

**Functional Specification  
of the  
OpenPGP application  
on  
ISO Smart Card Operating Systems**

Version 3.2

Author: Achim Pietig

Author:

Achim Pietig

Lippstädter Weg 14

32756 Detmold

Germany

Email: [openpgp@pietig.com](mailto:openpgp@pietig.com)

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references.

Many thanks to all people for advice, error correction and hints to this specification.

© 2016 Achim Pietig, Detmold

The author does not assume responsibility nor give a guarantee for the correctness and/or completeness of the features and functions described in this document.

The author is unable to accept any legal responsibility or liability for incorrect and/or incomplete details and their consequences.

Furthermore, the author reserves the right to revise these specifications for technical reasons and make amendments and/or updates to the same.

## History

### V1.0 to V1.1:

- Change of access rights for command GENERATE ASYMMETRIC KEY PAIR and P1=81 (reading of public key) to always
- Adjustment of the literature
- New Data Objects for private use, with different access conditions. This optional feature is announced in Extended capabilities
- New Data Objects for key generation date/time
- Data Object 'PW Status Bytes' (C4) mandatory for GET DATA as single object

### V1.1 to V2.0:

- Correction of AID description (RID of FSFE)
- Adjustment of the literature
- Alignment with changes and innovations in ISO 7816 and EN 14890
- Change in VERIFY command and password management
- Enhancement of Extended capabilities
- Enhancement of Algorithm attributes
- Reset functionality (life cycle management)
- Definition of key import according to ISO 7816-8
- Additional Hash algorithms and different behaviour of PSO:CDS command
- Improvements for cards without MF
- New Data Objects:
  - Cardholder certificate (ISO 7816-6)
  - Extended header list for key import, supporting
    - Cardholder private key template (ISO 7816-6/-8)
    - Cardholder private key (ISO 7816-6/-8)
  - Historical bytes (ISO 7816-4/-6)
  - Algorithm attributes for PUT DATA
  - Resetting Code for PUT DATA
  - SM keys for PUT DATA
- Deletion of DOs 'FF', 'E0'-'E2'
- Support for Secure Messaging
- Support for Command Chaining
- Support for different algorithm and key length (see PUT DATA for Algorithm attributes)
- Introduction of a Resetting Code for RESET RETRY COUNTER
- Simplification in password management
- Several editorial clarifications

### V2.01 to 2.1:

- Support for AES algorithm for PSO:DEC.
- New Data Objects:
  - AES-Key for PSO:DEC
- Change access conditions for TERMINATE
- Update list of Status Bytes
- INS for TERMINATE and ACTIVATE corrected

#### V2.1 to V3.0:

- Adjustment of the literature
- Alignment with changes and innovations from ISO 7816-4, -8 and EN 419212 (former 14890):
  - Introducing Extended length information (DO 7F66)
  - Introducing General feature management (DO 7F74)
  - Add new commands GET NEXT DATA and SELECT DATA from ISO 7816-4
  - Improved description of Secure Messaging
- Extend maximum length of several DOs with variable length and announce max. length in Extended Capabilities
- New Data Objects:
  - Cardholder certificate (ISO 7816-6) for Decipher and Sign
  - Extended length information (DO 7F66)
  - UIF-DOs (D6 - D8)
- Several corrections and editorial improvements
- Obsolete definitions from old versions are deleted
- 3DES is discarded for Secure Messaging
- RSA 1024 is removed
- SHA-1 and RIPEMD-160 are removed
- Modified and additional flow charts
- Enhancement of VERIFY (additional PIN format)
- Support for ECDSA/DH algorithm:
  - Enhancement of GENERATE PUBLIC KEY PAIR
  - Enhancement of key import
  - Enhancement of PSO:CDS, DEC and INT-AUT
  - Enhancement of Algorithm attributes
- Modification of Extended Capabilities
- Introducing User Interaction Flag

#### V3.0 to V3.0.1:

- Some error corrections
- Enhancement of User Interaction Flag for DOs D6 to D8

#### V3.0.1 to V3.1:

- Update of the literature
- Enhancement of the VERIFY command (reset and get status)

#### V3.1 to V3.2:

- Update of the literature
- Update of General feature management (DO 7F74)
- Key import for ECC added (Private Key Template, Algorithm Attributes)

# TABLE OF CONTENTS

|  |           |
|--|-----------|
| <b>1 Introduction</b> .....  | <b>7</b>  |
| 1.1 Definition of Abbreviations.....                                 | 8         |
| <b>2 General Requirements</b> .....                                  | <b>9</b>  |
| <b>3 Data Structure</b> .....  | <b>10</b> |
| <b>4 Directory and Data Objects of the OpenPGP Application</b> ..... | <b>11</b> |
| 4.1 Files and Objects under the MF.....                              | 11        |
| 4.1.1 <i>EF.DIR</i> .....  | 11        |
| 4.1.2 <i>Historical bytes</i> .....                                  | 12        |
| 4.1.3 <i>EF.ATR/INFO</i> .....                                       | 12        |
| 4.1.3.1 <i>Extended length information</i> .....                     | 12        |
| 4.1.3.2 <i>General feature management</i> .....                      | 13        |
| 4.2 <i>DF.OpenPGP</i> .....  | 13        |
| 4.2.1 <i>Application Identifier (AID)</i> .....                      | 14        |
| 4.3 User Verification in the OpenPGP Application.....                | 16        |
| 4.3.1 <i>Resetting Code</i> .....                                    | 17        |
| 4.4 Data Objects (DO).....   | 18        |
| 4.4.1 <i>DOs for GET DATA</i> .....                                  | 19        |
| 4.4.2 <i>DOs for PUT DATA</i> .....                                  | 22        |
| 4.4.3 <i>DOs in Detail</i> .....                                     | 24        |
| 4.4.3.1 <i>Private Use</i> .....                                     | 24        |
| 4.4.3.2 <i>Name</i> .....  | 24        |
| 4.4.3.3 <i>Language Preferences</i> .....                            | 24        |
| 4.4.3.4 <i>Sex</i> .....   | 24        |
| 4.4.3.5 <i>User Interaction Flag</i> .....                           | 25        |
| 4.4.3.6 <i>Extended Capabilities</i> .....                           | 25        |
| 4.4.3.7 <i>Algorithm Attributes</i> .....                            | 27        |
| 4.4.3.8 <i>Supported algorithms</i> .....                            | 28        |
| 4.4.3.9 <i>Private Key Template</i> .....                            | 29        |
| 4.4.4 <i>Length Field of DOs</i> .....                               | 30        |
| <b>5 Security Architecture</b> .....                                 | <b>31</b> |
| <b>6 Historical Bytes</b> .....                                      | <b>34</b> |
| 6.1 Card Capabilities (73).....                                      | 35        |
| 6.2 Card service data (31).....                                      | 36        |
| <b>7 Commands</b> .....  | <b>37</b> |
| 7.1 Usage of ISO Standard Commands.....                              | 37        |
| 7.2 Commands in Detail.....  | 39        |
| 7.2.1 <i>SELECT</i> .....  | 39        |
| 7.2.2 <i>VERIFY</i> .....  | 40        |
| 7.2.3 <i>CHANGE REFERENCE DATA</i> .....                             | 42        |
| 7.2.4 <i>RESET RETRY COUNTER</i> .....                               | 43        |
| 7.2.5 <i>SELECT DATA</i> .....                                       | 44        |
| 7.2.6 <i>GET DATA</i> .....  | 45        |
| 7.2.7 <i>GET NEXT DATA</i> .....                                     | 46        |
| 7.2.8 <i>PUT DATA</i> .....  | 47        |
| 7.2.9 <i>GET RESPONSE</i> .....                                      | 48        |

|  |           |
|--|-----------|
| 7.2.10 PSO: COMPUTE DIGITAL SIGNATURE.....                   | 49        |
| 7.2.10.1 Hash Algorithms.....                                | 50        |
| 7.2.10.2 DigestInfo for RSA.....                             | 51        |
| 7.2.11 PSO: DECIPHER.....                                    | 53        |
| 7.2.12 INTERNAL AUTHENTICATE.....                            | 56        |
| 7.2.12.1 Client/Server Authentication.....                   | 58        |
| 7.2.13 GENERATE ASYMMETRIC KEY PAIR.....                     | 59        |
| 7.2.14 GET CHALLENGE.....                                    | 61        |
| 7.2.15 TERMINATE DF.....                                     | 62        |
| 7.2.16 ACTIVATE FILE.....                                    | 63        |
| 7.3 Command Usage under Different I/O Protocols.....         | 64        |
| 7.4 Class Byte Definitions.....                              | 64        |
| 7.5 Secure Messaging (SM).....                               | 64        |
| 7.6 Logical Channels.....                                    | 70        |
| 7.7 Command Chaining.....                                    | 71        |
| 7.8 Life Cycle Management.....                               | 72        |
| 7.9 Status Bytes.....  | 73        |
| <b>8 Literature.....</b>                                     | <b>74</b> |
| <b>9 Flow Charts.....</b>                                    | <b>76</b> |
| 9.1 Application Selection reading main DOs.....              | 77        |
| 9.2 Reading optional Data objects.....                       | 78        |
| 9.3 Compute Digital Signature.....                           | 79        |
| 9.4 Decrypt Message.....                                     | 80        |
| 9.5 Generate Private Key.....                                | 81        |
| 9.6 Client/Server Authentication.....                        | 82        |
| 9.7 Storage of Card Holder Certificates.....                 | 83        |
| <b>10 Domain parameter of supported elliptic curves.....</b> | <b>84</b> |

## 1 Introduction

This functional specification describes the OpenPGP application based on the functionality of an ISO smart card operating system. In principle it defines the interface of the application between card and terminal, in this context the OpenPGP software with a standard card reader on PC/SC and/ or CCID basis.

The solution takes care of

- use of international standards,
- avoiding of patents,
- free usage under GNU General Public License,
- independence from specific smart card operating systems (second source),
- easy enhancement for future functionality,
- international use.

Consequently this specification does not deal with the description of the global commands and data fields of the card, the security functions generally provided by the card, any features that apply to more than one application, such as transmission protocols, nor with the description of the general mechanical and electrical characteristics of the card.

In particular, the specification provides a detailed description of the data objects directly related to the applications and their respective content formats. Contents of the application data are only prescribed if they represent a constant factor of the application.

Besides the definitions in this specification a card may support any other protocols, commands, data and variants. However, the OpenPGP application (e. g. GnuPG) should use only the defined features in this specification to be compatible to different implementations.

The encoding values mentioned in the specification are stated in hexadecimal form, unless otherwise indicated.

## 1.1 Definition of Abbreviations

|       |  |
|-------|--|
| AC    | Access Condition   |
| AES   | Advanced Encryption Standard   |
| AID   | Application Identifier   |
| APDU  | Application Data Unit  |
| ATR   | Answer To Reset  |
| AUT   | AUThentication   |
| BCD   | Binary Coded Decimal   |
| CLA   | CLAss byte   |
| CP    | Control Parameter  |
| CRT   | Control Reference Template   |
| DEC   | DECipher   |
| dec.  | Decimal  |
| DF    | Dedicated File   |
| DIR   | DIRectory  |
| DO    | Data Object  |
| DSI   | Digital Signature Input  |
| ECDH  | Elliptic Curve Diffie Hellman  |
| ECDSA | Elliptic Curve Digital Signature Algorithm   |
| EF    | Elementary File  |
| FCI   | File Control Information   |
| FCP   | File Control Parameter   |
| FID   | File IDentifier  |
| INS   | INStruction byte   |
| LCS   | Life Cycle Status  |
| MF    | Master File  |
| OS    | Operating System   |
| PK    | Public Key   |
| PW    | PassWord   |
| RC    | Resetting Code   |
| RFU   | Reserved for Future Use  |
| RSA   | Rivest-Shamir-Adleman  |
| SE    | Security Environment   |
| SIG   | SIGnature  |
| SK    | Secret Key   |
| SM    | Secure Messaging   |
| UID   | Unique card IDentifier   |
| UIF   | User Interaction Flag  |
| URL   | Uniform Resource Locator   |
| UTF-8 | UCS Transformation Format 8 (compatible with 7-bit US-ASCII for all characters < 80) |
| VR    | Virtual Root data object   |



## 2 General Requirements

The OpenPGP application is designed to run under several ISO-compatible card operating systems. The application can be developed on various chips and by different manufacturers. For all implementations, the following requirements should be fulfilled.

### Card ->

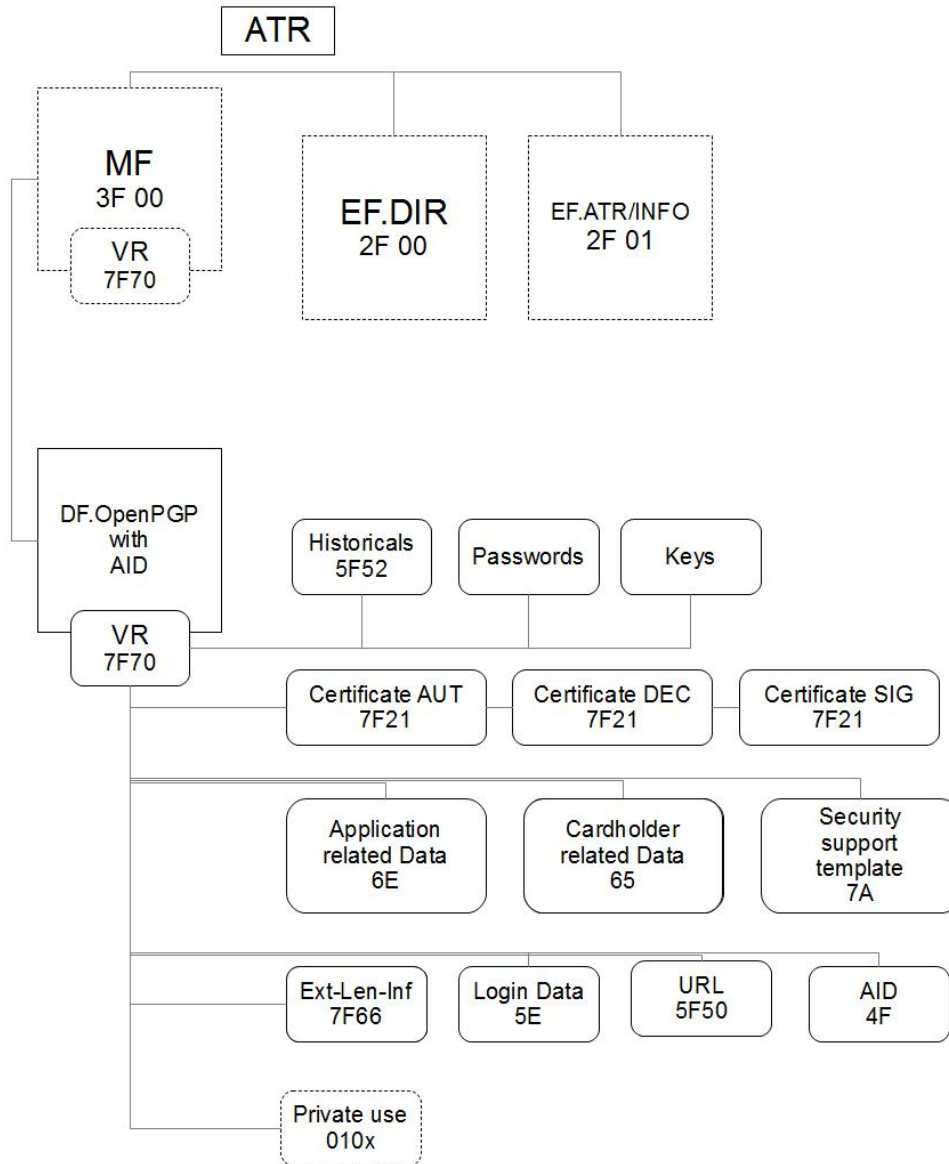
- The OpenPGP application does evaluate Historical bytes for 'Card capabilities'. For that reason the card shall provide a 'DO Historical bytes' at application level.
- The OpenPGP application may evaluate "Extended length information". For that reason the card should provide the content as DO at application level.
- As single transmission protocol any ISO protocol is allowed.
  - T=1 is preferred for cards with contacts (extended length, chaining for APDUs > 2048 bytes).
- The card may support different transmission protocols.
- High speed modes are requested (maximum as possible for the chip).
- Extended length (Lc and Le) fields are recommended (at least 2048 bytes for a single APDU under T=1)
  - The card shall announce this feature in 'Card capabilities'.
  - If Extended length is not supported (T=0), the card should support command chaining and/or ENVELOPE/GET RESPONSE for large command/response data.
- Contactless cards should support random UID (Unique card Identifier) to avoid collecting user profiles (Datenschutz).

### Reader (informative) ->

- A common driver (CCID, PC/SC or CT-API) shall be supported.
  - The driver should be available for several platforms (e. g. Win32/64, Linux, Macintosh).
- T=1 shall be supported for cards with contacts (T=0 optional).
- High-Speed protocols should be supported.
- Extended length should be supported with a minimum of 2048 bytes for APDUs for in- and output.
- Under T=0 ENVELOPE and GET RESPONSE are required to transport long APDUs. Command chaining should be supported.

### 3 Data Structure

The following diagram gives an overview over files and data objects (DO) relevant for the OpenPGP application. Security related data (e. g. keys, passwords) are stored in accordance with the used OS (files, data objects or other).



VR = Virtual Root DO



## 4 Directory and Data Objects of the OpenPGP Application

The DF.OpenPGP directory and the data objects contained therein constitute the OpenPGP application. On the card several other applications may exist in specific Dedicated Files (DF).

### 4.1 Files and Objects under the MF

The OpenPGP application uses its own set of data, including keys and passwords. No files/data of the MF or other DFs are needed for the application. However the operating system may store common data, like passwords, shareable in the card and use them for several applications.

#### 4.1.1 EF.DIR

This optional file under the MF (file identifier: '2F00') may contain one or several application templates and/or application identifiers as defined in ISO/IEC 7816-4. The data file is not requested and evaluated by the OpenPGP application, but may be used to declare the application to 3<sup>rd</sup> parties. The following entries should be added in that case:

- Application Identifier (tag '4F'), only the significant values should be used (6 bytes = 'D27600012401')
- Application label (tag '50'), the application label should contain the following UTF-8 encoded text: OpenPGP

Example:

An entry in EF.DIR is an application template (Tag 61) in most cases. The template is stored in a record or is appended to a previous template in case of a binary structure of the EF.DIR. An entry has the content:

```
61 11 4F 06 D27600012401 50 07 'OpenPGP'
```

If the card indicates DO handling for EF.DIR, then it should support the GET DATA command for reading all DOs in the EF at once ('00CB 2F00 02 5C00 00') directly after a reset.

### 4.1.2 Historical bytes

In the Answer To Reset (ATR) of a card with contacts Historical bytes may be present. These bytes are available as DO ('5F52') for all types of smart cards on application level also and are relevant only for the selected application in that case. In the OpenPGP card application the DO "Historical bytes" is available directly after SELECT as single DO or in between the "Application Related Data" ('6E').

### 4.1.3 EF.ATR/INFO

The optional file EF.ATR/INFO under the MF (file identifier: '2F01') contains "Extended length information" for APDUs, if extended length is announced in the Historical bytes.

If the card supports additional hardware like buttons or fingerprint sensors for special user interaction EF.ATR/INFO should contain the General feature management DO ('7F74') also. If no additional hardware is present this DO can be omitted.

If the card indicates DO handling for EF.ATR/INFO, then it should support the GET DATA command for reading all DOs in the EF at once ('00CB 2F01 02 5C00 00') directly after a reset.

