESTORE
2	RFU	RFU

Table 12: SEO AC coding

12.3.3 SEO Components

A SEO contains components for: CON (confidentiality), DS (Digital Signature) and TEST(Authentication).

Each component is of a given type, and is used by different commands, as shown in the following table:

SEO Component	BSO Class byte	BSO types	Commands
CON	20h	· RSA/RSA2 Private Key for ENC/DEC · 3DES key for ENC/DEC	PSO_DEC PSO_ENC
DS	20h	· RSA/RSA2 Private Key for Digital Signature	PSO_CDS
TEST	00h	· LOGICAL · PIN · 3DES key for authentication · RSA/RSA2 Public Key for authentication	EXT. AUTH VERIFY

Table 13: SEO Components

13. 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). Indeed, 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 BS objects and files are defined at object/file creation time by the commands CREATE FILE and PUT DATA (DATA_OCI).

The SM conditions for BS objects 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 a dedicated annex.

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 Get Challenge command.

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

The following sections define the SM conditions for files and BS objects.
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

13.1 SM Conditions

Byte No.	SM Condition	Commands involved
1..2	Rfu	-
3	ENC UPDATE/APPEND (objects)	PUT DATA DATA_OCI, PUT DATA DATA_SECI
4	SIG UPDATE/APPEND (objects)	PUT DATA DATA_OCI, PUT DATA DATA_SECI
5	Rfu	-
6	Rfu	-
7.. 12	Rfu	-
13	ENC ADMIN	PUT DATA DATA_FCI
14	SIG ADMIN	PUT DATA DATA_FCI
15	ENC CREATE	CREATE FILE
16	SIG CREATE	CREATE FILE
17..24	Rfu	-

Table 14: MF/DF SM coding

Byte No.	SM Condition	Commands involved
1	ENC READ OUT	READ BINARY, READ RECORD
2	SIG READ OUT	READ BINARY, READ RECORD
3	ENC UPDATE	UPDATE BINARY, UPDATE RECORD
4	SIG UPDATE	UPDATE BINARY, UPDATE RECORD
5	ENC APPEND	APPEND RECORD
6	SIG APPEND	APPEND RECORD
7..12	Rfu	-
13	ENC ADMIN	PUT DATA DATA_FCI
14	SIG ADMIN	PUT DATA DATA_FCI
15..22	Rfu	-
23	ENC READ IN	READ BINARY, READ RECORD
24	SIG READ IN	READ BINARY, READ RECORD

Table 15: EF SM coding

Byte No.	SM Condition	Commands involved
1	ENC USE IN	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY, EXTERNAL AUTH.
2	SIG USE IN	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY, EXTERNAL AUTH.
3	ENC CHANGE	CHANGE REFERENCE DATA/CHANGE KEY DATA
4	SIG CHANGE	CHANGE REFERENCE DATA/CHANGE KEY DATA
5	ENC UNBLOCK	RESET RETRY COUNTER
6	SIG UNBLOCK	RESET RETRY COUNTER
7..14	Rfu	-
15	ENC USE OUT	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY.
16	SIG USE OUT	PSO_DEC, PSO_ENC, PSO_CDS, VERIFY.

Table 16: BS SM coding

13.2 SM_SIG

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

The MAC computation 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	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

HEADER BLOCK structure:

Random Number	CLA	INS	P1	P2	Padding
N_RAND	**XCh**	xx	xx	xx	**ISO/IEC 9797 method 1**[1]
n*8 bytes	1 byte	1 byte	1 byte	1 byte	4 bytes
(n+1) * 8 bytes / 8 bytes					
HEADER BLOCK					

- CLA: should be set to XCh
- N_RAND: random number generated by GET CHALLENGE.
- P_BYTES: always 4 bytes used as padding

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

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	xx	xx	Xx	u	**Data**	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

Plain Text Block Structure:

P_T_O			
T	**L**	**V**	**Padding**
81h	U	**Data**	**ISO / IEC 9797 method 1**[2]
1 byte	1 byte	u bytes	S = 0…7 bytes
(u + 2 + s) bytes			

[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.

[2] The padding used is ISO/IEC 9797 method 1 for the DDU.

P_T_B

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

Compute the MAC:

HEADER BLOCK	P_T_B
(n+1)*8 bytes	(u + 2 + s) bytes
MAC	

MAC Object structure:

T	L	V
8Eh	08h	**MAC**
1 byte	1 byte	8 bytes
10 bytes		
MAC_OBJ		

Step 4: APDU command set up in SM format

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	xx	xx	xx	u	Data	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

APDU command structure in SM_SIG mode:

CLA	INS	P1	P2	Lc	Data Field		Le
XCh	xx	xx	xx	w=u+12	P_T_O	MAC_OBJ	xx
1 byte	1 byte	1 byte	1 byte	1 byte	(u + 2) bytes	10 bytes	1 byte
APDU Command in SM_SIG (u + 18) bytes							

Note: if u=0, for backward compatibility two formats for P_T_O are possible:

1. Plain Text Object = '81h' '00h' or
2. Plain Text Object = empty, ie 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.

13.3 APDU Command in SM_ENC

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	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

Compute the cipher text:

Data Field	ISO Padding
Data	ISO/IEC 9797 method 2[3]
u bytes	p = 1 ...8 bytes
(u + p) bytes	
ENC	

Cipher Text object structure:

T	L	V	
		Padding byte	**Cipher text**
87h	u + p + 1	01h	**ENC**
1 byte	1 byte	1 byte	(u + p) bytes
(u + p + 3) bytes			
C_T_O			

Step 2: APDU command set up in SM format

[3] 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.

APDU command structure in SM_ENC mode:

CLA	INS	P1	P2	Lc	Data Field	Le
XCh	xx	xx	xx	W	**C_T_O**	xx
1 byte	1 byte	1 byte	1 byte	1 byte	w = (u + p + 3) bytes	1 byte
APDU Command in SM_ENC						

13.4 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

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	xx	xx	xx	u	Data	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

HEADER BLOCK structure:

Random Number	CLA	INS	P1	P2	Padding
N_RAND	XCh	xx	Xx	Xx	ISO/IEC 9797 method 1[4]
n*8 bytes	1 byte	1 byte	1 byte	1 byte	4 bytes
(n+1) * 8 bytes					
HEADER BLOCK					

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

[4] The padding used is ISO/IEC 9797 method 1 for DDU.

