

**German
Health Professional Card
and
Security Module Card**

Specification

- Pharmacist & Physician -

Version 2.0

31.07.2003

Commissioned by:

Central Research Institute of Ambulatory Health Care in Germany

German Medical Association

National Association of Office Based Physicians

Werbe- u. Vertriebsgesellschaft Deutscher Apotheker mbH

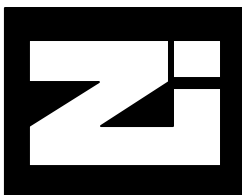
Editor: Bruno Struif

Co-editors: Alfred Giessler (X.509 Certificates) and Björn Schneider (CIOs)

Fraunhofer-Institute for Secure Telecooperation (FhG-SIT)

Rheinstrasse 75, 64295 Darmstadt

E-Mail: struif@sit.fraunhofer.de



Contents

1	Scope.....	1
2	General Card Requirements	1
3	References.....	1
4	Abbreviations and notations	3
4.1	Abbreviations	3
4.2	Notations	4
5	Technical Characteristics, Answer-to-Reset and Transmission Protocols	5
5.1	Technical Characteristics.....	5
5.2	Answer-to-Reset	5
5.3	Transmission Protocols.....	5
6	The Health Professional Card	5
6.1	General Structure.....	5
6.2	Elementary Files at MF-Level	5
6.2.1	EF.DIR.....	5
6.2.2	EF.GDO	6
6.2.3	EF.C.HPC.AUT	6
6.2.4	EF.PIN/KEYs.....	6
6.3	HPC opening	6
6.3.1	Command sequence after ATR.....	6
6.3.2	Reading of EF.DIR and EF.GDO	6
7	The Health Professional Application HPA.....	7
7.1	Certificate Service Provider and Certificates.....	7
7.2	File structure and file content.....	8
7.2.1	EF.PIN	8
7.2.2	EF.PrK.....	8
7.2.3	EF.PuK	9
7.2.4	EF.SK.....	9
7.2.5	EF.ARR	9
7.2.6	EF.HPD	9
7.2.7	EF.DM	10
7.2.8	EF.C.HP.ES.....	10
7.2.9	EF.C.HP.ES-AC1, -AC2 and -AC3	10
7.2.10	EF.C.HP.AUT.....	10
7.2.11	EF.C.CSP.CS	10
7.2.12	EF.C.RCSP.ES-SELF	10
7.2.13	EF.C.RCSP.AUT-SELF	10
7.3	Security Environments	10
7.4	Application selection	10
7.5	SE Selection.....	11
7.6	Reading and Updating of HP Data and Certificates.....	11
7.6.1	Reading of the Health Professional Data	11
7.6.2	Reading and Updating of HP Certificates.....	11
7.7	PIN Management.....	12
7.7.1	PIN Encoding	12
7.7.2	PIN Verification.....	12
7.7.3	PIN Change	12
7.7.4	Reset of Retry Counter and Setting a New PIN.....	13
7.8	Creation of an Electronic Signature.....	13
7.8.1	General aspects	13
7.8.2	Setting PrK.HP.ES.....	13
7.8.3	Delivery of the Hash Value	14
7.8.4	Initiation of the Electronic Signature Creation.....	14
7.9	Client / Server Authentication	14
7.10	Document cipher key decipherment.....	15
7.11	HPC / PDC Authentication	16

7.11.1	Mutual Authentication with a symmetric Algorithm	16
7.11.2	Mutual Authentication with an asymmetric Algorithm	17
7.11.3	Import of PuK.CSP.CS-CV	20
7.12	Establishment of a Trusted Channel between HPC and SMC	20
7.12.1	HPC and SMC know each other	20
7.12.2	HPC and SMC don't know each other	21
7.12.3	Reading and updating the Display Message	21
8	Cryptographic Information Application	22
8.1	General Structure.....	22
8.2	Application Selection.....	22
8.3	Reading the CIA Files	22
9	Channel Management	23
9.1	General Aspects.....	23
9.2	Opening a Logical Channel	23
9.3	Closing a Logical Channel	23
10	The Security Module Card SMC	24
10.1	General Structure.....	24
10.2	Files, PINs, Keys and Certificates	24
10.3	Application Selection.....	24
10.4	Command Sequences	24
10.5	Support of a Trusted Channel between an SMC and a remote HPC	25
10.5.1	TC establishment and handling of secured HPC commands	25
10.5.2	Production of secured commands using PSO commands	25
10.5.3	Processing of secured responses using PSO commands	26
10.5.4	Production of secured commands using the ENVELOPE command	26
10.5.5	Processing of secured responses using the ENVELOPE command	26
Annex A (normative): ATR		27
Annex B (normative): File Attributes, Access Conditions and Security Environments		29
Annex C (normative): Structure and Content of ES certificates		37
Annex D (normative): Certificate for Authentication and Key Encipherment		61
Annex E (normative): Algorithms and Input Formats for Security Operations		62
Annex F (normative): Card Verifiable Authentication Certificate		64
Annex G (normative): Device Authentication, Session Key Agreement and SM		69
Annex H (normative): Cryptographic Information Objects		75
Annex I (normative): SMC/HPC Interaction		108
Annex J (informative): Examples of Card-to-Card Communications		109

1 Scope

This specification defines the card interfaces to

- the Health Professional Card (HPC) for individuals from accredited health professions and
- the Security Module Card (SMC) usable in health care environments

and authentication procedures between a HPC or SMC and a Patient Data Card (PDC) for verifying the authenticity of a PDC and proving access rights.

In particular, this specification defines

- technical characteristics
- conventions of data transmission
- file and data structures
- security mechanisms
- commands for achieving the required card services and
- structure and content of certificates.

The specification is made in such a way that it is adaptable also to the needs of other health professionals.

The specification takes into account

- the German signature law and signature ordinance (Signatur-Gesetz SigG and Signatur-Verordnung SigV)
- the DIN specification for Digital Signature Cards
- the e-Sign K specification for electronic signatures
- the relevant ISO-Standards (especially ISO/IEC 7816 Part 1-4, 8, 9 and 15)
- other sources (e.g. requirements from trust centers).

The lifetime of a HPC and an SMC is up to 6 years (max. validity period of a HP certificate).

2 General Card Requirements

Cards compliant to this specification have to support

- **commands** related to
 - direct application selection
 - user verification
 - handling of transparent files and support of short file identifiers
 - handling of record files (linear structure)

- electronic signature creation
- client/server authentication
- card-to-card authentication
- document key decipherment
- channel management (optional, i.e. there is a need for HPCs with and without this function)

- **cryptographic functions** for

- RSA with 1024/1536 bits
- SHA-1 and HF2
- DES-3
- secure messaging.

The electronic signature function shall be evaluated according to the German signature ordinance for secure signature creation devices (EAL4+, strength of mechanism high).

As used in this document, the conformance key word "shall" denotes a mandatory feature, the key word "should" denotes a feature that is recommended, but not mandatory, while the keyword "may" denotes a feature whose presence or absence does not influence the conformance.

NOTE – It is recommended that the HPC supports a logical end-of-file so that no '00'-bytes or 'FF'-bytes are transmitted if the physical size of the file is greater than a stored certificate. The logical EOF value may be handled in such a way that it is set to the highest address of a byte written in the file.

3 References

[ALGCAT]

Usable Cryptographic Algorithms (Geeignete Kryptgorithmen in Erfüllung der Anforderungen nach SigG/SigV), 09.09.2002
See also www.regtp.de

[ANSI_X9.63]

Elliptic Curve Key Agreement and Transport Protocols

[DIN66291-1]

DIN V66291-1: 2000

Chipkarten mit Digitaler Signatur-Anwendung/-Funktion nach SigG/SigV
Teil 1: Anwendungsschnittstelle

[DIN66291-4]

DIN V66291-4: 2002

Chipkarten mit Digitaler Signatur-Anwendung/-Funktion nach SigG/SigV
Teil 4: Grundlegende Sicherheitsdienste

[ECDIR]

Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures

[EN1867]
prEN 1867: 1995
Machine readable cards - Health care applications - Numbering system and registration procedure for issuer identifiers

[ESIGN-F]
CEN/ISSS WS/E_Sign Draft CWA Group F:
Secure Signature Creation Devices
(Protection Profile)
CWA N137, 1st March 2001

[ESIGN-K]
CEN/ISSS WS/E_Sign Draft CWA Group K:
Application Interface for SmartCards used as
Secure Signature Creation Devices
Version 1.07, 10th July 2003

[ISIS-MTT OP]
T7, TeleTrusT: ISIS-MTT Specification,
Optional Profile "SigG-Profile", Version 1.0.2,
July 2002, www.teletrust.de

[ISIS-MTT P1]
T7, TeleTrusT: ISIS-MTT Specification, Part 1
"Certificate and CRL Profiles", Version 1.0.2,
July 2002, www.teletrust.de

[ISO11770]
ISO/IEC 11770: 1996
Information technology - Security techniques -
Key management
Part 3: Mechanisms using asymmetric techniques

[ISO7816-1]
ISO/IEC 7816-1: 1996 (2nd edition)
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 1: Physical characteristics

[ISO7816-2]
ISO/IEC 7816-2: 1996 (2nd edition)
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 2: Dimensions and location of contacts

[ISO7816-3]
ISO/IEC 7816-3: CD 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 3: Electronic signals and transmission protocols

[ISO7816-4]
ISO/IEC 7816-4: FCD 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 4: Interindustry commands for interchange

[ISO7816-5]
ISO/IEC 7816-5: FCD 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 5: Registration of application providers

[ISO7816-6]
ISO/IEC 7816-6: FCD 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 6: Interindustry data elements

[ISO7816-8]
ISO/IEC 7816-8: FCD 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 8: Interindustry commands for a cryptographic toolbox

[ISO7816-9]
ISO/IEC 7816-9: FCD 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 9: Interindustry commands for card and file management

[ISO7816-15]
ISO/IEC 7816-15: FDIS 2003
Information technology - Identification cards -
Integrated circuit(s) cards with contacts -
Part 15: Cryptographic information application

[ISO8825]
ISO/IEC 8825-1: 1995
Information technology - ASN.1 encoding rules -
Specification of Basic Encoding Rules (BER),
Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

[ISO9564]
ISO 9564-1, Banking – Personal Identification
Number management and Security, Part 1:
PIN protection principles and techniques, 1999

[ISO9796-2]
ISO9796-2: 1997, Information technology –
Security techniques – Digital signature
schemes giving message recovery –
Part 2: Mechanisms using a hash function

[ISO10118]
ISO 10118-2, Information technology – Security
techniques – Hash functions, Part 2: Hash
functions using an n bit block cipher algorithm,
1994

[ISO10918]
ISO/IEC 10918-1: 1994
Information technology - digital compression
and coding of continuous-tone still images: Requirements and guidelines

[NIST-SHS]
 NIST: FIPS Publication 180-1:
 Secure Hash Standard (SHS-1), Mai 1995

[PKCS#1]
 PKCS #1 V1.5: Nov. 1993, V2.0: July, 1998
 V2.1: June, 2002: RSA Cryptography Standard

[RFC1510]
 RFC 1510: May 1999
 Public Key Cryptography for Initial Authentication in Kerberos

[RFC2246]
 RFC 2246: Jan. 1999
 The TLS Protocol, Version 1.0

[RFC2279]
 UTF-8, a transformation format of ISO 10646,
 January 1998

[RFC2459]
 Internet X.509 Public Key Infrastructure -
 Certificate and CRL Profile, January 1999

[RFC3039]
 Internet X.509 Public Key Infrastructure
 Qualified Certificates Profile, January 2001

[RFC3280]
 Internet X.509 Public Key Infrastructure –
 Certificate and Certificate Revocation List
 (CRL) Profile, April 2002

[RSA]
 R. Rivest, A. Shamir, L. Adleman:
 A method for obtaining digital signatures and
 public key cryptosystems, Communications of
 the ACM, Vol. 21 No. 2, 1978

[SECCOS]
 Schnittstellenspezifikation für die ZKA-Karte.
 Secure Chip Card Operating System
 (SECCOS), Version 5.0, 06.06.2001

[SigG01]
 Law on the Conditions for Electronic
 Signatures (Gesetz über Rahmenbedingungen
 für elektronische Signaturen und zur Änderung
 weiterer Vorschriften), Bundesgesetzblatt Nr.
 22, 2001, S.876

[SigV01]
 Ordinance on Electronic Signatures
 (Verordnung zur elektronischen Signatur –
 SigV), 2001

[SSL]
 Netscape:
 SSL3.0 Specification

4 Abbreviations and notations

4.1 Abbreviations

AC = Attribute Certificate
 AID = Application Identifier
 AM = Access Mode
 AOD = Authentication Object Directory
 ARR = Access Rule Reference
 ASN.1 = Abstract Syntax Notation One
 ASS = Additional Security Services
 AT = Authentication Template
 ATR = Answer-to-Reset
 AUT = Authentication
 BCD = Binary Coded Decimal
 BER = Basic Encoding Rules
 C = Certificate
 CA = Certification Authority
 CAR = Certification Authority Reference
 CBC = Cipher Block Chaining
 CC = Cryptographic Checksum
 CD = Certificate Directory
 CE = Certificate Extensions
 CG = Cryptogram
 CH = Cardholder
 CHA = Certificate Holder Authorization
 CHR = Certificate Holder Reference
 CIA = Cryptographic Inform. Application
 CIO = Cryptographic Inform. Objects
 CLA = Class byte of a command
 COS = Card Operating System
 CPI = Certificate Profile Identifier
 CRT = Control Reference Template
 CS = CertSign (= CertificateSigning)
 CSP = Certificate Service Provider
 CT = Confidentiality Template
 CV = Card Verifiable (Certificate)
 CWA = CEN Working Agreement
 D, DIR = Directory
 DCOD = Data Container Object Directory
 DE = Data Element
 DER = Distinguished Encoding Rules
 DES = Data Encryption Standard
 DO = Data Object
 DF = Dedicated File
 DI = Baud rate adjustment factor
 DSI = Digital Signature Input
 DST = Digital Signature Template
 E = Evaluation
 EAL = Evaluation Assurance Level
 EF = Elementary File
 EOF = End-of-File
 ES = Electronic Signature
 FCI = File Control Information
 FI = Clock rate conversion factor
 FID = File Identifier
 FM = File Management
 GK = Group Key
 HB = Historical Bytes
 HF2 = Hash Function ISO 10118-2
 HI = Health Institution

HP = Health Professional
 HPA = Health Professional Application
 HPC = Health Professional Card
 HPD = Health Professional Data
 ICC = Integrated Circuit(s) Card
 ICCSN = ICC Serial Number
 ID = Identifier
 IFD = Interface Device
 IFSC = Information Field Size Card
 IFSD = Information Field Size Device
 IIN = Issuer Identification Number
 IK = Individual Key
 IV = Initial Value
 KD = Key Derivation data
 KE = Key Encipherment
 KEI = Key Encipherment Input
 MF = Master File
 MII = Major Industry Identifier
 MSE = MANAGE SECURITY ENVIRONMENT
 OID = Object Identifier
 P = Patient
 PA = Personalization Authority
 PCA = Policy Certification Authority
 PDC = Patient Data Card
 PHAR = Pharmacist
 PHYS = Physician
 PK,PuK = Public Key
 PI = Padding Indicator
 PIN = Personal Identification Number
 PIX = Proprietary Appl. Prov. Extension
 PPS = Protocol Parameter Selection
 PrK = Private Key
 PRND = Padding Random Number
 PSO = PERFORM SECURITY OPERATION
 R = Role ID
 RegTP = Regulation Authority for Telecom +
 Post (Regulierungsbehoerde für TP)
 RC = Retry Counter
 RCSP = Root Certificate Service Provider
 RD = Reference Data
 RFC = Request for Comment
 RID = Registered Application Provider Id.
 RND = Random Number
 RSA = Algorithm of Rivest,Shamir,Adleman
 S = Server
 SC = Security Condition
 SFID = Short EF Identifier
 SIG = Signature
 SK = Secret Key
 SM = Secure Messaging
 SMA = Security Module Application
 SMC = Security Module Card
 SMK = SM key
 SSC = Send Sequence Counter
 SSCD = Secure Signature Creation Device
 SSL = Security Sockets Layer
 SN = Serial Number
 TC = Trusted Channel
 TLS = Transport Layer Security
 UID = User Identification
 UQ = Usage Qualifier
 VD = Verification Data
 HPC Pharmacist & Physician V2.0

ZGW = CA for health care
 (Zertifizierungsstelle Gesundheitswesen)
 ZKA = Zentraler Kredit Ausschuss

4.2 Notations

For keys and certificates the following simplified Backus-Naur notation applies:

```

<object descriptor> ::= <key descriptor> |
<certificate descriptor>

<key descriptor> ::=
<key>.<keyholder>.<usage> | <SMkey>

<key> ::= <private key> | <public key>
| <group key> | <individual key>

<private key> ::= PrK (asym.)
<public key> ::= PuK (asym.)
<group key> ::= GK (sym.)
<individual key> ::= IK (sym.)

<keyholder> ::= <health professional>
| <patient> | <certificate service
provider> | <health professional
card> | <patient data card> |
<security module card> | <server>

<health professional> ::= HP
<patient> ::= P
<certificate service provider> ::= CSP
| RCSP
<health professional card> ::= HPC
<patient data card> ::= PDC
<security module card> ::= SMC
<server> ::= S

<usage> ::= <electronic signature> |
<key encipherment> | <authentication>

<electronic signature> ::= ES
<key encipherment> ::= KE
<authentication> ::= AUT

<certificate descriptor> ::=
<certificate>.<certificateholder>.<usage>

<certificate> ::= C

<certificateholder> ::=
<health professional> | <certificate
service provider> | <health
professional card> | <security
module card> | <server>

<SMkey> ::= SMK.ENC | SMK.MAC
  
```

For subsequent data items the following notation is used:

|| = Concatenation of data

5 Technical Characteristics, Answer-to-Reset and Transmission Protocols

5.1 Technical Characteristics

HPCs and SMCs are contact based smart-cards capable to process PK algorithms. The physical characteristics shall comply with ISO/IEC 7816-1 and related standards.

The dimensions and location of contacts shall be in accordance to ISO/IEC 7816-2.

The HPC shall support at least class B (3V).

An HPC is a normal size card (ID-001 card), the SMC (usually) a plug-in card (ID-000).

The description of the cover is out of scope of this document.

The size of the EEPROM shall allow the installation of the applications and their data described in this specification.

5.2 Answer-to-Reset

The coding for the ATR is shown in annex A. The characteristics are:

- Direct convention
- Since the cards shall support higher transmission rates the relevant values for FI and DI have to be indicated in TA1 (e.g. '13', see ISO/IEC 7816-3, table 7 and 8).
- The operation mode shall be the 'negotiable mode' (i.e. TA2 is absent). Protocol Parameter Selection (PPS) according to ISO/IEC 7816-3 shall be supported for negotiation of the FI/DI-values for higher speed (see annex A).
- Information Field Size Card (IFSC) has to be indicated (value at least '80')
- Block Waiting Time (BWI) and Character Waiting Time (CWI) shall be indicated
- The clock stop mode (XI) and the voltage class (UI) shall be indicated
- The Historical Bytes shall be in accordance with ISO/IEC 7816-4 (see annex A).

5.3 Transmission Protocols

The transmission protocol to be supported is T=1 (for further details see annex A).

6 The Health Professional Card

6.1 General Structure

The HPC contains

- some EFs at MF level for some general data objects and the CV certificate
- the HP application (HPA) providing the following services:
 - electronic identification of the health professional
 - electronic signature creation
 - client/server authentication
 - document decipherment
 - card-to-card authentication (HPC/PDC and HPC/SMC)
- the Cryptographic Information Application (CIA) providing information for the service system (e.g. doctor practice system) to support the communication between the service system and an HPC.

The general structure is shown in figure 1.

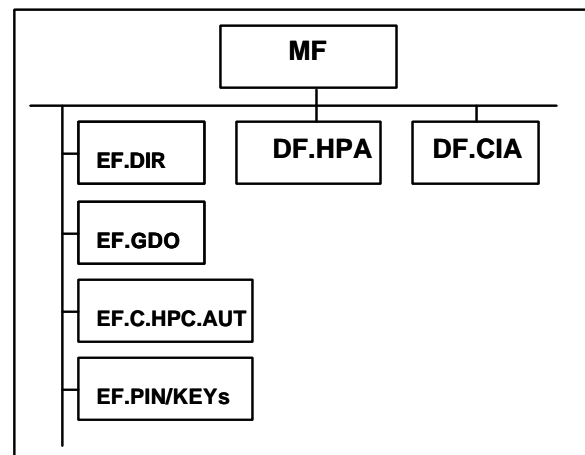


Figure 1 – General file structure of an HPC

NOTE – At MF-level, an EF.ARR may be present.

6.2 Elementary Files at MF-Level

6.2.1 EF.DIR

EF.DIR contains the application templates for DF.HPA and DF.CIA according to ISO/IEC 7816-4.

Tag	L	Application Template
'61'	'08'	'4F 06 D276 00004002'
'61'	'0D'	'4F 0B E828BD080F D276 00004002'

Table 1 – Application Templates in EF.DIR

6.2.2 EF.GDO

The EF.GDO contains DO ICC Serial Number (ICCSN, Tag '5A', see figure 2).