#### 4.1.3.1 Extended length information

In the OpenPGP card application the DO "Extended length information" ('7F66') shall be available directly after SELECT as single DO and in between the "Application Related Data" ('6E'). The DO contains the following data objects.

| 7F66 | 08 | <i>Extended length information</i> |   |
|------|----|------------------------------------|---|
|      | 02 | 02                                 | Maximum number of bytes in a command APDU<br>Unsigned Integer, 2 bytes (Most Significant Bit ... Least Significant Bit) |
|      | 02 | 02                                 | Maximum number of bytes in a response APDU<br>Unsigned Integer, 2 bytes   |

The maximum length of an APDU field means the total amount of bytes sent to or received from the card (including SM) in any command. If commands exceed the announced maximum length by definition, the card shall support chaining. If a response exceeds the maximum length (e. g. GET DATA), the card answers with status bytes 61xx and the remaining data can be read with GET RESPONSE.

If Extended Length is not supported by a card or if the Extended Length information is not present, then a maximum length of 256 (dec.) bytes is assumed (short length). The card should support chaining for command and response if any defined commands have this requirement.

#### 4.1.3.2 General feature management

In the OpenPGP card application the optional DO “General feature management” ('7F74') may be available directly after SELECT as single DO and in between the “Application Related Data” ('6E'). The DO announced additional hardware for user interaction, if present the User Interaction Flag (UIF) in the related DOs shall be evaluated.

The DO contains a data object with Tag '81' with the following content (only first byte):

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning  |
|----|----|----|----|----|----|----|----|--|
| 1  | -  | -  | -  | -  | -  | -  | -  | Display (defined by ISO/IEC 7816-4)                |
| -  | 1  | -  | -  | -  | -  | -  | -  | Biometric input sensor (defined by ISO/IEC 7816-4) |
| -  | -  | 1  | -  | -  | -  | -  | -  | Button   |
| -  | -  | -  | 1  | -  | -  | -  | -  | Keypad   |
| -  | -  | -  | -  | 1  | -  | -  | -  | LED  |
| -  | -  | -  | -  | -  | 1  | -  | -  | Loudspeaker  |
| -  | -  | -  | -  | -  | -  | 1  | -  | Microphone   |
| -  | -  | -  | -  | -  | -  | -  | 1  | Touchscreen  |

Actual only the behaviour for Button is defined, e. g. '7F74 03 81 01 20' announces a button. ISO defines more bytes for additional features, they are not used by the OpenPGP card at the moment.

## 4.2 DF.OpenPGP

The directory of the OpenPGP application is stored anywhere in the card. It has no fixed File Identifier (FID), so it is easy to integrate the application in any existing context. The FID (if needed) can be chosen by the card manufacturer or any other party. The directory contains all data objects of the application. The OpenPGP application can be selected with a SELECT command directly after a Reset of a card. The SELECT command may return FCPs (File Control Parameter), but the OpenPGP application in the terminal do not need to evaluate them. A valid SELECT of an application directory sets the curConstructedDO

pointer to the Virtual Root DO and adds all data objects in the application to the current template.

#### 4.2.1 Application Identifier (AID)

The OpenPGP application is selectable by a unique application identifier (see SELECT). The AID has a length of 16 bytes (dec.) and is coded in the following way. The AID is unique for each card and it is recommended to integrate this value in certificates, e.g. for client/server authentication. The RID in the AID is registered by FSF Europe e. V.

|               | RID            | PIX         |         |              |               |       |
|---------------|----------------|-------------|---------|--------------|---------------|-------|
| Coding        | D2 76 00 01 24 | 01          | xx xx   | xx xx        | xx xx xx xx   | 00 00 |
| Length (dec.) | 5              | 1           | 2       | 2            | 4             | 2     |
| Name          | RID of FSFE    | Application | Version | Manufacturer | Serial number | RFU   |

RID Registered application provider identifier (unique identification of FSFE), ISO 7816-5

PIX Proprietary application identifier extension (defined for OpenPGP application)

Application Indication of the application (OpenPGP)

Version Version number of the application

Manufacturer Unique code for the manufacturer of the application (card)

Serial number Unique serial number

RFU Reserved for Future Use

## Application

This value (1 byte binary) specifies the application. With this definition it is possible to design different applications under control of FSF Europe e. V. in the future. The following values are defined:

|     |                                |
|-----|--------------------------------|
| 00  | Reserved                       |
| 1   | OpenPGP application (standard) |
| 2   | SmartChess                     |
| ... |                                |
| FF  | Reserved                       |

## Version

The version number (2 bytes, BCD) gives information about the current status of the application. With this value it is possible to announce updates to the outside world. The version number is defined as follows:

|              |                   |                       |
|--------------|-------------------|-----------------------|
| Byte 1       | Byte 2            |                       |
| Main version | Secondary version | (values from 00 – 99) |

Example: A version

|      |          |       |
|------|----------|-------|
| 1.0  | is coded | 01 00 |
| 2.1  |          | 02 01 |
| 11.7 |          | 11 07 |

## Manufacturer

To identify a card in open networks (e. g. key servers) and for the purpose of Log-In in local or open networks or to a single computer, it is necessary to have unique application numbers (related to a specific card). For that reason, every card manufacturer or personaliser has a unique address. This manufacturer identification is controlled by FSF Europe e. V. and given to every interested manufacturer for free. Only registered manufactures are allowed to produce applications compatible with an OpenPGP application. The system works similar to MAC addresses on network cards. The 2 bytes are coded binary and the values '0000' and 'FFFF' are reserved for test purposes.

## Serial Number

Each OpenPGP application on a card from a manufacturer/personalizer has a unique serial number. The manufacturer/personalizer is responsible that no cards with duplicate numbers will occur in the outside world (like MAC addresses in networks). The number is 4 byte long (binary) and has the format MSB .. . LSB (Most Significant Bit ... Least Significant Bit). It should start with 00 00 00 01 for the first card with an OpenPGP application of a manufacturer and normally is incremented automatically by him. However gaps in the range of numbers are allowed.

### 4.3 User Verification in the OpenPGP Application

The OpenPGP application uses two local passwords for user verification, called PW (PW1 with 6 characters/digits minimum and PW3 with 8 characters/digits minimum). PW1 is also called user-password and PW3 admin-password. PW1 (user) is the password used for signing and decryption operations, PW3 (admin) is the security officers password. PW1 is needed for everyday use of the card, while PW3 is used to manage the card.

The format of the PWs is UTF-8 (case sensitive) by default or PIN block format 2 (digits only, useful for class 2/3 readers with PIN pad) as option. The format and maximum length for each PW is declared in the 'PW status' DO and can be changed by the user by option (announced in Extended capabilities). The card checks length and format of the passwords. The storage of the PWs is dependent on the card OS, global PWs (used by other applications as well) may be used but mapped to the application as local. PW1 is used as access condition for the command PSO:CDS, PSO:DEC, INT-AUT, GET DATA and PUT DATA. The OpenPGP application uses PW3 as access condition for the commands RESET RETRY COUNTER, PUT DATA, GENERATE ASYMMETRIC KEY PAIR and TERMINATE DF. All PWs use an error counter with an initial value of 3. This error counter is readable with GET DATA. After correctly verifying the PW, the access status of the corresponding PW remains valid up to a RESET of the card, a SELECT to a different DF or an internal resetting by specific commands.

The command PSO:CDS uses PW1 in a different mode than the other commands, it is valid for one command only and has to be presented again for the next signature calculation, for that reason terminals and software should not cache the passwords of a card! This behaviour can be changed by the user by option, so that the password remains valid in the



card up to a reset or changing the application. The other passwords are valid for the complete session by default.

If the card is delivered without personalisation and/or PW letter, then a default content is assumed (UTF-8): PW1 = "123456" (6 bytes, 313233343536); PW3 = "12345678" (8 bytes, 3132333435363738). It is highly recommended that the card holder change these values.

#### **4.3.1 Resetting Code**

If, for example, the card is issued by an authority or company, the users will get a complete personalised card with keys and password. The user should be able to work with the card, but is not permitted to change card data like keys and DOs under control of the issuer. He shall know his user-password (PW1), but is not aware of the admin-password (PW3). To reset PW1 in the case of a blocked error counter, a special Resetting Code (RC) is introduced. The issuer should give the RC to the user together with his password. The Resetting Code has the same format as the password and is stored in a DO 'RC'. The maximum length is announced in PW status bytes, the minimum length is 8 letters or digits. The format is the same as for PW1, the admin can change the values (optional). The Resetting Code can be used within the command RESET RETRY COUNTER instead of the admin-password (PW3). It is only valid for resetting PW1. By default DO 'RC' is empty and the related error counter is zero, so it cannot be used. The Resetting Code has an error counter with an initial value of 3. This error counter is readable with GET DATA. The DO 'RC' can be set to any value with a PUT DATA command after correct verification of the admin-password (PW3), the error counter then is set to 3.

#### **4.4 Data Objects (DO)**

To keep the interface to terminals simple and for the reason to integrate the OpenPGP application in different card OS, all relevant data elements for the application are stored as data objects. Terminals can run the application only with the SELECT (DATA), GET (NEXT) DATA, PUT DATA and cryptographic commands. Changing of any file identifier, short file identifier, file type or file structure has no influence on the terminal interface. DOs are stored in a TLV (Tag, Length, Value) format, whenever possible definitions of ISO (e. g. 7816-6) are used. This application structure is in compliance with the latest additions of ISO 7816-4, where each DF may have a Virtual Root data object (Tag '7F70') and any DO under this root may have its own access conditions. The OpenPGP card application does not use the DO '7F70' directly.

#### 4.4.1 DOs for GET DATA

The following DOs shall be supported by the GET (NEXT) DATA command. They can be accessed at least in the OpenPGP DF of the card. Simple DOs (S) return only the value with GET DATA. Constructed DOs (C, marked yellow) are returned including their tag and length. In constructed DOs additional DOs may occur (not defined here) but are not evaluated by the OpenPGP application in the terminal. The DOs in cursive letters cannot be accessed directly with GET DATA as single DO, the OpenPGP application in the terminal uses the non-cursive DOs (mostly constructed) only. Some of the DOs are optional (marked green), the occurrence is announced in Extended capabilities. DOs may appear several times in the table, if they are defined as single DO and occur in constructed DOs also. The order of DOs in a constructed DO may vary.

| Tag  | Format | Length        | Description  |
|------|--------|---------------|--|
| 0101 | S      | 0 – max.      | Optional DO for private use (binary, proprietary), can be used to store any information. The maximum length of this DO is announced in Extended capabilities.  |
| 0102 | S      | 0 – max.      | Optional DO for private use (binary, proprietary), can be used to store any information. The maximum length of this DO is announced in Extended capabilities.  |
| 0103 | S      | 0 – max.      | Optional DO for private use (binary, proprietary), can be used to store any information. The maximum length of this DO is announced in Extended capabilities.  |
| 0104 | S      | 0 – max.      | Optional DO for private use (binary, proprietary), can be used to store any information. The maximum length of this DO is announced in Extended capabilities.  |
| 4F   | S      | 5 - 16 (dec.) | Full Application identifier (AID), ISO 7816-4  |
| 5E   | S      | 0 – max.      | Login data (binary, proprietary)<br>This DO can be used to store any information used for the Log-In process in a client/server authentication (e.g. user name of a network). The maximum length of this DO is announced in Extended capabilities. |
| 5F50 | S      | 0 – max.      | Uniform resource locator (URL, as defined in RFC 1738). The URL should contain a Link to a set of public keys in OpenPGP format, related to the card. The maximum length of this DO is announced in Extended capabilities.                         |
| 5F52 | S      | 0 – 15 (dec.) | Historical bytes, Card service data and Card capabilities shall be included, mandatory for the OpenPGP application.  |
| 7F66 | C      | 08            | Extended length information (ISO 7816-4) with maximum number of bytes for command and response.  |
| 7F74 | C      | 03            | General feature management (optional)  |
| 65   | C      | var.          | Cardholder Related Data  |
| 5B   | S      | 0 – 39 (dec.) | Name according to ISO/IEC 7501-1)  |
| 5F2D | S      | 2 - 8         | Language preferences (according to ISO 639)  |
| 5F35 | S      | 1             | Sex (according to ISO 5218)  |

| Tag  | Format | Length        | Description   |
|------|--------|---------------|---|
| 6E   | C      | var.          | Application Related Data  |
| 4F   | S      | 5 – 16 (dec.) | Application identifier (AID), ISO 7816-4  |
| 5F52 | S      | 0 – 15 (dec.) | Historical bytes, Card service data and Card capabilities shall be included, mandatory for the OpenPGP application.   |
| 7F66 | C      | 08            | Extended length information (ISO 7816-4) with maximum number of bytes for command and response.   |
| 7F74 | C      | 03            | General feature management (optional)   |
| 73   | C      | var.          | Discretionary data objects  |
| C0   | S      | 10 (dec.)     | Extended capabilities<br>Flag list  |
| C1   | S      | var.          | Algorithm attributes signature<br>1 Byte Algorithm ID, according to RFC 4880/6637 further bytes depending on algorithm (e. g. length modulus and length exponent).  |
| C2   | S      | var.          | Algorithm attributes decryption   |
| C3   | S      | var.          | Algorithm attributes authentication   |
| C4   | S      | 7             | PW status Bytes<br>1 <sup>st</sup> byte: 00 = PW1 (no. 81) only valid for one PSO:CDS command<br>01 = PW1 valid for several PSO:CDS commands<br>2 <sup>nd</sup> byte: max. length and format of PW1 (user)<br>Bit 1-7 = max. length, for PIN format 2 always '08' (real PIN length up to 12 digits)<br>Bit 8 = 0 for UTF-8 passwords<br>1 for PIN block format 2<br>3 <sup>rd</sup> byte: max. length of Resetting Code (RC) for PW1<br>4 <sup>th</sup> byte: max. length and format of PW3 (admin), see PW1<br>Byte 5, 6, 7 (first byte for PW1, second byte for Resetting Code, third byte for PW3):<br>Error counter of PW1, RC and PW3. If 00 then the corresponding PW/RC is blocked. Incorrect usage decrements the counter, correct verification sets to default value = 03. |
| C5   | S      | 60 (dec.)     | Fingerprints (binary, 20 bytes (dec.) each for Sig, Dec, Aut in that order), zero bytes indicate a not defined private key.   |
| C6   | S      | 60 (dec.)     | List of CA-Fingerprints (binary, 20 bytes (dec.) each) of "Ultimately Trusted Keys". Zero bytes indicate a free entry. May be used to verify Public Keys from servers.  |
| CD   | S      | 12 (dec.)     | List of generation dates/times of public key pairs, binary. 4 bytes, Big Endian each for Sig, Dec and Aut. Each value shall be seconds since Jan 1, 1970. Default value is 00000000 (not specified).  |

| Tag   | Format | Length   | Description   |
|-------|--------|----------|---|
| D6    | S      | 02       | User Interaction Flag (UIF) for PSO:CDS (optional):<br>If not supported, DO is not available.<br>First byte =<br>00: UIF disabled (default)<br>01: UIF enabled<br>02: UIF permanently enabled (not changeable with PUT DATA, optional)<br>Second byte = Content from General feature management ('20' for button/keypad)  |
| D7    | S      | 02       | UIF for PSO:DEC (optional):<br>See UIF for PSO:CDS  |
| D8    | S      | 02       | UIF for PSO:AUT (optional):<br>See UIF for PSO:CDS  |
| 7A    | C      | 5        | Security support template   |
| 93    | S      | 3        | <i>Digital signature counter (counts usage of Compute Digital Signature command), binary, ISO 7816-4.</i>   |
| 7F 21 | C      | 0 – max. | Cardholder certificate (each for AUT, DEC and SIG)<br>These DOs are designed to store a certificate (e. g. X.509) for the keys in the card. They can be used to identify the card in a client-server authentication, where specific non-OpenPGP-certificates are needed, for S-MIME and other x.509 related functions. The maximum length of the DOs is announced in Extended capabilities. The content should be TLV-constructed, but is out of scope of this specification. The DOs are stored in the order AUT (1 <sup>st</sup> occurrence), DEC (2 <sup>nd</sup> occurrence) and SIG (3 <sup>rd</sup> occurrence). Storing the AUT certificate at first occurrence is for downward compatibility with older versions of this specification.   |
| C4    | S      | 7        | PW Status Bytes (binary)<br>1 <sup>st</sup> byte: 00 = PW1 (no. 81) only valid for one PSO:CDS command<br>01 = PW1 valid for several PSO:CDS commands<br>2 <sup>nd</sup> byte: max. length and format of PW1 (user)<br>Bit 1-7 = max. length, for PIN format 2 always '08' (real PIN length up to 12 digits)<br>Bit 8 = 0 for UTF-8 passwords<br>1 for PIN block format 2<br>3 <sup>rd</sup> byte: max. length of Resetting Code (RC) for PW1<br>4 <sup>th</sup> byte: max. length and format of PW3 (admin), see PW1<br>Byte 5, 6, 7 (first byte for PW1, second byte for Resetting Code, third byte for PW3):<br>Error counter of PW1, RC and PW3. If 00, then the corresponding PW/RC is blocked. Incorrect usage decrements the counter, correct verification sets to default value = 03. |

#### 4.4.2 DOs for PUT DATA

The following DOs are supported by the PUT DATA command. They can be accessed at least in the OpenPGP DF of the card. The content of possibly empty DOs can be deleted with PUT DATA and empty Lc (no data).

| Tag   | Format | Length        | Description  |
|-------|--------|---------------|--|
| 0101  | S      | 0 – max.      | Optional DO for private use (binary)   |
| 0102  | S      | 0 – max.      | Optional DO for private use (binary)   |
| 0103  | S      | 0 – max.      | Optional DO for private use (binary)   |
| 0104  | S      | 0 – max.      | Optional DO for private use (binary)   |
| 4D    | C      | var.          | Extended Header list (used for optional key import), uses:<br>7F48 (Cardholder private key template)<br>5F48 (Cardholder private key)  |
| 5B    | S      | 0 – 39 (dec.) | Name   |
| 5E    | S      | 0 – max.      | Login data   |
| 5F2D  | S      | 2 - 8         | Language preferences   |
| 5F35  | S      | 1             | Sex  |
| 5F50  | S      | 0 – max.      | Uniform resource locator (URL)   |
| 7F 21 | C      | 0 – max.      | Cardholder certificate (AUT, DEC, SIG)<br>These DOs are designed to store a certificate (e.g. X.509) for the AUT-, DEC- and SIG-key in the card. The maximum length of the DOs is announced in Extended capabilities. The content should be TLV-constructed, but is out of scope of this specification. The DOs are stored in the order AUT (1 <sup>st</sup> occurrence), DEC (2 <sup>nd</sup> occurrence) and SIG (3 <sup>rd</sup> occurrence).   |
| C1    | S      | var.          | Optional DO (announced in Extended capabilities).<br>Algorithm attributes signature  |
| C2    | S      | var.          | Optional DO (announced in Extended capabilities).<br>Algorithm attributes decryption   |
| C3    | S      | var.          | Optional DO (announced in Extended capabilities).<br>Algorithm attributes authentication   |
| C4    | S      | 1             | Optional DO (announced in Extended capabilities).<br>1 <sup>st</sup> PW Status Byte (1 byte binary):<br>00 = PW1 (No. 81) only valid for one PSO:CDS command<br>01 = PW1 valid for several PSO:CDS commands<br>2nd byte: max. length and format of PW1 (user)<br>Bit 1-7 = max. length, for PIN format 2 always '08'<br>(real PIN length up to 12 digits)<br>Bit 8 = 0 for UTF-8 passwords<br>1 for PIN block format 2<br>3rd byte: max. length of Resetting Code (RC) for PW1<br>4th byte: max. length and format of PW3 (admin), see PW1 |
| C7    | S      | 20 (dec.)     | Fingerprint (binary) for signature key, format according to RFC 4880   |
| C8    | S      | 20 (dec.)     | Fingerprint (binary) for decryption key  |

| Tag | Format | Length     | Description  |
|-----|--------|------------|--|
| C9  | S      | 20 (dec.)  | Fingerprint (binary) for authentication key  |
| CA  | S      | 20 (dec.)  | 1 <sup>st</sup> CA-Fingerprint in list (binary)  |
| CB  | S      | 20 (dec.)  | 2 <sup>nd</sup> CA-Fingerprint in list (binary)  |
| CC  | S      | 20 (dec.)  | 3 <sup>rd</sup> CA-Fingerprint in list (binary)  |
| CE  | S      | 4          | Generation date/time of signature key (Big Endian, format according to RFC 4880)   |
| CF  | S      | 4          | Generation date/time of decryption key (Big Endian)  |
| D0  | S      | 4          | Generation date/time of authentication key (Big Endian)  |
| D1  | S      | 16 / 32    | Optional DO (announced in Extended capabilities) for SM.<br>SM-Key-ENC for cryptogram (16 or 32 bytes in case of AES128/256). The stored SM-Key shall match the announced algorithm in Extended capabilities.  |
| D2  | S      | 16 / 32    | Optional DO (announced in Extended capabilities) for SM.<br>SM-Key-MAC for cryptographic checksum (16 or 32 bytes in case of AES128/256). The stored SM-Key shall match the announced algorithm in Extended capabilities.  |
| D3  | S      | 0 / 8 - xx | Resetting Code, 0 or 8 – xx bytes (dec.), binary   |
| D5  | S      | 16 / 32    | Optional DO (announced in Extended capabilities) for PSO:DEC with AES (32 bytes dec. in case of AES256, 16 bytes dec. in case of AES128).  |
| D6  | S      | 02         | User Interaction Flag (UIF) for PSO:CDS (optional):<br>If not supported, DO is not available.<br>First byte =<br>00: UIF disabled (default)<br>01: UIF enabled<br>02: UIF permanently enabled (optional, a PUT DATA command shall abort if the old content is 02.<br>Changing of a value 02 may be only possible with a factory reset – Terminate/Activate).<br>Second byte = Content from General feature management ('20' for button/keypad) |
| D7  | S      | 02         | UIF for PSO:DEC (optional):<br>See UIF for PSO:CDS   |
| D8  | S      | 02         | UIF for PSO:AUT (optional):<br>See UIF for PSO:CDS   |
| F4  | C      | var.       | Optional DO (announced in Extended capabilities) for SM.<br>Container for both SM keys (ENC and MAC) with Tags D1 and D2.<br>Useful for updating or deleting both keys simultaneous.   |

### 4.4.3 DOs in Detail

The following chapter describes some DOs in detail, especially the application specific DOs.