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	xx	xx	xx	u	Data	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

Compute the cipher text:

Data Field	ISO Padding
Data	ISO/IEC 9797 method 2
u bytes	p = 1 ...8 bytes
(u + p) bytes	
ENC	

Cipher Text object structure:

T	L	V	
		Padding byte	**Cipher text**
87h	u + p + 1	01	**ENC**
1 byte	1 byte	1 byte	(u + p) bytes
(u + p + 3) bytes			
C_T_O			

Cipher Text Block structure:

Cipher Text Object	Padding
C_T_O	ISO/IEC 9797 method 1 or 2[5]
(u + p + 3) bytes	s = 0 ...7 bytes
(u + p + 3 + s) bytes	
C_T_B	

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

Compute the MAC:

HEADER BLOCK	C_T_B

[5] The padding used is ISO/IEC 9797 method 1 for the DDU.

(n+1)*8 bytes	(u + p + 3 + s) bytes
MAC	

MAC Object structure:

T	L	V
8Eh	08h	**MAC**
1 byte	1 byte	8 bytes
10 bytes		
MAC_OBJ		

Step 4: APDU command set up in SM format

Original APDU command structure:

CLA	INS	P1	P2	Lc	Data Field	Le
X0h	xx	xx	xx	u	Data	
1 byte	1 byte	1 byte	1 byte	1 byte	u bytes	1 byte

APDU command structure in SM_SIG mode:

CLA	INS	P1	P2	Lc	Data Field		Le
XCh	xx	xx	xx	w=u+p+13	**C_T_O**	**MAC_OBJ**	xx
1 byte	1 byte	1 byte	1 byte	1 byte	(u + p + 3) bytes	10 bytes	1 byte
APDU Command in SM_SIG and SM_ENC (u + p + 19) bytes							

13.5 SM Response in SM_ENC

Response Data Field r bytes	SW1-SW2

Step1.

Response Data Field r bytes	Padding ISO 9797 mode 2
Data	80h…00h
r bytes	p = 1….8 bytes
Data to encipher (r+p) bytes	

Data input to DES/3DES algorithm to compute the ENC field used in the next step

Step2.

T	L	V	
		Padding Indicator	Enciphered Response
0x87	r+p+1	0x01	**ENC**
1 byte	1 byte	1 byte	(r+p) bytes
(r+p+3) bytes			
Cipher Text Object (r+p+3) bytes			

The card response in secure messaging mode SM_ENC is the Ciphered response object and the status words.

Cipher Text Object (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 of the data, the command sent is:

0Ch B0h 00h 00h 0Bh

13.6 SM Response in SM_SIG

Step1**. Building the Header Block as defined for the Command mode SM_SIG**

Step2. **Building the Plain Text Object and the Plain Text Block:**

Response Data Field r bytes	**SW1-SW2**

Plain Text Object			**Padding ISO 9797-mode 1**
T	**L**	**V**	**Padding Bytes**
81h	r byte (net length)	**Response Data Field**	00h…00h
1 byte	1 byte	r byte	s bytes
(r+s+2) bytes			
Plain Text Block (r+s+2) bytes			

Step3. **MAC computation and building the MAC Object :**

Header Block	**Plain Text Block**
(n+1) * 8 bytes	(r+s+2) bytes
Input to MAC computation	

T	**L**	**V**
8Eh	08h	**MAC**
1 byte	1 byte	8 byte
MAC Object 10 bytes		

Step4. **Card response building**

Plain Text Object (r+2) bytes	**MAC Object 10 bytes**	
Card Response in SM_SIG (r+12) bytes		**SW1-SW2**

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

Note:

Due to the 12bytes overhead, the maximum net length allowed is 256-12=244 bytes

13.7 SM Response in SM_ENC_SIG

Step1**. Building the Header Block as defined for the Command mode SM_SIG**

Step2**. Building the Cipher Text Object**

Response Data Field r bytes	SW1-SW2

Response Data Field r bytes	Padding ISO 9797 mode 2
Data	80h…00h
r bytes (net length)	p = 1….8 bytes
Data to encipher (r+p) bytes	

Data input to DES/3DES algorithm to compute the ENC field used in the next step

T	L	V	
		Padding Indicator	Enciphered Response
0x87	r+p+1	0x01	**ENC**
1 byte	1 byte	1 byte	(r+p) bytes
(r+p+3) bytes			
Cipher Text Object (r+p+3) bytes			

Step3**. Building the Cipher Text Block**

Cipher Text Object	Padding ISO 9797-mode 1
(r+p+3) byte	s bytes
(r+p+3+s) bytes	
Cipher Text Block (r+p+s+3) bytes	

Step4. **MAC computation and building the MAC Object :**

Header Block	Cipher Text Block
(n+1) * 8 bytes	(r+p+s+3) bytes
Input to MAC computation	

T	L	V
8Eh	08h	**MAC**
1 byte	1 byte	8 byte
MAC Object 10 bytes		

Step5. **Card response building**

Cipher Text Object (r+p+3) bytes	MAC Object 10 bytes	
Card Response in SM_ENC_SIG (r+p+13) bytes		**SW1-SW2**

14. APDU Reference

The DDU needs a subset of the APDU specified by the ISO norms. Not all the ISO specified operating modes of the supported commands are needed.

This is the list of the APDU commands used in the DDU. For detailed information on the operating modes see in the description of the single APDU command.