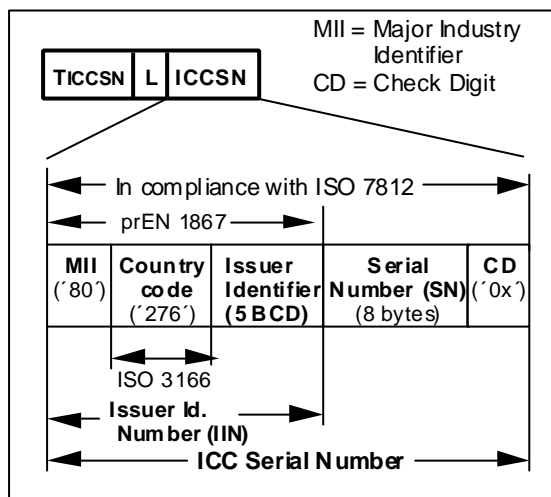


Figure 2 - ICC Serial No. for health cards

The IIN to be used (assigned by CCG - Centrale für Coorgansiation GmbH as registration authority, see www.ccg.de) as related national registration authority e.g. for the doctors or pharmacists chamber) is shown in table 2.

MII for Health care	Country Code Germany	Issuer Identifier
'80'	'276'	'000xx' (to be inserted)

Table 2 - Issuer Identification Number

6.2.3 EF.C.HPC.AUT

EF.C.HPC.AUT contains the card verifiable certificate. This certificate is used for card-to-card authentications, i.e. HPC/SMC or HPC/PDC. Structure and content are outlined in annex F.

6.2.4 EF.PIN/KEYs

Global PINs and keys may be stored in files attached to the MF (see Figure 4). Their names and usage is described in 7.2.1 and 7.2.2.

6.3 HPC opening

6.3.1 Command sequence after ATR

The command sequence to be performed prior to the HPA selection depends on the usage environment and its knowledge about the respective card. In principle the following environments may be distinguished:

- usual professional environment
- third party environment.

In an environment situation where everything is known, the HPA selection may immediately follow after reset. If several known HPCs are used at the same station, it may be necessary to read the ICCSN and to identify with it e.g. a card profile possibly consisting of the set of Cryptographic Information Objects. In third party environments, where a respective HPC is not known, it may be necessary first to read EF.DIR, EF.GDO and after that the CIOs. Subsequent the reading of EF.DIR and EF.GDO is described. How to read the CIOs is described in clause 8.

6.3.2 Reading of EF.DIR and EF.GDO

For reading EF.DIR and EF.GDO the ISO/IEC 7816-4 commands SELECT and READ BINARY are used.

CLA	As defined in ISO/IEC 7816-4
INS	'A4' = SELECT
P1	'02' = EF file selection
P2	'0C' = No FCI to return
Lc	'02' = Length of subsequent data field
Data field	- '2F00' = FID of EF.DIR - '2F02' = FID of EF.GDO
Le	Absent

Table 3 - SELECT command for EF selection

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 4 - SELECT response

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1,P2	'0000'
Lc	Absent
Data field	Absent
Le	'00' = Read until end-of-file

Table 5 - READ BINARY command without SFID

Data field	Data
SW1-SW2	'9000' or specific status bytes

Table 6 - READ BINARY response

The HPC shall support also the READ BINARY command with SFID, see table B.3. In this case, no SELECT command is necessary.

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1	'9E' = b8: 1 and SFID of EF.DIR '82' = b8: 1 and SFID of EF.GDO
P2	'00' = Offset
Lc	Absent
Data field	Absent
Le	'00' = Read until end-of-file

Table 7 - READ BINARY command with SFID

Data field	Data
SW1-SW2	'9000' or specific status bytes

Table 8 - READ BINARY response

7 The Health Professional Application HPA

7.1 Certificate Service Provider and Certificates

The HPC security mechanisms require different types of end user certificates:

- ES certificate(s) (X.509v3 electronic signature certificate(s), i.e. public key certificate and attribute certificate(s))
- AUT certificate (X.509v3 authentication certificate)
- KE certificate (X.509v3 key encipherment certificate).

End user certificates are produced by a Certificate Service Provider, addressed in this context as "CSP Health Care". The root CSP produces a certificate for the CSP Health Care, whereby the certified key pair is only used for the production of end user certificates for electronic signatures. Therefore the CSP Health Care produces a self-certificate, and the respective private key is used for the production of AUT and KE certificates.

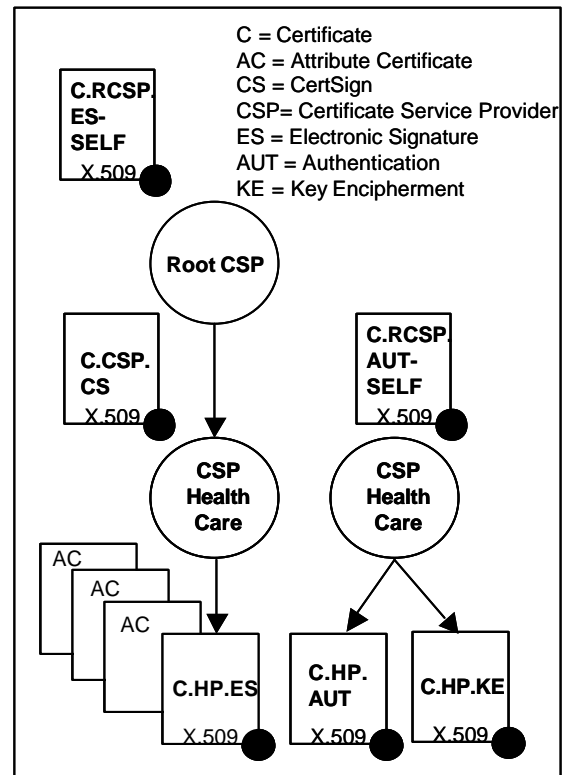


Figure 3 - CSPs and certificates (example)

NOTE – The role of a CSP Health Care can be performed by several institutions.

An AUT certificate (Card Verifiable authentication certificate) for trusted channel support and proving access rights to patient data cards will also be present.

7.2 File structure and file content

The file organization in an HPC shall be according to ISO/IEC 7816-4. The file structure of DF.HPA is shown in fig. 4.

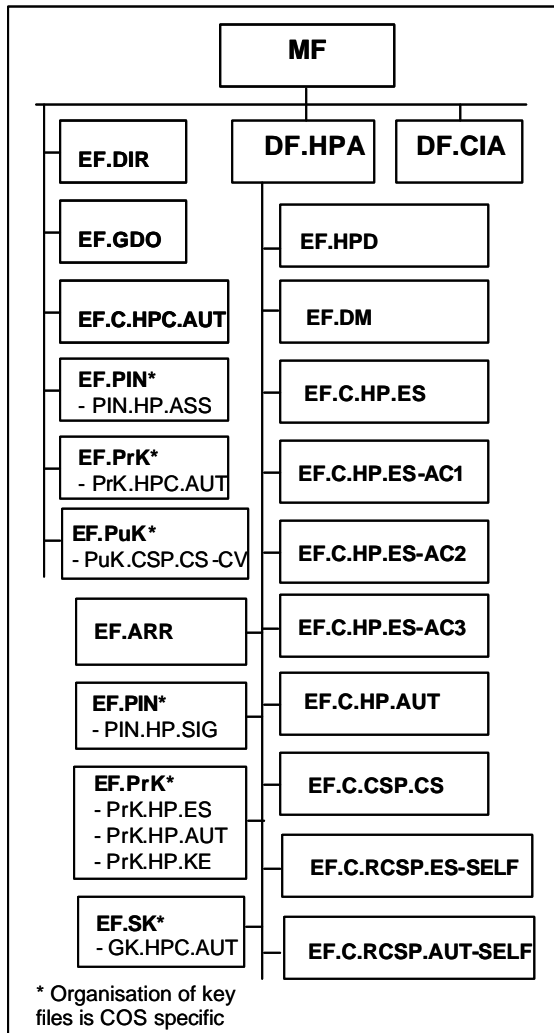


Figure 4 - File structure of DF.HPA

NOTE – The certificate C.HP.KE is not stored on the HPC, but can be retrieved from a certificate server prior to enciphering a document.

The file identifiers (FIDs) and the access conditions to the elementary files are outlined in annex B.

EF.PIN, EF.PrK, EF.PuK and EF.SK are examples of internal files. The real key organization in the HPC is COS specific.

7.2.1 EF.PIN

The HPA provides 2 different PINs:

- PIN.HP.SIG is a DF-specific PIN and only used for the protection of the SigG/SigV-related private electronic signature key of the health professional PrK.HP.ES. The PIN consists of at least 6 digits and is changeable. The retry counter shall have the initial value 3.
- PIN.HP.ASS is a global PIN and used for the protection of the additional security services. The PIN consists of at least 5 digits and is changeable. The retry counter shall have the initial value 3.

Both PINs have an own resetting code of 8 digits (the value may be the same). The usage of the resetting code is limited by a usage counter with an initial value of 5. The usage counter is decremented independent whether the resetting code was correct or not. The successful presentation resets the retry counter and sets a new PIN, if the respective option was chosen.

The PIN references as used in the VERIFY, CHANGE RD and RESET RC command are shown in the following table.

PIN Name	PIN Ref.	Resetting Code (Usage Counter = 5)
PIN.HP.SIG	'81'	8 digits
PIN.HP.ASS	'01'	8 digits

Table 9 – PIN references and resetting code

The initialization of PIN.HP.SIG e.g. by using a transport PIN is COS dependent and has to comply to RegTP regulations. For the initialization of PIN.HP.ASS the same principle may be applied.

7.2.2 EF.PrK

The health professional uses the following private keys in combination with the related X.509 certificates:

- PrK.HP.ES for signature creation
- PrK.HP.AUT for client/server authentication
- PrK.HP.KE for document cipher key decipherment.

The key references and the protection of the private keys are shown in the following table.

Key Name	KeyRef	Protection
PrK.HP.ES	'91'	PIN.HP.SIG
PrK.HP.AUT	'92'	PIN.HP.ASS
PrK.HP.KE	'93'	PIN.HP.ASS

Table 10 – Key references and protection of HP-related private keys

For card-to-card authentication procedures based on card verifiable certificates, a further private key is needed (it is a global key possibly stored in a key file located under the MF):

- PrK.HPC.AUT for HPC/PDC and HPC/SMC authentication.

The authentication procedures are different (with/without session key establishment) as well as the usage condition. The key reference and the protection of the related private key is shown in the following table.

Key Name	KeyRef	AlgRef	SE
PrK.HPC.AUT (used for pro-ving access right to a PDC and verification of HPC authenticity)	'11'	'1E', see table E.2	#2
PrK.HPC.AUT (Used for TC establishment)	'11'	'1F', see table E.2	#3

Table 11 – Key reference and protection of PrK.HPC.AUT

As public key algorithm, RSA is applied. In addition the support of the hash algorithm SHA-1 is required.

With respect to RSA key length for legally binding electronic signatures the algorithm catalog [ALGCAT] has to be observed.

Algorithm	Key Length	Validity
RSA	1024	2006
RSA	1536	2007

Table 12 – Key length of PrK.HP.ES

The allowed padding schemes are presented in annex E.

For the RSA keys PrK.HP.AUT, PrK.HP.KE and PrK.HPC.AUT 1024 bits key length will be used.

The HPC shall ensure, that the respective keys are only usable for the service to which the key is dedicated (internal binding of the key to its HPC Pharmacist & Physician V2.0

purpose; the way to ensure this is COS dependent).

7.2.3 EF.PuK

For CV certificate verification, the PuK of the CSP issuing the CV certificate must be present in the HPC, so that a CV certificate from another card can be verified.

EF.PuK contains therefore PuK.CSP.CS-CV (RSA key). The PuK reference is contained in the CV certificate (see annex F, field CAR) and has to be used when setting the PuK for certificate verification.

7.2.4 EF.SK

As symmetrical algorithm, the HPC has to support DES-3. The key reference of the group key GK possibly used for HPC/PDC authentication is referenced as shown in the subsequent table.

GK.HPC.AUT-HP consists of 2 parts:

- GK.HPC.ENC-HP (16 byte)
- GK.HPC.MAC-HP (16 byte).

Key Name	KeyRef	Present in	Protection
GK.HPC.AUT-PHYS1	'9A'	HPC-PHYS	PIN.HP.ASS
GK.HPC.AUT-PHYS2	'9B'	HPC-PHYS	PIN.HP.ASS
GK.HPC.AUT-PHAR1	'9D'	HPC-PHAR	PIN.HP.ASS
GK.HPC.AUT-PHAR2	'9E'	HPC-PHAR	PIN.HP.ASS

Table 13 – Key references of GK.HPC.AUT

NOTE – A health professional specific key reference allows to take the same key reference for the Group Key GK in a HPC and the associated Individual Key IK in the PDC.

7.2.5 EF.ARR

For storing access rules, EF.ARR (linear structure with records of variable size) is used.

7.2.6 EF.HPD

The EF.HPD contains DOs providing a description of the health professional (see table 14). All DOs are encapsulated in the Cardholder Related Data Template (Tag '65', see ISO/IEC 7816-6). If the authenticity of the presented HPC shall be verified (e.g. prior to a medica-

tion delivery to a health professional), then an authentication procedure between HPC and SMC shall be performed, see clause 7.12. The EF.HPD cannot be modified after the card has been issued.

Tag	L	Value	Status
'65'	var.	Cardholder related data	mandatory
'5B'	var.	Name (ASCII characters, national version)	mandatory
'53'	var.	Discretionary data, used for profession indication: e.g. PHARMACIST / PHYSICIAN	mandatory
'42'	var.	Issuer authority, e.g. the related professional chamber (ASCII characters, national version)	mandatory
'5F30'	var.	Service code, in this context used for Registration No. (ASCII characters, national version)	mandatory
'5F2B'	8	Date of birth (YYYYMMDD)	mandatory
'5F40'	var.	Cardholder portrait image (format e.g. as defined in ISO/IEC 10918-1)	optional
'5F3D'	var.	Digital signature (DSI-Input: all bytes preceding '5F3D')	optional

Table 14 - DOs for describing the HP

The digital signature – if present – shall be produced by the CA issuing the ES certificate of the health professional.

7.2.7 EF.DM

The EF.DM contains the display message which indicates to the health professional that a trusted channel has been successfully established. It consists of 8 bytes (ASCII characters). The display message can be modified by the HP.

7.2.8 EF.C.HP.ES

The EF.C.HP.ES contains the X.509v3 public key certificate of the health professional for the electronic signature service according to SigG/SigV.

7.2.9 EF.C.HP.ES-AC1, -AC2 and -AC3

The EF.C.HP.ES-AC1, -AC2 and -AC3 may contain an X.509v3 attribute certificate from a profession chamber (e.g. physicians chamber, pharmacists chamber), professional organizations (e.g. doctors association) or governmental institutions. CIOs describe the existence of attribute certificates.

7.2.10 EF.C.HP.AUT

EF.C.HP.AUT contains the X.509v3 authentication certificate of the health professional.

7.2.11 EF.C.CSP.CS

EF.C.CSP.CS contains the X.509v3 certificate of the health care CSP issued by the root CSP.

7.2.12 EF.C.RCSP.ES-SELF

EF.C.RCSP.CS contains the X.509v3 self certificate of the root CSP for electronic signatures.

7.2.13 EF.C.RCSP.AUT-SELF

EF.C.RSP.CS contains the X.509v3 self certificate of the root CSP for authentication (the health care CSP).

7.3 Security Environments

The HPA shall support 2 security environments:

- SE #2 contains all security services (for details see table B.10). The access conditions require the presentation of PIN.HP.ASS prior to the usage of PrK.HPC.AUT, see 7.2.2. The HPC/PDC interactions are therefore performed in SE #2.
- SE #3 contains all security services (for details see table B.10). The access conditions don't require the presentation of PIN.HP.ASS prior to the usage of PrK.HPC.AUT, see 7.2.2. The HPC/SMC interactions are therefore performed in SE #3.

SE #1 (default SE) is not used.

7.4 Application selection

The ISO/IEC 7816-4 command for 'Direct Application Selection' is shown in the subsequent two tables.

CLA	As defined in ISO/IEC 7816-4
INS	'A4' = SELECT
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'06' = Length of subsequent data field
Data field	'D276 00004002' = AID of DF.HPA
Le	Absent

Table 15 - SELECT command for DF.HPA selection with AID

NOTE – The AID of the version 1.0 of the HPA was 'D27600004001'. The RID 'D276000040' is assigned to the German Health Care.

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 16 - SELECT response

7.5 SE Selection

Prior to any command processing after application selection, the relevant SE shall be set.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'F3' = RESTORE
P2	'0x' = SE # : If HPC/SMC interaction is performed, SE #3 shall be set, in other cases SE #2
Lc	Absent
Data field	Absent
Le	Absent

Table 17 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 18 - MSE response

7.6 Reading and Updating of HP Data and Certificates

7.6.1 Reading of the Health Professional Data

Reading of the HP Data shall be always possible. However, if the HPC is presented in a pharmacy, then a CV-based authentication procedure between HPC and SMC should be

performed (see clause 7.12) before reading the data possibly in SM mode.

For reading the Health Professional Data the ISO/IEC 7816-4 command READ BINARY is used.

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1	'82' = b8: 1 and SFID of EF.HPD
P2	'00' = Offset
Lc	Absent
Data field	Absent
Le	'00' = Read until end-of-file* or 'xx' = length of expected data

Table 19 - READ BINARY command for reading the HPD data

* If the length of the data in the file is greater than 256, then it may be necessary, that the command has to be repeated with a respective offset. Since the first READ BINARY command with the SFID selects the respective EF, subsequent READ BINARY commands use P1,P2 for offset. Cards supporting extended Le fields (i.e. Le = '000000') shall indicate this characteristic in the card capabilities, see Annex A.

Data field	HPD data
SW1-SW2	'9000' or specific status bytes

Table 20 - READ BINARY response

7.6.2 Reading and Updating of HP Certificates

For reading a health professional X.509 certificate, the READ BINARY command as described in Tables 19 and 20 with the respective SFID shall be used.

For updating attribute certificates (see access conditions in Annex B), the ISO/IEC 7816-4 command UPDATE BINARY shall be used.

CLA	As defined in ISO/IEC 7816-4
INS	'D6' = UPDATE BINARY
P1	'82' = b8: 1 and SFID of EF.C.x
P2	'00' = Offset
Lc	'xx' = Length of subsequent data field
Data field	Data
Le	Absent

Table 21 - UPDATE BINARY command for updating

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 22 - UPDATE BINARY response

If the certificate is longer than 255 byte, then the UPDATE BINARY command shall be repeated specifying the respective offset in P1-P2.

7.7 PIN Management

7.7.1 PIN Encoding

The HPA supports PIN.HP.SIG and PIN.HP.ASS. A PIN is always encoded as "Format 2 PIN Block" according to ISO 9564-1:

6-digit PIN (PIN.HP.SIG):

C	L	P	P	P	P	P	P	F	F	F	F	F	F	F	F
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

5-digit PIN (PIN.HP.ASS, if 5 digits are used):

C	L	P	P	P	P	P	F	F	F	F	F	F	F	F	F
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

C = Control field, value 2
L = Length of PIN in BCD
P = PIN digit in BCD
F = Filler with value 'F'

The coding format has therefore always 8 byte.

7.7.2 PIN Verification

For PIN verification the ISO/IEC 7816-4 command VERIFY is used.

CLA	As defined in ISO/IEC 7816-4
INS	'20' = VERIFY
P1	'00'
P2	'81' = PIN.HP.SIG reference '01' = PIN.HP.ASS reference
Lc	'08' = Length of subsequent data field
Data field	PIN
Le	Absent

Table 23 - VERIFY command for health professional authentication

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 24 - VERIFY response

After successful presentation of a PIN, the associated retry counter is automatically reset to its initial value (3).

The following Status Bytes are of special importance and shall be supported:

- '63Cx': Warning - verification failed (x indicates the number of further allowed retries)
- '6983': Checking error: authentication method blocked (these status bytes shall be delivered, if the VERIFY command is sent and the RC is zero).

NOTE – The return code '6300' is not allowed.

7.7.3 PIN Change

For PIN change the ISO/IEC 7816-4 command CHANGE RD may be used at any time convenient for the HP.

CLA	As defined in ISO/IEC 7816-4
INS	'24' = CHANGE REFERENCE DATA
P1	'00' = Exchange reference data
P2	'81' = PIN.HP.SIG reference '01' = PIN.HP.ASS reference
Lc	'10' = Length of subsequent data field
Data field	PIN_old PIN_new
Le	Absent

Table 25 - CHANGE RD command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 26 - CHANGE RD response

7.7.4 Reset of Retry Counter and Setting a New PIN

For resetting the retry counter to its initial value and – if used in this way – setting a new PIN, the RESET RETRY COUNTER command as specified in ISO/IEC 7816-4 is used.

The command does not set or influence the security status, i.e. a VERIFY command is still needed e.g. prior to signature creation.

CLA	As defined in ISO/IEC 7816-4
INS	'2C' = RESET RETRY COUNTER
P1	'00' or '01'
P2	'81' = PIN.HP.SIG reference '01' = PIN.HP.ASS reference
Lc	'10' or '08' = Length of subsequent data field
Data field	If P1 = '00': Resetting code (8 byte) followed by new PIN (8 byte) If P1 = '01': Resetting code (8 byte)
Le	Absent

Table 27 – RESET RC command for RC reset and setting a new PIN

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 28 - RESET RC response

7.8 Creation of an Electronic Signature

7.8.1 General aspects

An electronic signature process consists of several steps as shown in figure 5.