#### 4.4.3.1 Private Use

These optional DOs can be used by the card holder, administrator or any application for proprietary data (e. g. password list). The difference between the DOs are the access conditions. The presence and maximum length of the DOs is announced in Extended capabilities.

#### 4.4.3.2 Name

This interindustry data element consists of 0 to 39 bytes, each byte is a character from ISO 8859-1 (Latin 1) alphabet (identical to 7-bit-US-ASCII for characters < 80). The data element consists of surname (e. g. family name and given name(s)) and forename(s) (including name suffix, e. g., Jr. and number). Each item is separated by a '<' filler character (3C), the family- and fore-name(s) are separated by two '<<' filler characters.

#### 4.4.3.3 Language Preferences

This data element consists of 1 to 4 pairs of bytes (e. g. 2 bytes or 6 bytes) with coding according to ISO 639-1, ASCII lower case (e. g. de = german; en = english; nl = dutch; fr = french). At least one entry (2 bytes) should be present, the first entry has highest priority. The information can be used by the terminal for the user interface (e. g. language of text).

#### 4.4.3.4 Sex

This data element of 1 byte (binary) represents the 'Sex' of a person according to ISO 5218. The following values are defined for the OpenPGP application:

|               |    |
|---------------|----|
| Male          | 31 |
| Female        | 32 |
| Not announced | 39 |

The terminal can use the information for the user interface.



#### 4.4.3.5 User Interaction Flag

The optional feature User Interaction Flag adds a special behaviour to the related commands. If the flag is enabled (or permanently enabled) and a button or keypad is present, the related function will only run if a special button (on a keypad ENTER button) on the card or device is pressed by the user. This is a security function against viruses/trojans that try to call the functions on the card without knowledge of the user. The flags are evaluated by the card and can be changed by the card holder with the admin password (optional). Cards may support a permanently enabling for a flag, in that case the value cannot be changed any more (except factory reset). If the user did not finish the action (abort, timeout) the card should answer with 6600.

#### 4.4.3.6 Extended Capabilities

The Extended capabilities consists of 10 bytes (dec.) with the following meaning. The first byte is a table that indicates additional features to the terminal. A set bit (1) means that the function is available, a value equalling zero means that the function is not available. Bits can be set simultaneous.

Coding of byte 1 of Extended capabilities:

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning   |
|----|----|----|----|----|----|----|----|---|
| 1  | -  | -  | -  | -  | -  | -  | -  | Secure Messaging supported  |
| -  | 1  | -  | -  | -  | -  | -  | -  | Support for GET CHALLENGE<br>The maximum supported length of a challenge can be found in Extended capabilities. |
| -  | -  | 1  | -  | -  | -  | -  | -  | Support for Key Import  |
| -  | -  | -  | 1  | -  | -  | -  | -  | PW Status changeable (DO C4 available for PUT DATA)   |
| -  | -  | -  | -  | 1  | -  | -  | -  | Support for Private use DOs (0101-0104)   |
| -  | -  | -  | -  | -  | 1  | -  | -  | Algorithm attributes changeable with PUT DATA   |
| -  | -  | -  | -  | -  | -  | 1  | -  | PSO:DEC supports AES  |
| -  | -  | -  | -  | -  | -  | -  | 0  | RFU   |

The other bytes are coded as follows:

| Byte    | Length | Value   |
|---------|--------|---|
| 02      | 01     | Secure Messaging Algorithm<br>01 = AES 128 bit, dec.<br>02 = AES 256 bit, dec.<br>If SM is not supported (see 1 <sup>st</sup> byte), the coding is 00   |
| 03 - 04 | 02     | Maximum length of a challenge supported by the command GET CHALLENGE (unsigned integer, Most Significant Bit ... Least Significant Bit). If GET CHALLENGE is not supported (see 1 <sup>st</sup> byte), the coding is 0000 |
| 05 - 06 | 02     | Maximum length of Cardholder Certificates (DO 7F21, each for AUT, DEC and SIG), coded as unsigned integer (Most Significant Bit ... Least Significant Bit).   |
| 07 - 08 | 02     | Maximum length of special DOs (Private Use, Login data, URL), coded as unsigned integer (Most Significant Bit ... Least Significant Bit).   |
| 09      | 01     | PIN block 2 format<br>0 = not supported<br>1 = supported  |
| 0A      | 01     | RFU (00)  |

#### 4.4.3.7 Algorithm Attributes

This DO announces information related to the supported algorithm of the card. The terminal shall use this information for the key import functionality (if available). The formats are used by the key generation in the card and are also related to the output format of the corresponding command.

The content of the DO is optionally changeable (announced in Extended capabilities) with PUT DATA. This is useful if the card supports several algorithms or different key length. The attributes can be changed independent for each key, so it is possible for example to use different key length for signing and decrypting. A card should reject unsupported values in the DO. The supported values are manufacturer specific, please ask your card vendor for the correct parameters if the card supports changing of the attributes.

If the attributes of an existing key are changed and do no longer match with the stored key, the card should delete this key internally or prevent it from usage.

RSA:

| Byte    | Length | Value  |
|---------|--------|--|
| 01      | 01     | Algorithm ID (RFC 4880)<br>01 = RSA (Encrypt or Sign)  |
| 02 - 03 | 02     | Length of modulus n in bit (e. g. 2048 bit decimal = 0800), binary   |
| 04 - 05 | 02     | Length of public exponent e in bit (e. g. 32 bit decimal = 0020), binary   |
| 06      | 01     | Import-Format of private key<br>00 = standard (e, p, q)<br>01 = standard with modulus (n)<br>02 = crt (Chinese Remainder Theorem)<br>03 = crt with modulus (n) |

ECDSA:

| Byte    | Length | Value  |
|---------|--------|--|
| 01      | 01     | Algorithm ID (RFC 4880 and 6637)<br>19 = ECDSA for PSO:CDS and INT-AUT<br>18 = ECDH for PSO:DEC                                      |
| 02 - xx | var.   | OID of relevant curve, binary (see annex)  |
| last    | 01     | Import-Format of private key, optional<br>standard (private key only) if Format byte is not present<br>FF = standard with public key |

#### 4.4.3.8 Supported algorithms

An OpenPGP smart card V3.0 or higher shall support the following algorithms with their specific details.

Mandatory:

RSA 2048 or higher for PSO:CDS, PSO:DEC and INTERNAL AUTHENTICATE

Optional:

ECDSA 256 bit or higher for PSO:CDS and INTERNAL AUTHENTICATE

ECDH 256 bit or higher for PSO:DEC

AES 128 bit or higher for Secure Messaging and/or PSO:DEC

For ECDSA/ECDH the following curves are recommended (EN 419212):

Domain parameters standardized by

- "SEC 2: Recommended Elliptic Curve Domain Parameters", Version 2.0, by Certicom or
- ECC Brainpool working group.

From this recommendation the following curves should be used for an OpenPGP smart card and will be available in common smart card products:

The prime field curves

- ansix9p256r1
- ansix9p384r1
- ansix9p521r1

from [ANSI X9.62], compliant with [FIPS-186-3] and

- brainpoolP256r1
- brainpoolP384r1
- brainpoolP512r1

from [RFC5639].

At least one of this curves shall be supported by an OpenPGP smart card if EC is available. Details of the curve parameters can be found in the annex.

#### 4.4.3.9 Private Key Template

If the card supports key import (see Extended Capabilities), the terms of the corresponding private key are coded in the following way. The function does not matter how the key is stored in the card internally. The function does not set the value of the corresponding fingerprint. The key import uses a PUT DATA command with odd INS (DB) and an Extended header list (DO 4D) as described in ISO 7816-8.

The DOs in the Extended Header list start with a Control Reference Template (CRT) of the referenced key.

Digital signature: B6 00 (Tag, Length)  
 Confidentiality: B8 00  
 Authentication: A4 00

The next DO consists of a Cardholder private key template (7F48), that describes the input and the length of the content of the following DO. The last DO (Cardholder private key, 5F48) represents a concatenation of the key data elements according to the definitions in DO '7F48'.

The order of the DOs is mandatory (the card may support any order), xx means a length field (1, 2 or 3 bytes). Unnecessary DOs in between DO 7F48 should be stripped off (see format of private in Algorithm attributes).

| 4D | xx             | <i>Extended Header list for RSA</i> |   |                           |                                      |
|----|----------------|-------------------------------------|---|---------------------------|--------------------------------------|
|    | B6 or B8 or A4 | 00                                  | <i>Control Reference Template to indicate the private key</i> |                           |                                      |
|    | 7F48           | xx                                  | <i>Cardholder private key template</i>                        |                           |                                      |
|    |                | 91                                  | xx  | <i>Public exponent: e</i> | <i>key format: standard and crt</i>  |
|    |                | 92                                  | xx  | <i>Prime1: p</i>          | <i>standard and crt</i>              |
|    |                | 93                                  | xx  | <i>Prime2: q</i>          | <i>standard and crt</i>              |
|    |                | 94                                  | xx  | <i>PQ: 1/q mod p</i>      | <i>crt</i>                           |
|    |                | 95                                  | xx  | <i>DP1: d mod (p - 1)</i> | <i>crt</i>                           |
|    |                | 96                                  | xx  | <i>DQ1: d mod (q - 1)</i> | <i>crt</i>                           |
|    |                | 97                                  | xx  | <i>Modulus: n</i>         | <i>optional for standard and crt</i> |
|    | 5F48           | xx                                  | <i>Concatenation of key data as defined in DO 7F48</i>        |                           |                                      |

Some smart cards with RSA algorithm support only one coding for the Public Exponent (e.g. 65537 dec.). The card may reject an imported key, if e does not match the internal requirements. But the card shall accept 65537 dec. (010001) as value for e.

Smart cards may not be able to calculate a public key. In that case the public key can be integrated in the key import. This feature is announced in 'Algorithm attributes'.

The length of the key data shall match the values given in the DO 'Algorithm attributes' (C1 – C3). E.g., if the Modulus n has a length of 2048 bit (dec.), then p and q have a fixed length of 1024 bits each.

| 4D | xx             | <i>Extended Header list for ECDSA and ECDH</i> |   |                              |
|----|----------------|--|---|------------------------------|
|    | B6 or B8 or A4 | 00   | <i>Control Reference Template to indicate the private key</i> |                              |
|    | 7F48           | xx   | <i>Cardholder private key template</i>                        |                              |
|    |                | 92   | xx  | <i>Private key</i>           |
|    |                | 99   | xx  | <i>Public key (optional)</i> |
|    | 5F48           | xx   | <i>Concatenation of key data as defined in DO 7F48</i>        |                              |

The private key for EC is always stored in big-endian format and zero-padded to the adjusted underlying field size. The adjusted underlying field size is the underlying field size that is rounded up to the nearest 8-bit boundary.

In case of a signature key (CRT B6), the card internally resets the signature counter to zero.

#### 4.4.4 Length Field of DOs

According to ISO 7816-4 the length field in TLV-structures has the following format:

| Number of bytes | First byte | Second byte | Third byte | Value (dec.) |
|-----------------|------------|-------------|------------|--------------|
| 1               | 00 – 7F    | -           | -          | 0 – 127      |
| 2               | 81         | 00 - FF     | -          | 0 – 255      |
| 3               | 82         | 0000 - FFFF |            | 0 - 65535    |

## 5 Security Architecture

All commands and data of a smart card are under control of the security of the card operating system. ISO defines mechanisms, attributes (e. g. in FCP) and environments for security purposes. Because these features are quite complex and may differ from card to card (depending on mask developer), the OpenPGP application does not evaluate security related items of a card.

This chapter is informative for the card developer and defines the access conditions for all commands and data objects of the application in a common way. The described security features are mandatory for the card, but the coding or the way of implementation is up to the card developer, manufacturer or personaliser. Private keys and passwords shall not be readable from the card with any command or function. Commands and data have access conditions to be fulfilled.

The following tables show all access conditions for the OpenPGP application. READ is a synonym for all functions and commands of the operations system that present data to the external world, WRITE is a synonym for all functions and commands that change data in the Eeprom of the chip. If constructed DOs are processed, the access conditions of each single DO shall be fulfilled.

Access conditions for relevant commands:

| Command                        | Access condition   | Description   |
|--------------------------------|--|---|
| SELECT (DATA)                  | Always   |   |
| GET (NEXT) DATA                | Various  | Depending on Data Objects   |
| VERIFY                         | Always   |   |
| CHANGE REFERENCE DATA          | Always   |   |
| RESET RETRY COUNTER            | VERIFY of PW3 or Resetting Code                            |   |
| PUT DATA                       | Various  | Depending on Data Objects   |
| GENERATE ASYMMETRIC KEY PAIR   | Always (read pub-key)<br>VERIFY of PW3 (generate key-pair) |   |
| PSO: COMPUTE DIGITAL SIGNATURE | VERIFY of PW1  | With PW no. 81  |
| PSO: DECIPHER                  | VERIFY of PW1  | With PW no. 82  |
| INTERNAL AUTHENTICATE          | VERIFY of PW1  | With PW no. 82  |
| GET CHALLENGE                  | Always   |   |
| TERMINATE DF                   | VERIFY of PW3 or<br>Always if PW3 is blocked               | If PW3 is blocked or not usable (e. g. eeprom error), the access condition is always. |
| ACTIVATE FILE                  | Always   | Has only affect if the application is in termination state.                           |
| Other commands                 | Various  | e. g. commands for personalisation  |

Access conditions for Data Objects:

| Data Object                | READ       | WRITE      | Description   |
|----------------------------|------------|------------|---|
| Private use (0101)         | Always     | Verify PW1 | With PW no. 82  |
| Private use (0102)         | Always     | Verify PW3 |   |
| Private use (0103)         | Verify PW1 | Verify PW1 | With PW no. 82  |
| Private use (0104)         | Verify PW3 | Verify PW3 |   |
| AID (4F)                   | Always     | Never      | Writing possible only during personalisation (manufacturer) |
| Login data (5E)            | Always     | Verify PW3 |   |
| Name (5B)                  | Always     | Verify PW3 |   |
| Language preference (5F2D) | Always     | Verify PW3 |   |
| Sex (5F35)                 | Always     | Verify PW3 |   |
| URL (5F50)                 | Always     | Verify PW3 |   |



| <b>Data Object</b>                          | <b>READ</b> | <b>WRITE</b> | <b>Description</b>   |
|---|-------------|--------------|--|
| Card holder private key (5F48)              | Never       | Verify PW3   | Relevant for all private keys in the application (signature, decryption, authentication) |
| Cardholder certificates (7F21)              | Always      | Verify PW3   |  |
| DS-Counter (93)                             | Always      | Never        | Internal Reset during key generation   |
| Extended capabilities (C0)                  | Always      | Never        | Writing possible only during personalisation   |
| Algorithm attributes (C1 – C3)              | Always      | Verify PW3   |  |
| PW1 Status bytes (C4)                       | Always      | Verify PW3   | Only 1 <sup>st</sup> byte can be changed, other bytes only during personalisation        |
| Fingerprints (C5, C7 – C9)                  | Always      | Verify PW3   |  |
| CA-Fingerprints (C6, CA – CC)               | Always      | Verify PW3   |  |
| Generation date/time of key pairs (CD – D0) | Always      | Verify PW3   |  |
| SM-Key-ENC (D1)                             | Never       | Verify PW3   |  |
| SM-Key-MAC (D2)                             | Never       | Verify PW3   |  |
| Resetting Code (D3)                         | Never       | Verify PW3   |  |
| AES-Key for PSO:DEC (D5)                    | Never       | Verify PW3   |  |
| SM-Key-Container (F4)                       | Never       | Verify PW3   |  |
| General feature management (7F74)           | Always      | Never        |  |
| Extended length information (7F66)          | Always      | Never        |  |
| User Interaction Flag PSO:CDS (D6)          | Always      | Verify PW3   |  |
| User Interaction Flag PSO:DEC (D7)          | Always      | Verify PW3   |  |
| User Interaction Flag PSO:AUT (D8)          | Always      | Verify PW3   |  |

In case that the card supports Secure Messaging, a correct command in SM mode is equivalent with reaching the access condition for PW3 (Verify PW3).

## 6 Historical Bytes

Historical bytes are part of the ATR (Answer To Reset) from a card with contacts, the 'format byte (T0)' indicates the presence of Historical bytes in bits 1- 4, according to ISO 7816-3. An alternative way of reading Historical bytes is a special DO with the Tag '5F52'. Because not all cards provide Historical bytes with an appropriate content, the OpenPGP application uses this DO. In the Historical bytes the presence of the DOs 'Card service data byte' and 'Card capabilities' are relevant. The DO is coded according to ISO 7816-4.

The DO 'Historical bytes' ('5F52') is part of the "Application related data" and can be read with the GET DATA command from within the OpenPGP card application. It is relevant for the application and may differ from Historical bytes in the ATR.

The first Historical byte is the "category indicator byte". If the category indicator byte is set to '00', '10' or '80', then the format is according to ISO. Any other value indicates a proprietary format. The OpenPGP application assumes a category indicator byte set to '00'. The remaining Historical bytes consist of optional consecutive COMPACT-TLV data objects followed. The last 3 bytes of the Historical bytes are a status indicator byte and two processing status bytes SW1/SW2 (normally '9000').

The status indicator byte is evaluated by the OpenPGP application as follows:

- 00 = No information given  
Card does not offer life cycle management, commands TERMINATE DF and ACTIVATE FILE are not supported
- 03 = Initialisation state  
OpenPGP application can be reset to default values with an ACTIVATE FILE command
- 05 = Operational state (activated)  
Card supports life cycle management, commands TERMINATE DF and ACTIVATE FILE are available

The COMPACT-TLV format has a Tag in the first nibble of a byte (bit 5-8) and a length in the second nibble (bit 1-4). For the OpenPGP application a TL with '73' is relevant, it announces a DO 'Card capabilities' with 3 bytes. In addition a TL with '31' announces a DO 'Card service data' with 1 byte, that is interpreted by the OpenPGP application also. Other DOs may appear.