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

APDU	CLA	INS
PUT DATA	0Xh	DAh
CREATE FILE	0Xh	E0h
SELECT FILE	00h	A4h
APPEND RECORD	0Xh	E2h
CHANGE KEY DATA	9Xh	24h
CHANGE REFERENCE DATA	0Xh	24h
EXTERNAL AUTHENTICATE	0Xh	82h
GENERATE KEY PAIR	0Xh	46h
GET CHALLENGE	00h	84h
MSE	00h	22h
PSO_CDS	0Xh	2Ah
PSO_DEC	0Xh	2Ah
PSO_ENC	0Xh	2Ah
READ BINARY	0Xh	B0h
READ RECORD	0Xh	B2h
RESET RETRY COUNTER	0Xh	2Ch
UPDATE BINARY	0Xh	D6h
UPDATE RECORD	0Xh	DCh
VERIFY	0Xh	20h
GIVE RANDOM	80h	86h

Table 17: APDUs supported by DDU

14.1 PUT DATA

CLA	0xh
INS	DAh
P1	See *Table 19*
P2	See *Table 19*
P3	According to P1 and P2
Data Field	DATA_OCI/DATA_FCI/DATA_SECI

Table 18: PUT DATA command

P1	P2	Description
01h	6Eh	Creation and administration of the following types of BSO: • *RSA/RSA2 KPRI EXP-CRYPT/DECRYPT* • *RSA/RSA2 KPRI MOD-CRYPT/DECRYPT* • *RSA/RSA2 KPRI EXP-SIGN* • *RSA/RSA2 KPRI MOD-SIGN* • *RSA/RSA2 KPUB EXP-EXT AUTH* • *RSA/RSA2 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.
01h	6Fh	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.
01h	6Dh	Creation of SE Objects. For the explanation on "Data Field" please refer to the

		DATA_SECI paragraph.

Table 19: PUT DATA, description of P1 and P2

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.

14.1.1 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 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.

Note:

The following definitions doesn’t apply to BSO of type RSA with 2048 bits key length (RSA2). In such case the format of the command is proprietary.

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

DATA_OCI Template														
OCI1 (M)			OCI2 (M)			OCI3 (M)			OCI4 (O)			OCI5 (M/E)		
T	L	V	T	L	V	T	L	V	T	L	V	T	L	V

Table 20: DATA_OCI template

For the definition of “T” and “L” of the DATA_OCI template, please refer to the following table:

OCI m / o	T	L	Value	Description
1 M	83h	2	BSO Address	BYTE 1: Object Class BYTE 2: Object ID
2 M	85h	8	BYTE 1: OPTIONS BYTE 2: FLAGS BYTE 3: ALGORITHM BYTE 4: ERROR COUNT BYTE 5: USE COUNT BYTE 6: FFh BYTE 7: VALIDITY COUNTER BYTE 8: MINIMUM LENGTH	
3 M	86h	7	Bytes used for the AC definition	
4 O	CBh	8	Bytes used for the Secure Messaging definition	
5 M	8Fh	n	Data to be inserted into the BSO: • 3DES key • Module/Exp of a private key • PIN	

Table 21: DATA_OCI description

T	L	Byte N°	Description	V (value)
83h	2	1	Object Class: **	
			RSA KPRI EXP-CRYPT/DECRYPT	21h
			RSA KPRI MOD-CRYPT/DECRYPT	20h
			RSA KPRI EXP-SIGN	21h

			RSA KPRI MOD-SIGN	20h
			RSA KPUB EXP-EXT AUTH	01h
			RSA KPUB MOD-EXT AUTH	00h
			3DES CRYPT/DECRYPT	20h
			3DES -SM	10h
			3DES -EXT AUTH	00h
			PIN	00h
			LOGICAL_OBJECT	00h
		2	Object ID: 5 significant bit only. Objects type Mod and Exp of a given key have to have the same object ID.	000xxxxxb (from 01h to 1Fh)
85h	8	1	Options: **	
			RSA KPRI EXP-CRYPT/DECRYPT	02h
			RSA KPRI MOD-CRYPT/DECRYPT	22h
			RSA KPRI EXP-SIGN	02h
			RSA KPRI MOD-SIGN	22h
			RSA KPUB EXP-EXT AUTH	01h
			RSA KPUB MOD-EXT AUTH	21h
			3DES CRYPT/DECRYPT	83h
			3DES -SM	83h
			3DES –EXT AUTH	83h
			PIN	02h
			LOGICAL_OBJECT	02h
		2	Preset Maximum Retry Counter for TEST objects: preset maximum number of consecutive wrong attempts	from 01h to 0Fh

			other objects: 00h	
		3	Algorithm: ** 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 Authentication 3DES -SM Cipher 3DES -EXT AUTH PIN LOGICAL_OBJECT	 0Ch 0Ch 88h 88h 88h 88h 03h 82h 03h 82h 87h 7Fh
		4	for PIN objects: current number of consecutive wrong attempts Other objects: 0Fh	From 00h to 0Fh
		5	Unlimited Use	FFh
		6	No Meaning	FFh
		7	Validity Counter	TEST BSO: 00h to FFh* 00h and FFh mean 'no limitation Other BSO: 00h
		8	If the object is a PIN: PIN len Other objects: 00h	
8Fh		--	BSO value	

Table 22: DATA_OCI TLV description

*typically 00h or 01h.
**for RSA2 component see Card Capabilities

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 23: DATA_OCI class, option and algorithm

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

Table 24: BSO Value (tag 8F)

14.1.2 PUT DATA - DATA_FCI Description:

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 25 and Table 26.

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.

Access Condition: the access condition to satisfy is AC_ADMIN.

The “Data Field” is composed of sequence of 3 FCI, in TLV format.

DATA_FCI template								
FCI1			FCI2			FCI3		
T	L	V	T	L	V	T	L	V

Table 25: DATA_FCI template

For the definition of “T” and “L” of the DATA_FCI template, please refer to the following table:

FCI	T	L	V (Value)	Description
1	84h	p max 10h	AID byte string	DF Name Optional
2	86h	q	Bytes for the AC definition (see tables ADMF, ACDF and ACEF).	Access Condition Optional
3	CBh	r	Bytes for the SM definition (see tables ADMF, ACDF and ACEF).	Secure Messaging Optional

Table 26: DATA_FCI description

14.1.3 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).