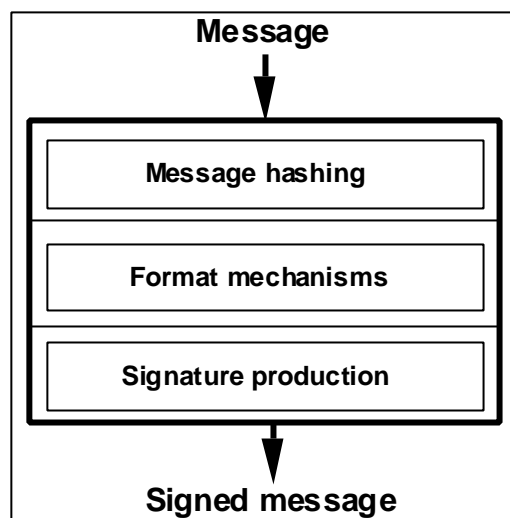


Figure 5 - Signature process according to ISO/IEC 9796-2

Hashing is done outside the HPC except the last text block, i.e. the intermediate hash value concatenated with the last text block is delivered to the HPC.

Since the hash value by applying RSA is shorter than the Digital Signature Input (DSI) for signature production, the hash value has to be padded, i.e. to be formatted. The padding is done inside the HPC and the padding scheme to be applied shall be set with the MSE command.

7.8.2 Setting PrK.HP.ES

Before the first usage of the electronic signature by the health professional, the key and algorithm references shall be set by using the ISO/IEC 7816-4 command MSE. The key remains set until power off or change of the DF.HP, i.e. the MSE command with SET DST is only sent once a session.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for computation
P2	'B6' = DST
Lc	'xx' = Length of subsequent data field
Data field	'84 01 91' '80 01 xx' = DO for KeyRef of PrK.HP.ES DO AlgRef, value see table E.1
Le	Absent

Table 29 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 30 - MSE response

7.8.3 Delivery of the Hash Value

For delivering the hash value, the ISO/IEC 7816-8 command PSO: HASH is used. For the required security level (see [ESIGN-F]), the intermediate hash value followed by the final text block is sent to the card, which computes the final hash value.

CLA	As defined in ISO/IEC 7816-4
INS	'2A' = PERFORM SECURITY OPERATION: HASH
P1	'90' = Compute hash value
P2	'A0' = Data field contains DOs relevant for hashing
Lc	'xx' = Length of subsequent data field
Data field	'90' - L – Intermediate hash value '80' - L – Final block
Le	Absent

Table 31 - PSO: HASH command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 32 - PSO: HASH response

7.8.4 Initiation of the Electronic Signature Creation

After hashing is done, the ISO/IEC 7816-8 command PSO: COMPUTE DIGITAL SIGNATURE has to be sent.

CLA	As defined in ISO/IEC 7816-4
INS	'2A' = PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE
P1	'9E' = Return digital signature
P2	'9A' = Data field – if present – contains data to be signed or integrated in the DSI
Lc	Absent
Data field	Absent (i.e. DSI already present in the card)
Le	'00' or 'xx' = length of expected digital signature

Table 33 - PSO: COMPUTE DS command

Data field	Electronic signature
SW1-SW2	'9000' or specific status bytes

Table 34 - PSO: COMPUTE DS response

7.9 Client / Server Authentication

For proving access rights to components such as servers, a PK based authentication procedure has to be performed. The key pair used is that one of the cardholder (PrK.HP.AUT, PuK.HP.AUT) and the public key together with the distinguished name of the cardholder is certified by an X.509 certificate. Relevant authentication procedures are e.g.

- the PK Kerberos protocol (for logon authentication)
- the TLS protocol (for authentication on the client side; covers the SSL protocol)
- the WTLS protocol.

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. In case of RSA, the authentication input T is formatted according to [PKCS #1], Version 2.0:

$$'01' || PS || '00' || T$$

where PS is an octet string consisting of octets with value 'FF'. The formatted octet string must consist of k-1 octets where k is the length in octets of the modulus of the private key for authentication.

The other parts at the client side have to be performed by the software in the cardholders system.

Before the INTERNAL AUTHENTICATE command can be executed, the related PrK has to be selected.

Furthermore, the related algorithm reference denoting the type of authentication protocol may be set additionally.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for internal authentication
P2	'A4' = AT
Lc	'06' = Length of subsequent data field
Data field	'84 01 92' '80 01 xx' = DO for KeyRef of PrK.HP.AUT DO AlgRef, see table E.2
Le	Absent

Table 35 – MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 36 - MSE response

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'xx' = Length of subsequent data field
Data field	<ul style="list-style-type: none"> - DigestInfo, used for KERBEROS - H_MD5 H_SHA1, used for SSL/TLS - H_SHA1, used for WTLS and NETSCAPE <p>The formatting of the authentication input is according to PKCS#1 (the length of the data field shall not exceed 40% of the length of the modulus)</p>
Le	'00' or 'xx' = length of expected digital signature

Table 37 - INT. AUTHENTICATE command

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

Table 38 - INT. AUTHENTICATE response

7.10 Document cipher key decipherment

For confidential document exchange, the following scheme is applied:

- key transport is organised by enciphering the document cipher key with the receiver's PuK.HP.KE
- document enciphering with a symmetrical algorithm.

If an enciphered document is produced, an HPC is not involved: the software in the PC of the document sender computes the document ciphering key, enciphers the document and finally enciphers the document cipher key by applying the receiver's public key taken from the receiver's x.509 KE certificate retrieved e.g. from a certificate server.

Before on the receiver site key decipherment can be performed, the private key PrK.HP.KE has to be selected with the ISO/IEC 7816-4 command MSE.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for decipherment
P2	'B8' = CT
Lc	'03' = Length of subsequent data field
Data field	'84 01 93' = DO for KeyRef of PrK.HP.KE
Le	Absent

Table 39 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 40 - MSE response

After the key is set, the decipher operation can be executed with the ISO/IEC 7816-8 command PSO: DECIPHER.

CLA	As defined in ISO/IEC 7816-4
INS	'2A' = PERFORM SECURITY OPERATION: DECIPHER
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 followed by cryptogram (format see tab. E.3 and E.4)
Le	'00' or 'xx' = Length of document cipher key

Table 41 – PSO: DECIPHER command

Data field	Document cipher key
SW1-SW2	'9000' or specific status bytes

Table 42 – PSO: DECIPHER response

7.11 HPC / PDC Authentication

For the HPC/PDC interaction two actions are required:

- the PDC has to prove its authenticity
- the health professional has to prove his access rights.

When proving access rights, an authentication procedure has to be performed, so that in the PDC the related security status can be set, i.e.

- in symmetrical case: group key x has been successfully presented
- in asymmetrical case: certificate holder authorization y (see table F.7) has been successfully presented.

In the authentication procedure, no SM keys are established or not used afterwards, since in the subsequent communication with the PDC (reading/writing data) an HPC is not involved.

In the following, the command sequences and the keys needed for these services are described with respect to PDCs supporting symmetric algorithms and/or asymmetric algorithms. The HPC has to ensure that the HPC/PDC-authentication procedure can only be performed after HP authentication (PIN.HP.ASS presentation).

7.11.1 Mutual Authentication with a symmetric Algorithm

The mutual authentication scheme (identical to [E-Sign-K]) is shown in figure 6 and should be performed in this sequence. The SM key derivation data KD.HPC and KD.PDC (each 32 byte) may be used for SM key computation, however, the SM keys are subsequently not used in this case.

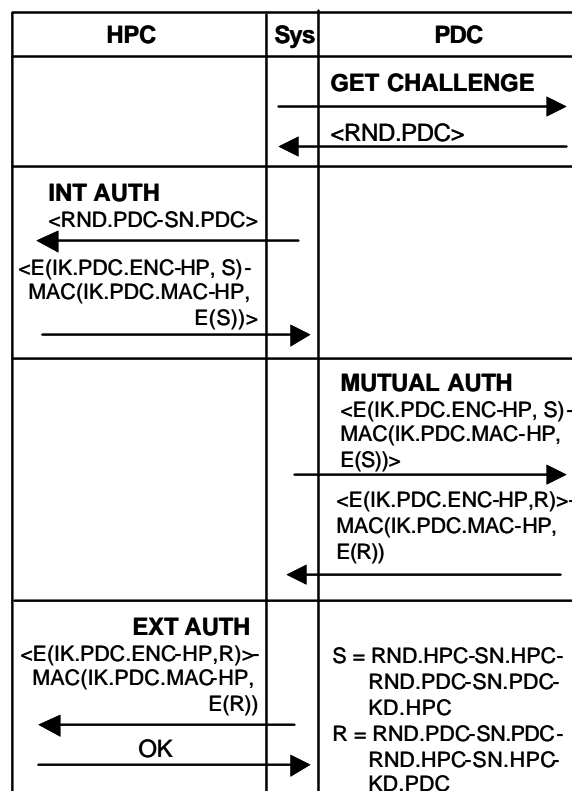


Figure 6 - Mutual authentication HPC / PDC

The individual key IK to be used in the PDC for this authentication procedure is HP-dependent, i.e. there is at least

- IK.PDC.AUT-PHYS and
- IK.PDC.AUT-PHAR

present in a PDC.

The derivation of the individual key of the PDC (IK.PDC.AUT-HP, consisting of the 2 parts IK.PDC.ENC-HP and IK.PDC.MAC-HP) to be computed in the HPC with the group key GK is performed as follows:

$$IK.PDC.ENC-HP = d * GK.HPC.ENC-HP(HF2(I, SN))$$

$$IK.PDC.MAC-HP = d * GK.HPC.MAC-HP(HF2(I, SN))$$

where

- IK is the individual DES-3 key (16 byte)
- d*GK is the DES-3 decipherment operation (ECB mode) performed on each hash-value block of 8 byte, i.e. d*GK(H₁) | d*GK(H₂)
- HF2 (I, SN) is the HF2 hash function performed on the 8 byte serial number SN padded with zeros to the length of 16 byte and the initial value I (16 byte) = '5252525252525252 2525252525252525'

A parity adjustment function may be performed on IK, if required by the COS.

NOTE – The key derivation mechanisms is identical to that one valid for cards with the ZKA card operating system SECCOS.

For encryption and decryption of the challenge, DES-3 (CBC mode) is applied as figure 7 shows. The data to be processed, is a block of 8 byte.

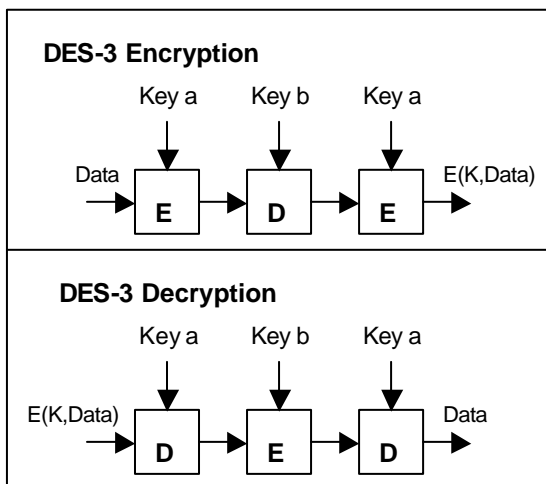


Figure 7 - Encryption/Decryption with DES-3

The first command to be send to the HPC – after RND.PDC has been requested from the PDC – is the INTERNAL AUTHENTICATE command.

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00'
P2	'xx' = GK.HPC.AUT-x (see table 13)
Lc	'10' = Length of subsequent data field
Data field	RND.PDC SN.PDC
Le	'00'

Table 43- INTERNAL AUTHENTICATE command for proving access rights to a PDC

Data field	E(IK.PDC.ENC-HP, S) MAC (IK.PDC.MAC-HP, E(IK,S)) with S = RND.HPC SN.HPC RND.PDC SN.PDC KD.HPC
SW1-SW2	'9000' or specific status bytes

Table 44 - INTERNAL AUTHENTICATE response

NOTE – The token S has to be computed by the HPC and includes a RND.HPC which remains valid for the subsequent EXTERNAL AUTHENTICATE command. The data item for deriving IK.PDC.AUT-HP is the SN.PDC.

After the MUTUAL AUTHENTICATE command has been sent to the PDC, the response data of this command have to be delivered to the HPC in the EXTERNAL AUTHENTICATE command.

CLA	As defined in ISO/IEC 7816-4
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00'
P2	'xx' = GK.HPC.AUT-x (see table 13)
Lc	'10' = Length of subsequent data field
Data field	E (IK.PDC.ENC-HP, R) MAC (IK.PDC.MAC-HP, E(IK,R)) with R = RND.PDC SN.PDC RND.HPC SN.HPC KD.PDC
Le	Absent

Table 45 - EXT. AUTHENTICATE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 46 - EXT. AUTHENTICATE response

NOTE – Token S and R contain the same RNDs but in a different order.

7.11.2 Mutual Authentication with an asymmetric Algorithm

This service is only defined in the Security Environment #2, which has to be set as current SE using the MSE command. If SE #2 is the current one, then MSE command may be skipped.

Before performing the authentication procedure, the certificate C.HPC.AUT has to be presented to the PDC. It is assumed that the

PuK.CSP.CS-CV to be applied for verifying the HPC CV certificate is present in the PDC (if not, a cross CV certificate has to be presented to import the respective PuK for HP CV certificate verification, see 7.11.3).

For reading the C.HPC.AUT certificate out of the HPC the READ BINARY command is used.

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1	'81' = b8:1 and SFID of EF.C.HPC.AUT
P2	'00' = Offset
Lc	Absent
Data field	Absent
Le	'00' = Read until end-of-file

Table 47 - READ BINARY command for reading the CV certificate

Data field	CV certificate
SW1-SW2	'9000' or specific status bytes

Table 48 - READ BINARY response

Reading the CV certificate of the health professional may be performed only once and then stored in the health professional's PC environment for saving time.

The real procedure starts with verifying the PDC CV-certificate. The public key of the CSP for verification is expected to be in the HPC (if not then this key has to be delivered in a CV certificate which can be verified with the PuK.CSP.CS-CV present in the HPC). For CV-certificate verification the following commands have to be performed:

- MANAGE SECURITY ENVIRONMENT for setting the public key of the CSP (i.e. PuK.CSP.CS-CV)
- VERIFY CERTIFICATE for checking the CV-certificate of the PDC.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'81' = SET for verification
P2	'B6' = DST
Lc	'0A' = Length of subsequent data field
Data field	'83 08 ...' = DO for KeyRef of PuK.CSP.CS-CV (value taken from the CAR field in the CV certificate)
Le	Absent

Table 49 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 50 - MSE response

After the PuK.CSP.CS-CV is set, then the VERIFY CERTIFICATE command is sent whereby command chaining is used (for command chaining, see ISO/IEC 7816-8). In the first step the certificate signature is presented, in the second step the PK-Remainder (see DIN V66291-1, clause 18.3.1, a).

CLA	As defined in ISO/IEC 7816-4, command chaining bit set to 1
INS	'2A' = PERFORM SECURITY OPERATION: VERIFY CERTIFICATE
P1	'00'
P2	'AE' = Certificate in the data field, signed signature input consists of non-BER-TLV-coded data, i.e. the certificate content is a concatenation of DEs
Lc	'xx' = Length of subsequent data field
Data field	'5F37'-L-SIG.CA (see annex F)
Le	Absent

Table 51 - PSO: VERIFY CERTIFICATE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 52 - PSO: VERIFY CERTIFICATE response

CLA	As defined in ISO/IEC 7816-4, command chaining bit set to 0
INS	'2A' = PERFORM SECURITY OPERATION: VERIFY CERTIFICATE
P1	'00'
P2	'AE' = Certificate in the data field, signed signature input consists of non-BER-TLV-coded data, i.e. the certificate content is a concatenation of DEs
Lc	'xx' = Length of subsequent data field
Data field	'5F38'-L-PK remainder (see annex F)
Le	Absent

Table 53 - PSO: VERIFY CERTIFICATE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 54 - PSO: VERIFY CERTIFICATE response

Before the EXTERNAL AUTHENTICATE command is performed the key to be applied by the HPC for this command has to be set. Furthermore, the PIN.HPC.ASS must have been successfully presented as required by the access conditions in SE #2.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'81' = SET for external authentication
P2	'A4' = AT
Lc	'11' = Length of subsequent data field
Data field	'83 0C xx ... xx 80 01 1E' = DO for KeyRef of PuK.PDC.AUT, i.e. ICCSN.PDC DO AlgRef, see table E.2
Le	Absent

Table 55 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 56 - MSE response

CLA	As defined in ISO/IEC 7816-4
INS	'84' = GET CHALLENGE
P1, P2	'0000'
Lc	Absent
Data field	Absent
Le	'08'

Table 57 - GET CHALLENGE command

Data field	RND.HPC (8 byte)
SW1-SW2	'9000' or specific status bytes

Table 58 - GET CHALLENGE response

After GET CHALLENGE follows the command EXTERNAL AUTHENTICATE, which delivers the digital signature of the PDC to the HPC (see also Annex G). To receive this digital signature, an INTERNAL AUTHENTICATE command with RND.HPC || ICCSN.HPC (least significant 8 byte) in the data field has to be sent to the PDC. The HPC has to verify the PDC signature.

CLA	As defined in ISO/IEC 7816-4
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'xx' = Length of subsequent data field
Data field	Authentication related data: digital signature of PDC (DSI format ISO/IEC 9796-2, see DIN V66291-1, tab. D.1: '6A' PRND1 h(PRND1 RND.HPC ICCSN.HPC, least significant 8 byte) 'BC' with h = SHA-1)
Le	Absent

Table 59 - EXT. AUTHENTICATE command

NOTE – The token contains no DEs for session key exchange

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 60 - EXT. AUTHENTICATE response

After the CV certificate of the health professional has been presented to the PDC, the software system will require a challenge from the PDC prior to sending the subsequent commands. Before the INTERNAL AUTHENTICATE command is performed the key to be applied by the HPC for this command has to be set.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for internal authentication
P2	'A4' = AT
Lc	'06' = Length of subsequent data field
Data field	'84 01 11' '80 01 1E' = DO for KeyRef of PrK.HPC.AUT DO AlgRef, see table E.2
Le	Absent

Table 61 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 62 - MSE response

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'10' = Length of subsequent data field
Data field	RND.PDC (8 byte) ICCSN.PDC (least significant 8 byte)
Le	'00' or 'xx' = length of expected signature

Table 63 - INT. AUTHENTICATE command

Data field	Digital signature (DSI format ISO/IEC 9796-2, see DIN V66291-1, tab. D.1: '6A' PRND2 h(PRND2 RND.PDC ICCSN.PDC, least significant 8 byte) 'BC' with h = SHA-1)
SW1-SW2	'9000' or specific status bytes

Table 64 - INT. AUTHENTICATE response

7.11.3 Import of PuK.CSP.CS-CV

If the CV certificate of the PDC has to be verified with another PuK.CSP.CS-CV than present in the HPC, the respective PuK must be imported, i.e. the software has to fetch a cross-certificate, where the PuK to be imported is certified and the verification can be performed by using the Puk.CSP.CS-CV present in the HPC. This leads to a 2 step verification process:

- setting the PuK.CSP.CS-CV present in the HPC
- verification of cross-certificate and temporary storing of the new PuK.CSP.CS-CV
- setting the PuK.CSP.CS-CV for the verification of C.PDC.AUT
- verification of C.PDC.AUT as described in 7.9.2.

7.12 Establishment of a Trusted Channel between HPC and SMC

This service can only successfully performed in the Security Environment #3, which has to be set as current SE using the MSE command (RESTORE function). If SE #3 is already the current one, then MSE command may be skipped.

Two cases are to be distinguished:

1. HPC and SMC know each other
2. HPC and SMC don't know each other

The IFD usually starts with the first case.

7.12.1 HPC and SMC know each other

In a first step, the SMK.SMC, consisting of SMK.SMC.ENC and SMK.SMC.MAC, has to be set (the Secure Messaging keys may be stored in an appropriate COS specific cyclic key file in the HPC and SMC after once a successful CV-based authentication procedure (see 7.11.2) between these 2 partners has been performed).

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'81' = SET for external authentication
P2	'A4' = AT
Lc	'0A' = Length of subsequent data field
Data field	'83'-08-'xx ... xx' } DO Key Ref for SMK.SMC: ICCSN.SMC (least significant 8 bytes)
Le	Absent

Table 65 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 66 - MSE response

NOTE – In the SMC, the SMK.HPC has to be referenced with ICCSN.HPC

If the SMK.SMC is known, then a symmetric authentication procedure as described in 7.11.1 shall be performed.

The SM key derivation shall be based on [ANSI-X9.63], i.e. the following steps are performed:

1. KD.HPC_SMC: KD.HPC and KD.SMC are exclusively or-ed (32 byte)
2. H_1(KD.HPC_SMC || C1), whereby H is SHA_1 and C1 a counter of 4 byte with value 1
3. H_2(KD.HPC_SMC || C2), whereby H is SHA_1 and C2 a counter of 4 byte with value 2
4. SMK.HPC_SMC.ENC: first 16 bytes of H_1
5. SMK.HPC_SMC.MAC: first 16 bytes of H_2

If the referenced SMK.SMC is not known, then the status bytes '6A88' = "Referenced data not

found” shall be returned. In this case, the procedure according to 7.12.2 applies.

7.12.2 HPC and SMC don't know each other

If HPC and SMC don't know each other (first time, a TC is established between them), then CV certificates have to be exchanged and verified as described in 7.11.2.