## 6.1 Card Capabilities (73)

This interindustry data element consists of three software function tables (1 byte each) according to ISO 7816-4. The first software function table indicates selection methods supported by the card. The second software function table is the "data coding byte". The third software function table indicates the ability to chain commands, the support for Extended Lc and Le fields and to handle logical channels. A set Bit (1) means that the function is available (unless otherwise specified), a value equal zero means that the function is not available. Bits can be set simultaneous. For the OpenPGP application only the third table (byte 3) is relevant (yellow marked functions should be evaluated in this version).

Command chaining, length fields and logical channels (third byte):

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning                                    |
|----|----|----|----|----|----|----|----|--|
| x  | -  | -  | -  | -  | -  | -  | -  | Command chaining                           |
| -  | x  | -  | -  | -  | -  | -  | -  | Extended Lc and Le fields                  |
| -  | -  | x  | -  | -  | -  | -  | -  | Extended Length Information in EF.ATR/INFO |
| -  | -  | -  | x  | x  | -  | -  | -  | Logical channel number assignment          |
| -  | -  | -  | -  | -  | y  | z  | t  | Maximum number of logical channels         |

If Extended Length is announced in bit 7, then the DO "Extended length information" shall be present in the OpenPGP card application.

## 6.2 Card service data (31)

This interindustry data element consists of 1 byte according to ISO 7816-4. A set Bit (1) means that the function is available (unless otherwise specified), a value equal zero means that the function is not available. Bits can be set simultaneous. For the OpenPGP application yellow marked functions should be evaluated in this version.

Command chaining, length fields and logical channels (third byte):

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning  |
|----|----|----|----|----|----|----|----|--|
| x  | -  | -  | -  | -  | -  | -  | -  | Application Selection by full DF name (AID)  |
| -  | x  | -  | -  | -  | -  | -  | -  | Application Selection by partial DF name   |
| -  | -  | x  | -  | -  | -  | -  | -  | DOs available in EF.DIR  |
| -  | -  | -  | x  | -  | -  | -  | -  | DOs available in EF.ATR/INFO<br>Shall be set to 1, if Extended Length is supported   |
| -  | -  | -  | -  | x  | x  | x  | -  | EF.DIR and EF.ATR/INFO access services by the GET DATA command (BER-TLV)<br>Shall be set to 010, if Extended Length is supported |
| -  | -  | -  | -  | -  | -  | -  | x  | Card with MF (0)<br>Card without MF (1)  |

## 7 Commands

The OpenPGP application is based on the functionality of ISO 7816-4, -8 and -9. Thus the standard OS commands are available to the 'external environment'.

### 7.1 Usage of ISO Standard Commands

The following table shows all standard commands of an ISO operating system, which are used by the OpenPGP application. Only the given subsets (P1/P2) of a command shall be implemented, however the card may provide other functions.

| Command                          | INS         | P1           | P2       | Comment  |
|----------------------------------|-------------|--------------|----------|--|
| SELECT                           | A4          | 04           | 00       | AID = 1-16 Byte (partial AID is recommended)<br>P2 = 00 for first or only occurrence   |
| SELECT DATA                      | A5          | 01/02<br>/03 | 04       | P1 = Occurrence number of DOs with same tag (DO 7F21)<br>P2 = First or only occurrence of a DO after skipping P1 occurrences and Return the data control Information (DO 62)     |
| GET DATA                         | CA          | xx           | xx       | Supported as defined for specified DOs   |
| GET NEXT DATA                    | CC          | xx           | xx       | Supported as defined for specified DOs   |
| VERIFY                           | 20          | 00/FF        | 81/82/83 | Local PW1 or PW3   |
| CHANGE REFERENCE DATA            | 24          | 00           | 81/83    | Change of PW1 or PW3   |
| RESET RETRY COUNTER              | 2C          | 00/02        | 81       | Resets the retry counter of PW1 and sets a new value for PW1. In the command data the new PW1 is present.  |
| PUT DATA                         | DA or<br>DB | xx           | xx       | Supported as defined for specified DOs   |
| GENERATE ASYMMETRIC KEY PAIR     | 47          | 80/81        | 00       | P1 = 80: Generation of internal private key, public key in response (DO 7F49)<br>P1 = 81: Reading of actual public key<br>Relevant key is addressed by a CRT in the command data |
| PERFORM SECURITY OPERATION (PSO) | 2A          | xx           | xx       | As defined in the next lines   |
| <i>COMPUTE DIGITAL SIGNATURE</i> | 2A          | 9E           | 9A       | Input are plain data (e. g. hash code), length shall match the algorithm and key length of the   |

| Command               |                 | INS | P1 | P2 | Comment  |
|-----------------------|-----------------|-----|----|----|--|
|                       |                 |     |    |    | card, digital signature in response for RSA  |
|                       | <i>DECIPHER</i> | 2A  | 80 | 86 | Input: Padding indicator byte (00 or 02) and encrypted data (length of encrypted data shall match algorithm and key length) for RSA; DO for ECDH<br>Response: Plain data / Shared secret   |
| INTERNAL AUTHENTICATE |                 | 88  | 00 | 00 | Authentication input related to algorithm  |
| GET RESPONSE          |                 | C0  | 00 | 00 | Used under T=0 and for retrieving long DOs with GET DATA under any protocol  |
| GET CHALLENGE         |                 | 84  | 00 | 00 | Fully supported (Le defines length of random number), optional command. If supported the card shall provide any length according to simple Le. If extended Le is supported the maximum length is announced in Extended capabilities  |
| TERMINATE DF          |                 | E6  | 00 | 00 | The command puts the application into the termination state. After termination only SELECT and ACTIVATE FILE are available   |
| ACTIVATE FILE         |                 | 44  | 00 | 00 | In the OpenPGP application the termination state is equivalent to the initialisation state. An ACTIVATE FILE command in this stage resets all DOs to default values and sets the application into the operational state. The usage of this command in operational state (not terminated) has no effect to any data |

Additional commands for production, personalisation and other applications are out of scope of this specification.

## 7.2 Commands in Detail

The following section describes some of the commands in more detail. In all examples short Lc/Le is used. If the card provides extended Lc/Le than the terminal may extend the fields to a length of 2 or 3 bytes. Commands that support Secure Messaging (option) are marked with a Class byte of 0C.

### 7.2.1 SELECT

With this command the OpenPGP application in the terminal selects the corresponding application on the card. Only the significant bytes of the AID are presented in the command data. Possible response data (FCI) don't need to be evaluated by the application. A valid SELECT of the OpenPGP application sets the curConstructedDO pointer to the Virtual Root DO and adds all data objects in the application to the current template. The curDO pointer is undefined, so a following GET/PUT DATA command will always reference the first occurrence of a DO.

Command:

|            |                   |
|------------|-------------------|
| CLA        | 00                |
| INS        | A4                |
| P1         | 04                |
| P2         | 00                |
| Lc         | 06                |
| Data field | D2 76 00 01 24 01 |
| Le         | 00                |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | FCI or empty                  |
| SW1-SW2    | 9000 or specific status bytes |

### 7.2.2 VERIFY

The VERIFY command is used to check the correct password given in the command data and set an appropriate access status for the relevant password. If the command is called without data, the actual access status of the addressed password is returned or the access status is set to 'not verified'.

With P1 = 00 and password data in the command one of the passwords of the application is verified. PW1 has two modes and can be used with 2 different numbers (in P2). PW 1 with P2 = 81 sets the access conditions for a PSO:CDS command only. Depending on the PW1 status byte (see Extended capabilities) this access condition is only valid for one PSO:CDS command or remains valid for several attempts. PW1 with P2 = 82 sets the access condition for several other functions and remains valid up to a reset or SELECT of a different application.

If the command is called with P1 = 00 and no data (Lc empty), then the actual access status of the addressed password in P2 is returned. If the password is still verified the cards answers with normal status bytes (SW1-SW2 = 9000). If the password is not checked and the verification is required, then the card answers with the status bytes 63CX, where 'X' encodes the number of further allowed retries.

If the command is called with P1 = FF, then Lc shall be empty (no data in the command data field). The addressed password in P2 is resetted to the status 'not verified'. To use commands that are protected by this password, VERIFY with correct password data has to be called again.



Command:

|            |   |
|------------|---|
| CLA        | 00 / 0C   |
| INS        | 20  |
| P1         | 00 / FF   |
| P2         | 81 (PW1) or 82 (PW1) or 83 (PW3)  |
| Lc         | xx (min. 06 for PW1, min. 08 for PW3, max. see DO 'C4') in case of UTF-8 format<br>08 in case of PIN block format 2<br>Absent for status check or Reset |
| Data field | Corresponding PW in correct format or absent  |
| Le         | Empty (means not present in command)  |

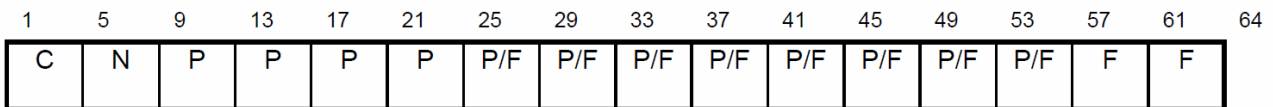
Response:

|            |                               |
|------------|-------------------------------|
| Data field | None                          |
| SW1-SW2    | 9000 or specific status bytes |

### PIN block format 2

This format can be set in the PW status bytes (optional, announced in Extended capabilities) and has the following structure (taken from ISO 9564-3):

Bit



where

- C is the control field, 4-bit field value 0010 (2);
- N is the PIN length, 4-bit binary number with permissible values 0100 (4) to 1100 (12);
- P is the PIN digit, 4-bit field with permissible values 0000 (zero) to 1001 (9);
- P/F is the PIN/Fill digit, with P or F determined by PIN length;
- F is the fill digit, 4-bit field value 1111 (15).

### 7.2.3 CHANGE REFERENCE DATA

With this command the passwords of the application can be changed. In compliance with EN 419212 only one P1 variation from ISO 7816-4 is defined for signature cards. The command can be accessed without restrictions, the actual PW is part of the command data and will be verified first by the card. The length of the existing password is known in the card, so that neither a delimiter nor padding for filling up fixed formats is necessary for UTF-8. The length of the new UTF-8 password therefore computes  $L_{\text{new}} = L_c - L_{\text{old}}$ . PIN block format 2 passwords have a fixed length of 8 bytes.

Command:

|            |  |
|------------|--|
| CLA        | 00 / 0C  |
| INS        | 24   |
| P1         | 00   |
| P2         | 81 (PW1) or 83 (PW3)   |
| Lc         | xx   |
| Data Field | Actual PW + New PW<br>Length of new UTF-8 PW: min. 06 for PW1, min. 08 for PW3<br>For max. length see DO 'C4'<br>For PIN block format 2 two blocks (8 bytes each) for old and new PIN are concatenated |
| Le         | Empty (means not present in command)   |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | None                          |
| SW1-SW2    | 9000 or specific status bytes |

## 7.2.4 RESET RETRY COUNTER

With this command the error counter and the value of PW1 can be resetted, that means the new value is stored and the error counter is set to the default value (3). RESET RETRY COUNTER can be used after correct verification of PW3 (P1 = 02) or by presenting the Resetting Code (DO D3) in the command data (P1 = 00). Usage of secure messaging is equivalent to PW3. The length of the Resetting Code is known in the card, so that neither a delimiter nor padding for filling up fixed formats is necessary for UTF-8. The length of the new UTF-8 password therefore computes  $L_{new} = L_c - L_{RC}$ . PIN block format 2 passwords have a fixed length of 8 bytes.

Command:

|            |   |
|------------|---|
| CLA        | 00 / 0C   |
| INS        | 2C  |
| P1         | 00 / 02   |
| P2         | 81 (PW1)  |
| Lc         | xx (min. 06 for PW1, max. see DO 'C4')                |
| Data field | New PW (P1 = 02)<br>Resetting Code + New PW (P1 = 00) |
| Le         | Empty (means not present in command)                  |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | None                          |
| SW1-SW2    | 9000 or specific status bytes |

## 7.2.5 SELECT DATA

With this command a DO in the current template can be selected. The command is needed to address DOs with several instances with the same Tag (e. g. 7F21, Card holder certificate). A successful command sets the curDO pointer to the addressed DO and makes it the current DO, a following GET DATA or PUT DATA will access this current DO. Setting the first instance (P1 = 00) will work with every accessible DO, higher values will only work for DOs with several instances.

Possible response data (Control Parameter = CP) don't need to be evaluated by the application.

Command:

|            |   |
|------------|---|
| CLA        | 00  |
| INS        | A5  |
| P1         | xx = Occurrence number of an instance to skip<br>00 – 02 is defined for the OpenPGP application |
| P2         | 04 = First occurrence after skipping P1 occurrences with CP in response                         |
| Lc         | xx  |
| Data Field | 60 xx (General reference DO)<br>5C xx (Tag list)<br>Tag of the DO to be selected                |
| Le         | 00  |

Response:

|            |                                  |
|------------|----------------------------------|
| Data field | Control Parameters (CP) or empty |
| SW1-SW2    | 9000 or specific status bytes    |

Examples:

' Selects the 1<sup>st</sup> occurrence of the Card holder certificate (AUT)

00 A5 00 04 06 60 04 5C 02 7F21 00

' Selects the 2nd occurrence of the Card holder certificate (DEC)

00 A5 01 04 06 60 04 5C 02 7F21 00

' Selects the 3rd occurrence of the Card holder certificate (SIG)

00 A5 02 04 06 60 04 5C 02 7F21 00

### 7.2.6 GET DATA

With this command DOs can be read from the card. The Tag (simple or constructed) is given in P1/P2 (e. g. 5F50 for URL or 006E for Application Related Data). For simple DOs only the value is in the response field (e. g. 5F50 = URL returns a byte string representing the URL without leading Tag/Length). For constructed DOs all values returned are encapsulated with Tag/Length (e. g. 0065 = Cardholder Related Data returns the concatenation of following DOs (L = Length): 5B L Name 5F2D L Language Preferences 5F35 L Sex).

If the DO is longer than the maximum supported length for a response of a card, then the card may answer with status bytes 61xx and return only the first part of the data, xx indicate the remaining data in the card. The data may be truncated at any position and shall be concatenated later in the terminal. The terminal can read the missing data with a following GET RESPONSE command and Le = 00 (or 0000 for extended Le). This can be repeated several times (another status byte 61xx). The reading of data is complete if any command (GET DATA or GET RESPONSE) answers with status bytes 9000.

A valid GET DATA makes the first occurrence of the addressed DO to the current DO, if the curDO pointer was undefined. Otherwise (e. g. previous SELECT DATA) the current DO with the addressed Tag is returned.

Command:

|            |  |
|------------|--|
| CLA        | 00 / 0C                                      |
| INS        | CA   |
| P1         | xx (00 if Tag has a length of only one byte) |
| P2         | xx   |
| Lc         | Empty  |
| Data Field | None   |
| Le         | 00   |

Response:

|            |   |
|------------|---|
| Data field | Addressed data or DOs (maybe partially) |
| SW1-SW2    | 9000 or 61xx or specific status bytes   |

### 7.2.7 GET NEXT DATA

With this command other instances of DOs with the same Tag can be read from the card. The command is only valid for the user certificate (7F21) that has 3 occurrences. The command can be used only if the curDO pointer is still set to a valid occurrence of DO '7F21'. It selects the next occurrence automatically (internal SELECT DATA) and returns the data field of the DO.

With the following commands in a sequence all certificates can be read from the card:

GET DATA with P1P2= 7F21 returns the 1st occurrence (AUT certificate)

GET NEXT DATA with P1P2= 7F21 returns the 2nd occurrence (DEC certificate)

GET NEXT DATA with P1P2= 7F21 returns the 3rd occurrence (SIG certificate)

This way is quicker than reading all certificates with SELECT DATA and following GET DATA, but it is still possible to do it in that way. If you only want to read the 3rd occurrence (SIG certificate), for e. g., then a SELECT DATA with following GET DATA is more useful.

Command:

|            |         |
|------------|---------|
| CLA        | 00 / 0C |
| INS        | CC      |
| P1         | 7F      |
| P2         | 21      |
| Lc         | Empty   |
| Data Field | None    |
| Le         | 00      |

Response:

|            |                                       |
|------------|---------------------------------------|
| Data field | Addressed DO                          |
| SW1-SW2    | 9000 or 61xx or specific status bytes |

### 7.2.8 PUT DATA

With this command DOs can be written to the card. The Tag is given in P1/P2 (e. g. 5F50 for URL or 005B for Name). For simple DOs only the value is in the data field (without leading Tag/Length). The command can only be used after correct presentation of PW3 (except DO 0101 and DO 0103 after correct verification of PW1 with No. 82).

For key import in compliance with ISO 7816-8 an Extended header list is supported with an odd INS of the command. In that case P1/P2 has the value 3FFF (references the current DF for the DOs in the Extended header list).

An empty data field (Lc absent) deletes the content of a DO with variable length (e. g. URL), but not the DO itself.

A valid PUT DATA makes the first occurrence of the addressed DO to the current DO, if the curDO pointer was undefined. Otherwise (e. g. previous SELECT DATA) the content of the current DO with the addressed Tag is replaced.

Command:

|            |   |
|------------|---|
| CLA        | 00 / 0C   |
| INS        | DA / DB   |
| P1         | xx = 00 if Tag has a length of one byte only<br>= 3F in case of odd INS |
| P2         | xx (FF in case of odd INS)  |
| Lc         | xx or empty   |
| Data Field | Addressed data or Extended header list or empty                         |
| Le         | Empty   |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | None                          |
| SW1-SW2    | 9000 or specific status bytes |

### 7.2.9 GET RESPONSE

This command is needed under T=0 for some command cases according to ISO 7816-3 and under any protocol (e.g. T=1) for receiving long data that cannot be transmitted in one response.

After receiving status bytes with 61 xx, the terminal should send a GET RESPONSE command with xx or 00 in the Le field.

Command:

|            |                               |
|------------|-------------------------------|
| CLA        | 00 / 0C                       |
| INS        | C0                            |
| P1         | 00                            |
| P2         | 00                            |
| Lc         | Empty                         |
| Data field | Empty                         |
| Le         | 00 (xx given in previous SW2) |

Response:

|            |                                       |
|------------|---------------------------------------|
| Data field | Data                                  |
| SW1-SW2    | 9000 or 61xx or specific status bytes |



### 7.2.10 PSO: COMPUTE DIGITAL SIGNATURE

The command for digital signature computation is shown in the table below. The hash value (ECDSA) or the DigestInfo is delivered in the data field of the command. Signature key as well as signature algorithm and the related Digital-Signature-Input formats are implicitly selected.

The command is only possible after correct presentation of PW1 with No. 81. The command internally checks the PW1 Status byte DO (first byte), if the value is 00, then the access condition (No. 81 of PW1) is reset and the PW has to be verified again for the following command.

It is possible to use the command outside of the OpenPGP environment (e. g. S-MIME), for that reason it is possible to store a related certificate (Tag 7F21) as 3rd occurrence into the card.

Command:

|            |  |
|------------|--|
| CLA        | 00   |
| INS        | 2A   |
| P1         | 9E   |
| P2         | 9A   |
| Lc         | Length of subsequent data field                            |
| Data field | Data to be integrated in the DSI: hash value or DigestInfo |
| Le         | 00   |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | Digital signature             |
| SW1-SW2    | 9000 or specific status bytes |

The DSI format for RSA:

According to PKCS #1 the DSI is generated internally by the card and has the following structure:

| Description         | Length | Value      |
|---------------------|--------|------------|
| Start byte          | 1      | 00         |
| Block type          | 1      | 01         |
| Padding string (PS) | N-3-L  | FF ... FF  |
| Separator           | 1      | 00         |
| Data field          | L      | DigestInfo |

In compliance with PKCS #1, the card checks that the DigestInfo in the command data field is not longer than 40% of the length of the modulus of the signature key, otherwise the command is rejected.

DSI for ECDSA:

The DSI consists of the hash value which was calculated (28, 32, 48 or 64 bytes, dec.). If the required DSI for the computation is longer than the hash value, then the DSI is filled with leading zero bits by the card.