Access Condition: the access condition to satisfy is the AC_APPEND of the current DF for the creation of objects or AC_UPDATE for the administration

DATA_SECI								
SECI1			SECI2			SECI3		
T	L	V	T	L	V	T	L	V

Table 27: DATA_SECI template

For the definition of "T" and "L" of the DATA_SECI template, please refer to the following table:

SECI	T	L	V (Value)
1	83h	1	SEO ID (00h-FEh)
2	86h	2	Bytes for the AC definition
3	8Fh	6	Data to be inserted into the object: · 3DES key ID · ID of Modulo/Exp of a private key

Table 28: DATA_SECI description

The detailed explanation of the bytes contained in the "V" field is explained in the following table:

T	L	Byte N°	V (value)
83h	1	1	SEO ID
86h	2	1	AC for MSE RESTORE
		2	RFU

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

Table 29: DATA_SECI TLV description

14.2 CREATE FILE

CLA	0xh
INS	E0h
P1	00h
P2	00h
P3	Lc = data field length
Data	Data (see *Table 31*)

Table 30: CREATE FILE command

Create File Data							
Tag	**Length**	**Value**					
62h	N	FCI1	FCI2	FCI3	FCI4	FCI5	FCI6 (optional)

Table 31: CREATE FILE data

FCI byte	**Tag**	**Length**	**Value**		**Description**
1	80h Or 81h	02h	if Tag=80h (Transparent EF only) File size in bytes	if Tag=81h (DF only) DF Size	File size

2	82h	03h	Byte1: 01h TRANSPARENT EF 02h LINEAR FIXED EF 05h LINEAR VARIABLE TLV 06h CYCLIC 38h DF Byte2: rfu Byte3: Meaningful for Fixed Record EF only: Record Size	File type RFU Record Size
3	83h	02h	File ID	File ID
4	85h	01h	MUST be set to 01h	Reserved
5	86h	09h	Access condition	File AC
6	CBh	18h	Secure Messaging (Optional)	File SM

Table 32: CREATE FILE data FCI

Description:

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 Table 32). The sixth FCI is optional.

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.

14.3 SELECT FILE

CLA	00h
INS	A4h
P1	00h, 03h, 04h, 08h, 09h
P2	00h
P3	Lc = data field length
Data Field	Empty / FID /AID / Path
Le	00h [6]

Table 33: SELECT FILE command

Description

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

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

Table 34: SELECT FILE: P1 coding

The partial ID match selection is not required.

[6] Le = 0 means that all the available FCI data can be returned if available.

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

FCI Description

File Control Information has a TLV (Tag Length Value) format. That means that each information returned in the response data is preceded by a description Tag and the length of data.

FCI has the following format:

Tag	L	Value
6Fh (or 62h)	n	FCI1 || FCI2 || FCI3 || ...

Only the following FCI object is mandatory in case of EF selection, while all other tags in the FCI can be proprietary implementations and will be ignored by DDU applications.

Tag	Len	Description
80h or 81h	2	EF Body Length

14.4 READ BINARY

CLA	0Xh
INS	B0h
P1	Offset high
P2	Offset low
P3	Le = Number of bytes to read

Table 35: READ BINARY command

Description:

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=00h 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:

In previous implementations of DDU one of the following situations may occour:

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 the application implementation 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 00).

14.5 UPDATE BINARY

CLA	0Xh
INS	D6h
P1	Offset high
P2	Offset low
P3	Lc = Length of data field
Data Field	Data

Table 36: UPDATE BINARY command

Description:

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 access conditions for UPDATE on the current EF are satisfied.

14.6 APPEND RECORD

CLA	0Xh
INS	E2h
P1	00h
P2	00h
P3	Lc = number of bytes to be written
Data Field	Record Data

Table 37: APPEND RECORD command

Data	Empty
SW	Status condition

Table 38: APPEND RECORD answer

Description:

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 records at the end of the file, provided there is enough memory available in it. If there is not enough memory, it will be generated an error.

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 APPENO for the current EF is satisfied.

14.7 READ RECORD

CLA	0Xh
INS	B2h
P1	Record number or record identifier (00h indicates the current record)
P2	Mode to access to the file
Le	Number of byte to be read
Data Field	Empty

Table 39: READ RECORD command

Data	Data of record to read
SW	Status condition

Table 40: READ RECORD answer

Description:

This command reads the contents of one record from a EF previously selected with a SELECT FILE.

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

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 parameter 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=00h or P2 =04h)

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

P2=04h: 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.

P2=00h: read first

read the record number 1; set the record pointer to 1

P1 has to be 00h

P2=01h: read last

read the record with the highest record number; set the record pointer to the maximum

P1 has to be 00h

P2=02h: read next

read the record whose record number is one more than the current one; set the record pointer

P1 has to be 00h

P2=03h: read previous

read the record whose record number is one less than the current one; set the record pointer

P1 has to be 00h

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

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

P2=00h: read first occurrence

read the record with the tag given in P1 with the smallest record number; set the record pointer to point to the record found

P2=01h: read last occurrence

read the record with the tag given in P1 with the highest record number; set the record pointer to point to the record found

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 point to the record found

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 point to the record found

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 00).

14.8 UPDATE RECORD

CLA	0Xh
INS	DCh
P1	Record number or record identifier
P2	Designates the access-mode to the file
Lc	Number of bytes to be written
Data Field	Record Data

Table 41: UPDATE RECORD command

Data	Empty
SW	Status condition

Table 42: UPDATE RECORD answer

This command replaces the contents 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 will be generated an error.

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

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

P2=04h: update current/absolute

P1=0 : update the current record;

P1=n : update the record number n;

n has to be different from FFh.

P2=00h: update first

update the record number 1;
set the record pointer to 1;

P1 has to be 00h;

P2=01h: update last

update the record with the highest record number;

set the record pointer to the highest record number;

P1 has to be 00h;

P2=02h: update next

update the record whose record number is one more than the current one;

set the record pointer;

P1 has to be 00h;

P2=03h: update previous

update the record whose record number is one less than the current one;

set the record pointer;

P1 has to be 00h;

The use of this command on cyclic files with P2 different from 03h 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 access condition for the UPDATE function of the EF is satisfied.