After that the procedure as described in Annex G, which is according to DIN 66291-4, is performed.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'C1' = SET for int./ext. authentication
P2	'A4' = AT
Lc	'xx' = Length of subsequent data field
Data field	'83 0C xx ... xx' '84 01 11' '80 01 1F' = DO for KeyRef of PuK.SMC.AUT, i.e. ICCSN.SMC DO for KeyRef of PrK.HPC.AUT DO AlgRef, see table E.2
Le	Absent

Table 67 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 68 - MSE response

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'10' = Length of subsequent data field
Data field	RND.SMC (8 byte) ICCSN.SMC (least significant 8 byte)
Le	'00' or 'xx' = length of expected signature

Table 69 - INT. AUTHENTICATE command

Data field	E.PuK.SMC.AUT (SIGMIN) with SIGMIN = min (SIG, SIG*), see table G.1
SW1-SW2	'9000' or specific status bytes

Table 70 - INT. AUTHENTICATE response

CLA	As defined in ISO/IEC 7816-4
INS	'84' = GET CHALLENGE
P1, P2	'0000'
Lc	Absent
Data field	Absent
Le	'08'

Table 71 - GET CHALLENGE command

Data field	RND.HPC (8 byte)
SW1-SW2	'9000' or specific status bytes

Table 72 - GET CHALLENGE response

CLA	As defined in ISO/IEC 7816-4
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'xx' = Length of subsequent data field
Data field	E.PuK.HPC.AUT (SIGMIN) with SIGMIN = min (SIG, SIG*), see table G.2
Le	Absent

Table 73 - EXT. AUTHENTICATE command

NOTE – The token contains DEs for session key exchange.

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 74 - EXT. AUTHENTICATE response

The SM keys are computed as described in Annex G and stored persistent in a COS dependant way (min. 5 set of keys, e.g. in a cyclic key file with 5 records) under the key reference SMK.SMC with the value ICCSN.SMC (least significant 8 bytes). These keys are used for SM key derivation next time.

7.12.3 Reading and updating the Display Message

As described in 7.2.7, the display message shall be presented to the HP after successful TC establishment so that the HP is aware that a TC has been properly established. For reading the READ BINARY command in SM mode is used. Update is possible with the UPDATE BINARY command after PIN.HP.ASS presentation.

8 Cryptographic Information Application

8.1 General Structure

The general structure of the Cryptographic Information Application (CIA) is shown in figure 8. The logical file names, the FIDs, the SFIDs and the content of the files (see annex H) are in compliance with ISO/IEC 7816-15.

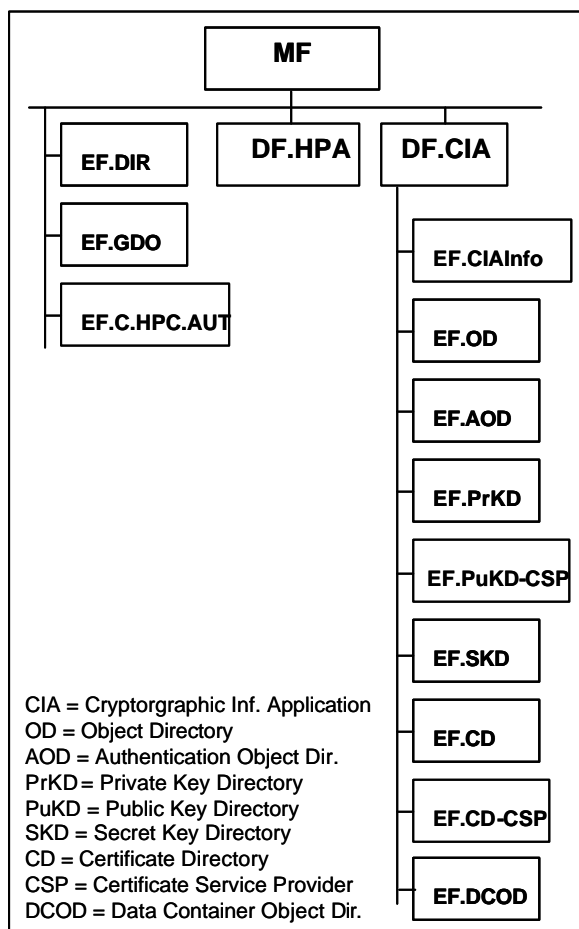


Figure 8 – DF.CIA and substructure

8.2 Application Selection

The ISO/IEC 7816-4 command for 'Direct Application Selection' is shown in the subsequent tables.

CLA	'00'
INS	'A4' = SELECT
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'0B' = Length of subsequent data field
Data field	'E828BD080F D276 00004002' = AID of DF.CIA related to DF.HPA
Le	Absent

Table 75 – SELECT command for DF.CIA selection with AID

NOTE – The RID of DF.CIA is according to ISO/IEC 7816-15. The RID is followed by the AID of the application to which the CIOs belong.

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 76 – SELECT response

8.3 Reading the CIA Files

For reading the CIA files the ISO/IEC 7816-4 command READ BINARY is used.

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1	'92' = b8: 1 and SFID of EF.CIAInfo '91' = b8: 1 and SFID of EF.OD '94' = b8: 1 and SFID of EF.AOD '95' = b8: 1 and SFID of EF.PrKD '96' = b8: 1 and SFID of EF.PuKD-CSP '97' = b8: 1 and SFID of EF.SKD '98' = b8: 1 and SFID of EF.CD '99' = b8: 1 and SFID of EF.CD-CSP '9A' = b8: 1 and SFID of EF.DCOD
P2	'00' = Offset
Lc	Absent
Data field	Absent
Le	'00' = Read until end-of-file or 'xx' = length of expected data

Table 77 – READ BINARY command for reading the CIA files

NOTE – SFID '13' cannot be used, see ISO/IEC 7816-15.

Data field	CIOs
SW1-SW2	'9000' or specific status bytes

Table 78- SELECT response

9 Channel Management

9.1 General Aspects

HPCs supporting channel management (indication in the card capabilities, see Annex A) shall be able to support 4 logical channels as defined in ISO/IEC 7816-4. If the HPC is used in an environment, where no channel mechanism is required, then the basic channel with no. 0 is used for the processing of all commands.

In environments, where the channel mechanism is needed (e.g. the HPC shall remain in the same card reader, but the functionality of the HPC is used by several workstations in a doctor practice or a pharmacy, then the basic channel is only used for reading the EF.GDO and opening a channel.

Within such a channel, the HPA will be selected, having its own independent security status (no PIN-sharing over channels).

Furthermore, for

- the PIN.HP.SIG presentation and
- the usage of the basic security services (electronic signature, client/server authentication and key decipherment)

SM is required, i.e. after DF.HPA selection the presentation of PIN.HP.ASS has to be performed to enable the CV-based authentication procedure with session key agreement (the usage of PrK.HPC.AUT is protected by PIN.HP.ASS).

9.2 Opening a Logical Channel

For opening a logical channel the ISO/IEC 7816-4 command MANAGE CHANNEL with "open" function is used. The command is always send in channel #0.

CLA	'00'
INS	'70' = MANAGE CHANNEL
P1,P2	'0000' = Open a logical channel, channel no. in response data field
Lc	Absent
Data Field	Absent
Le	'01'

Table 79 – MANAGE CHANNEL command for logical channel selection

Data field	Logical channel no. assigned by the HPC (1 byte)
SW1-SW2	'9000' or specific status bytes

Table 80- MANAGE CHANNEL response

9.3 Closing a Logical Channel

For closing a logical channel the ISO/IEC 7816-4 command MANAGE CHANNEL with "close" function is used.

CLA	As defined in ISO/IEC 7816-4
INS	'70' = MANAGE CHANNEL
P1,P2	'8000' = Closing a logical channel, channel no. in CLA
Lc	Absent
Data field	Absent
Le	Absent

Table 81 – MANAGE CHANNEL command for closing a logical channel

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 82 – MANAGE CHANNEL response

10 The Security Module Card SMC

10.1 General Structure

An SMC provides similar functions as an HPC, but the X.509 certificates are not related to a single person but to a health institution (e.g. doctor practice, pharmacy). It contains therefore no health professional data. But an SMC has to support additional functions due to its role as security module and support of trusted channels to HPCs. With respect to evaluation and certification, there are no special evaluation requirements.

An SMC is used in a specific environment and remains in the respective card terminal. The main application is the Security Module Application SMA. A DF.CIA is not needed. The general structure is shown in figure 9.

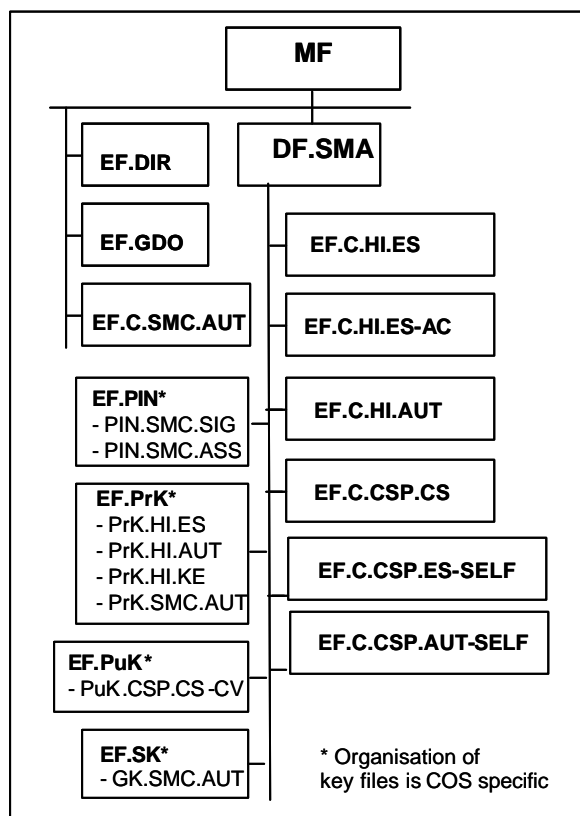


Figure 9 – General file structure of a SMC

10.2 Files, PINs, Keys and Certificates

Files, PINs, Keys and Certificates are similar to an HPC. The main differences are, that

- the certificate owner is the respective health institution HI (e.g. doctor practice, pharmacy)

- the SMC related signature service does not require the SSCD level (i.e. the level for qualified electronic signatures).

An SMC in a health professional environment can interact with a PDC for proving the respective access right. For the authentication with CV certificates, the role id for SMCs is defined in table F.7. For a symmetrical authentication procedure, the SMC needs a group key as defined in the subsequent table.

Key Name	Key Ref	Present in	Protection
GK.SMC.AUT-PHYS	'9C'	HPC-PHYS	PIN.SMC.ASS
GK.SMC.AUT-PHAR	'9F'	HPC-PHAR	PIN.SMC.ASS

Table 83– Key references of GK.SMC.AUT

In an SMC there is only the SE #1 (default SE) present.

10.3 Application Selection

The ISO/IEC 7816-4 command for 'Direct Application Selection' is shown in the 2 subsequent tables.

CLA	As defined in ISO/IEC 7816-4
INS	'A4' = SELECT
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'06' = Length of subsequent data field
Data field	'D276 00004003' = AID of DF.SMA
Le	Absent

Table 84 – SELECT command for DF.SMA selection with AID

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 85 – SELECT response

10.4 Command Sequences

The command sequences for

- signing,
- client/server authentication,
- document cipher key decryption and
- SMC/PDC interaction

are equivalent to the command sequences to be supported by an HPC.

10.5 Support of a Trusted Channel between an SMC and a remote HPC

10.5.1 TC establishment and handling of secured HPC commands

For TC establishment an authentication procedure between SMC and HPC with session key agreement as defined in 7.12 has to be performed.

For the production of a secured HPC command and the processing of a secured HPC response, the SMC shall support either

- PSO commands as specified in 10.5.2 and 10.5.3 or
- the ENVELOPE command as specified in 10.5.4 and 10.5.5 or
- both concepts.

In any case, the secured HPC command has to be constructed using the SM-DOs delivered by the SMC.

10.5.2 Production of secured commands using PSO commands

The general principle is shown in annex I. Special care may be suitable to control this command sequence.

For the SM-DO production the following commands are used:

- PSO: COMPUTE CC
- PSO: ENCIPHER

Before, the SM keys for the PSO commands have to be set with MSE commands.

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'C1' = SET for computation and verification
P2	'B4' = CCT
Lc	'03' = Length of subsequent data field
Data field	'83 01 F1' = DO for KeyRef of SK.SMC.MAC
Le	Absent

Table 86 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 87 - MSE response

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'C1' = SET for encipherment and decipherment
P2	'B8' = CT
Lc	'03' = Length of subsequent data field
Data field	'83 01 F2' = DO for KeyRef of SK.SMC.ENC
Le	Absent

Table 88 - MSE command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 89 - MSE response

CLA	As defined in ISO/IEC 7816-4
INS	'2A' = PSO: COMPUTE CC
P1	'8E' = CC in response data field
P2	'80' = PV in command data field
Lc	'xx' = Length of subsequent data field
Data field	Data for which the cryptographic checksum shall be computed
Le	'00'

Table 90 – PSO: COMPUTE CC command

Data field	Cryptographic checksum
SW1-SW2	'9000' or specific status bytes

Table 91 – PSO: COMPUTE CC response

CLA	As defined in ISO/IEC 7816-4
INS	'2A' = PSO: ENCIPHER
P1	'86' = PI-cryptogram in response data field
P2	'80' = PV in command data field
Lc	'xx' = Length of subsequent data field
Data field	Data to encipher
Le	'00'

Table 92 – PSO: ENCIPHER command

Data field	'01' (=PI) – enciphered data
SW1-SW2	'9000' or specific status bytes

Table 93 – PSO: ENCIPHER response

10.5.3 Processing of secured responses using PSO commands

For the verification of a cryptographic checksum, the PSO operation VERIFY CC shall be used.

CLA	As defined in ISO/IEC 7816-4
INS	'2°' = PSO: VERIFY CC
P1	'00'
P2	'A2' = PV in command data field
Lc	'xx' = Length of subsequent data field
Data field	'80'-L-PV '8E'-L-CC
Le	Absent

Table 94 – PSO: VERIFY CC command

Data field	Absent
SW1-SW2	'9000' or specific status bytes

Table 95 – PSO: VERIFY CC response

If the response data contain a cryptogram, then the deciphered data are obtained with the PSO operation DECIPHER.

CLA	As defined in ISO/IEC 7816-4
INS	'2A' = PSO: DECIPHER
P1	'80' = PV in response data field
P2	'86' = PI-cryptogram in command data field
Lc	'xx' = Length of subsequent data field
Data field	PI - cryptogram
Le	'00'

Table 96 – PSO: DECIPHER command

Data field	Deciphered data
SW1-SW2	'9000' or specific status bytes

Table 97 – PSO: DECIPHER response

10.5.4 Production of secured commands using the ENVELOPE command

An ENVELOPE command with odd instruction code is send in normal mode to the SMC. In the data field, the unsecured HPC command in the DO Command-to-perform (tag '52') and an SM template (tag '7D') is present containing the Response Descriptor (tag 'BA') which denotes,

what shall be returned. The SM template enforces the usage of the SM keys.

NOTE – The ENVELOPE command with odd instruction code allows the construction of such special services.

CLA	As defined in ISO/IEC 7816-4
INS	'C3' = ENVELOPE
P1-P2	'0000'
Lc	'xx' = Length of subsequent data field
Data field	If the HPC command data – if any – shall be transmitted in DO PV (Case 1): '52'-L-HPC command-to-perform '7D'-L-(BA' -L- ['8E'-'00']) If the HPC command data shall be transmitted in a DO CG (Case 2): '52'-L-HPC command-to-perform '7D'-L-(BA' -L- ['87'-'00' '8E'-'00'])
Le	'00'

Table 98 – ENVELOPE command

Data field	- Case 1: '7D'-L-(8E' -'04'-CC) - Case 2: '7D'-L-(87' -L- '01'-CG '8E' -'04'-CC)
SW1-SW2	'9000' or specific status bytes

Table 99 – ENVELOPE response

The Cryptographic Checksum CC has to be computed according to the rules for SM-protected commands as defined in [ISO7816-4], whereby the command header shall be always integrated in the CC.

NOTE – If the verification data (PIN or - in a later HPC version - biometric verification data) shall be delivered in enciphered form from an input unit (crypto-PIN pad / crypto sensor) to the SMC, then an MSE command and a PSO: DECIPHER command (response data field absent!) prior to the ENVELOPE command should be sent to the SMC, whereby the data are enciphered by a secret key known to the crypto-PIN pad / crypto sensor and the SMC and set with the MSE command. The DO command-to-perform in the data field of the ENVELOPE command contains the VERIFY command in the form <CLA'-INS-P1-P2-Lc-T.CG-'00'>, whereby '00' denotes, that the verification data are already present in the card, and the RD denotes that the DO CG and the DO CC should be returned.

10.5.5 Processing of secured responses using the ENVELOPE command

The HPC will return secured responses, whereby 3 cases occur:

- response with DO Processing status (tag '99')

- response with DO Plain value (tag '81')
- response with DO Cryptogram (tag '87').

All secured responses are protected by a cryptographic checksum CC.

NOTE – The cases addressed here are application cases and should not be mixed up with transmission cases as described in ISO/IEC 7816-3.

The CC must be verified. If a cryptogram is present, then the plain value has to be returned by the SMC after successful verification of the CC.

CLA	As defined in ISO/IEC 7816-4
INS	'C3' = ENVELOPE
P1-P2	'0000'
Lc	'xx' = Length of subsequent data field
Data field	Case 1: '7D'-L-('99'-02'- SW1-SW2 '8E'-04'-CC) Case 2: '7D'-L-('81'-L- Data '8E'-04'-CC) Case 3: '7D'-L-('87' –L- '01'-CG '8E'-04'-CC 'BA' –L- ['80'-00'])
Le	Case 1, 2: Absent Case 3: '00'

Table 100 – ENVELOPE command

Data field	- Case 1, 2: Absent - Case 3: '7D'- L-('80'-L-Data)
SW1-SW2	'9000'or specific status bytes

Table 101 – ENVELOPE response

Annex A

(normative)

ATR

A.1 ATR Coding

Table A.1 shows the coding of the ATR for HPC and SMC (T=1 cards).

Character	Value	Meaning
TS	'3B'	Initial Character (direct convention)
T0	'9x'	Format Character (TA1/TD1 indication, x = no. of HB)
TA1	'xx'	Interface Character (FI/DI value)
TD1	'81'	Interface Character (T=1, TD2 indication)
TD2	'B1'	Interface Character (T=1, TA3/TB3/TD3 indication)
TA3	'xx'	Interface Character (IFSC coding)
TB3	'45'	Interface Character (BWI/CWI coding)
TD3	'1F'	Interface Character (T=15, TA4 indication)
TA4	'xx'	Interface Character (XI/UI coding)
Ti	HB	Historical Bytes (HB, imax. = 15)
TCK	XOR	Check Character (exclusive OR)

Table A.1 – ATR coding (sequence from top to down)

For the coding of the Historical Bytes the following conventions in compliance with ISO/IEC 7816-4 apply:

- CI = '00' according to ISO/IEC 7816-4
- TPI = '6x' according to ISO/IEC 7816-4 (x codes the length of the DO)
- ICM = IC Manufacturer Id (see Table A.2)
- ICT = Coding manufacturer specific
- OSV = Coding manufacturer specific
- DD = Coding manufacturer specific (usually not used)
- TCS = '31' according to ISO/IEC 7816-4
- CS = Card Service Data Byte according to ISO/IEC 7816-4
- TCC = '73' according to ISO/IEC 7816-4
- CCB = Card Capabilities Data Bytes according to ISO/IEC 7816-4 (indication of supported logical channels, extended Le field, ...)
- CLS = Card Life Cycle (Default value '00')
- SW1-SW2 = '9000'

Historical Bytes

CI	PIDO	CPDO	CLS	SW1-SW2
----	------	------	-----	---------

CI = Category Indicator ('00')
 PIDO = Pre-Issuing Data Object
 CPDO = Card Profile Data Object
 CLS = Card Life Status
 SW1-SW2 = Status bytes

Pre-Issuing Data Object

TPI	ICM	ICT	OSV	DD
-----	-----	-----	-----	----

TPI = Tag/Length Pre-Issuing DO ('6x')
 ICM = IC Manufacturer ID
 ICT = IC Type (1 byte, if b8=0;
 2 byte, if b8=1 of first byte)
 OSV = Operating System Version (2 byte)
 DD = Discretionary Data (n byte)

Card Profile Data Objects

TCS	CS	TCC	CCB
-----	----	-----	-----

TCS = Tag/Length Card Service DO ('31')
 CS = Card Service Data Byte
 TCC = Tag/Length Card Capabilities DO ('73')
 CCB = Card Capability Bytes

Figure A.1 – Structure of the Historical Bytes

Table A.2 shows the actual values for chip manufacturer ICM.

answered by an HPC with S-Block IFS Response. The value for IFSD shall be 254 Bytes.

ICM	IC Manufacturer (see ISO/IEC 7816-6 and www.sc17.com , Standing Document 4)
'01'	Motorola, US
'02'	STMicroelectronics, FR
'03'	Hitachi, JP
'04'	Philips Semiconductors, DE
'05'	Infineon, DE
'06'	Cylinco, US
'07'	Texas Instruments, US
'08'	Fujitsu, JP
'09'	Matsushita, JP
'0A'	NEC, JP
'0B'	Oki, JP
'0C'	Toshiba, JP
'0D'	Mitsubishi, JP
'0E'	Samsung, KR
'0F'	Hyundai, KR
'10'	LG Semiconductors, KR
'11'	Emosyn-EM, US
'12'	Inside Technologies, FR
'13'	ORGA Kartensysteme, DE
'14'	Sharp, JP
'15'	ATMEL, US
'16'	EM Microelectronic, CH
'17'	KSW Microtec, DE
'18'	ZMD, DE
'19'	XICOR, US
'1A'	Sony, JP
'1B'	Malaysia Microelectronic Solutions, MY
'1C'	Emosyn, US
'1D'	Shanghai Fudan Microelectronics, CN
'1E'	Magellan Technology, AU
'1F'	Melexis, CH

Table A.2 – ICM coding

HPC and SMC shall support the asynchronous half-duplex block transmission protocol T=1. The asynchronous half-duplex character transmission protocol T=0 may be present, but shall not be indicated in the ATR.