### 7.2.10.1 Hash Algorithms

The following hash algorithms are supported by RFC 4880 and can be used as input in the DSI. Older SHA-1 and RIPEMD-160 are not allowed for DSI in actual standards and should be avoided for the OpenPGP smart card also. However the card may not check the integrity of a DSI.

| <i>Hash algorithm</i> | <i>Length of hash-code (dec.)</i> |
|-----------------------|-----------------------------------|
| SHA-224               | 28                                |
| SHA-256               | 32                                |
| SHA-384               | 48                                |
| SHA-512               | 64                                |

### 7.2.10.2 DigestInfo for RSA

The following tables show the contents of defined DigestInfos for RSA. The card may not check the correctness of the DigestInfo.

SHA-224:

| <b>Tag</b> | <b>Length</b> | <b>Value</b>       | <b>Description</b>                          |
|------------|---------------|--------------------|---|
| 30         | 2D            |                    | Tag/Length of Sequence                      |
| 30         | 0D            |                    | Tag/Length of Sequence                      |
| 06         | 09            | 608648016503040204 | OID of the SHA-224 {2 16 840 1 101 3 4 2 4} |
| 05         | 00            | -                  | TLV coding of ZERO                          |
| 04         | 1C            | xx...xx            | hash-code                                   |

SHA-256:

| <b>Tag</b> | <b>Length</b> | <b>Value</b>       | <b>Description</b>                          |
|------------|---------------|--------------------|---|
| 30         | 31            |                    | Tag/Length of Sequence                      |
| 30         | 0D            |                    | Tag/Length of Sequence                      |
| 06         | 09            | 608648016503040201 | OID of the SHA-256 {2 16 840 1 101 3 4 2 1} |
| 05         | 00            | -                  | TLV coding of ZERO                          |
| 04         | 20            | xx...xx            | hash-code                                   |

SHA-384:

| <b>Tag</b> | <b>Length</b> | <b>Value</b>       | <b>Description</b>                          |
|------------|---------------|--------------------|---|
| 30         | 41            |                    | Tag/Length of Sequence                      |
| 30         | 0D            |                    | Tag/Length of Sequence                      |
| 06         | 09            | 608648016503040202 | OID of the SHA-384 {2 16 840 1 101 3 4 2 2} |
| 05         | 00            | -                  | TLV coding of ZERO                          |
| 04         | 30            | xx...xx            | hash-code                                   |

## SHA-512:

| <b>Tag</b> | <b>Length</b> | <b>Value</b>       | <b>Description</b>                          |
|------------|---------------|--------------------|---|
| 30         | 51            |                    | Tag/Length of Sequence                      |
| 30         | 0D            |                    | Tag/Length of Sequence                      |
| 06         | 09            | 608648016503040203 | OID of the SHA-512 {2 16 840 1 101 3 4 2 3} |
| 05         | 00            | -                  | TLV coding of ZERO                          |
| 04         | 40            | xx...xx            | hash-code                                   |

### 7.2.11 PSO: DECIPHER

The command is used by the application as key decipherment service. The command can be used after correct presentation of PW1 (with No. 82) only. For confidential document exchange, the following scheme is applied:

- The key transport is organised by enciphering the content encryption key with the receivers public key.
- The document enciphering is done with a symmetrical algorithm (e. g. AES).

The card is not involved in the enciphering of the document. The software computes the content encryption key, enciphers the document and finally enciphers the content encryption key by using the receivers public key. The card performs a key decryption applying the private key for decryption in a DECIPHER command to the cryptogram contained in the data field of the command.

By option (announced in Extended capabilities) the card supports the decryption of a plain text with an AES-key stored in a special DO (D5). This is useful if no certificate or public key exists and the external world has a common secret with the card. The functionality can be used e. g. for storing an encrypted transport key or seed on a flash card/stick or hard disc, only the relevant card is able to decrypt the secret if the AES-key is deleted in the outside world after encryption and storing into the card.

In case of the RSA algorithm the command input (except padding indicator byte) shall be formatted according to PKCS#1 before encryption:

| Description         | Length | Value                 |
|---------------------|--------|-----------------------|
| Start byte          | 1      | 00                    |
| Block type          | 1      | 02                    |
| Padding string (PS) | N-3-L  | Non-zero-random-bytes |
| Separator           | 1      | 00                    |
| Data field          | L      | Message               |

PS is a byte string consisting of randomly generated non-zero bytes. The length of PS shall be at least 8 bytes. The formatted string shall consist of N bytes where N is the length of the modulus of the private key for decryption. The Padding indicator byte and the encrypted message is given to the command in the command data. The card decrypts all bytes after the padding indicator byte, checks the conformance of correct PKCS#1 padding and returns the plain text (length = message) in the response.

In case of the AES algorithm the command input (except padding indicator byte) shall be exact 16 bytes. A key shall be present in DO (D5), the Padding indicator byte (02) and the encrypted message is given to the command in the command data. The card decrypts all bytes after the padding indicator byte and returns the plain text (16 bytes) in the response.

In case of ECDH the card supports a partial decrypt only. The input is a cipher DO with the following data:

|    |      |                  |               |                            |
|----|------|------------------|---------------|----------------------------|
| A6 | xx   | <i>Cipher DO</i> |               |                            |
|    | 7F49 | xx               | Public Key DO |                            |
|    |      | 86               | xx            | <i>External Public Key</i> |

The external public key for ECDH consists of of two raw big-endian integers with the same length as a field element each. In compliance with EN 419212 the format is

$$04 || x || y$$

where the first byte (04) indicates an uncompressed raw format.

The card shall abort the command if the given OID does not match the defined curve for the command (see Algorithm attributes).

With its own private key and the given public key the card calculates a shared secret in compliance with the Elliptic Curve Key Agreement Scheme from Diffie-Hellman. The shared secret is returned in the response, all other calculation for deciphering are done outside of the card.

Command:

|            |   |
|------------|---|
| CLA        | 00  |
| INS        | 2A  |
| P1         | 80 = Return plain value   |
| P2         | 86 = Enciphered data present in the data field  |
| Lc         | xx = Length of subsequent data field  |
| Data field | Padding indicator byte (00) for RSA or (02) for AES followed by cryptogram<br>Cipher DO 'A6' for ECDH |
| Le         | 00  |

Response:

|            |  |
|------------|--|
| Data field | Plain data (RSA or AES) / Shared secret (ECDH) |
| SW1-SW2    | 9000 or specific status bytes                  |

It is possible to use the command outside of the OpenPGP environment, for that reason it is possible to store a related certificate (Tag 7F21) as 2nd occurrence into the card.

### 7.2.12 INTERNAL AUTHENTICATE

The INTERNAL AUTHENTICATE command can be used for Client/Server authentication. The usage is up to the terminal, the card only provides this command for asymmetric algorithms. The input data shall be an Authentication Input (AI) compliant with PKCS#1 for RSA, the card does an internal PKCS#1-padding and calculates a signature with the corresponding secret key for authentication. For ECDSA the input is a hash only. The command can be used after correct presentation of PW1 (with No. 82) only.

In compliance with PKCS #1, the card checks in case of RSA that the AI in the command data field is not longer than 40% of the length of the modulus of the signature key, otherwise the command is rejected.

PKCS#1-Padding for Authentication Input used with RSA:

| Description         | Length | Value                     |
|---------------------|--------|---------------------------|
| Start byte          | 1      | 00                        |
| Block type          | 1      | 01                        |
| Padding string (PS) | N-3-L  | FF...FF                   |
| Separator           | 1      | 00                        |
| Data field          | L      | Authentication Input (AI) |

The resulting input for the signature in case of RSA has the length N. The card calculates the signature with the private key for authentication:  $\text{sign}(\text{SK}_{\text{Aut}})[00 | 01 | \text{PS} | 00 | \text{AI}]$  and returns the result as authentication data in the response.

For ECDSA the AI consists of the hash value which was calculated (28, 32, 48 or 64 bytes, dec.). If the required AI for the computation is longer than the hash value, then the AI is filled with leading zero bits by the card.



Command:

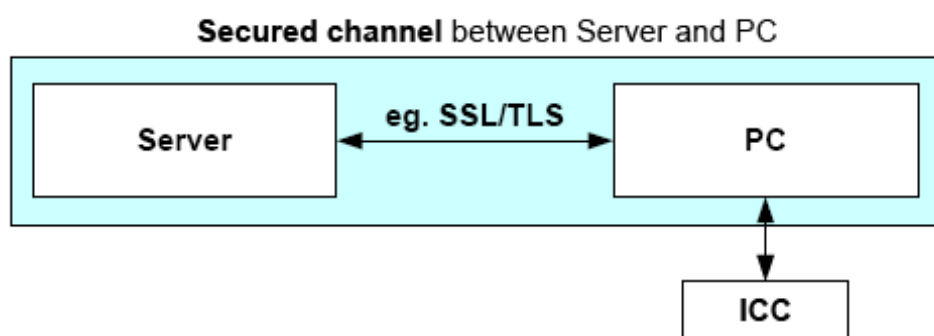
|            |  |
|------------|--|
| CLA        | 00   |
| INS        | 88 = INTERNAL AUTHENTICATE   |
| P1         | 00   |
| P2         | 00   |
| Lc         | xx = Length of subsequent data field   |
| Data Field | Authentication Input (AI)<br>for RSA: $Lc \leq 0,4 * N$ , e.g.:<br>$Lc \leq 102$ for 2048 bit modulus<br>for ECDSA: Hash value<br>(values are decimal) |
| Le         | 00   |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | Signature                     |
| SW1-SW2    | 9000 or specific status bytes |

### 7.2.12.1 Client/Server Authentication

This specification covers only the case where the card performs a digital signature computation applying the private key for authentication in an INTERNAL AUTHENTICATE command to the authentication input contained in the data field of the command after formatting the input. The mechanism can be used, for example, with Secure Shell (SSH), PK Kerberos, SSL/TLS or WTLS. All these protocols base on the same cryptographic algorithms. In particular they are all using PKCS #1 padding format in the case of RSA.



In the above example (taken from EN 419212-5), client/server authentication establishes a secured channel between a remote server and a PC. The ICC will be used as a cryptographic toolbox in order to provide the cryptographic functionality to the PC. If a process needs a certificate from the card that is not provided by OpenPGP (e. g. x.509), then it can be stored in the DO 'Cardholder certificate' (7F21 AUT) as 1st occurrence.

### 7.2.13 GENERATE ASYMMETRIC KEY PAIR

This command either initiates the generation and storing of an asymmetric key pair, i. e., a public key and a private key in the card, or returns the public key of an asymmetric key pair previously generated in the card or imported. The card should use 65537 dec. (10001 bin.) as default value for e, but other values are accepted by the outside world. In case of key pair generation the command does not set the values of the corresponding fingerprint. After receiving the public key the terminal has to calculate the fingerprint and store it in the relevant DO. The generation of a key pair for digital signature resets the digital signature counter to zero (000000). The command can only be used after correct presentation of PW3 for the generation of a key pair. Reading of a public key is always possible.

Command:

|            |   |
|------------|---|
| CLA        | 00 / 0C   |
| INS        | 47  |
| P1         | 80 = Generation of key pair<br>81 = Reading of actual public key template |
| P2         | 00  |
| Lc         | 02  |
| Data field | CRT for relevant function   |
| Le         | 00  |

Response:

|            |                                     |
|------------|-------------------------------------|
| Data field | Public key as a set of data objects |
| SW1-SW2    | 9000 or specific status bytes       |

Defined CRTs (Control Reference Template) for the command (generation of key pair or reading of public key).

Digital signature: B6 00 (Tag, Length)  
Confidentiality: B8 00  
Authentication: A4 00

Defined DOs for response (xx = Length):

7F49 xx

Set of public key data objects for RSA

81 xx Modulus (a number denoted as n coded on x bytes)

82 xx Public exponent (a number denoted as v, e.g. 65537 dec.)

Set of public key data objects for ECDSA/ECDH

86 xx Public key (a point denoted as PP on the curve, equal to x times PB where x is the private key, coded on 2z or z+1 bytes)

The public key for ECDSA/DH consists of two raw big-endian integers with the same length as a field element each. In compliance with EN 419212 the format is

04 || x || y

where the first byte (04) indicates an uncompressed raw format.

A card may send other DOs in compliance with ISO 7816-8, but only the mandatory DOs defined above need to be evaluated.

### 7.2.14 GET CHALLENGE

This optional command (announced in Extended Capabilities) generates a random number of the length given in Le. It is a service to the terminal application because smart cards often generate high sophisticated random numbers by certified hardware. Several smart card implementations have limitations for the length of the random number, so the maximum length is announced in Extended capabilities.

The command is mandatory if the cards supports Secure Messaging.

Command:

|            |   |
|------------|---|
| CLA        | 00  |
| INS        | 84  |
| P1         | 00  |
| P2         | 00  |
| Lc         | Empty   |
| Data field | Empty   |
| Le         | xx (01-FF for Short Le, 0001-maximum for Extended Le) |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | Challenge with length xx      |
| SW1-SW2    | 9000 or specific status bytes |

### 7.2.15 TERMINATE DF

This optional command (announced in Life Cycle Status indicator in Historical bytes) sets the Life Cycle Status indicator to 03 and puts the card into initialisation state. The behaviour of the application is similar to the termination state, no commands can be used except SELECT FILE, which return specific status bytes (6285). After a selection an ACTIVATE FILE command is possible. That command is designed to renew a card in case of blocked passwords or other problems. The command is allowed after correct verification of PW3 (Admin-PW) or secure messaging (SM) if PW3 is disabled.

If PW3 is blocked or not accessible (e. g. memory failure) and SM is disabled, then <sup>ICV = RND</sup>TERMINATE can be used without access conditions. This prevents the card from any status that makes it unusable.

Command:

|            |         |
|------------|---------|
| CLA        | 00 / 0C |
| INS        | E6      |
| P1         | 00      |
| P2         | 00      |
| Lc         | Empty   |
| Data field | Empty   |
| Le         | Empty   |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | Empty                         |
| SW1-SW2    | 9000 or specific status bytes |

### 7.2.16 ACTIVATE FILE

This optional command (announced in Life Cycle Status indicator in Historical bytes) can be used to reset all values (DOs, Keys, PWs etc.) to their default values (after production/initialisation). The command has effect only, if the life cycle status is in initialisation state (indicator byte set to 03). If the command is used in operational state, it will end with status bytes 9000, but nothing in the application will be changed (dummy command).

Command:

|            |       |
|------------|-------|
| CLA        | 00    |
| INS        | 44    |
| P1         | 00    |
| P2         | 00    |
| Lc         | Empty |
| Data field | Empty |
| Le         | Empty |

Response:

|            |                               |
|------------|-------------------------------|
| Data field | Empty                         |
| SW1-SW2    | 9000 or specific status bytes |

### 7.3 Command Usage under Different I/O Protocols

For the OpenPGP application the T=1 protocol (ISO 7816-3) is recommended for cards with contacts. However other protocols (one or more) in a card are possible too. The OpenPGP application is designed to run under every protocol (e. g. T=0, contactless) that is provided by the card reader. If contactless protocols (e. g. ISO 14443) are implemented, card and reader should support Secure Messaging (e. g. VERIFY) to avoid data tracking 'over the air'.

### 7.4 Class Byte Definitions

For the OpenPGP application all standard commands are used with a class byte (CLA) coding according to ISO. The following values are defined (CLA 10 and 1C are only relevant if the card supports command chaining).

| CLA | Description  |
|-----|--|
| 00  | CLA without SM (last or only command of a chain)                                   |
| 0C  | CLA with SM and header authentication (last or only command of a chain)            |
| 10  | CLA without SM (command is not the last command of a chain)                        |
| 1C  | CLA with SM and header authentication (command is not the last command of a chain) |

### 7.5 Secure Messaging (SM)

The OpenPGP application defines secure messaging with static keys in this version and uses a variant of SM from EN 419212 with AES without Send Sequence Counter (CBC mode). Command data are encrypted first and then protected by a cryptogram. SM is optional and announced in Extended capabilities. Only a sub-set of the commands may support SM (see CLA-Definition in the command section). SM is designed as a replacement of the Admin-PW (PW3) and can be used to change data objects online (helpful for company cards, admin = company; user = employee). It is highly recommended to use SM for security related commands (e. g. VERIFY) if the OpenPGP application runs in contactless mode, because all data on the interface can be traced easily within a range of several meters.

SM is defined for the algorithm AES with 128 or 256 bit. The supported algorithm is announced in Extended capabilities, the related keys are stored in the application DOs 'SM Key-ENC' and 'SM Key-MAC'. By default the content of the DOs is empty and SM will

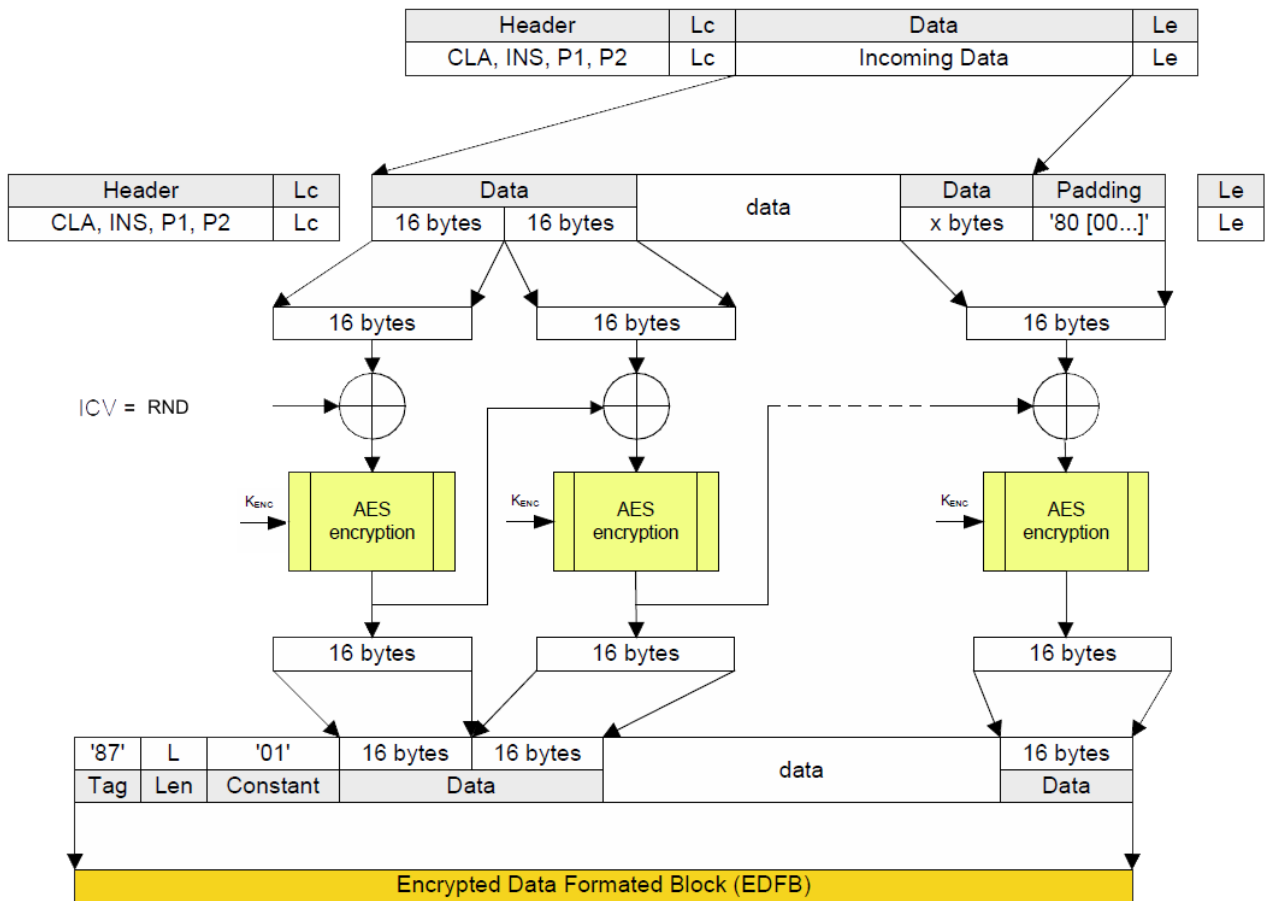


not work. The keys can be set with a PUT DATA command after correct presentation of the Admin-PW (PW3). From that point on all related commands can be used with SM. A command with correct SM has the same access condition to data in the application than a VERIFY with PW3.