14.9 VERIFY

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

Table 43Table 43: VERIFY command

Data	Empty
SW	Status condition

Table 44: VERIFY answer

Description:

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 BS object is decremented. When this number of retries reaches the value 0, the authentication mechanism is blocked.

When the comparison succeeds, the security status of the card changes. The number of retries of the BS object 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

14.10 CHANGE REFERENCE DATA

CLA	0xh
INS	24h
P1	00h (implicit test) / 01h (explicit test)
P2	PIN ID to be changed (BSO ID)
	b7 = 0 PIN under the MF b7 = 1 PIN search with backtracking b6 = 0 b5 = 0 b4 --- b0 PIN Identifier (BSO ID)
P3	Lc = m/empty (old) + n (new)
Data Field	Verify data (P1=0) or empty (P1=1) || New reference data

Table 45: CHANGE REFERENCE DATA command

Data	Empty
SW	Status condition

Table 46: CHANGE REFERENCE DATA answer

Description:

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

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 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 (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.

14.11 CHANGE KEY DATA

CLA	9xh
INS	24h
P1	Key Class (see P1 description below)
P2	b7 = 0 BSO under the MF b7 = 1 BSO search with backtracking b6 = 0 b5 = 0 b4 --- b0 BSO Identifier (BSO ID)
P3	N bytes
Data Field	New key data (see data description below)

Table 47: CHANGE KEY DATA command

Data	Empty
SW	Status condition

Table 48: CHANGE KEY DATA answer

Description:

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.

P1 is the object description.

The table below gives details:

P1 description								
Bit Number								Description
7	6	5	4	3	2	1	0	
X	X							
		1	0	0	0	0	1	RSA KPRI EXP-CRYPT/DECRYPT
		1	0	0	0	0	0	RSA KPRI MOD-CRYPT/DECRYPT
		1	0	0	0	0	1	RSA KPRI EXP-SIGN
		1	0	0	0	0	0	RSA KPRI MOD-SIGN
		0	0	0	0	0	1	RSA KPUB EXP-EXT AUTH
		0	0	0	0	0	0	RSA KPUB MOD-EXT AUTH
		1	0	0	0	0	0	3DES - CRYPT/DECRYPT
		0	1	0	0	0	0	3DES – SM
		0	0	0	0	0	0	3DES – EXT AUTH

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.

Access Condition: the access condition to satisfy is AC_ CHANGE of relevant BSO

Note:

The definitions above doesn't apply to BSO of type RSA with 2048 bits key length (RSA2). In such case the format of the command is proprietary.

14.12 EXTERNAL AUTHENTICATE

CLA	0xh
INS	82h
P1	00h
P2	BSO ID used for the Authentication =00h refer to CSE component TEST
	b7 = 0 BSO under the MF b7 = 1 BSO search with backtracking b6 = 0 b5 = 0 b4 --- b0 BSO Identifier (BSO ID)
P3	Lc = data field length
Data field	C/R Response

Table 49: EXTERNAL AUTHENTICATE command

Data	Empty
SW	Status condition

Table 50: EXTERNAL AUTHENTICATE answer

Description:

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.

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

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.

The BSO 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, the error “Conditions not satisfied” 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 his 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.

Note:

The definitions above doesn’t apply to BSO of type RSA with 2048 bits key length (RSA2). In such case the format of the command is proprietary.

14.13 RESET RETRY COUNTER

CLA	0xh
INS	2Ch
P1	=00h for PIN =XXh for other BSO
P2	BSO ID to be changed
	b7 = 0 BSO under the MF b7 = 1 BSO search with backtracking b6 = 0 b5 = 0 b4 --- b0 BSO Identifier (BSO ID)
P3	Lc = m/empty (old) + n/empty (new)
Data field	Verify data or empty + New reference data or empty

Table 51: RESET RETRY COUNTER command

Data	Empty
SW	Status condition

Table 52: RESET RETRY COUNTER answer

Description:

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.

The table below gives details:

P1 description								
Bit Number								Description
7	6	5	4	3	2	1	0	
0	0							
		0	0	0				RSA KPUB EXP-EXT AUTH
		0	0	0				RSA KPUB MOD-EXT AUTH
		0	0	0				3DES – EXT AUTH
		0	0	0				PIN
					0	0	0	The data field contains “Verification Data” and “New Reference Data”. Valid only when the object referred to by P2 has as access condition for “AC UNBLOCK” and a reference to a PIN object.
					0	0	1	The data field contains only “Verify Data”. Valid only when the object referred to by P2 has as access condition for “AC UNBLOCK” and a reference to a PIN object.
					0	1	1	The data field is empty

Table 53: RESET RETRY COUNTER, coding of P1

Note1: the mode with P1=00h is allowed for PIN objects only.

Note2: 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:

condition: P2 has to be the reference to a PIN object, whose data len is m.

condition: the object referenced by P2 has an access condition for UNBLOCK.

condition: the object referenced by P2 has a reference to a PIN object.

the first m bytes of the command Data Field are compared with the object data of the PIN object referenced by the object referenced in P2

if the comparison succeeds the reset counter is set to its maximum value, and the object data are replaced by the next n bytes in the command DataField.

P1=xxxx x001:

condition: P2 has to be the reference to a PIN object, whose data len is m.

condition: the object referenced by P2 has an access condition for UNBLOCK.

condition: the object referenced by P2 has a reference to a PIN object.

the command Data Field is compared with the object data field of the PIN object referenced by 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 Unblock of the referenced BSO has to be verified before the command is issued

14.14 GET CHALLENGE

CLA	00h
INS	84h
P1	00h
P2	00h
P3	Le
Data field	Empty

Table 54: GET CHALLENGE command

Data	Random Number (Le bytes)
SW	Status condition

Table 55: GET CHALLENGE answer

Description:

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. 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.

Security:

no access condition to satisfy.