The T=1 implementation shall be in compliance with ISO/IEC 7816-3. Chaining is mandatory.

The following simplifications are allowed:

- NAD Byte: not interpreted (NAD shall be set to '00')
- S-Block ABORT: not used
- S-Block VPP state error: not used

The Information Field Size Card (IFSC) shall be indicated in the ATR (Character TA3, recommended value: at least '80' = 128 Bytes).

The Information Field Size Device (IFSD) shall be transmitted by the interface device (IFD) immediately after answer to reset and PPS, i.e. the IFD shall send at once after answer to reset the S-Block IFS Request which has to be HPC Pharmacist & Physician V2.0

Annex B

(normative)

File Attributes, Access Conditions and Security Environments

B.1 Elementary files

B.1.1 Naming concept for certificate files

The naming concept for certificate files is shown in figure B.1.

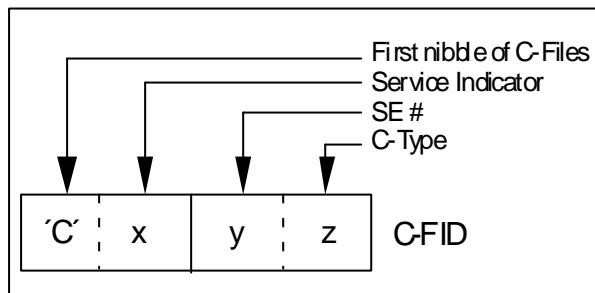


Figure B.1 – FID naming concept for certificate files

The coding for the service indicator shows table B.1.

Value	Service
'0'	Digital Signature acc. to SigG (X.509v3 Certificate)
'1'	Entity Authentication (CV Certificate)
'2'	Key Encipherment (X.509v3 Certificate)
'3'	Data Encipherment (X.509v3 Certificate)
'4'	Key Agreement (X.509v3 Certificate)
'5'	Entity Authentication (X.509v3 Certificate)
	Other values RFU

Table B.1 – Coding of the service indicator

The SE# is applied if security environments are used and have impact on the FID. In the current case the third nibble of the FID is set to '0'.

The certificate type denotes, to whom the certificate is related.

Value	C-Type
'0'	C.CH (PK Certificate of Cardholder) or C.ICC
'1' - '7'	C.CH (Business or Professional (Attribute-) Certificate of Cardholder)
'8' - 'D'	C.CSP (Certificate for a CSP issued by RCSP)
'E'	C.RCA (self certificate of RCSP)
'F'	RFU

Table B.2 – Coding of the certificate type

B.1.2 HPC file characteristics

Only those EFs are described, which are accessible with read/updated commands after personalization of the HPC. The access rules are described in Table B.11.

B.1.2.1 EFs at MF Level

File	FID / SFID	File structure	File size (length of data)	Access condition	Pre- sence
EF.DIR (Application Directory)	'2F00' '1E'	transparent	64 byte	Read: always Update: never ARR #1	man- datory
EF.GDO (Global Data Objects)	'2F02' '1D'	transparent	64 byte	Read: always Update: never ARR #1	man- datory
EF.C.HPC.AUT (Card Verifiable Certificate)	'2F03' '01'	transparent	256 byte or length of C.ICC.AUT	Read: always Update: never ARR #1	man- datory

Table B.3 – EFs at MF level and their characteristics

B.1.2.2 EFs under DF.HPA

File	FID / SFID	File structure	File size (length of data)	Access condition	Pre- sence
EF.HPD (Health Professional Data)	'D001' '02'	trans- parent	2 k byte or length of HP data	Read: Always Update: never ARR #1	man- datory
EF.DM (Display Message)	'D002' '03'	trans- parent	8 byte	Read: PIN.HP.ASS or CHA=AID.R-SMC.HP, see tab. F.7 Update: PIN.HP.ASS ARR #4	man- datory
EF.C.HP.ES (Electronic Signature Certificate)	'C000' '04'	trans- parent	1.5 k byte or length of certificate	Read: PIN.HP.ASS Update: never ARR #2	man- datory
EF.C.HP.ES-AC1 (Electronic Signature Attri- bute Certificate 1)	'C001' '05'	trans- parent	1k byte or length of certificate	Read: PIN.HP.ASS Update: PIN.HP.ASS ARR #3	man- datory
EF.C.HP.ES-AC2 (Electronic Signature Attri- bute Certificate 2)	'C002' '06'	trans- parent	1k byte or length of certificate	Read: PIN.HP.ASS Update: PIN.HP.ASS ARR #3	man- datory
EF.C.HP.ES-AC3 (Electronic Signature Attri- bute Certificate 3)	'C003' '07'	trans- parent	1k byte or length of certificate	Read: PIN.HP.ASS Update: PIN.HP.ASS ARR #3	man- datory
EF.C.HP.AUT (Authentication Certificate)	'C500' '08'	trans- parent	1.5 k byte or length of certificate	Read: PIN.HP.ASS Update: never ARR #2	man- datory
EF.C.CSP.CS (Digital Signature Certifi- cate for CSP)	'C008' '09'	trans- parent	1k byte or length of certificate	Read: always Update: never ARR #1	man- datory
EF.C.RCSP.ES-SELF (Self Certificate of RCSP for Electronic Signature)	'C00E' '0A'	trans- parent	1k byte or length of certificate	Read: always Update: never ARR #1	man- datory
EF.C.RCSP.AUT-SELF (Self Certificate of RCSP for Authentication)	'C50E' '0B'	trans- parent	1k byte or length of certificate	Read: always Update: never ARR #1	man- datory

Table B.4 – HPA files and their characteristics

B.1.2.3 EFs under DF.CIA

If the DF.CIA is present, then the CIA files and their characteristics according to the subsequent table shall be installed.

File	FID / SFID	File structure	File size (length of data)	Access condition	Pre- sence
EF.CIAInfo (CIA Information)	'5032' '12'	transparent	256 byte or length of CIOs	Read: always Update: never	man- datory
EF.OD (Object Directory)	'5031' '11'	transparent	64 byte or length of CIOs	Read: always Update: never	man- datory
EF.AOD (Authent. Object Directory)	'5034' '14'	transparent	256 byte or length of CIOs	Read: always Update: never	man- datory
EF.PrKD (Private Key Directory)	'5035' '15'	transparent	512 byte or length of CIOs	Read: always Update: never	man- datory
EF.PuKD-CSP (Public Key Directory)	'5036' '16'	transparent	128 byte or length of CIOs	Read: always Update: never	man- datory
EF.SKD (Secret Key Directory)	'5037' '17'	transparent	128 byte or length of CIOs	Read: always Update: never	man- datory
EF.CD (HP Certificate Directory)	'5038' '18'	transparent	256 byte or length of CIOs	Read: always Update: never	man- datory
EF.CD-CSP (CSP Certificate Directory)	'5039' '19'	transparent	128 byte or length of CIOs	Read: always Update: never	man- datory
EF.DCOD (Data Container Object Directory)	'503A' '1A'	transparent	128 byte or length of CIOs	Read: always Update: never	man- datory

Table B.5 – CIA files and their characteristics

NOTE: SFID '13' not used (see ISO/IEC 7816-15)

For all EFs, the access rule #1 (see table B.11) applies.

B.2 Security Environments and Control Reference Templates for DF.HPA

CRT-Name	Tag	L	Tag	L	Value	Meaning
AT_1	'A4'	'06'				AT for user authentication
			'95'	'01'	'08'	UQ – user authentication, knowledge based
			'83'	'01'	'01'	Reference of PIN.HP.ASS
AT_2	'A4'	'06'				AT for user authentication
			'95'	'01'	'08'	UQ – user authentication, knowledge based
			'83'	'01'	'81'	Reference of PIN.HP.SIG
AT_3	'A4'	'xx'				AT for internal device authentication with PDC (asym)
			'95'	'01'	'40'	UQ – internal authentication
			'84'	'01'	'11'	Reference of PrK.HPC.AUT
			'80'	'01'	'1E'	AlgID
AT_4	'A4'	'xx'				AT for int./ext. device authentication with SMC (asym)
			'95'	'01'	'C0'	UQ – internal/external authentication
			'84'	'01'	'11'	Reference of PrK.HPC.AUT
			'80'	'01'	'1F'	AlgID
AT_5	'A4'	'09'				AT for client-server authentication (asym)
			'95'	'01'	'40'	UQ – internal authentication
			'84'	'01'	'92'	Reference of PrK.HP.AUT
			'80'	'01'	'xx'	AlgID
AT_6	'A4'	'06'				AT for int./ext. device authentication with PDC (sym)
			'95'	'01'	'C0'	UQ – internal/external authentication
			'84'	'01'	'xx'	Reference of GK.HPC.AUT-HP

Table B.6 – Authentication Templates (AT)

CRT-Name	Tag	L	Tag	L	Value	Meaning
DST_1	'B6'	'09'				DST for electronic signature
			'95'	'01'	'40'	UQ – computation (DST)
			'84'	'01'	'91'	Reference of PrK.HP.ES
			'80'	'01'	'xx'	AlgID, see Table E.1
DST_2	'B6'	'09'				DST for signature verification (CV certificate)
			'95'	'01'	'80'	UQ – verification (DST)
			'83'	'01'	'xx'	Reference of PuK.CSP.CS-CV
			'80'	'01'	'xx'	AlgID

Table B.7 – Digital Signature Templates (DST)

CRT-Name	Tag	L	Tag	L	Value	Meaning
CT_1	'B8'	'09'				CT for key encipherment/decipherment
			'95'	'01'	'40'	UQ – decipherment (CT)
			'84'	'01'	'93'	Reference of PrK.HP.KE
			'80'	'01'	'xx'	AlgID
CT_2	'B8'	'06'				CT for SM in commands
			'95'	'01'	'10'	UQ – SM in command data fields
			'84'	'01'	'xx'	Reference of the computed SM key SMK.ENC
CT_3	'B8'	'06'				CT for SM in responses
			'95'	'01'	'20'	UQ – SM in response data fields
			'84'	'01'	'xx'	Reference of the computed SM key SMK.ENC

Table B.8 – Confidentiality Templates (CT)

CRT-Name	Tag	L	Tag	L	Value	Meaning
CCT_1	'B4'	'08'				CT for SM in commands
			'95'	'01'	'10'	UQ – SM in command data fields
			'84'	'01'	'xx'	Reference of the computed SM key SMK.MAC
			'87'	'00'		Previous initial value block plus one
CCT_2	'B4'	'08'				CT for SM in responses
			'95'	'01'	'20'	UQ – SM in response data fields
			'84'	'01'	'xx'	Reference of the computed SM key SMK.MAC
			'87'	'00'		Previous initial value block plus one

Table B.9 – Cryptographic Checksum Templates (CCT)

B.2.2 Security Environments

The following table describes which security mechanisms are available in the respective security environment (SE#1 not explicitly used).

SE	L	SE#	L	V	CRTs	Meaning
'7B'	var.					
		'80'	'01'	'02'		SE #2
					AT_1	AT for user verification (PIN.HP.ASS)
					AT_2	AT for user verification (PIN.HP.SIG)
					AT_3	AT for external device authentication (HPC/PDC)
					AT_5	AT for client-server authentication
					AT_6	AT for internal/external device authentication (sym)
					DST_1	DST for electronic signature
					DST_2	DST for signature verification (CV certificate)
					CT_1	CT for key encipherment/decipherment
'7B'	var.					
		'80'	'01'	'03'		SE #3
					AT_1	AT for user verification (PIN.HP.ASS)
					AT_2	AT for user verification (PIN.HP.SIG)
					AT_4	AT for internal device authentication (HPC/SMC)
					AT_5	AT for client-server authentication
					DST_1	DST for electronic signature
					DST_2	DST for signature verification (CV certificate)
					CT_1	CT for key encipherment/decipherment
					CT_2	CT for SM in commands
					CT_3	CT for SM in responses
					CCT_1	CCT for SM in commands
					CCT_2	CCT for SM in responses

Table B.10 – Security Environments

The SEs may be stored in EF.SE, referenced in the FCP of the DF.HPA in the DO "FID of EF.SE" (tag '8D'). The SE templates may contain additional information, e.g. life cycle status information.

B.3 Access Conditions

In the subsequent specification of the access rules, allowed operations and their access conditions are described. If a command, e.g. UPDATE BINARY, is never allowed, then it is not described in the respective ARR, since the default is "never" (in cards which have not this default value, an additional

AM-DO must be present – addressing all other possible commands – followed by the SC-DO '9700' = never).

The Tables B.11 – B.13 show possible codings of the access rules in EF.ARR and Table B.14 denotes the security data objects referencing the expanded format and pointing to these access rules.

Rec-No.	Value	Meaning
1	'800101' '9000'	AM: READ BINARY SC: Always
2	'800101' 'A406' '950108' '830101'	AM: READ BINARY SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01'
3	'800103' 'A406' '950108' '830101'	AM: UPDATE / READ BINARY SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01'
4	'800101' 'A0xx' 'A406' '950108' '830101' 'AFxx' CCT_1 (see Table B.9) CT_2 (see Table B.8) CCT_2 (see Table B.9) CT_3 (see Table B.8) 'A40D' '950180' '5F4C 07 D27600004000xx' (xx = see Table F.7) '800102' 'A406' '950108' '830101'	AM: READ BINARY SC: OR-Template (AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01') or (AND-Template CCT for SM in commands CT for SM in commands CCT for SM in responses CT for SM in responses AT UQ: External Authentication CHA with value AID (6 msb byte) R-SMC.PHAR/PHYS) AM: UPDATE BINARY AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01'

Table B.11 – Access conditions for EFs in EF.ARR

NOTE – Instead of CHA, key references may be used representing the different roles.

Rec- No.	Value	Meaning
5	'87032A9E9A' 'A406' '950108' '830181'	AM: PSO: COMPUTE DS (SE #2) SC: AT UQ: Knowledge based user auth. PIN.HP.SIG referenced by '81'
6	'87032A9E9A' 'A406' '950108' '830181' CCT_1 (see Table B.9) CT_2 (see Table B.8) CCT_2 (see Table B.9) CT_3 (see Table B.8)	AM: PSO: COMPUTE DS (SE #3) SC: AT UQ: Knowledge based user auth. PIN.HP.SIG referenced by '81' CCT for SM in commands CT for SM in commands CCT for SM in responses CT for SM in responses
7	'840188' 'A406' '950108' '830101'	AM: INT. AUTH. (SE #2, Client-Server) SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01'
8	'840188' 'A406' '950108' '830101' CCT_1 (see Table B.9) CT_2 (see Table B.8) CCT_2 (see Table B.9) CT_3 (see Table B.8)	AM: INT. AUTH. (SE #3, Client-Server) SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01' CCT for SM in commands CT for SM in commands CCT for SM in responses CT for SM in responses
9	'87032A8086' 'A406' '950108' '830101'	AM: PSO: DECIPHER (SE #2) SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01'
10	'87032A8086' 'A406' '950108' '830101' CCT_1 (see Table B.9) CT_2 (see Table B.8) CCT_2 (see Table B.9) CT_3 (see Table B.8)	AM: PSO: DECIPHER (SE #3) SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01' CCT for SM in commands CT for SM in commands CCT for SM in responses CT for SM in responses
11	'840188' 'A406' '950108' '830101'	AM: INT. AUTH. (SE #2, HPC/PDC authent.) SC: AT UQ: Knowledge based user auth. PIN.HP.ASS referenced by '01'
12	'870188' '9000'	AM: INT. AUTH. (SE #3, HPC/SMC authent.) SC: Always

Table B.12 – Access conditions for private and secret keys in EF.ARR

Rec- No.	Value	Meaning
13	'840120' '9000'	AM: VERIFY (SE #2) SC: Always
14	'840120' CCT_1 (see Table B.9) CT_2 (see Table B.8) CCT_2 (see Table B.9) CT_3 (see Table B.8)	AM: VERIFY (SE #3) CCT for SM in commands CT for SM in commands CCT for SM in responses CT for SM in responses

Table B.13 – Access conditions for PINs in EF.ARR

Key Name	Key-Ref	ARR	Meaning
Prk.HP.ES	'91'	'8B-06-EFID.ARR-02-05-03-06	SE #2: Rec. 5 SE #3: Rec. 6
Prk.HP.AUT	'92'	'8B-06-EFID.ARR-02-07-03-08	SE #2: Rec. 7 SE #3: Rec. 8
Prk.HP.KE	'93'	'8B-06-EFID.ARR-02-09-03-0A	SE #2: Rec. 9 SE #3: Rec. 10
Prk.HPC.AUT	'11'	'8B-06-EFID.ARR-02-0B-03-0C	SE #2: Rec. 11 SE #3: Rec. 12
GK.HPC.AUT-HP	'9A' '9B'	'8B-04-EFID.ARR-02-0B	SE #2: Rec. 11
PIN.HP.ASS	'01'	'8B-06-EFID.ARR-02-0D-03-0E	SE #2: Rec. 13 SE #3: Rec. 14
PIN.HP.SIG	'81'	'8B-06-EFID.ARR-02-0D-03-0E	SE #2: Rec. 13 SE #3: Rec. 14

Table B.14 – Security attribute data objects referencing expanded format

Annex C

(normative)

Structure and Content of ES Certificates

C.1 Electronic signature public key certificate (X.509v3 ES-certificate)

C.1.1 General aspects

The coding scheme of HP ES-certificates shall comply with

- standardised coding schemes for certificates (X.509v3),
- legal requirements (the German digital signature act SigG and the associated signature ordinance SigV), and
- ISIS-MTT specification for interoperable PKI applications (especially Part 1 "Certificate and CRL Profiles" [ISIS-MTT P1], and its optional profile "SigG-Profile" [ISIS-MTT OP]).

Certificate fields that are qualified as mandatory in the above mentioned documents, regulations and specifications have to be present in an ES certificate for health professionals. In addition there are further mandatory certificate fields due to the intended usage of the HP ES-certificates in national and international environments.

The ES certificates consist of

- the ES public key certificate, and
- one or more ES attribute certificates.

The ES public key certificate is described subsequently as a subset of the ISIS-MTT specification for certificates. The attribute certificates are described in clause C.7.

C.1.2 Certificate structure

The general certificate structure consists of three mandatory parts:

- the certificate content to be signed,
- the signature algorithm, and
- the signature of the certificate issuer.

In ASN.1 notation the general certificate structure is defined by the type *Certificate*, defined below.

```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signature           BIT STRING }
```

C.2 ToBeSignedCertificate

C.2.1 General structure

The general structure contains the following fields (in ASN.1 type definition):

```
TBSCertificate ::= SEQUENCE {
HPC Pharmacist & Physician V2.0
```

version	[0] EXPLICIT Version DEFAULT v1,
serialNumber	CertificateSerialNumber,
signature	AlgorithmIdentifier,
issuer	Name,
validity	Validity,
subject	Name,
subjectPublicKeyInfo	SubjectPublicKeyInfo,
issuerUniqueID	[1] IMPLICIT UniqueIdentifier OPTIONAL,
subjectUniqueID	[2] IMPLICIT UniqueIdentifier OPTIONAL,
extensions	[3] EXPLICIT Extensions Optional }

NOTE – In X.509v3 extensions are optional, but for ES-certificates, some extensions are mandatory.

C.2.2 Version

The version denotes the coding scheme of a certificate.

ASN.1 definition of type *Version*:

```
Version ::= INTEGER { v1(0), v2(1), v3(2) }
```

For health professional ES-certificates, only X.509v3 is relevant.

C.2.3 Serial Number

The *serialNumber* in combination with the *issuer* name fields of ES certificates uniquely identifies a ES certificate [RFC2459]. This uniqueness requirement is extended in [ISIS-MTT P1] to all kinds of certificates, i.e. for ES as well as attribute certificates (ACs).

ASN.1 definition of type *CertificateSerialNumber* for the *serialNumber* field:

```
CertificateSerialNumber ::= INTEGER
```

The [RFC2459] definition of serial number does not constrain the value or the length of this field. However, [RFC3280] requires that the serial number MUST be a positive integer, not longer than 20 octets ($1 \leq SN < 2^{159}$, MSB=0 indicates the positive sign!). These requirements on length also apply for [ISIS-MTT P1] and have to taken into account for HP ES-certificates.

C.2.4 Signature

The *signature* field of type *AlgorithmIdentifier* identifies the signature algorithm used by the certification authority (CA) to sign this certificate. Its content must be the same as that of the field *signatureAlgorithm* (see C.3).

ASN.1 definition of type *AlgorithmIdentifier* for the *signature* field:

```
AlgorithmIdentifier ::= SEQUENCE {
    algorithm      OBJECT IDENTIFIER,
    parameters    ANY DEFINED BY algorithm OPTIONAL }
```

C.2.5 Issuer

The *issuer* field of the type *Name* specifies the CA which issued the certificate.