If existing keys are replaced with PUT DATA, the card shall use the new values for the calculation of the response. The deletion of SM-keys should be done simultaneous with the DO 'F4', otherwise the functionality of SM is undefined. A key is deleted by writing an empty data string with PUT DATA.

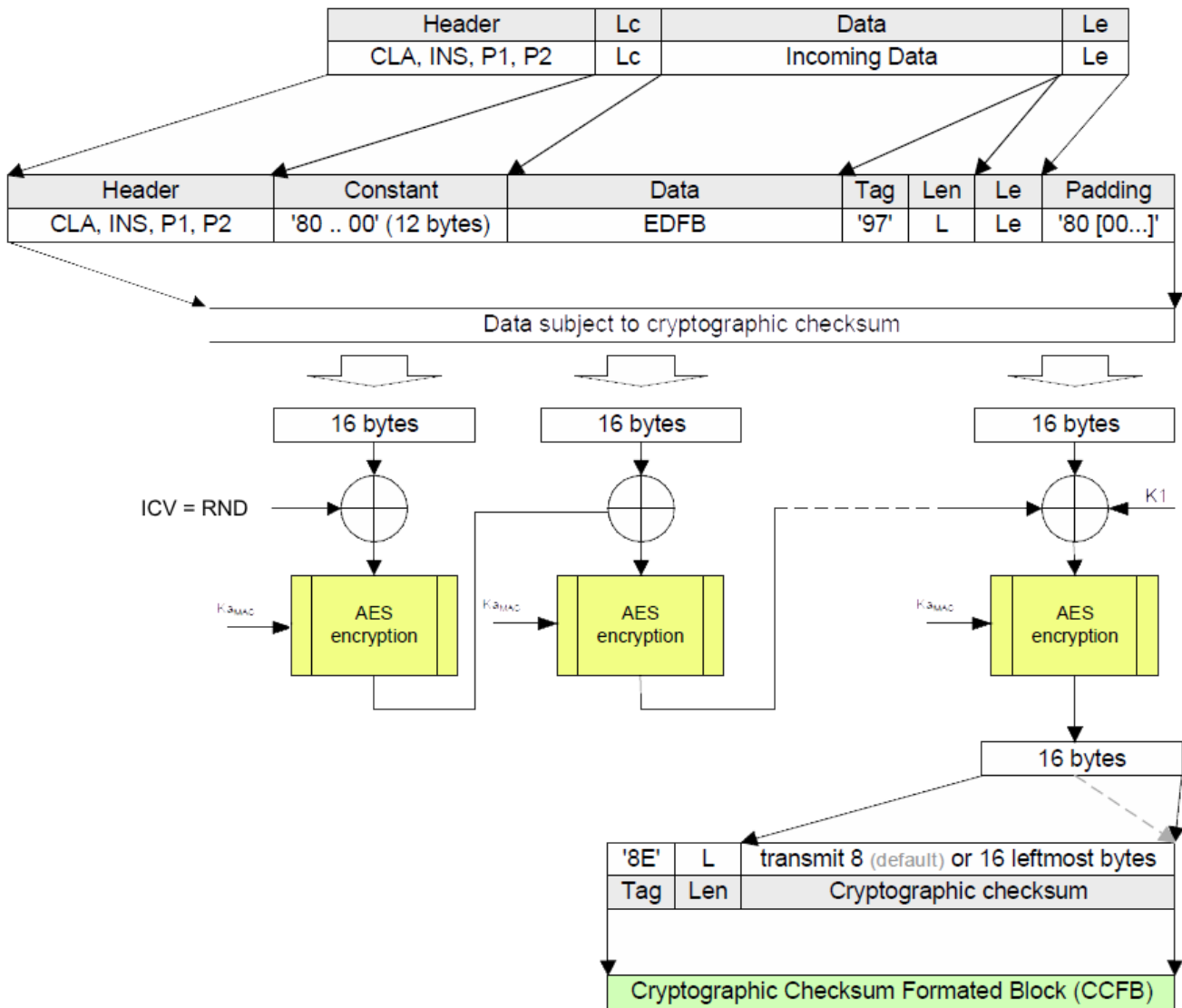
Rules for SM usage:

- For a command with even INS, any command/response data is encrypted and encapsulated in a Tag 87 with padding indicator (01).
- For a command with odd INS, any command/response data is encrypted and encapsulated in a Tag 85 without padding indicator.
- Commands with response (Le field not empty) have a protected Le-field (Tag 97) in the command data.
- Commands with response have a protected processing status (Tag 99) in the response.
- Commands without response (e. g. Case 3) have no processing status and/or MAC in the response field (SW1SW2 only).
- All SM-commands have at least an unencrypted MAC (Tag 8E) in the command data field.
- To protect the SM commands against replay attacks, a random number (RND) of 16 bytes is used as Initial Chaining Value (ICV). Before each command with SM the external world shall read a RND with 16 bytes from the card (Get Challenge). The last challenge from the card is internally used as ICV and is valid for one command and the response only.
- Encryption is done first in CBC-Mode with ICV = RND.
- MACing is done on the encrypted data with ICV = RND in CMAC-Mode.

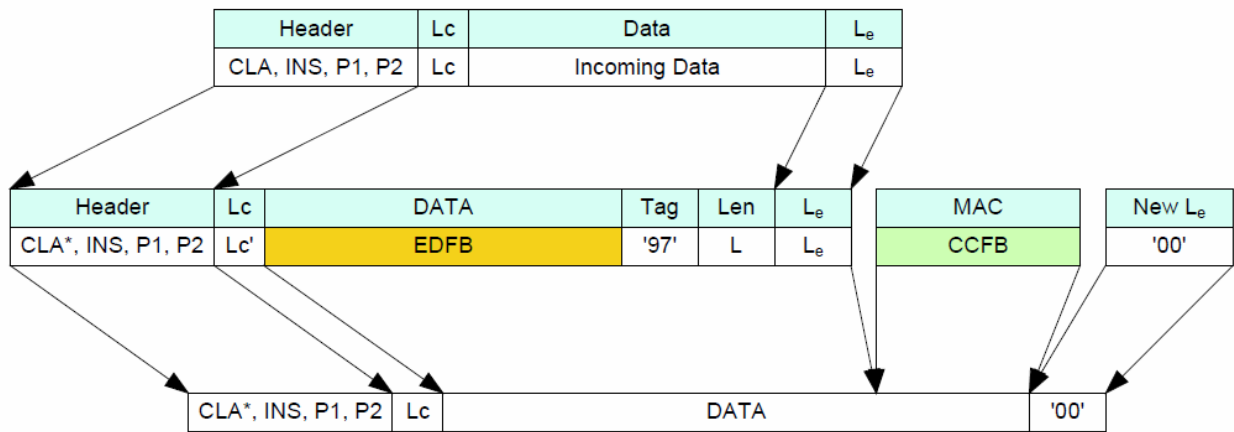


### APDU command data encryption (even INS) with AES in CBC mode

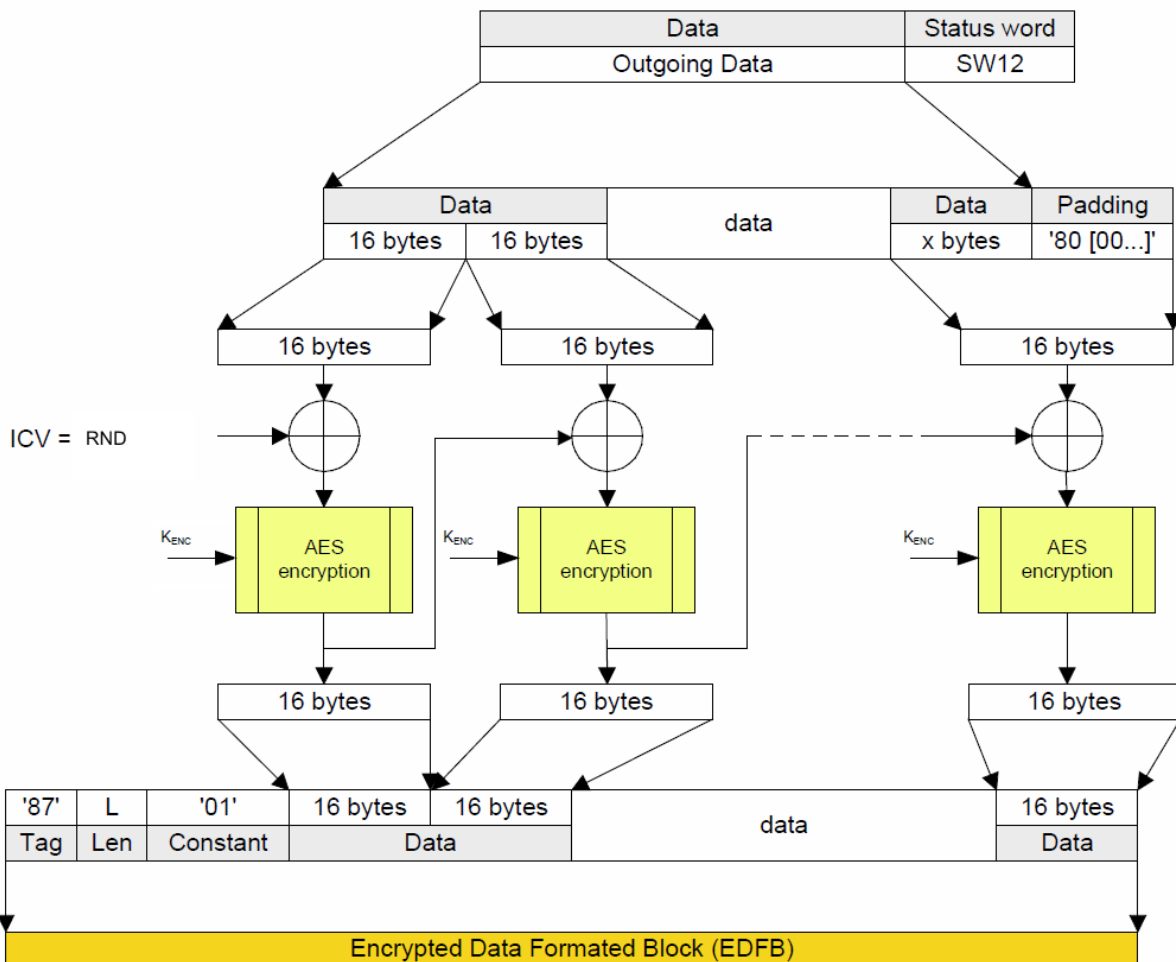
For odd INS the Tag for EDFB is '85' and the padding indication (constant '01') is omitted.



### APDU computation of AES cryptographic checksum in CMAC mode

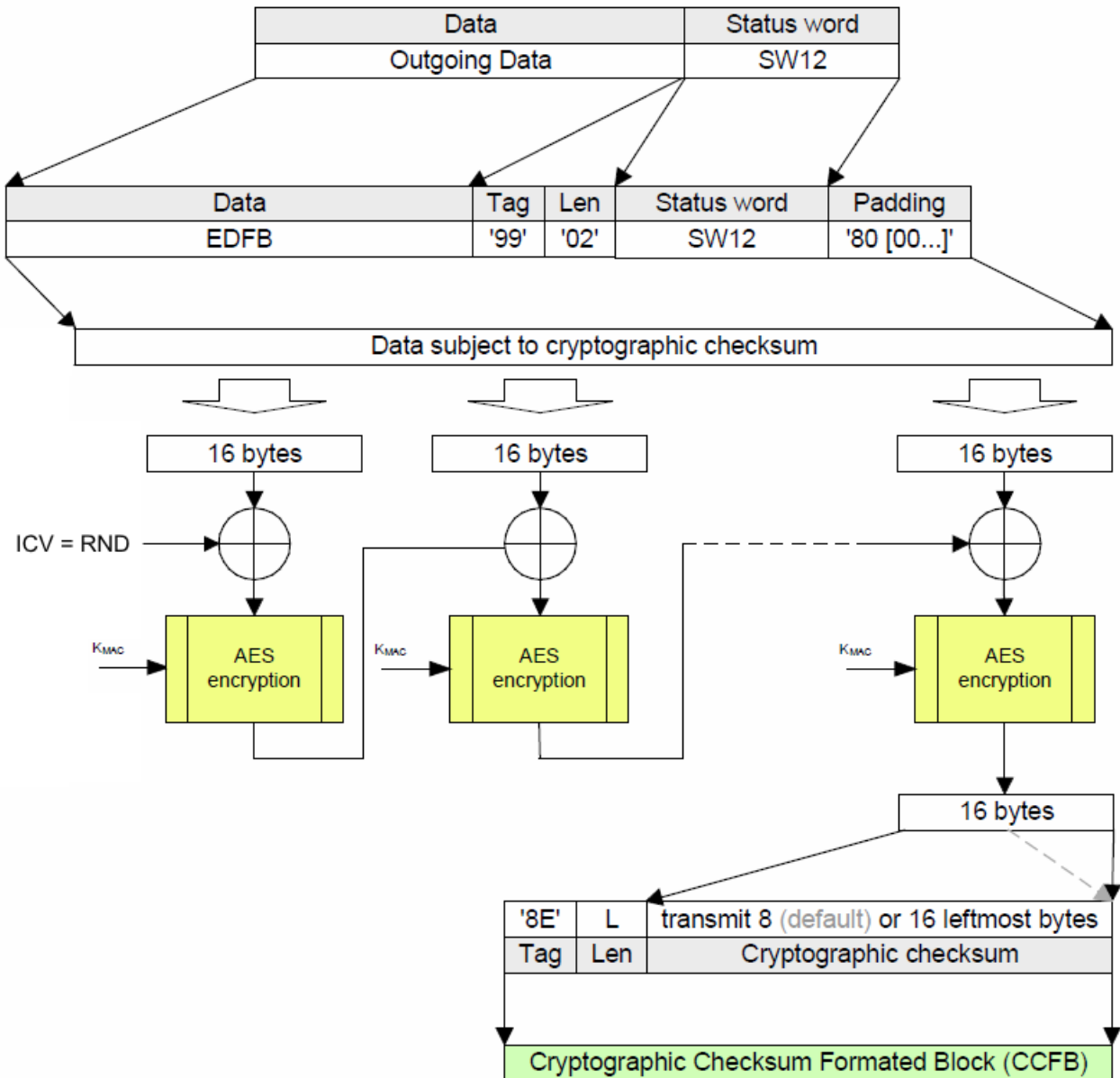


### Final APDU command construction

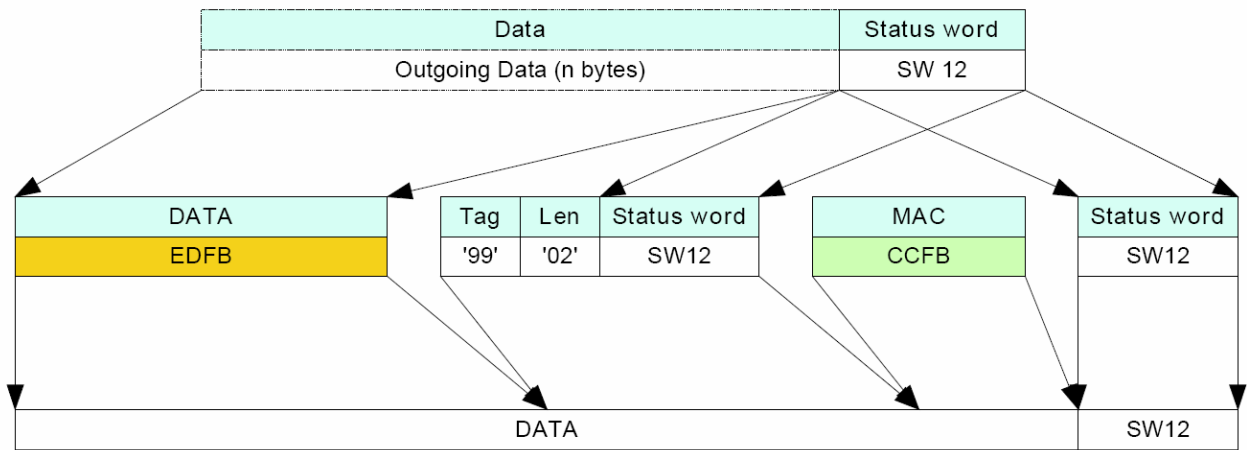


### Response APDU data encryption using AES in CBC-mode

For odd INS the Tag for EDFB is '85' and the padding indication (constant '01') is omitted.



**Response APDU computation of cryptographic checksum with AES in CMAC-Mode**



### Final response APDU construction

#### 7.6 Logical Channels

The OpenPGP application does not use logical channels in this version. Channel number zero is assumed for all commands.

## 7.7 Command Chaining

If command data are too long for a single command (e. g. PSO:DEC with RSA 2048 and support of Short length only) a card may provide command chaining. The feature is announced in card capabilities in the Historical bytes.

If chaining is used, the CLA of a command shall be set to the appropriate value and the command data consist of the first block of the complete command data. The card stores the data block internally and wait for a follow-up command with the same INS/P1/P2. If sufficient the next data block is concatenated by the card, up to an equal command with no chaining bit in the CLA. Then the command is executed with the whole data from all previous commands in this chain.

The length of a data field in a command with chaining bit in CLA should be equal to the maximum input buffer of the card announced in ATR/INFO.

Chaining can be combined with Secure Messaging, the complete length of a single command shall not exceed the maximum input buffer of the card announced in ATR/INFO.

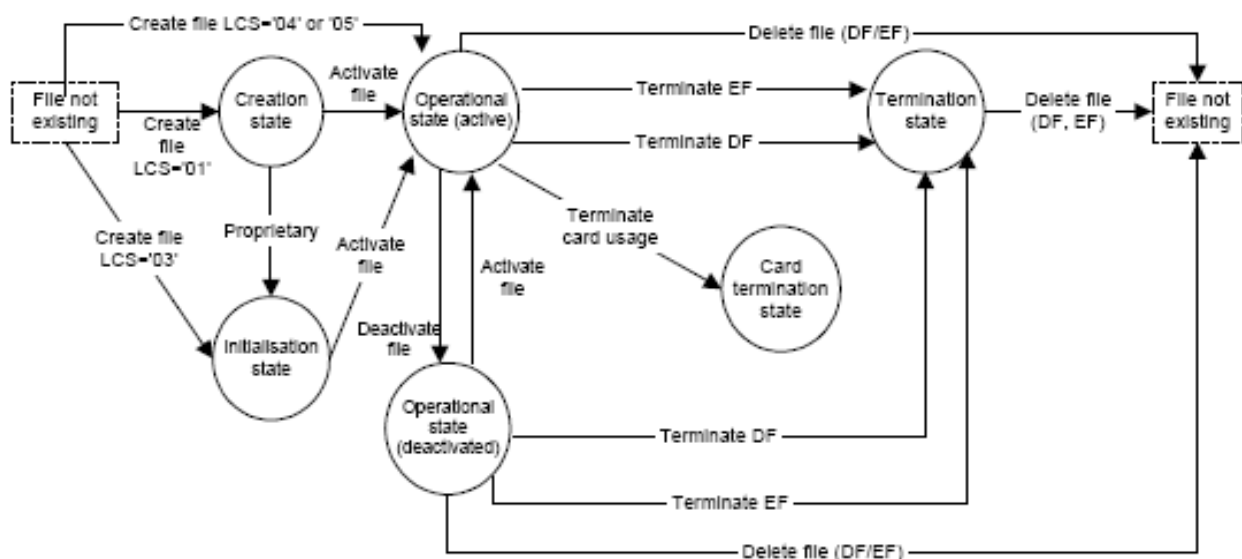
Commands that may support chaining have an appropriate coding in their CLA definition.