14.15 MSE (Manage Security Environment)

CLA	00h
INS	22h
P1	see command description
P2	see command description
P3	Lc = data field length
Data Field	Data to be used in the Current Security Environment (CSE) in TLV format

Table 56: MSE command

Data	Empty
SW	Status condition

Table 57: MSE answer

Description :

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	F3h	Security Environment Object ID (SEO ID)	Empty
SET	F1h	B8h/A4h/B6h. See Table 59	See Table 59

Table 58: MSE MODES

CSE	Command MSE SET				Related Commands
CSE Component	P2 (Component TAG)	Data Field T	L	V	
Confidentiality component (CON)	B8h	83h 84h	01h	Object ID	PSO_DEC PSO_ENC
Authentication component (TEST)	A4h	83h 84h	01h	Object ID	EXT AUTH
Digital Signature component (DS)	B6h	83h 84h	01	Object ID	PSO_CDS

Table 59: MSE SET

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 several components:

Component	Used object	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	EXTERNAL AUTHENTICATION

Table 60: CSE. Used objects and related commands

14.15.1 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 which ID is in P2
- If the wanted SE object is found, this one becomes the CSE. Afterward the CSE can be used for execute the commands PSO_DEC, PSO_ENC, PSO_CDS, EXTERNAL AUTHENTICATION.

- 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

14.15.2 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

14.16 GENERATE KEY PAIR

CLA	00h
INS	46h
P1	00h
P2	00h
P3	Lc = Length of data field (Proprietary data)
Data field	Pvk (2 bytes) Pbk (2 bytes) ARMT (1 bytes) Dif_pq (1 bytes) Pub_Exp (2 bytes)
Le	Empty

Table 61: GENERATE KEY PAIR command

Data	Empty
SW	Status Condition

Table 62: GENERATE KEY PAIR answer

Description:

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 generated empty in the current DF. The GENERATE KEY PAIR will create two TLV records with tags:

  - ¾ 10h for the key module
  - ¾ 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 access condition for APPEND of the file that will contain the public key has to be verified.

Note:

The definitions above doesn't apply to BSO of type RSA with 2048 bits key length (RSA2). In such case the format of the command is proprietary.

14.17 PSO_DEC

CLA	0xh
INS	2Ah
P1	80h
P2	86h
P3	Lc = Length of data to be deciphered + 1byte for padding indicator Or
Data field	00h (padding indicator) || enciphered data Or FID of elementary file used for writing data to be deciphered
Le	Length of the deciphered data

Table 63: PSO_DEC command

Data	The deciphered data
SW	Status condition

Table 64: PSO_DEC answer

Description:

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 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

Notes:

When PSO_DEC 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).

When card has not the capability of using command with extended length nor command chaining and the PSO_DEC command must use a BSO of type RSA Key with 2048 bits key length (RSA2), then an Elementary File of type Transparent must be used in order to send data to be deciphered to the card, using the UPDATE BINARY command, and then execute the PSO DEC command. The terminal gets the deciphered data with a READ BINARY command from the same Elementary File. The data to be written into the elementary file is the data to be deciphered with the same as described in the data field of the PSO_DEC command when used without the elementary file.

14.18 PSO_ENC

CLA	0xh
INS	2Ah
P1	86h
P2	80h
P3	Lc = Length of data to be enciphered (plain text) Or Lc = 2
Data field	Input data to be enciphered Or FID of elementary file used for writing data to be ciphered
Le	Length of the enciphered data

Table 65: PSO_ENC command

Data	00h (padding indicator) || The enciphered data
SW	Status condition

Table 66: PSO_ENC answer

Description:

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.

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

Notes:

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

When card has not the capability of using command with extended length nor command chaining and the PSO_ENC command must use a BSO of type RSA Key with 2048 bits key length (RSA2), then an Elementary File of type Transparent must be used in order to send data to be ciphered to the card, using the UPDATE BINARY command, and then execute the PSO ENC command. The terminal gets the ciphered data with a READ BINARY command from the same Elementary File. The data to be written into the elementary file is the data to be enciphered.

14.19 PSO_CDS

CLA	0xh
INS	2Ah
P1	9Eh
P2	9Ah
P3	Lc = Length of data to be signed Or Lc = 2
Data field	Input data to be signed Or FID of elementary file used for writing data to be signed
Le	Length of the signed data

Table 67: PSO_CDS command

Data	Le bytes of the sign
SW	Status condition

Table 68: PSO_CDS answer

Description:

This command computes the digital signature (DS) of the input data. The DS is given in the response data field.

To use this command 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 to be signed is the DER encoding of the DigestInfo value, as stated in the PKCS#1 specification (RFC 3447).

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.

Notes:

When card has not the capability of using command with extended length nor command chaining and the PSO_CDS command must use a BSO of type RSA Key with 2048 bits key length (RSA2), then an Elementary File of type Transparent must be used in order to send data to be ciphered to the card, using the UPDATE BINARY command, and then execute the PSO CDS command. The terminal gets the ciphered data with a READ BINARY command from the same Elementary File. The data to be written into the elementary file is the data to be signed.

14.20 GIVE RANDOM

CLA	80h
INS	86h
P1	00h
P2	00h
P3	Lc = Length of data field
Data field	Data from the terminal

Table 69: Give random

Description:

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:

None.

14.21 GET RESPONSE

CLA	00h
INS	C0h
P1	00h
P2	00h
P3	Le
Data field	Empty

Data	Le bytes retrieved
SW	Status condition

Table 70: GET Response

Description:

This command allows the terminal to retrieve Le byte data when the response of a particular command doesn't fit in the maximum length of a response data field.

15. Annex A– Cryptographic Algorithms

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

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

15.1 RSA (Rivest, Shamir, Adleman)

15.1.1 Acronyms

p	First secret prime
q	Second secret prime
N	Modulus
e	Public exponent
d	Private exponent
M	Plaintext. Data before encryption
C	Cipher text. Data after encryption
RSA	Rivest Shamir Adleman

Table 71: RSA symbols legend

15.1.2 RSA description

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

$$C = M^e \bmod N.$$

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

$$M = C^d \bmod N.$$

The command GENERATE KEY PAIR in the DDU generates a key pair with 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).

14.1.3 RSA keys

The keys are stored in the DDU in a special data objects called Base security Object (BSO). Please refer to the BSO description section for a complete object description.