ASN.1 definition of type *Name* for the *issuer* field:

```
Name ::= CHOICE { RDNSequence }
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
```

```

RelativeDistinguishedName ::= SET OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
    type      AttributeType,
    value     AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType

DirectoryString ::= CHOICE {
    printableString PrintableString (SIZE (1..maxSize))
    teletexString   TeletexString (SIZE (1..maxSize))
    utf8String      UTF8String (SIZE (1..maxSize))
    bmpString       BMPString (SIZE (1..maxSize))
    universalString UniversalString (SIZE (1..maxSize)) }

```

For attribute types that are based on the type *DirectoryString* the following notes shall be regarded:

- A printable string contains characters with ASCII coding, international version.
- A teletex string contains characters according to the T.61 alphabet, able to encode German specific characters like ä and ß.
- The UTF8 encoding method, defined in [RFC2279], is the preferred one among all CHOICE options and MUST be used in all certificates issued after December 31, 2003 [RFC2459]: Until that date, strings MAY be represented as the most restrictive one of PrintableString or BMPString. TeletexString and UniversalString are only included for backward compatibility and SHOULD NOT be used in new certificates. [ISIS-MTT P1] recommends to use a subset of the UTF8 character set, including only the ANSI/ISO 8859-1 characters (Unicode Latin-1 page).

Concerning attribute types to be used in the issuer field for HP ES-certificates according to this specification, only the following attributes are mandatory:

- country name, and
- organization name.

The use of other attribute types, e.g. serial number, is optional.

The coding scheme for all *issuer* attribute types in HP ES-certificates is defined in [ISIS-MTT P1, Table 7].

C.2.5.1 Country name

The country name identifies the country in which the CA is located.

ASN.1 definition of attribute type *countryName*:

```

id-at                ::= { 2 5 4 }
id-at-countryName AttributeType ::= { id-at 6 }
CountryName          ::= PRINTABLE STRING (SIZE(2))

```

The printable string of the country name for Germany is accordingly to ISO 3166 „DE“.

C.2.5.2 Organization name

The organization name identifies the CA.

ASN.1 definition of attribute type *organizationName*:

```
id-at-organizationName AttributeType ::= { id-at 10 }
OrganizationName          ::= DirectoryString (SIZE(1..64))
```

For the printable string variant of the organization name for CAs the same convention as for card verifiable certificates may be used:

- 1.-2. character: „DE“
- 3.-5. character: CA acronym (e.g. ZGW for „Zertifizierungsstelle GesundheitsWesen“)

C.2.6 Validity

The validity field denotes the validation period of the certificate.

ASN.1 definition of type *Validity*.

```
Validity          ::= SEQUENCE {
    notBefore      Time,
    notAfter       Time }

Time              ::= CHOICE {
    utcTime        UTCTime,
    generalizedTime GeneralizedTime }
```

Validity dates before and through 2049 shall be encoded as *UTCTime*, dates in 2050 and later as *GeneralizedTime*. Processing components shall be able to interpret both data formats. Data values shall be given in the following format:

YYMMDDhhmmssZ for *UTCTime*, and
YYYYMMDDhhmmssZ for *GeneralizedTime*.

with

- Y = Year
- M = Month
- D = Day
- h = Hour
- m = Minute
- s = Second

Z is the symbol for Greenwich Mean Time GMT.

Example:
"20030101000000Z"

The maximum validity period is determined by the CA and legal regulations depending on the strength of algorithms used.

C.2.7 Subject

The *subject* field identifies the certificate holder in the sense of technical identification (the juridical identification can be provided according to ISIS-MTT in the *subjectDirectoryAttributes* extension, see C.2.9.5). The general substructure of the *subject* field is *Name*, which is the same as for the *issuer* field. The subject name MUST at least contain the attributes *commonName* and *countryName*.

For HP ES-certificates according to this specification, the following attributes shall be present:

- countryName
- commonName.

The use of other attribute types, e.g. stateOrProvince, serialNumber, is optional.

The coding scheme for these *subject* attribute types in HP ES-certificates is defined in [ISIS-MTT P1, Table 7].

C.2.7.1 Country name

The country name identifies the country, in which the certificate holder is registered. The conventions as described in clause C.2.5.1 apply.

C.2.7.2 State or province

The state name identifies the state, in which the certificate holder is registered (e.g. Bayern).

ASN.1 definition of attribute type *stateOrProvinceName*:

```
id-at-StateOrProvinceName AttributeType ::= { id-at 8 }
StateOrProvinceName          ::= DirectoryString (SIZE(1..128))
```

The preferred encoding variant for *DirectoryString* is UTF8 subset with only ANSI/ISO 8859-1 characters (Unicode Latin-1 page) (see also notes in C.2.5).

C.2.7.3 Common name

The common name specifies the name of a person in such a way as it is commonly used. In qualified certificates it MUST contain the legal name of the card holder.

ASN.1 definition of attribute type *commonName*:

```
id-at-commonName AttributeType ::= { id-at 3 }
CommonName          ::= DirectoryString (SIZE(1..64))
```

The preferred encoding variant for *DirectoryString* is UTF8 subset with only ANSI/ISO 8859-1 characters (Unicode Latin-1 page) (see also notes in C.2.5).

NOTE:

Personal data information, e.g. the gender of a person, may be specified in the *subjectDirectoryAttributes* certificate extension (in the *gender* attribute, see C.2.9.5).

C.2.7.4 Serial number

This field contains a serial number assigned by the CA, if necessary, for achieving a unique name of the subject.

ASN.1 definition of attribute type *serialNumber*:

```
id-at-serialNumber AttributeType ::= { id-at 5 }
SerialNumber          ::= PRINTABLE STRING (SIZE(1..64))
```

C.2.8 Subject public key info

This field contains the public key of the certificate holder, i.e. the health professional, together with the identifier of the related algorithm.

ASN.1 definition of type *SubjectPublicKeyInfo*:

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm           AlgorithmIdentifier,
    subjectPublicKey    BIT STRING }

AlgorithmIdentifier ::= SEQUENCE {
    algorithm          OBJECT IDENTIFIER,
    parameters        ANY DEFINED BY algorithm OPTIONAL }
```

The OID for the algorithm can denote in principle

- the signature algorithm (mandatory),
- the hash algorithm (conditional), and
- the padding format, i.e. the DSI format (conditional).

A signature verifier needs the complete information, but not necessarily denoted by the OID in the certificate. In specific environments (e.g. Internet) the signature algorithm, the DSI format and the hash algorithm may be (additionally) indicated in the message or together with the message. However, since the availability of this information can not be guaranteed (especially when storing only a signed document which does not contain this information), a system design cannot rely on that. Furthermore, this additional information would be unprotected if not covered by the user's signature. Therefore this information has no relevance for the OID content in a certificate.

Relevant for the HPC is

- the signature algorithm RSA,
- the hash algorithms SHA-1, and
- the padding formats, i.e. the DSI formats PKCS#1 and ISO/IEC 9796-2rnd.

In a verification process first the signature algorithm has to be known and applied, i.e. the signature algorithm has to be denoted in the OID in the certificate.

In the case of RSA, after retransformation of the signature the padding scheme has to be applied in the decoding process. Since the defined padding schemes can be easily distinguished e.g. by looking on the first byte, there is no need for indication in the OID.

If in the DSI the hash algorithm is indicated, then the signed message can be hashed by applying the respective hash function and the computed hash value compared with the hash value as part of the DSI. In this case the OID in the certificate shall not indicate the hash algorithm to allow flexibility. The hash algorithm indication is supported by PKCS#1 (the digestinfo of PKCS#1 contains the OID of the hash algorithm) and the hash algorithm indication is also provided in the DSI format ISO/IEC 9796-2rnd, when the trailer of 2 bytes is used.

If in the DSI the hash algorithm is not denoted (this is true for the DSI-format ISO/IEC 9796-2rnd with trailer 'BC'), then the OID in the certificate shall denote the hash algorithm in addition to the signature algorithm.

The consequence is:

- in an HPC with RSA and DSI = PKCS#1, the OID has to denote only RSA
- in an HPC with RSA and DSI = ISO/IEC 9796-2rnd (trailer 'BC'), the OID has to denote RSA and the hash algorithm SHA-1 or RIPEMD-160
- in an HPC supporting the DSI-formats PKCS#1 and ISO/IEC 9796-2rnd (trailer 'BC'), the OID has to denote RSA and the hash algorithm SHA-1 or RIPEMD-160 (DSI = PKCS#1 or ISO/IEC 9796-2rnd (trailer 'BC')). If PKCS#1 is applied then the hash algorithm indicated in the OID has to be used.

C.2.9 Extensions

Extensions are optional in X.509v3 and can be used for the inclusion of additional information. An extension needs always an object identifier to identify the type of the respective extension. The importance of an extension has to be indicated by its boolean criticality flag „true“ or „false“. „True“ means, that the extension MUST be known and processable by an application, or the complete certificate has to be rejected during the verification. An extension value is defined as string of octets.

```
Extensions ::= SEQUENCE (1..MAX) OF Extension
```

```
Extension ::= SEQUENCE {
    extnId          OBJECT IDENTIFIER,
    critical        BOOLEAN DEFAULT FALSE,
    extnValue       OCTET STRING }
```

For HP DS-certificates according to this specification, the following categories of certificate extensions are relevant:

- X.509 basic extensions,
- private X.509 extensions for qualified certificates [RFC3039], and
- private X.509 extensions from [ISIS-MTT OP].

X.509 basic extensions include:

- basic constraints,
- key usage,
- certificate policies,
- subject alternative name,
- authority key identifier,
- subject key identifier, and
- subject directory attributes.

These basic X.509 certificate extensions are identified by the initial object identifier arc

```
id-ce OBJECT IDENTIFIER ::= { 2 5 29 }
```

Private X.509 extension for qualified certificates:

- qualified certificate statements

Private X.509 certificate extensions are identified by the initial object identifier arc

```
id-pe OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 1 }
```

ISIS-MTT SigG Private X.509 Extensions:

- admission

ISIS-MTT SigG private X.509 certificate extensions are identified by the initial object identifier arc

```
id-isismtt OBJECT IDENTIFIER ::= { 1 3 36 8 }
id-isismtt-at OBJECT IDENTIFIER ::= { id-isismtt 3 }
```

C.2.9.1 Basic Constraints

This field specifies constraints relevant to the certificate holder. It is required to indicate that an HP is not allowed to issue certificates, i.e. that he cannot perform the role of a CA. The basic constraints
HPC Pharmacist & Physician V2.0

extension, if used in an HP certificate MUST be marked critical, and the value of the *cA* component MUST be set to FALSE.

```
id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
```

```
BasicConstraints ::= SEQUENCE {  
    cA BOOLEAN DEFAULT FALSE,  
    pathLenConstraint INTEGER (0..MAX) OPTIONAL }
```

C.2.9.2 Key usage

This field specifies the usage of the PK certified.

ASN.1-Definition of type *KeyUsage*:

```
id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }
```

```
KeyUsage ::= BIT STRING {  
    digitalSignature (0),  
    nonRepudiation (1),  
    keyEncipherment (2),  
    dataEncipherment (3),  
    keyAgreement (4),  
    keyCertSign (5),  
    cRLSign (6),  
    encipherOnly (7),  
    decipherOnly (8) }
```

For HP ES-certificate, only *nonRepudiation* is relevant and has to be set. The key usage *digitalSignature* is valid for AUT-certificates, which are used in authentication procedures with digital signatures in the technical sense and not in the legal sense as a signature of a person. The extension MUST be marked critical.

C.2.9.3 Certificate Policies

This field specifies the certificate policy that was applied when issuing the certificate. The extension MUST be present, and SHOULD NOT be marked critical. The values have to be set complying with [ISIS-MTT P1]. Legacy systems use the *CertificatePolicies* extension to mark qualified certificates and to recognize this fact in components. For the sake of *vertical interoperability*, these extensions SHOULD NOT be marked critical, in spite of the fact that their contents restrict the usability of the certificate in some way. As these informations are extremely relevant in verifying the legal validity of the signature, SigG-conforming components MUST evaluate them.

ASN.1-Definition of type *CertificatePolicies*:

```
id-ce-certificatePolicies OBJECT IDENTIFIER ::= { id-ce 32 }
```

```
CertificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation
```

```
PolicyInformation ::= SEQUENCE {  
    policyIdentifier CertPolicyId,  
    policyQualifiers SEQUENCE SIZE (1..MAX) OF  
        PolicyQualifierInfo OPTIONAL }
```

```
PolicyQualifierInfo ::= SEQUENCE {  
    policyQualifierId PolicyQualifierId,  
    qualifier ANY DEFINED BY policyQualifierId }
```

```

CertPolicyId          ::= OBJECT IDENTIFIER
-- isis-mtt branch for certificate policies
id-isismtt-cp OBJECT IDENTIFIER ::= { id-isismtt 1 }
id-isismtt-cp-sigGconform OBJECT IDENTIFIER ::= { id-isismtt-cp 1 }

```

C.2.9.4 Subject Alternative Name

This field contains one or more „alternative technical names“ of the certificate holder such as a unique e-mail address or a unique www-address denoting the homepage of the respective user. Since the *subject* field uniquely identifies the card holder, the *SubjectAltNames* extension SHOULD NOT be marked critical.

ASN.1-Definition of Type *SubjectAltName*:

```

id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }

SubjectAltName ::= GeneralNames

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
    otherName          [0] IMPLICIT OtherName,
    rfc822Name         [1] IMPLICIT IA5String,
    dNSName            [2] IMPLICIT IA5String,
    x400Address        [3] IMPLICIT ORAddress,
    directoryName      [4] EXPLICIT Name,
    ediPartyName       [5] IMPLICIT EDIPartyName,
    uniformResourceIdentifier [6] IMPLICIT IA5String,
    iPAddress          [7] IMPLICIT OCTET STRING,
    registeredID       [8] IMPLICIT OBJECT IDENTIFIER }

OtherName ::= SEQUENCE {
    type-id          OBJECT IDENTIFIER
    value            [0] EXPLICIT ANY DEFINED type-id }

ORAddress ::= SEQUENCE {
    built-in-standard-attributes ANY,
    built-in-domain-defined-attributes SEQUENCE OF ANY OPTIONAL,
    extension-attributes SET OF ANY OPTIONAL }

EDIPartyName ::= SEQUENCE {
    NameAssigner     [0] EXPLICIT DirectoryString OPTIONAL,
    partyName        [1] EXPLICIT DirectoryString }

```

RFC822 is the convention, in which e-mail addresses for Internet are defined.

Example of e-mail address:

```
"fritz.meyer@kvb.de"
```

C.2.9.5 Subject Directory Attributes

This extension may contain further X.500 attributes of the card holder. Qualified certificates MAY store legal identification data (e.g. of a personal identification card, passport or similar) in this extension. This optional extension MUST NOT be marked critical.

ASN.1-Definition of Type *SubjectDirectoryAttributes*:

```
id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::= { id-ce 9 }
SubjectDirectoryAttributes ::= Attributes
```

Standard Attributes

Title Attribute

Optional attribute, defined by X.520 that contains the value of the card holder's title.

```
id-at OBJECT IDENTIFIER ::= { 2 4 5 }
id-at-title OBJECT IDENTIFIER ::= { id-at 12 }
Title ::= DirectoryString (SIZE(1..64))
```

QC Private Extensions [RFC3039] for personal data are identified by the initial object identifier arc

```
id-pkix OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 }
id-pda OBJECT IDENTIFIER ::= { id-pkix 9 }
```

Date of Birth Attribute

Optional attribute that contains the value of the date of birth of the card holder.

```
id-pda-dateOfBirth AttributeType ::= { id-pda 1 }
DateOfBirth ::= GeneralizedTime
```

Place of Birth Attribute

Optional attribute that contains the value of the place of birth of the card holder.

```
id-pda-placeOfBirth AttributeType ::= { id-pda 2 }
PlaceOfBirth ::= DirectoryString (SIZE(1..128))
```

Gender Attribute

Optional attribute that contains the value of the gender of the card holder.

```
id-pda-gender AttributeType ::= { id-pda 3 }
Gender ::= PrintableString (SIZE(1)) - "M", "F", "m", "f"
```

Country of Citizenship Attribute

Optional attribute that contains the value of the card holder's claimed countries of citizenship at the time when the certificate was issued.

```
id-pda-countryOfCitizenship AttributeType ::= { id-pda 4 }
CountryOfCitizenship ::= PrintableString (SIZE(2)) - ISO 3166
```

Country of Residence Attribute

Optional attribute that contains the value of the card holder's claimed countries in which the holder is a resident.

```
id-pda-countryOfResidence AttributeType ::= { id-pda 5 }
CountryOfResidence ::= PrintableString (SIZE(2)) - ISO 3166
```

Name at Birth Attribute

Optional attribute, defined by [ISIS-MTT P1], that contains the value of the card holder's name at his/her birth.

```
id-isismtt-at-nameAtBirth OBJECT IDENTIFIER ::= { id-isismtt-at 14 }
NameAtBirth ::= DirectoryString (SIZE(1..64))
```

C.2.9.6 Authority Key Identifier

This field is mandatory according to [ISIS-MTT P1], and denotes the reference to the PK of the CA to be applied for verification of the CA signature. This extension MUST NOT be marked critical.

ASN.1-Definition of Type *AuthorityKeyIdentifier*:

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier [0] IMPLICIT KeyIdentifier OPTIONAL,
    authorityCertIssuer [1] IMPLICIT GeneralNames OPTIONAL,
    authorityCertSerialNumber [2] IMPLICIT CertificateSerialNumber
        OPTIONAL }
KeyIdentifier ::= OCTET STRING
```

Notes:

Both identification methods, described in [RFC2459] MAY be used in the same certificate. [ISIS-MTT P1] requires that the *keyIdentifier* field MUST contain exactly the same ID as the *subjectKeyIdentifier* of the CA certificate (see C.2.9.7 below). If *authorityCertIssuer* is present, it MUST contain exactly one *directoryName* element filled with the *subject* DName of the issuing CA certificate.

C.2.9.7 Subject Key Identifier

This field denotes a reference to the PK of the certificate holder and is optional in ISIS-MTT, however recommended to use. The subject key Identifier in form of an ICCSN (in combination with key usage) may be used in HPCs supporting signature verification on the basis of stored PKs of defined partners. The extension MUST NOT be marked critical.

ASN.1-Definition of Type *SubjectKeyIdentifier*:

```
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 14 }
SubjectKeyIdentifier ::= OCTET STRING
```

C.2.9.8 Qualified Certificate Statements

This field provides statements to indicate the fact that the certificate is a qualified certificate in accordance with a particular legal system. Based on the argumentation presented for certificate policies (see C.2.9.3) the extension SHOULD NOT be marked critical.

ASN.1-Definition of Type *QCStatements*

```
id-pe-qcStatements OBJECT IDENTIFIER ::= { id-pe 3 }
QCStatements ::= SEQUENCE OF QCStatement

QCStatement ::= SEQUENCE {
    statementId OBJECT IDENTIFIER,
    statementInfo ANY DEFINED BY statementId OPTIONAL }

```

Predefined QC Statement [RFC3039]

```
id-qcs OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 11 }
id-qcs-pkixQCSyntax-v1 OBJECT IDENTIFIER ::= { id-qcs 1 }
```

This OID is to be used as *statementId* indicating conformance with the syntax and semantics defined in [RFC3039]. It refers to the data type *SemanticsInformation* below.

```
SemanticsInformation ::= SEQUENCE {
```

```
semanticsIdentifier      OBJECT IDENTIFIER OPTIONAL,
nameRegistrationAuthorities NameRegistrationAuthorities OPTIONAL }
```

```
NameRegistrationAuthorities ::= SEQUENCE SIZE(1..MAX) OF GeneralName
```

The *semanticsIdentifier* SHALL contain an OID defining semantics for attributes and names in certificate fields.

The *nameRegistrationAuthorities* SHALL contain a name of one or more registration authorities responsible for registration of attributes and names associated with the subject.

Some *registeredID* of the semantics or of a certificate policy may occur here.

NOTE:

[RFC3039] requires that at least one of *semanticsIdentifier* and *nameRegistrationAuthorities* must be present.

ETSI QC Statements

Compliance with the EU directive

```
id-etsi-qcs OBJECT IDENTIFIER ::= { 0 4 0 1862 1 }
id-etsi-qcs-QcCompliance OBJECT IDENTIFIER ::= {id-etsi-qcs 1}
```

This OID is to be used as *statementId*, in order to indicate that the certificate has been issued in accordance with the EU-directive [ECDIR] as implemented in the country under which law the issuer CA operates. When inserting this OID, the *statementInfo* field is to be omitted.

Limitation of value of transaction

```
id-etsi-qcs-QcLimitValue OBJECT IDENTIFIER ::= {id-etsi-qcs 2}
```

This OID is to be used as *statementId* in conjunction with the *QcEuLimitValue* statement below.

```
QcEuLimitValue ::= MonetaryValue

MonetaryValue ::= SEQUENCE {
    currency      Iso4217CurrencyCode,
    amount        INTEGER,
    exponent      INTEGER }

Iso4217CurrencyCode ::= CHOICE {
    alphabetic    PrintableString,
    numeric       INTEGER(1..999) }
```

The limit value of an transaction is given by $\text{amount} * 10^{\text{exponent}}$

Retention Period

CAs or a relevant name registration authority retain external information about the owner of qualified certificates. This information allows identifying the physical person in case of dispute.

```
id-etsi-qcs-QcRetentionPeriod OBJECT IDENTIFIER ::= {id-etsi-qcs 3}
```

This OID is to be used as *statementId* in conjunction with the *QcRetentionPeriod* statement below.

```
QcRetentionPeriod ::= INTEGER
```

This field indicates how many years after the expiry date of the certificate such information will be retained.

C.2.9.9 CRL Distribution Points

This field identifies how CRL information can be obtained. The extension should be marked non-critical.