For performance reasons chaining should be avoided, all modern card readers support APDUs with 2 KB and more.

## 7.8 Life Cycle Management

Many users of an OpenPGP card asked me if it would be possible to RESET a card to the origin state at delivery time (all DOs empty or with default values). ISO 7816 offers no command directly for that purpose, but it can be done with a behaviour of the card called Life Cycle Status (LCS). A card can have a life cycle from 'creation state' to 'initialisation state' to 'operational state' (that can be activated or deactivated) up to 'termination state'. ISO defines two types of LCS: primary state, that allows only transitions in one direction and is irreversible and secondary state, that is application specific and reversible.

The following figure from ISO 7816-9 demonstrates it:



The solution for the OpenPGP application uses an application specific secondary state, that has some additions to the defined behaviour in ISO. In the phase from initialisation to operational state a card normally gets its data. The ISO command ACTIVATE FILE (related to a DF) can set this state. So the solution for the OpenPGP application is to use the ACTIVATE FILE command to reset all values in the card to their default values (user DOs empty, PWs to default values, keys/fingerprints empty etc.).

ISO defines no direct way from operational to initialisation state. The solution for the secondary state in the OpenPGP application is a transition of the termination state. After termination, the OpenPGP application falls back to initialisation state. Then the card can be resetted (re-newed) by an ACTIVATE FILE command.



Because this behaviour is proprietary and cannot be implemented on all existing platforms, it is optional and announced in the Life Cycle Status indicator in the Historical bytes.

## 7.9 Status Bytes

After a command the chip returns a pair of status bytes (return code). All codings of ISO 7816-4 are valid for the card and may occur in a specific context.

The following table shows possible coding for status bytes:

|    |       |   |
|----|-------|---|
| 61 | xx    | Command correct, xx bytes available in response (normally used under T=0 or for commands under any protocol with long response data that cannot be transmitted in one response) |
| 62 | 85    | Selected file in termination state  |
| 63 | CX    | Password not checked, 'X' encodes the number of further allowed retries   |
| 64 | 02-80 | Triggering by the card<br>0E = Out of Memory (BasicCard specific)   |
| 65 | 81    | Memory failure  |
| 66 | 00    | Security-related issues (reserved for UIF in this application)  |
| 67 | 00    | Wrong length (Lc and/or Le)   |
| 68 | 81    | Logical channel not supported   |
| 68 | 82    | Secure messaging not supported  |
| 68 | 83    | Last command of the chain expected  |
| 68 | 84    | Command chaining not supported  |
| 69 | 82    | Security status not satisfied<br>PW wrong<br>PW not checked (command not allowed)<br>Secure messaging incorrect (checksum and/or cryptogram)                                    |
| 69 | 83    | Authentication method blocked<br>PW blocked (error counter zero)  |
| 69 | 85    | Condition of use not satisfied  |
| 69 | 87    | Expected SM data objects missing (e. g. SM-key)   |
| 69 | 88    | SM data objects incorrect (e. g. wrong TLV-structure in command data)   |
| 6A | 80    | Incorrect parameters in the data field  |
| 6A | 82    | File or application not found   |
| 6A | 88    | Referenced data not found   |
| 6B | 00    | Wrong parameters P1-P2  |
| 6D | 00    | Instruction (INS) not supported   |
| 6E | 00    | Class (CLA) not supported   |
| 90 | 00    | Command correct   |

## 8 Literature

European Standard (2015):

FprEN 419212-1, Application Interface for Secure Elements for Electronic Identification, Authentication and Trusted Services, Part 1: Introduction and common definitions

European Standard (2015):

FprEN 419212-3, Application Interface for Secure Elements for Electronic Identification, Authentication and Trusted Services, Part 3: Device Authentication Protocols

European Standard (2015):

FprEN 419212-4, Application Interface for Secure Elements for Electronic Identification, Authentication and Trusted Services, Part 4: Privacy specific Protocols

European Standard (2015):

FprEN 419212-5, Application Interface for Secure Elements for Electronic Identification, Authentication and Trusted Services, Part 5: Trusted eServices

European Standard (2016):

FprEN 419212-2, Application Interface for Secure Elements for Electronic Identification, Authentication and Trusted Services, Part 2: Signature and Seal Services

gematik (2016):

Einführung der Gesundheitskarte – Spezifikation des Card Operating System (COS), Version 3.9.0

ISO/IEC (2003):

ISO 9564-3, Personal Identification Number management and security, Part 3: Requirements for offline PIN handling in ATM and POS systems

ISO/IEC (2004):

ISO/IEC 7816-9, Identification cards - Integrated circuit cards, Part 9: Commands for card management

ISO/IEC (2006):

ISO/IEC 7816-3, Identification cards - Integrated circuit cards - Part 3: Cards with contacts: Electrical interface and transmission protocols

ISO/IEC (2013):

ISO/IEC 7816-4, Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange

ISO/IEC (2014):

ISO/IEC 7816-4, Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange; Technical Corrigendum 1

ISO/IEC (2016):

ISO/IEC FDIS 18328-3, Identification cards – ICC managed devices, Part 3: Organization, security and commands for interchange

ISO/IEC (2016):

ISO/IEC FDIS 7816-6.2, Identification cards - Integrated circuit cards - Part 6: Interindustry data elements for interchange

ISO/IEC (2016):

ISO/IEC FDIS 7816-8, Identification cards - Integrated circuit cards, Part 8: Commands for security operations

RSA Laboratories (2002):

PKCS #1 v2.1: RSA Encryption Standard

The Internet Society (2007):

RFC 4880: OpenPGP Message Format

The Internet Society (2012):

RFC 6637: Elliptic Curve Cryptography (ECC) in OpenPGP

## ANNEX

### 9 Flow Charts

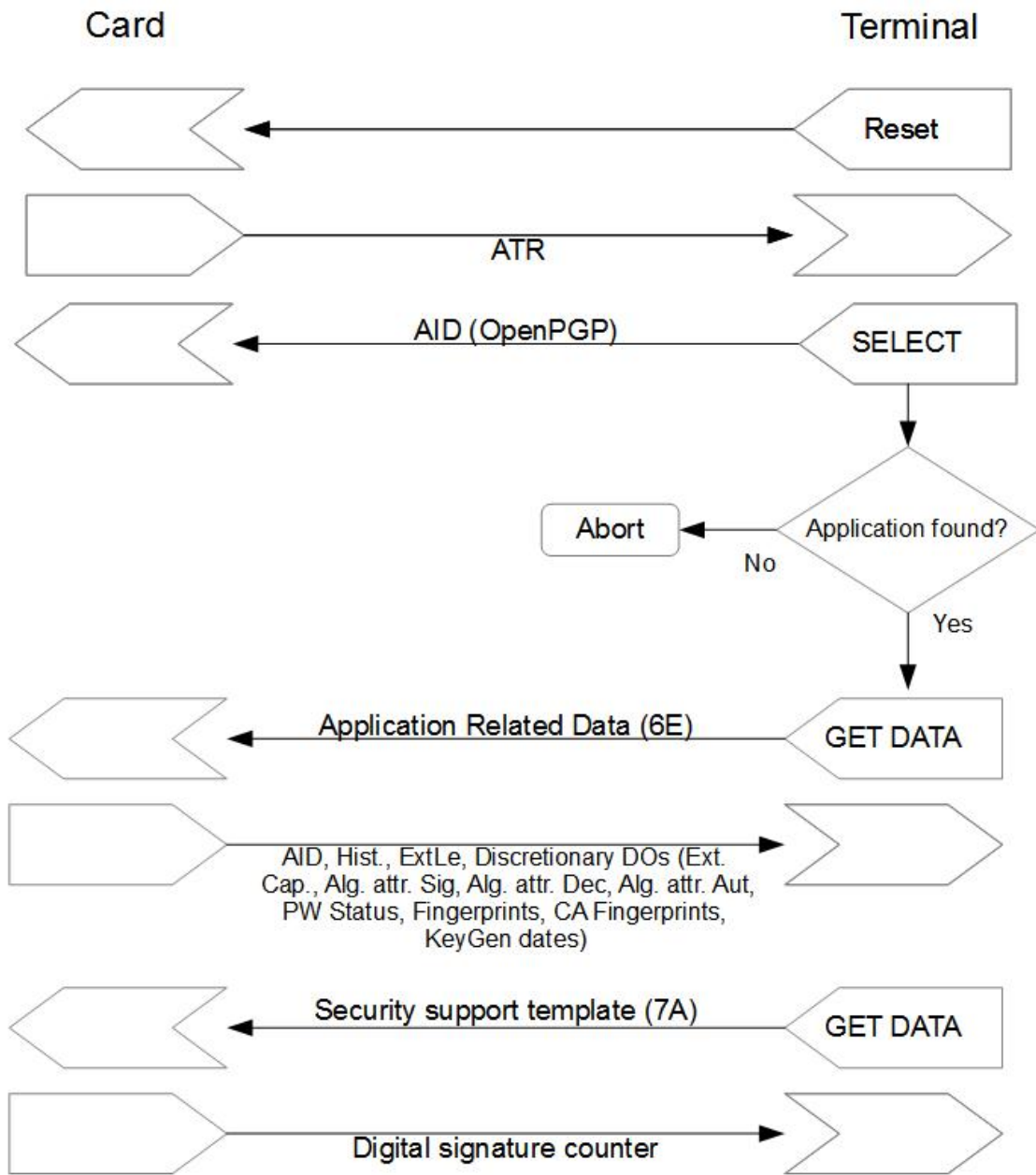
The communication scenarios illustrate some possibilities for the use of the OpenPGP application. Only a few functions are described, there are several additional functions available. The data in the commands may not be complete and are informative only.

In principle, the application sequences to be realised apply to the application structure described in the specification. The realisation of the application sequences is generally made possible by the global commands provided to the card by the operating system, taking account of the security structure.

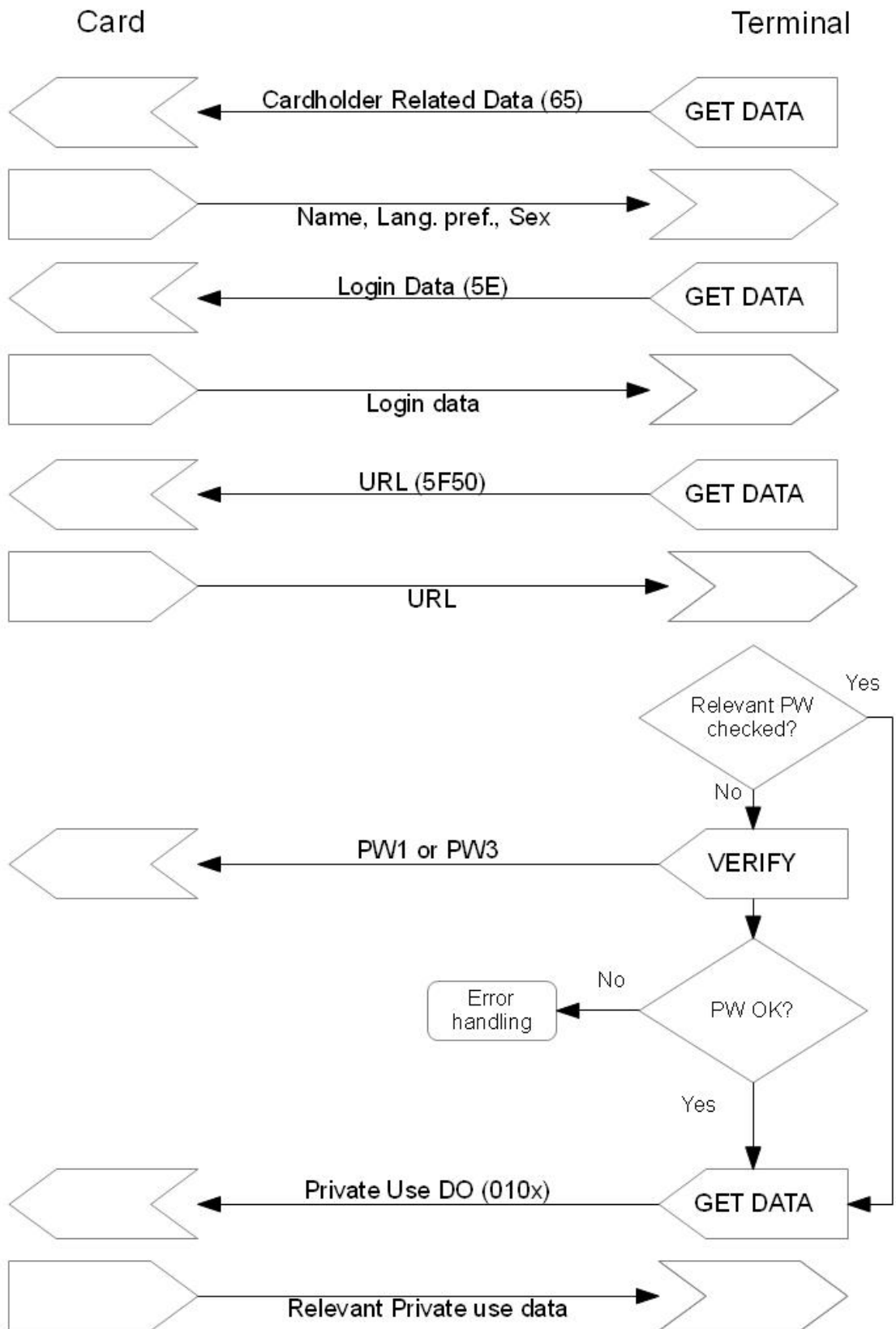
With respect to the sequences, only those application data are considered that are relevant at the interface between card and terminal. Standard return codes, header information and error events are not included for reasons of clarity. The scenarios are intended to clarify the essential mechanisms of the application and are used to facilitate a better understanding of the entire specification. They are not intended to serve as the only basis for the realisation of terminal programs.

As long as the security guidelines required by the applications are observed, the modification of the following scenarios is possible.

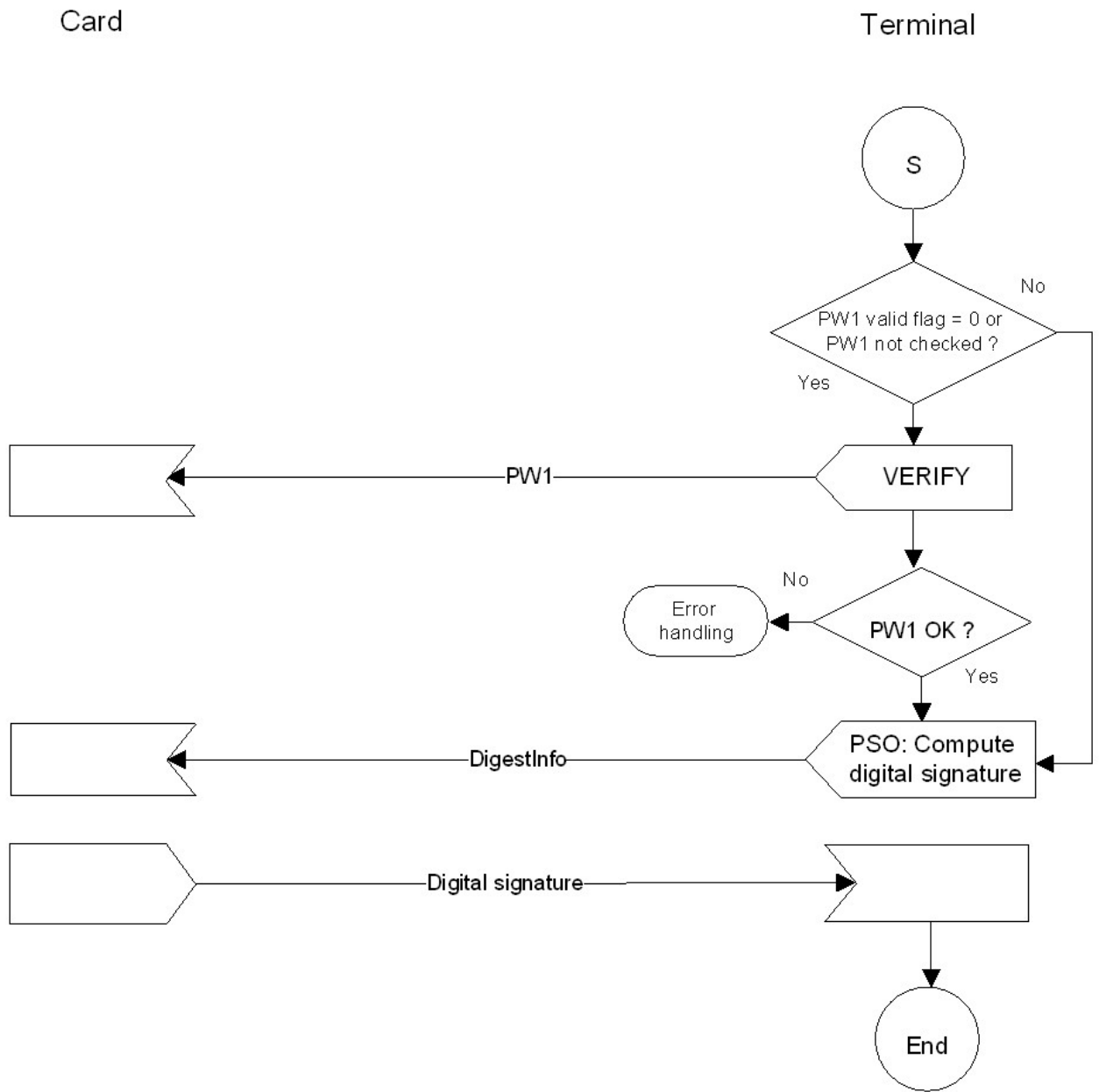
## 9.1 Application Selection reading main DOs