Since there is no way to retrieve data from BSOs, the public key is also copied in a file to allow an external access.

15.2 DES (Data Encryption Standard)

15.2.1 Acronyms

DES	Data Encryption Standard
3DES	Triple DES
K	Key for DES
K_1	The first key of 3DES
K_2	The second key of 3DES
K_3	The third key of 3DES
DK	DES with key K
$D^{-1}K$	Inverse DES with key K
$3DK_1K_2$	Triple DES with keys K_1 and K_2
CBC	Cipher Block Chaining – mode of operation for n-bit block cipher
M	Plaintext. Data before encryption
C	Cipher text. Data after encryption
EK	Encipher mechanism with key K
DK	Decipher mechanism with key K
FIPS	Federal Information Processing Standard

Table 72: DES acronyms

15.2.2 DES

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 DDU the DES is used only as Triple DES

15.2.3 Triple DES

It's a stronger variation of DES. The following figures show the of Triple DES for encipher a plaintext **M** with 2 or 3 keys.

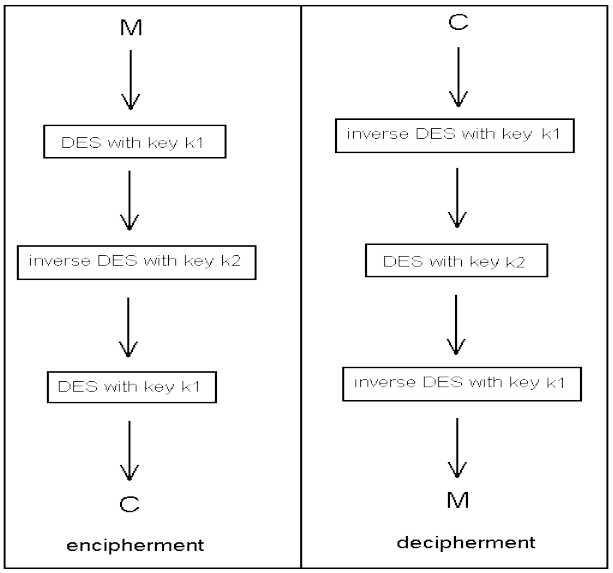

Figure 2: Triple DES with 2 keys

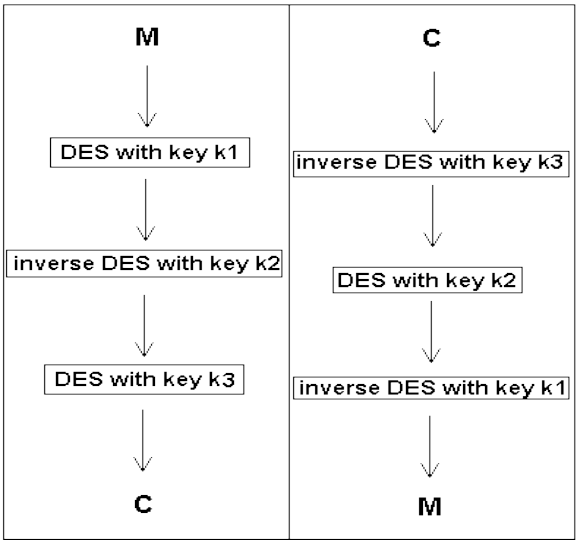

Figure 3: Triple DES with 3 keys

In other words:

$$C = Dk1\ (Dk2\text{-}1(Dk1(M)))\ M$$

$$= Dk1\text{-}1\ (Dk2(Dk1\text{-}1(C)))$$

15.2.4 Cipher Block Chaining - 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 mode 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 $\mathbf{P_1, P_2, ..., P_q}$ each of 64 bits
- A key **K**
- A Starting Vector (***SV***) of 64 bits. For the DDU it is a string of 00h.
- A sequence of **q** cipher text blocks $\mathbf{C_1, C_2, ..., C_q}$ each of 64 bits

The cipher operation:

$$\mathbf{C_1 = eK\ (P_1 \oplus SV)}$$

$$\mathbf{Ci = eK\ (P_i \oplus Ci_{-1})\ for\ i = 2, 3... q}$$

NOTE: for Message Authentication Code computation **(MAC**) the output is only the last block $\mathbf{C_q}$.

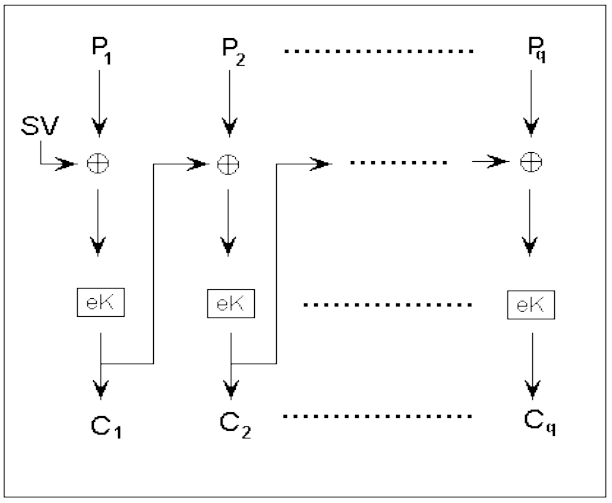

Figure 4: DES encipherment in CBC mode

The decipherment operation:

$$\mathbf{P1 = dK\ (C1) \oplus SV}$$

$$\mathbf{Pi = dK\ (Ci) \oplus Ci\text{-}1\ for\ i = 2, 3 \ldots q}$$

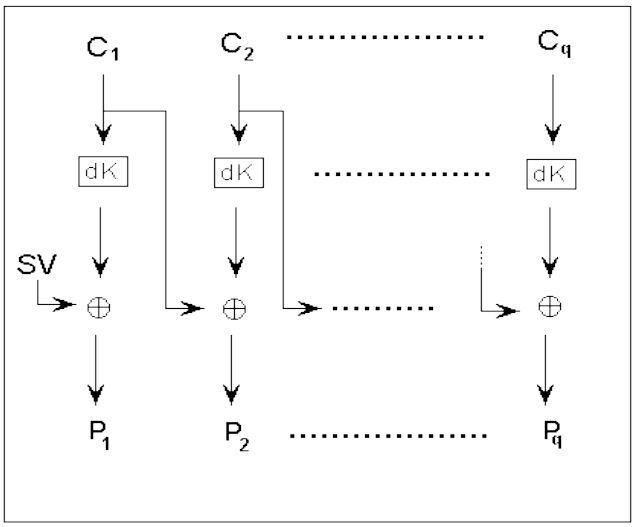

Figure 5: DES decipherment in CBC mode

15.2.5 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.