ASN.1-Definition of Type *CRLDistributionPoints*

```

id-ce-cRLDistributionPoints OBJECT IDENTIFIER ::= { id-ce 31 }
CRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::= SEQUENCE {
    distributionPoint [0] DistributionPointName OPTIONAL,
    reasons [1] ReasonFlags OPTIONAL,
    cRLIssuer GeneralNames OPTIONAL }

DistributionPointName ::= CHOICE {
    fullName [0] GeneralNames,
    nameRelativeToCRLIssuer RelativeDistinguishedName }

ReasonFlags ::= BIT STRING {
    unused (0),
    keyCompromise (1),
    cACompromise (2),
    affiliationChanged (3),
    superseded (4),
    cessationOfOperation (5),
    certificateHold (6),
    privilegeWithdrawn (7),
    aACompromise (8) }

```

This field is mandatory according to [ISIS-MTT P1], if indirect CRLs issued, and it should be present if direct CRLs are issued.

C.2.9.10 Professional Admission

Professional admission information can be expressed either in the base ES certificate by means of the [ISIS-MTT OP] private extension *admission* of the type *AdmissionSyntax*, or in one or more attribute certificates by means of the attribute *admission*, also of the type *AdmissionSyntax*, defined by [ISIS-MTT OP]. The decision where this information has to be included is a matter of the authorities of professional groups.

ASN.1 Definition of type *AdmissionSyntax*:

```

id-isismtt-at-admission OBJECT IDENTIFIER ::= { isismtt-at 3 }
id-isismtt-at-namingAuthorities OBJECT IDENTIFIER ::= { isismtt-at 11 }

AdmissionSyntax ::= SEQUENCE {
    admissionAuthority GeneralName OPTIONAL,
    contentsOfAdmissions SEQUENCE OF Admissions}

Admissions ::= SEQUENCE {
    admissionAuthority [0] EXPLICIT GeneralName OPTIONAL,
    namingAuthority [1] EXPLICIT NamingAuthority OPTIONAL,
    professionInfos SEQUENCE OF ProfessionInfo}

NamingAuthority ::= SEQUENCE {
    namingAuthorityId OBJECT IDENTIFIER OPTIONAL,
    namingAuthorityUrl IA5String OPTIONAL,
    namingAuthorityText DirectoryString (SIZE(1..128)) OPTIONAL }

```

```

ProfessionInfo          ::= SEQUENCE {
    namingAuthority      [0] EXPLICIT NamingAuthority OPTIONAL,
    professionItems      SEQUENCE OF DirectoryString (SIZE(1..128)),
    professionOIDs       SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
    registrationNumber   PrintableString (SIZE(1..128)) OPTIONAL,
    addProfessionInfo    OCTET STRING OPTIONAL }

```

The components of the relatively complex structure of *AdmissionSyntax* supports the following concepts and requirements:

- The data field *admissionAuthority* is used to denote the entity responsible for the assignment of the values of the related components. Admission authorities are institutions (e.g. professional associations, chambers, unions, administrative bodies, companies, etc.), which are responsible for granting and verifying professional admissions. An admission authority is indicated by a *GeneralName* object (see C.2.9.4).
- The data field *namingAuthority* is used to provide the names of authorities which are responsible for the administration of title registers. Naming authorities define code lists and allowed values for specific professional groups. The name of the authority can be identified by an object identifier in the field *namingAuthorityId*, by means of a text string in the field *namingAuthorityText*, by means of a URL address in the field *namingAuthorityUrl*, or by a combination of them. For example, the text string can contain the name of the authority, the country and the name of the title register. The URL-option refers to a web page which contains lists with „officially“ registered professions (text and possibly OID) as well as further information on these professions. Object identifiers for the component *namingAuthorityId* are grouped under the OID-arc *id-isis-at-namingAuthorities*. The registration of new OIDs must be applied for at TeleTruST.
- The data type *ProfessionInfo* is used to specify certain professions, specializations, disciplines, or fields of activity. A profession is represented by one or more text strings, resp. profession OIDs in the fields *professionItems* and *professionOIDs*, and by a registration number in the field *registrationNumber*. An indication in text form must always be present, whereas the other indications are optional. The component *addProfessionInfo* may contain additional application-specific information in DER-encoded form.

By means of different *namingAuthority*-OIDs or profession OIDs hierarchies of professions, specializations, disciplines, fields of activity, etc. can be expressed. The issuing admission authority should always be indicated in the field *admissionAuthority*, whenever a registration number is presented. However, information on admissions can be given without indicating an admission or a naming authority by the exclusive use of the component *professionItems*. In this case the certification authority is responsible for the verification of the admission information.

This professional *admission* attribute is *single-valued*. However, several admissions can be captured in the sequence structure of the component *contentsOfAdmissions* of *AdmissionSyntax* or in the component *professionInfos* of *Admissions*.

The component *admissionAuthority* of *AdmissionSyntax* serves as default value for the component *admissionAuthority* of *Admissions*. Within the latter component the default value can be overwritten, in case that another authority is responsible.

The component *namingAuthority* of *Admissions* serves as a default value for the component *namingAuthority* of *ProfessionInfo*. Within the latter component the default value can be overwritten, in case that another naming authority needs to be recorded.

The length of the string objects is limited to 128 characters. It is recommended to indicate a *namingAuthorityURL* in all issued attribute certificates. If a *namingAuthorityURL* is indicated, the field *professionItems* of *ProfessionInfo* should contain only registered titles. If the field *professionOIDs* exists, it has to contain the OIDs of the professions listed in *professionItems* in the same order. In general, the field *professionInfos* should contain only one entry, unless the admissions that are to be listed are logically connected (e.g. they have been issued under the same admission number).

Profession OIDs should always be defined under the OID branch of the responsible naming authority.

As already mentioned before, the decision where the profession information has to included is a matter of the authorities of professional groups. The following minimum requirements have been specified by the professional groups *physicians* and *pharmacists*.

Professional group "physicians":

In the base ES certificate it shall be expressed that the professional is a physician. The minimum information to be provided in the *admission* extension of the base ES certificate is "PHYSICIAN" in the component *professionItems* of *AdmissionSyntax*.

```
{ ..., professionItems {"PHYSICIAN"}, ... }
```

Note: Further professional information has to be provided in an attribute certificate for qualifications (see C.7.1), and in an attribute certificate for authorizations (see C.7.2).

Professional group "pharmacists":

In the base ES certificate the qualification of the card holder shall be included, i.e. it shall be expressed that the professional either is a "Apotheker", "Apothekerassistent", "Pharmazieingenieur", "PTA", "PKA/Helferin", "Apothekenassistent", "Pharmazeutische Assistentin", "Apothekenfacharbeiter", "Pharmaziepraktikant", "Stud.pharm. oder Famulant", "PTA-Praktikanten" or "Azubis".

For example, the minimum information to be provided for an "Apotheker" (pharmacist) is "Apotheker" in *professionItems*.

```
{ ..., professionItems {"Apotheker"}, ... }
```

Note: Only the group members "Apotheker", "Apothekerassistent", "Apothekenassistent", "PTA" and "Pharmazieingenieur" are allowed to digitally sign electronic prescriptions. However, all group members are allowed to use the signature function for purposes other than signing electronic prescriptions. Further professional information can also be provided either in this extension of the base ES certificate, or in the *admission* attribute in one or more attribute certificates (see C.7).

Besides specifying the qualification of a card holder, the professional group "pharmacists" also has defined the following categories for further professional information:

- category "Tätigkeitsbereich" (field of activity),
- category "berufliche Rolle" (professional role),
- category "Fachapothekereigenschaft" (speciality type),
- category "Zusatzqualifikationen zu Fachapothekereigenschaft" (dedicated specialty type),
- category "Fach-PTA-Eigenschaft" (PTA specialty type), and
- category "Zugehörigkeit zu Kammern und Fachorganisationen" (membership in chambers and special organizations).

Concrete values of the different categories that are allowed to be used in the admission extension of the base ES certificate will be described in a separate document.

The following example shall illustrate how detailed professional information can be included in the *admission* extension of the base ES certificate. Note, that this information can also be included in the *admission* attribute of attribute certificates.

```
{ admissionAuthority "NameOfAuthorityResponsibleForCompleteContent",
  contentsOfAdmissions {
    admission_1 {
      admissionAuthority "NameOfAuthorityResponsibleForContentOfAdmission1",
      namingAuthority {
        namingAuthorityId id-OIDOfNamingAuthority,
        namingAuthorityUrl "URLToListOfQualifications",
        namingAuthorityText "NameOfAuthority" },
      professionInfos {
        professionInfo {
          professionItems { "Apotheker" } } } },
    admission_2 {
      professionInfos {
        professionInfo {
          namingAuthority {
            namingAuthorityUrl "URLToListOfProfessionalRole" },
          professionItems {"Apothekenleiter" } } } } }
```

C.3 Signature algorithm

This field contains the same coding as the signature field in the to be signed sequence.

C.4 Signature

This field contains the signature of the CA which issues the certificate.

C.5 ES certificate in table form and in complete syntax form

The following table shows the ES certificate in table form. Only fields relevant to a ES certificate for a health professional are listed. In such a certificate all fields classified as mandatory according to ISIS-MTT are also qualified as mandatory for a HP-ES certificate. A cross in column ISIS-MTT and HPC indicates that the respective field is mandatory.

Certificate Field	Content	ISISMTT	HPC
version	X.509v3	X	X
serialNumber	Serial number of certificate	X	X
signature	Algorithm identifier for CA signature	X	X
issuer	Certification authority	X	X
countryName	DE for Germany	X	X
organizationName	DEZGW for German CA in health care	X	X
validity	Validation period	X	X
notBefore	Generalized Time	X	X
notAfter	Generalized Time	X	X
subject	Certificate holder	X	X
countryName	DE for Germany	X	X
stateOrProvince	Name of German state	X	X
commonName (CN)	Common name in its full form	X	X
serialNumber	Serial number for unique naming of subject	X	X
subjectPublicKeyInfo	PK data	X	X
algorithm	OID of algorithm incl. parameters if any	X	X
subjectPublicKey	Coding of PK with modulus and publicExponent	X	X
extensions	Extensions	X	X
basicConstraints	Classification as end user certificate	X	X
keyUsage	nonRepudiation, i.e. usage of certificate restricted to digital signatures according to SigG requirement	X	X
certificatePolicies	Indication of SigG Conformance	X	X
subjectAltName	alternative technical name with possibly e-Mail address of certificate holder	X	X
subjectDirectoryAttributes	X.500 attributes for additional personal data as for example date of birth, place of birth, gender, country of citizenship, country of residence, and name at birth	X	X
authorityKeyIdentifier	Reference to PK of the CA for verification of the CA signature	X	X
subjectKeyIdentifier	Reference to PK of the certificate holder	X	X
qcStatements	Indication that certificate is a qualified certificate	X	X
cRLDistributionPoints	Identification of how CRL information can be obtained	X	X
admission	Professional admission information	X	X
signatureAlgorithm	Algorithm identifier for CA signature (Value identical to signature field in Certificate Content)	X	X
signature	Signature of CA	X	X

Tab. C.1: Content of the ES-certificate of the health professional

The complete syntax form of the ES public key certificate is:

```
-- certificate
```

```

Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signature           BIT STRING }

-- to be signed certificate
TBSCertificate ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    serialNumber        CertificateSerialNumber,
    signature           AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID      [1] IMPLICIT UniqueIdentifier OPTIONAL,
    subjectUniqueID     [2] IMPLICIT UniqueIdentifier OPTIONAL,
    extensions          [3] EXPLICIT Extensions Optional }

-- version
Version ::= INTEGER { v1(0), v2(1), v3(2) }

-- serial number
CertificateSerialNumber ::= INTEGER

-- signature
AlgorithmIdentifier ::= SEQUENCE {
    algorithm           OBJECT IDENTIFIER,
    parameters         ANY DEFINED BY algorithm OPTIONAL }

-- issuer
Name ::= CHOICE { RDNSSequence }

RDNSSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
    type               AttributeType,
    value              AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType

-- directory string
DirectoryString ::= CHOICE {
    printableString    PrintableString (SIZE (1..maxSize))
    teletexString      TeletexString (SIZE (1..maxSize))
    utf8String         UTF8String (SIZE (1..maxSize))
    bmpString          BMPString (SIZE (1..maxSize))
    universalString    UniversalString (SIZE (1..maxSize)) }

-- country name attribute
CountryName ::= PRINTABLE STRING (SIZE(2))

-- organization name attribute
OrganizationName ::= DirectoryString (SIZE(1..64))

-- validity

```

```

Validity                               ::= SEQUENCE {
    notBefore      Time,
    notAfter       Time }

Time                                    ::= CHOICE {
    utcTime        UTCTime,
    generalizedTime GeneralizedTime }

-- state or province name attribute
StateOrProvinceName ::= DirectoryString (SIZE(1..128))

-- common name attribute
CommonName           ::= DirectoryString (SIZE(1..64))

-- serial number attribute
SerialNumber         ::= PRINTABLE STRING (SIZE(1..64))

-- subject public key info
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm      AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

-- extensions
Extensions           ::= SEQUENCE (1..MAX) OF Extension
Extension             ::= SEQUENCE {
    extnId         OBJECT IDENTIFIER,
    critical       BOOLEAN DEFAULT FALSE,
    extnValue      OCTET STRING }

-- basic constraints extension
BasicConstraints     ::= SEQUENCE {
    cA             BOOLEAN DEFAULT FALSE,
    pathLenConstraint INTEGER (0..MAX) OPTIONAL }

-- key usage extension
KeyUsage             ::= BIT STRING {
    digitalSignature      (0),
    nonRepudiation       (1),
    keyEncipherment      (2),
    dataEncipherment     (3),
    keyAgreement         (4),
    keyCertSign          (5),
    cRLSign              (6),
    encipherOnly         (7),
    decipherOnly         (8) }

-- certificate policies extension
CertificatePolicies  ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation    ::= SEQUENCE {
    policyIdentifier    CertPolicyId,
    policyQualifiers    SEQUENCE SIZE (1..MAX) OF
        PolicyQualifierInfo OPTIONAL}

PolicyQualifierInfo ::= SEQUENCE {
    policyQualifierId    PolicyQualifierId,
    qualifier            ANY DEFINED BY policyQualifierId }

```

```

CertPolicyId          ::= OBJECT IDENTIFIER

-- subject alternative name extension
SubjectAltName        ::= GeneralNames

GeneralNames          ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName           ::= CHOICE {
    otherName          [0] IMPLICIT OtherName,
    rfc822Name         [1] IMPLICIT IA5String,
    dNSName            [2] IMPLICIT IA5String,
    x400Address        [3] IMPLICIT ORAddress,
    directoryName      [4] EXPLICIT Name,
    ediPartyName       [5] IMPLICIT EDIPartyName,
    uniformResourceIdentifier [6] IMPLICIT IA5String,
    iPAddress          [7] IMPLICIT OCTET STRING,
    registeredID       [8] IMPLICIT OBJECT IDENTIFIER }

OtherName             ::= SEQUENCE {
    type-id            OBJECT IDENTIFIER
    value              [0] EXPLICIT ANY DEFINED type-id }

ORAddress             ::= SEQUENCE {
    built-in-standard-attributes ANY,
    built-in-domain-defined-attributes SEQUENCE OF ANY OPTIONAL,
    extension-attributes SET OF ANY OPTIONAL }

EDIPartyName         ::= SEQUENCE {
    nameAssigner [0] EXPLICIT DirectoryString OPTIONAL,
    partyName [1] EXPLICIT DirectoryString }

-- subject directory attributes extension
SubjectDirectoryAttributes ::= Attributes

Title                ::= DirectoryString (SIZE(1..64))

DateOfBirth          ::= GeneralizedTime

PlaceOfBirth         ::= DirectoryString (SIZE(1..128))

Gender               ::= PrintableString (SIZE(1)) - "M", "F", "m", "f"

CountryOfCitizenship ::= PrintableString (SIZE(2)) - ISO 3166

CountryOfResidence  ::= PrintableString (SIZE(2)) - ISO 3166

NameAtBirth         ::= DirectoryString (SIZE(1..64))

-- authority key identifier extension
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier      [0] IMPLICIT KeyIdentifier OPTIONAL,
    authorityCertIssuer [1] IMPLICIT GeneralNames OPTIONAL,
    authorityCertSerialNumber [2] IMPLICIT
        CertificateSerialNumber OPTIONAL}

KeyIdentifier        ::= OCTET STRING

-- subject key identifier extension
SubjectKeyIdentifier ::= OCTET STRING

```

```

-- qualified certificate statements extension
QCStatements ::= SEQUENCE OF QCStatement

QCStatement ::= SEQUENCE {
    statementId OBJECT IDENTIFIER,
    statementInfo ANY DEFINED BY statementId OPTIONAL }

-- predefined RFC3039 qualified certificate statements
SemanticsInformation ::= SEQUENCE {
    semanticsIdentifier OBJECT IDENTIFIER OPTIONAL,
    nameRegistrationAuthorities NameRegistrationAuthorities OPTIONAL }

NameRegistrationAuthorities ::= SEQUENCE SIZE(1..MAX) OF GeneralName

-- ETSI QC statement on limitation of transaction value
QcEuLimitValue ::= MonetaryValue

MonetaryValue ::= SEQUENCE {
    currency Iso4217CurrencyCode,
    amount INTEGER,
    exponent INTEGER }

Iso4217CurrencyCode ::= CHOICE {
    alphabetic PrintableString,
    numeric INTEGER(1..999) }

-- ETSI QC statement on retention period
QcRetentionPeriod ::= INTEGER

-- CRL distribution points extension
id-ce-cRLDistributionPoints OBJECT IDENTIFIER ::= { id-ce 31 }
CRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::= SEQUENCE {
    distributionPoint [0] DistributionPointName OPTIONAL,
    reasons [1] ReasonFlags OPTIONAL,
    cRLIssuer [2] GeneralNames OPTIONAL }

DistributionPointName ::= CHOICE {
    fullName [0] GeneralNames,
    nameRelativeToCRLIssuer [1] RelativeDistinguishedName }

ReasonFlags ::= BIT STRING {
    unused (0),
    keyCompromise (1),
    cACompromise (2),
    affiliationChanged (3),
    superseded (4),
    cessationOfOperation (5),
    certificateHold (6),
    privilegeWithdrawn (7),
    aACompromise (8) }

-- ISIS-MTT private extension/attribute admission
AdmissionSyntax ::= SEQUENCE {
    admissionAuthority GeneralName OPTIONAL,
    contentsOfAdmissions SEQUENCE OF Admissions}

```

```

Admissions ::= SEQUENCE {
    admissionAuthority [0] EXPLICIT GeneralName OPTIONAL,
    namingAuthority [1] EXPLICIT NamingAuthority OPTIONAL,
    professionInfos SEQUENCE OF ProfessionInfo}

NamingAuthority ::= SEQUENCE {
    NamingAuthorityId OBJECT IDENTIFIER OPTIONAL,
    namingAuthorityUrl IA5String OPTIONAL,
    namingAuthorityText DirectoryString (SIZE(1..128)) OPTIONAL }

ProfessionInfo ::= SEQUENCE {
    namingAuthority [0] EXPLICIT NamingAuthority OPTIONAL,
    professionItems SEQUENCE OF DirectoryString (SIZE(1..128)),
    professionOIDS SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
    registrationNumber PrintableString (SIZE(1..128)) OPTIONAL,
    addProfessionInfo OCTET STRING OPTIONAL }

```

C.6 Object Identifiers Used in this Specification

Object Identifier		Meaning
Name	Value	
id-at	{ 2 5 4 }	arc for X.500 attributes
id-at-title	{ 2 5 4 12 }	title
id-ce	{ 2 5 29 }	arc for certificate extensions
id-ce-subjectDirectoryAttributes	{ id-ce 9 }	provides additional X.500 attributes of the card holder to be used for legal identification data
id-ce-subjectKeyIdentifier	{ id-ce 14 }	identifies the card holder's certificate that contains a specific public key
id-ce-keyUsage	{ id-ce 15 }	defines the purpose of the private key to be used only for signature generation corresponding to non-repudiation service
id-ce-subjectAltName	{ id-ce 17 }	indicates alternative technical names of the HP card holders
id-ce-basicConstraints	{ id-ce 19 }	indicates that the HP card holder is not allowed to perform the role of a CA
id-ce-certificatePolicies	{ id-ce 32 }	indicates the policy under which the certificate has been issued, and the purpose for which it is to be used
id-ce-authorityKeyIdentifier	{ id-ce 35 }	identifies the public key of the issuing CA
id-pkix	{ 1 3 6 1 5 5 7 }	arc for pkix
id-pe	{ id-pkix 1 }	arc for private certificate extensions
id-pe-qcStatements	{ id-pe 3 }	indicates that the certificate is a qualified certificate in accordance with a particular legal system
id-pda	{ id-pkix 9 }	arc for QC personal data attributes
id-pda-dateOfBirth	{ id-pda 1 }	date of birth
id-pda-placeOfBirth	{ id-pda 2 }	place of birth
id-pda-gender	{ id-pda 3 }	gender
id-pda-countryOfCitizenship	{ id-pda 4 }	country of citizenship
id-pda-countryOfResidence	{ id-pda 5 }	country of residence
id-isismtt	{ 1 3 36 8 }	arc for ISIS-MTT
id-qcs	{ id-pkix 11 }	arc for QC statements
id-qcs-pkixQCSyntax-v1	{ id-qcs 1 }	RFC3039 QC statement
id-isismtt-at	{ id-isismtt 3 }	arc for ISIS-MTT attributes and extensions
id-isismtt-at-admission	{ id-isismtt-at 3 }	Indicates professional admission information
id-isismtt-at-namingAuthorities	{ id-isismtt-at 11 }	arc for naming authorities for professions
id-isismtt-at-nameAtBirth	{ id-isismtt-at 14 }	name at birth
id-isismtt-cp	{ id-isismtt 1 }	arc for ISIS-MTT certificate policies
id-isismtt-cp-sigGconform	{ id-isismtt-cp 1 }	indicates that the certificate is a qualified certificate according to [ECDIR], complies with the special requirements of SigG, and has been issued by an accredited CA
id-etsi-qcs	{ 0 4 0 1862 1 }	arc for ETSI QC statements
id-etsi-qcs-QcCompliance	{ id-etsi-qcs 1 }	indicates that the certificate has been issued in accordance with the [ECDIR]
id-etsi-qcs-QcLimitValue	{ id-etsi-qcs 2 }	indicates a limitation of the transaction value in accordance with the [ECDIR]
id-etsi-qcs-QcRetentionPeriod	{ id-etsi-qcs 3 }	indicates how many years after the expiry date of a certificate external information about the card holder of the QC will be retained

Table C.2 – Overview of OIDs

C.7 Electronic signature attribute certificate (X.509v3 certificate)

Attribute certificates are issued to denote special attributes of the certificate holder not presented in its ES base certificate. If a document is signed by a cardholder and he wants to use one or more of his attribute certificates in a given context, then the attribute certificate shall be integrated in the sequence to be signed at the end of the document.