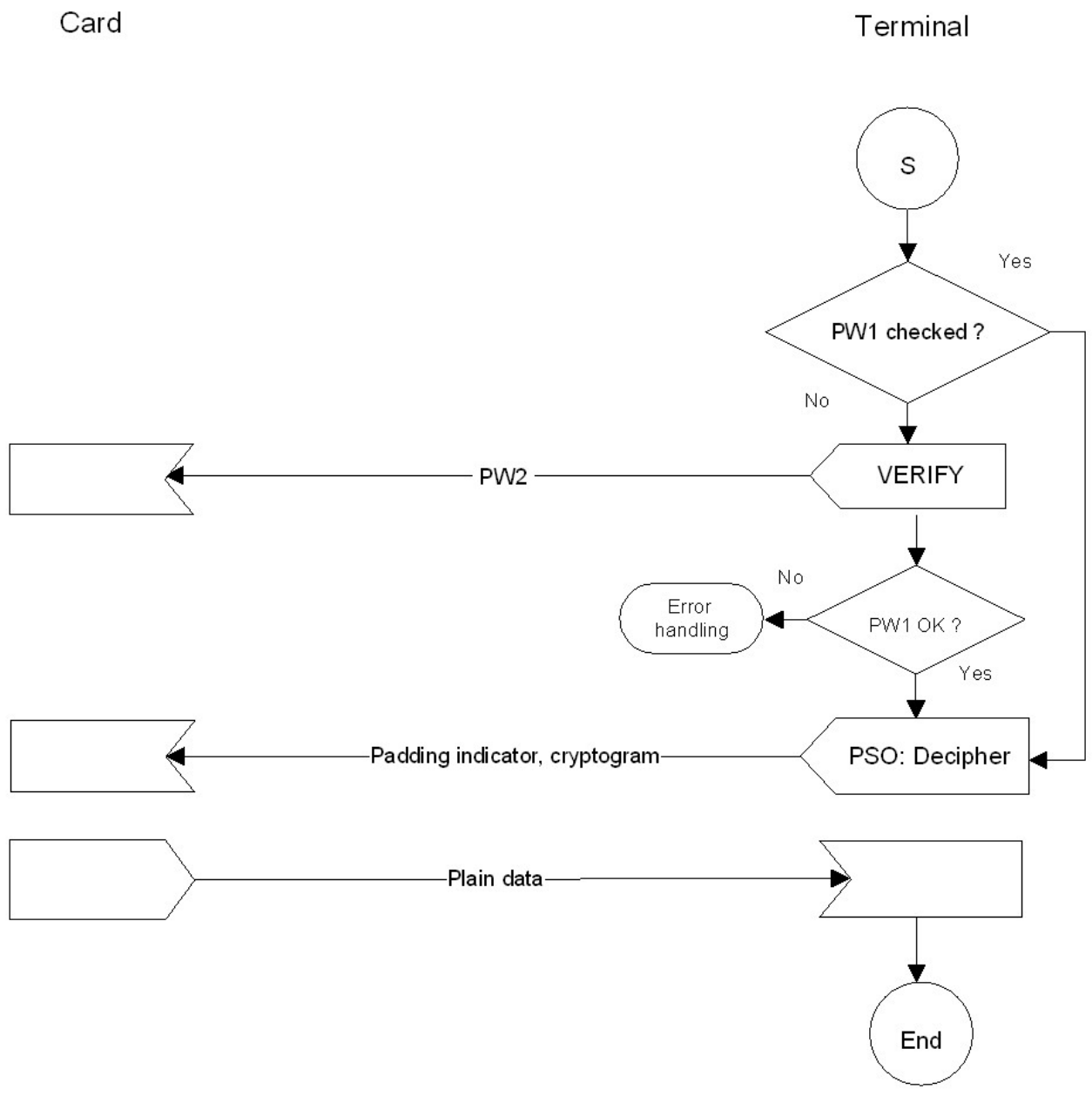
## 9.2 Reading optional Data objects



### 9.3 Compute Digital Signature

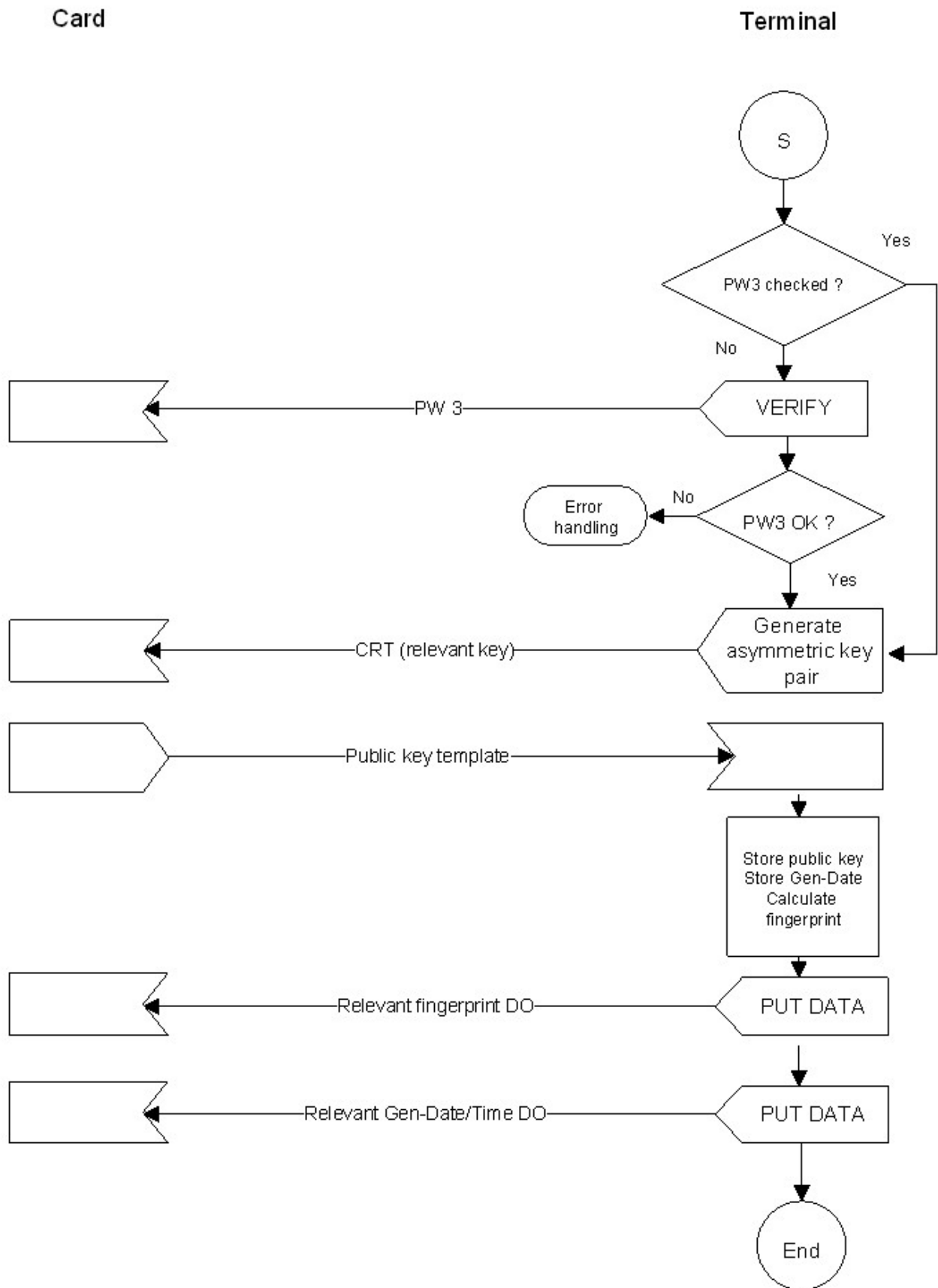


## 9.4 Decrypt Message

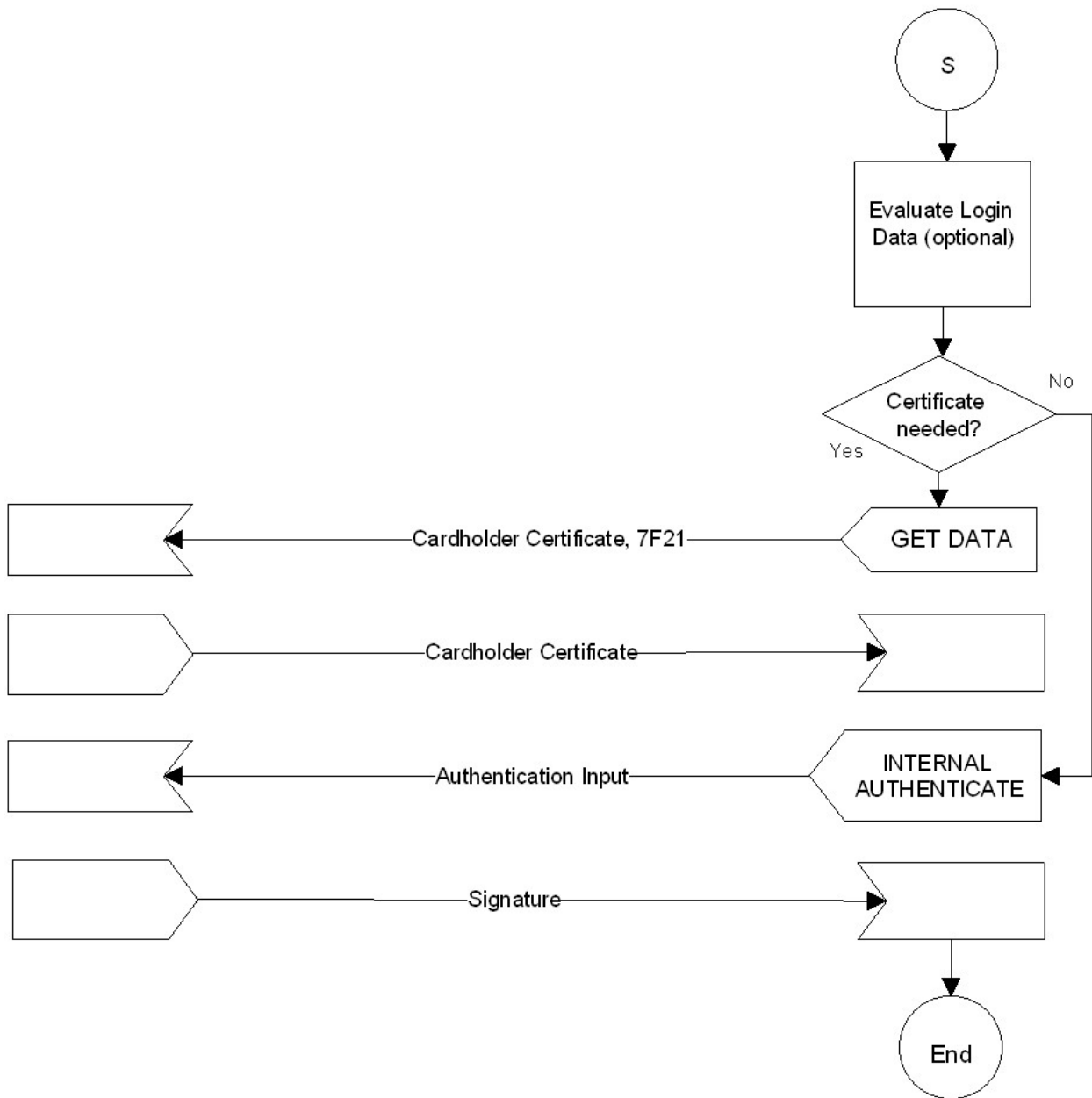




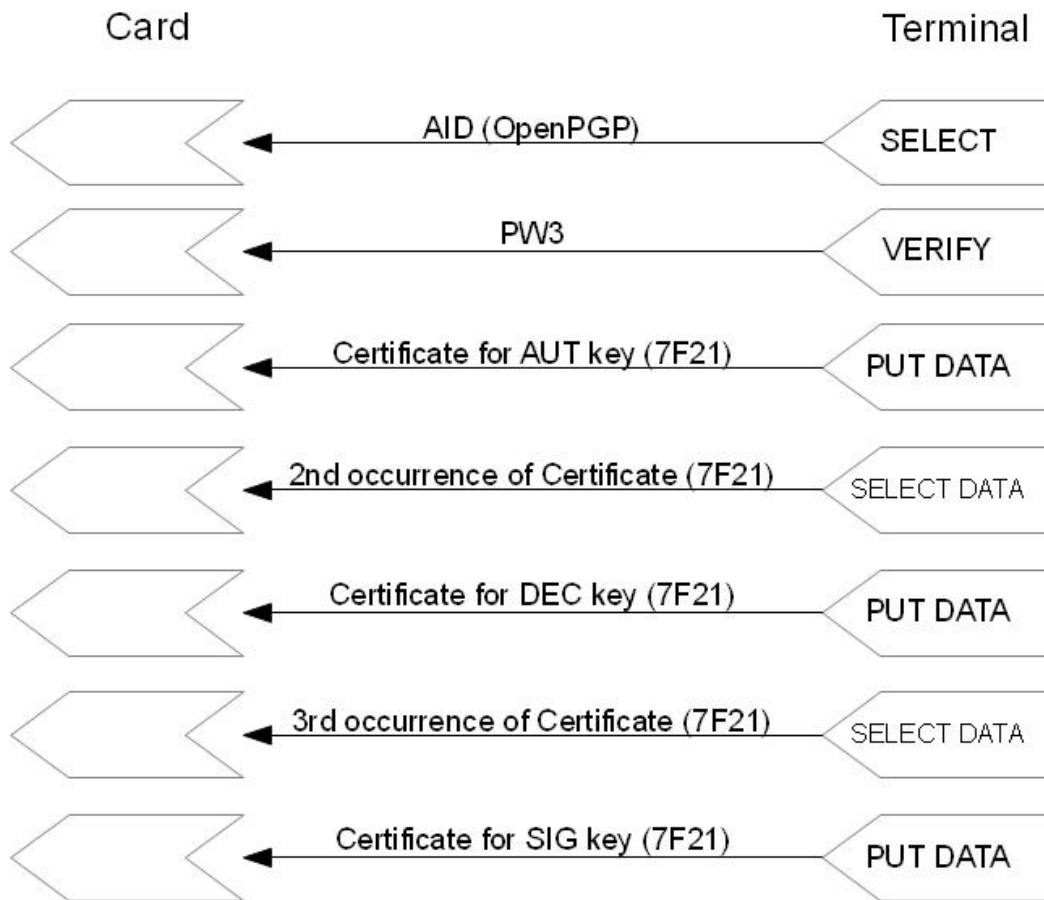
## 9.5 Generate Private Key



## 9.6 Client/Server Authentication



## 9.7 Storage of Card Holder Certificates



## 10 Domain parameter of supported elliptic curves

Most tables and values are taken from gematik specification "Spezifikation des Card Operation System (COS), Version 3.7.0".

**ansix9p256r1, OID = {1.2.840.10045.3.1.7} = '2A8648CE3D030107'**

The used domain parameters are from [ANSI X9.62] and are identical with [FIPS 186-4].

*p* 'FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF  
FFFFFFFF FFFFFFFF'  
*a* 'FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF  
FFFFFFFF FFFFFFFF C'  
*b* '5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0  
CC53B0F6 3BCE3C3E 27D2604B'  
*G* compressed:  
'03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81  
2DEB33A0 F4A13945 D898C296'  
as coordinates:  
xg = '6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81  
2DEB33A0 F4A13945 D898C296'  
yg = '4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357  
6B315ECE CBB64068 37BF51F5'  
*n* 'FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD  
A7179E84 F3B9CAC2 FC632551'  
*h* 1

**ansix9p384r1, OID = {1.3.132.0.34} = '2B81040022'**

*p* 'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000  
FFFFFFFF'  
*a* 'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000  
FFFFFFFF C'  
*b* 'B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112  
0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF  
'  
*G* compressed:  
'03 AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62  
8BA79B98 59F741E0 82542A38 5502F25D BF55296C 3A545E38  
72760AB7'  
as coordinates:  
xg = 'AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62  
8BA79B98 59F741E0 82542A38 5502F25D BF55296C 3A545E38  
72760AB7'  
yg = '3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD  
289A147C E9DA3113 B5F0B8C0 0A60B1CE 1D7E819D 7A431D7C  
90EA0E5F'  
*n* 'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A  
CCC52973'  
*h* 1

**ansix9p521r1, OID = {1.3.132.0.35} = '2B81040023'**

*p* '01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF'  
*a* '01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF'  
*b* '0051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B  
99B315F3 B8B48991 8EF109E1 56193951 EC7E937B 1652C0BD  
3BB1BF07 3573DF88 3D2C34F1 EF451FD4 6B503F00'  
*G* compressed:  
'0200C6 858E06B7 0404E9CD 9E3ECB66 2395B442 9C648139  
053FB521 F828AF60 6B4D3DBA A14B5E77 EFE75928 FE1DC127  
A2FFA8DE 3348B3C1 856A429B F97E7E31 C2E5BD66'  
as coordinates:  
xg = '000000C6 858E06B7 0404E9CD 9E3ECB66 2395B442  
9C648139 053FB521 F828AF60 6B4D3DBA A14B5E77 EFE75928  
FE1DC127 A2FFA8DE 3348B3C1 856A429B F97E7E31 C2E5BD66'  
yg = '00000118 39296A78 9A3BC004 5C8A5FB4 2C7D1BD9  
98F54449 579B4468 17AFBD17 273E662C 97EE7299 5EF42640  
C550B901 3FAD0761 353C7086 A272C240 88BE9476 9FD16650'  
*n* '01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  
FFFFFFFF FFFFFFFF FFFFFFFF 51868783 BF2F966B 7FCC0148  
F709A5D0 3BB5C9B8 899C47AE BB6FB71E 91386409'  
*h* 1

**brainpoolP256r1, OID={1.3.36.3.3.2.8.1.1.7} = '2B2403030208010107'**

The used domain parameters are from [RFC5639].

*p* 'A9FB57DB A1EEA9BC 3E660A90 9D838D72 6E3BF623 D5262028  
2013481D 1F6E5377'  
*a* '7D5A0975 FC2C3057 EEF67530 417AFFE7 FB8055C1 26DC5C6C  
E94A4B44 F330B5D9'  
*b* '26DC5C6C E94A4B44 F330B5D9 BBD77CBF 95841629  
5CF7E1CE 6BCCDC18 FF8C07B6'  
*G* as coordinates:  
xg = '8BD2AEB9 CB7E57CB 2C4B482F FC81B7AF B9DE27E1  
E3BD23C2 3A4453BD 9ACE3262'  
yg = '547EF835 C3DAC4FD 97F8461A 14611DC9 C2774513  
2DED8E54 5C1D54C7 2F046997'  
*n* 'A9FB57DB A1EEA9BC 3E660A90 9D838D71 8C397AA3 B561A6F7  
901E0E82 974856A7'  
*h* 1

**brainpoolP384r1, OID={1.3.36.3.3.2.8.1.1.11} = '2B240303020801010B'**

*p* '8CB91E82 A3386D28 0F5D6F7E 50E641DF 152F7109  
ED5456B4 12B1DA19 7FB71123 ACD3A729 901D1A71  
87470013 3107EC53'  
*a* '7BC382C6 3D8C150C 3C72080A CE05AFA0 C2BEA28E  
4FB22787 139165EF BA91F90F 8AA5814A 503AD4EB  
04A8C7DD 22CE2826'  
*b* '04A8C7DD 22CE2826 8B39B554 16F0447C 2FB77DE1  
07DCD2A6 2E880EA5 3EEB62D5 7CB43902 95DBC994  
3AB78696 FA504C11'  
*G* as coordinates:  
xg = '1D1C64F0 68CF45FF A2A63A81 B7C13F6B 8847A3E7  
7EF14FE3 DB7FCAFE 0CBD10E8 E826E034 36D646AA  
EF87B2E2 47D4AF1E'  
yg = '8ABE1D75 20F9C2A4 5CB1EB8E 95CFD552 62B70B29  
FEEC5864 E19C054F F9912928 0E464621 77918111 42820341  
263C5315'  
*n* '8CB91E82 A3386D28 0F5D6F7E 50E641DF 152F7109  
ED5456B3 1F166E6C AC0425A7 CF3AB6AF 6B7FC310  
3B883202 E9046565'  
*h* 1

**brainpoolP512r1, OID={1.3.36.3.3.2.8.1.1.13} = '2B240303020801010D'**

*p* 'AADD9DB8 DBE9C48B 3FD4E6AE 33C9FC07 CB308DB3  
B3C9D20E D6639CCA 70330871 7D4D9B00 9BC66842  
AECDA12A E6A380E6 2881FF2F 2D82C685 28AA6056 583A48F3  
'  
*a* '7830A331 8B603B89 E2327145 AC234CC5 94CBDD8D  
3DF91610 A83441CA EA9863BC 2DED5D5A A8253AA1  
0A2EF1C9 8B9AC8B5 7F1117A7 2BF2C7B9 E7C1AC4D  
77FC94CA'  
*b* '3DF91610 A83441CA EA9863BC 2DED5D5A A8253AA1  
0A2EF1C9 8B9AC8B5 7F1117A7 2BF2C7B9 E7C1AC4D  
77FC94CA DC083E67 984050B7 5EBAE5DD 2809BD63  
8016F723'  
*G* as coordinates:  
xg = '81AEE4BD D82ED964 5A21322E 9C4C6A93 85ED9F70  
B5D916C1 B43B62EE F4D0098E FF3B1F7 8E2D0D48 D50D1687  
B93B97D5F 7C6D5047 406A5E68 8B352209 BCB9F822'  
yg = '7DDE385D 566332EC C0EABFA9 CF7822FD F209F700  
24A57B1A A000C55B 881F8111 B2DCDE49 4A5F485E  
5BCA4BD8 8A2763AE D1CA2B2F A8F05406 78CD1E0F  
3AD80892'  
*n* 'AADD9DB8 DBE9C48B 3FD4E6AE 33C9FC07 CB308DB3  
B3C9D20E D6639CCA 70330870 553E5C41 4CA92619 41866119  
7FAC1047 1DB1D381 085DDADD B5879682 9CA90069'  
*h* 1