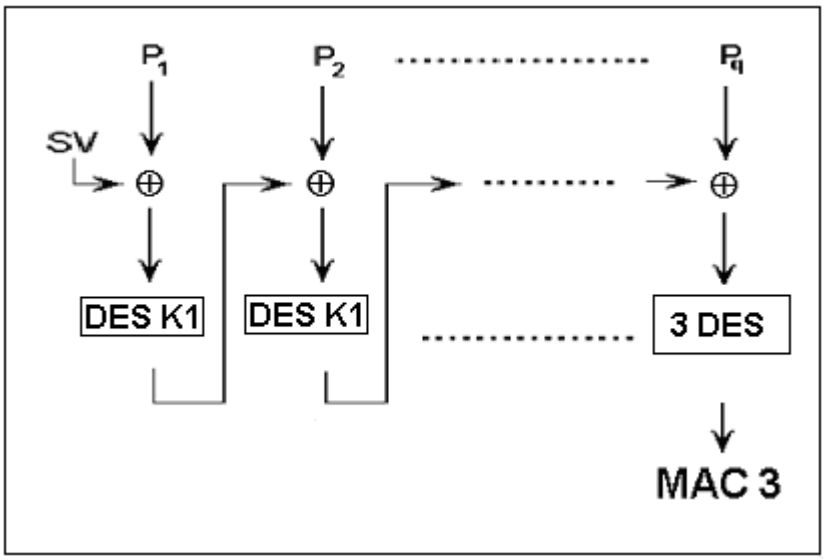

Figure 6: The scheme of MAC3

15.3 Padding schemes

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

MODE / FUNCTION		DES	RSA
	Padding Indicator		
PSO_DEC PSO_ENC	0x00	ISO/IEC 9797 Mode 2 The padding is removed from the deciphered text	PKCS#1 V1.5 Block-type 2. The padding is removed from the deciphered text
PSO_CDS		-	PKCS#1 V1.5 Block-type 1
SM_ENC		ISO/IEC 9797 Mode 2	-
SM_SIG		ISO/IEC 9797 Mode 1	-
External Authentication		ISO/IEC 9797 Mode 1	PKCS#1 V1.5 Block-type 1

16. Annex B – Status word list

SW1	SW2	Description
63h	00h	Failed Authentication xxx
65h	81h	Memory Error
67h	00h	Lc not valid
68h	81h	Logic Channel not supported
69h	81h	File type inconsistent with command
69h	82h	Security Status not satisfied
69h	83h	Authentication method blocked – BSO blocked
69h	84h	Referenced BSO is invalid
69h	85h	Condition of use not satisfied
69h	86h	No current EF selected
69h	87h	Expected SM data object missing
69h	88h	SM data object invalid
6Ah	80h	Incorrect parameters in the data field
6Ah	81h	Function not supported
6Ah	82h	File not found
6Ah	83h	Record not found
6Ah	84h	Non enough memory
6Ah	85h	Lc inconsistent with TLV structure
6Ah	86h	Incorrect P1-P2
6Ah	87h	Lc inconsistent with P1-P2
6C	00h	Le inconsistent with expected data
6D	00h	INS not valid
6E	00h	CLA not valid
6Fh	00h	General Error
6Fh	86h	Key object not found
6Fh	87h	Chaining Error
6Fh	FFh	Internal Error
90h	00h	Command successful

17. Annex C – Optional commands for Digital Signature

The DDU Functional Specifications have been issued with no specific Digital Signature Application.

According several Digital Signature schemes the digital signature key has to be delivered locked to the signatory, and can be unlocked only by the signatory. Different card manufacturers propose different solutions for this requirement. This specification suggests the use of the commands Activate/Deactivate (ISO standard), that are thus introduced in the specification as optional commands. To introduce these commands, some additions are needed in the Create File command as well.

17.1 Create File changes

After creation files are in the "Activated" status: this means that the file can be selected, read, updated, etc as normal.

Create File can set the AC for the Activate/Deactivate command as follows:

Byte No.	Access Condition	Protected Command
1	RFU	-
2	AC UPDATE	PUT DATA DATA_OCI PUT DATA DATA_SECI
3	AC APPEND	PUT DATA DATA_OCI PUT DATA DATA_SECI
4	AC Deactivate	DEACTIVATE
5	AC Activate	ACTIVATE
6	RFU	-
7	AC ADMIN	PUT DATA DATA_FCI
8	AC CREATE	CREATE FILE
9	RFU	-

Table 73: MF/DF AC coding

Byte No.	Access Condition	Protected Command
1	AC READ	READ BINARY, READ RECORD
2	AC UPDATE	UPDATE BINARY, UPDATE RECORD
3	AC APPEND	APPEND RECORD
4	AC Deactivate	DEACTIVATE
5	AC Activate	ACTIVATE
4,5,6	RFU	-
7	AC ADMIN	PUT DATA DATA_FCI
8,9	RFU	-

Table 74: EF AC coding

17.2 Deactivate File

CLA	0xh
INS	04h
P1	00h
P2	00h
P3	Lc=00h
Data Field	empty

Table 75: Deactivate File

Description

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 for Deactivate must have been fulfilled

17.3 Activate File

CLA	0xh
INS	44h
P1	00h
P2	00h
P3	Lc=00h
Data Field	empty

Table 76: Activate File

Description

This command acts on the currently selected file. It changes the status of the current file to "Activated".

Security:

AC for Activate must have been fulfilled

********** End of document **********