The general definition of an attribute certificate is

```
AttributeCertificate ::= SEQUENCE {
    tbsAttributeCertificate TBSAttributeCertificate,
    signatureAlgorithm      AlgorithmIdentifier,
    signature               BIT STRING }

TBSAttributeCertificate ::= SEQUENCE {
    version                Version DEFAULT v1,
    subject                CHOICE {
        baseCertificateID [0] IssuerSerial,
        subjectName       [1] GeneralNames },
    issuer                 GeneralNames,
    signature              AlgorithmIdentifier,
    serialNumber           CertificateSerialNumber,
    attrCertValidityPeriod AttCertValidityPeriod,
    attributes             SEQUENCE OF Attribute,
    issuerUniqueID         UniqueIdentifier OPTIONAL,
    extensions             Extensions OPTIONAL }

Attribute ::= SEQUENCE {
    type                   AttributeType,
    values                 SET OF AttributeValue }
```

The reference to the ES base certificate to which the attribute certificate belongs is made by specifying the issuer and the certificate serial number of the ES base certificate.

```
IssuerSerial ::= SEQUENCE {
    issuer                 GeneralNames,
    serial                 CertificateSerialNumber,
    issuerUID              UniqueIdentifier OPTIONAL }

AttCertValidityPeriod ::= SEQUENCE {
    notBeforeTime          GeneralizedTime,
    notAfterTime           GeneralizedTime }
```

Professional admission information can be expressed in an attribute certificate by means of the attribute *admission* of the type *AdmissionSyntax*, defined by [ISIS-MTT OP]. This professional *admission* attribute is *single-valued*. Still, several admissions can be captured in the sequence structure of the component *contentsOfAdmissions* of *AdmissionSyntax* or in the component *professionInfos* of *admissions*. For further details of *AdmissionSyntax* see C.2.9.9.

In the following, requirements for the use of the *admission* attribute in attribute certificates are specified. The attributes for a health professional are related to his professional admissions. The following two categories may be distinguished with respect to the authorities guaranteeing these admissions:

- category "qualifications", and
- category "authorizations".

Note that both kinds of attribute certificates are based on the same structure using the attribute *admission* of the ASN.1 type *AdmissionSyntax*.

The usage and content of attribute certificates and their possible categories with the professional specific admissions will be detailed in separate documents. The two subsequent clauses describe exemplary implementations.

C.7.1 Attribute certificate for qualifications

The attribute certificate for qualifications encodes e.g. the following information:

- profession,
- specialty type, and
- dedicated specialty.

For each specialty type a separate attribute certificate may be issued.

The authority guaranteeing these qualifications is the related chamber. An example of an attribute certificate for qualifications is as follows.

```
{ admissionAuthority "BAYERISCHE LANDESAERZTEKAMMER",
  contentsOfAdmissions {
    { professionInfos {
      { namingAuthority {
          namingAuthorityId id-OIDForBAEK,
          namingAuthorityUrl "UrlToListProfession" },
        professionItems { "Arzt" },
        registrationNumber "10.530" },
      { namingAuthority {
          namingAuthorityId id-OIDForBAEK,
          namingAuthorityUrl "UrlToListSpecialityType" },
        professionItems { "GB" } },
      { namingAuthority {
          namingAuthorityId id-OIDForBAEK,
          namingAuthorityUrl "UrlToListDedicatedSpeciality" },
        professionItems { "Anatomie" } } } } }
```

C.7.2 Attribute certificate for authorizations

The attribute certificate for authorizations encodes e.g. the following information:

- general authorization,
- authorization type, and
- dedicated authorization.

For each authorization type a separate attribute certificate may be issued.

The authority guaranteeing these authorizations is the related professional association. An example of an attribute certificate for authorizations is as follows.

```
{ admissionAuthority "KASSENÄRZTLICHE VEREINIGUNG BAYERN",
  contentsOfAdmissions {
    { professionInfos {
      { namingAuthority {
          namingAuthorityId id-OIDForKBV,
          namingAuthorityUrl "UrlToGeneralAuthorization" },
        professionItems { "Vertragsarzt" } },
      { namingAuthority {
          namingAuthorityId id-OIDFor,
          namingAuthorityUrl "UrlToAuthorizationType" },
        professionItems { "VF" } },
      { namingAuthority {
          namingAuthorityId id-OIDForBAEK,
          namingAuthorityUrl "UrlToDedicatedAuthorization" },
        professionItems { "Sonographie" } } } } }
```

Annex D

(normative)

Certificates for Authentication and Key Encipherment

D.1 Structure and content

The AUT-certificate and the KE-certificate are X.509v3 public key certificates.

The AUT-certificate contains information suitable for

- user identification, if access rights at the server side are UID oriented, and
- proving access rights, if no UID is registered and access rights are bound to authorizations denoted in the certificate (i.e. the profession indication PHYSICIAN or PHARMACIST).

The KE-certificate contains information about the receiver of a confidential document.

D.2 Coding

The coding is identical to the ES public key certificate with the following changes:

- there is no policy field indicating SigG conformance, but an appropriate identifier for the applied policy
- key usage is set to "digital signature" in the AUT certificate and to "key encipherment" in the KE certificate.
- the algorithm OID of the subject public key info field has to be set according to the use of the PK certified.

Annex E

(normative)

Algorithms and Input Formats for Security Operations

E.1 RSA-DSI-Formats

E.1.1 DSI according to ISO/IEC 9796-2 with random number

The digital signature input based on ISO/IEC 9796-2 and integrating a random number has the following structure:

- Header: 2 bits (= 01)
- More-data bit = 1 (Mn not empty)
- Padding field : bits equal to 0 (amount depending on length of modulus) followed by a single bit set to 1
- Data field: random no. inserted by the card (8 bytes)
- Hash field: hash-code (for SHA-1: 160 bits)
- Trailer: 1 byte: 'BC'

Contrary to the original algorithm ISO/IEC 9796-2 the random number as internal message is not integrated into the calculation of the hash value. Also a recoverable string (see ISO/IEC 9796-2, chapter. 6.3.4) is not produced, i.e. the intermediate string is used directly as DSI.

E.1.2 DSI according to PKCS #1

The DSI format according to PKCS #1 V1.5 (Block Type 1) has the following structure:

- Startbyte: '00'
- Block type: '01'
- Padding-String: 'FF ...FF'
- Separator: '00'
- DigestInfo: ASN.1-Sequence of digestAlgorithm (ASN.1-Sequence of OID and parameter) and digest (ASN.1-DO hash value)

The DigestInfo to be delivered to the card has the following coding:

SHA-1 with OID: { 1 3 14 3 2 26 }

DigestInfo: 3021 3009 06052B0E03021A 0500 0414 || hash value (20 bytes)

E.1.3 Algorithm Identifier for COMPUTE DS

In Table E.1 the algorithm reference to be set with a MSE command prior to the first signature creation is shown.

AlgRef	Meaning	OID
'11'	SHA-1 and RSA with DSI according to ISO/IEC 9796-2 with RND	{1 3 36 3 4 3 2 1} (TTT)
'12'	SHA-1 and RSA with DSI according to PKCS#1	{1 2 840 113549 1 1 5} (RSADSI)

Table E.1 – Algorithm References for ES

E.2 Algorithm Identifier for INTERNAL AUTHENTICATE

Tab. E.2 shows the algorithm references relevant for INTERNAL AUTHENTICATION.

AlgRef	Meaning
'0x'	No hash function
'1x'	SHA-1, OID = { 1 3 14 3 2 26 }
'x1'	RSA with DSI acc. to ISO/IEC 9796-2 with RND
'x2'	RSA with DSI acc. to PKCS #1
'05'	RSA framing according to PKCS#1 without OID
'1E'	RSA with SHA-1 without session key exchange
'1F'	RSA with SHA-1 with session key exchange and storing of SM keys

Table E.2 – Algorithm References for hash and signature algorithms for INTERNAL AUTHENTICATE used for device authentication and client/server authentication

E.3 Decryption of document cipher key

Tab. E.3 and E.4 show the algorithm reference relevant for decipherment.

AlgRef	Meaning
'0x'	$x \geq A$, Padding method not indicated
'0A'	RSA , PI = '00'
'1A'	RSA , PI \neq '00', see tab. E.4

Table E.3 - Algorithm References for decipherment

PI	KEI	Specification
'00'	No further information	ISO/IEC 7816-4
'81'	'02' RND (all bytes \neq '00', number key length dependent) '00' (= separator) document cipher key	PKCS #1 V2.0

Table E.4 - Key Encipherment Input Formats

Annex F

(normative)

Card Verifiable Authentication Certificate (CV AUT Certificate)

F.1 Principle structure

This certificate is used in PK-based authentication procedures as described in [DIN 66291-4] and applied in HPC/PDC and HPC/SMC interworking. In this annex the abbreviations CA (Certification Authority) and CSP (Certificate Service Provider) are equivalent.

The principle structure of a card verifiable certificate (CV certificate) shows the subsequent figure. The sequence of data elements can be described by a headerlist as defined in ISO/IEC 7816-8. This requires a fixed length of each data element.

Certificate Content	Certificate Profile Identifier (1 B)	Certification Authority Reference (8 B)	Certificate Holder Reference (12 B)	Certificate Holder Authorization (7 B)	OID.PuK (x B)	PuK (modulus tag '81', exponent tag '82') (x B)
Headerlist Content	'5F29 01'	'42 08'	'5F20 0C'	'5F4C 07'	'06 0x'	'7F49 xx 81 xx 82 xx'

Table F.1 - Certificate content and certificate headerlist

NOTE – The tag of CHA is '5F4C' and not '5F4B', see ISO/IEC 7816-6.

F.1.1 Certificate Profile Identifier

The "Certificate Profile Identifier (CPI)" has the purpose to denote the exact structure of a CV certificate. It can be considered as an identifier of a card internal headerlist describing the concatenation of the data elements including their length so that e.g. the PuK in a CV certificate can be found by the certificate verifying card (patient data card).

F.1.2 Certification Authority Reference (Authority Key Identifier)

The „Certification Authority Reference (CAR)“ has the purpose of identifying the certificate issuing CA with a distinguished name in such a way that the DE can be used as an authority key identifier for referencing the PK to be applied for the certificate verification. The CAR consists of

- the CA name (country code according to ISO 3166 (2 Bytes, DE = Deutschland) followed by an acronym of the CA (3 Bytes, ASCII characters) and
- an extension for key referencing (3 Bytes).

CA Name (5 B)	Extension for key referencing (3 B)
---------------	-------------------------------------

Table F.2 - Structure of the Certification Authority Reference (Authority Key Identifier)

The extension has the following structure:

Service Indicator (1 BCD)	Discretionary Data (1 BCD)	Algorithm Reference (2 BCD)	Date (last two digits of key generation year) (2 BCD)
---------------------------	----------------------------	-----------------------------	---

Table F.3 - Structure of the extension for key referencing

The Service Indicator has the value 0 = entity authentication according to the key usage in x.509v3 certificates.

The Discretionary Data may have a value at the discretion of the related CA.

The Algorithm Reference can be individually assigned by a CS for distinguishing different PK algorithms.

The Date consist of the last two digits of the year, in which the key pair for certificate signing was produced. If more than one key pair has been generated, it may be distinguished by using the discretionary data field.

F.1.3 Certificate Holder Reference (Subject Key Identifier)

The „Certificate Holder Reference (CHR)“ has the purpose to denote the certificate holder uniquely in such a way that the data element can be used as a subject key identifier for referencing the PK of the certificate holder. The CHR (12 bytes) consists of

- a CA name (5 Bytes) || Extension for key referencing (3 Bytes), if the certificate holder is a CA
- the ICCSN (12 Bytes), if the certificate holder is the card itself.

Filler (4 B)	CA Name (5 B)	Extension for key referencing (3 B)
-----------------	------------------	--

Table F.4 - Structure of the Certificate Holder Reference, if certificate holder is a CA

The „Extension for key referencing“ has the same structure as shown in tab. F.3. The field „date“ contains the last two digits of the year, in which the PK certified in the certificate (i.e. the PuK.CA.CS_AUT) is issued.

ICCSN.ICC (12 B)

Table F.5 - Structure of Certificate Holder Reference, if the certificate holder is the card itself.

NOTE – The ICCSN is a unique identifier and can also be interpreted as an identifier of the health professional holding the card.

F.1.4 Certificate Holder Authorization

The „Certificate Holder Authorization (CHA)“ has the purpose to denote the access rights of the card holder, e.g. the access rights of the health professional with respect to data stored in files in a patient data card. The meaning of CHA can be compared with a role based group key when applying symmetrical algorithms.

The CHA consists of

- a prefix denoting the entity assigning the role id ('D27600004000') and
- the role identifier of the health professional, see table F.7.

Prefix 'D27600004000' (6 B)	Role ID, see table F.7 (1B)
--------------------------------	--------------------------------

Table F.6 - Structure of Certificate Holder Authorization

NOTE – The last byte of the prefix is set to '00' (no differentiation between AID of DF.HPA and DF.SMA)

CHA Role ID	Owner
'00'	CSP (No authorization, used in higher level certificates or cross-certificates)
'01'	HPC Physician 1
'02'	HPC Physician 2 (reserved for future use)
'03'	SMC Physician
'04'	HPC Pharmacist 1
'05'	HPC Pharmacist 2 (reserved for future use)
'06'	SMC Pharmacist

Table F.7 - CHA role ID coding

NOTE – If different Health Professionals have different access rights to a PDC, then different role identifiers have to be assigned.

F.1.5 Object identifier for signature algorithm of the certificate holder

The Object Identifier has to be taken in compliance with DIN V66291-1.

F.1.6 PK of certificate holder

F.1.6.1 General construction

The Public Key in a certificate consists of a concatenation of parameters. These parameters, which have a context specific tag, belong to the DO PK (Tag '7F49', constructed) and have to be coded as octet string. In the CV certificate verifying entity (i.e. in a HPC, PDC or SMC) the occurrence of such a parameter and its length can be described in an appropriate headerlist.

F.1.6.2 Public key RSA

- Tag '81': Modulus
- Tag '82': Public exponent (e.g. 65537)

F.1.7 Signature formats for the CA signature

F.1.7.1 General aspects

The data to be signed are the certificate content. The hash function used and the digital signature input (DSI) format are denoted by the OID. For CV AUT certificates in an HPC the RSA algorithm is used for certificate signing.

F.1.7.2 RSA and DSI according to ISO/IEC 9796-2 and annex E

In the DSI for CV AUT certificates based on RSA no random padding is needed, since there are no attacks with dynamically produced DSIs. Therefore the original DSI format according to ISO 9796-2 can be used:

- Header: 2 bits (= 01)
- More-data bit = 0
- Padding field according to 9796-2
- Hash field: hash-code (for SHA-1: 160 bits)
- Trailer: 1 byte: 'BC'

F.1.8 Coding of the CV certificates

The following table shows the coding of the CV certificates.

CPI (1 B)	CAR (8 B)	CHR (12 B)	CHA (7 B)	OID	PK	Remark
'03'	AID (see F.6) '00'	'2B240304020201'	Modulus (128 B) Exponent (4 B)	C.CA: RSA, 1024 bit, SHA-1, 9796-2
'04'	see tables F.6 & F.7	'2B2407020101' '2B2407020101'	Modulus (128 B) Exponent (4 B)	C.ICC: RSA, 1024 bit, SHA-1,9796-2 C.IFD: RSA, 1024 bit, SHA-1,9796-2

Table F.8 – CPI values and CV field values

F.2 Structure and content of EF.C.HPC.AUT

EF.C.HPC.AUT contains a constructed certificate data object with tag '7F21' (RSA certificate with message recovery), see Table F.9.

Tag	L	Value		
'7F21'	'81CD'	CV certificate (205 byte)		
		Tag	L	Value
		'5F37'	'8180'	SIG.CA (128 byte)
				'6A' = Padding according to ISO 9796-2
				'04' = CPI
				'xx..xx' = CAR (8 byte)
				'xx..xx' = CHR (12 byte)
				'xx..xx' = CHA (7 byte)
				'xx..xx' = OID (6 byte)
				'xx..xx' = PK part 1 (first part of modulus, 72 byte)
				'xx..xx' = Hash (20 byte)
				'BC' = Trailer
		'5F38'	'3C'	'xx..xx' = PK remainder (rest of modulus followed by exponent '00010001', 60 byte)
		'42'	'08'	'xx..xx' = CAR (8 byte)

Table F.9 – Structure and content of EF.C.HPC.AUT

Annex G

(normative)

Device Authentication, Session Key Agreement and Secure Messaging

G.1 Device Authentication with RSA

Fig. G.1 shows the general procedure of mutual authentication with RSA including session key agreement.

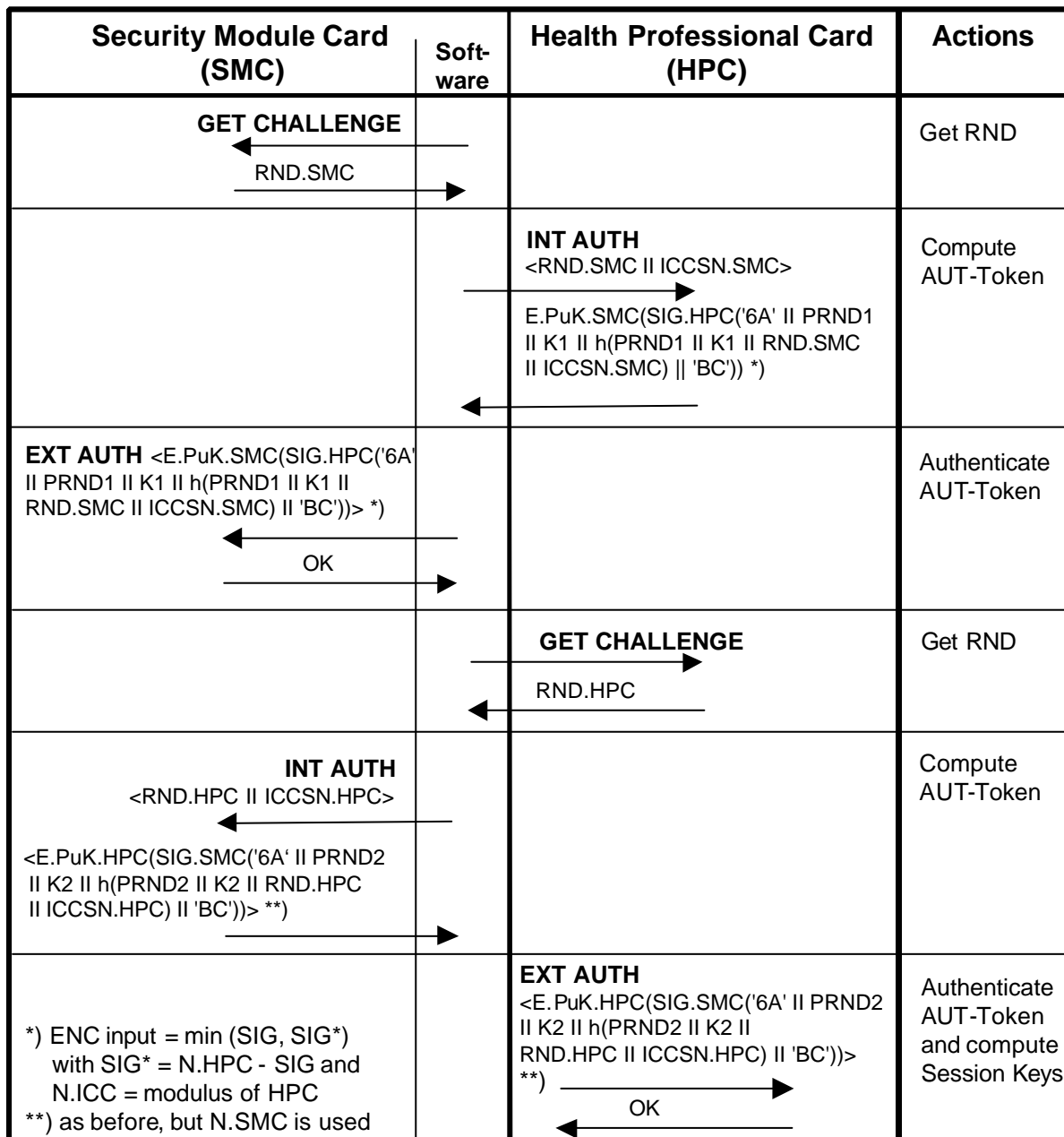


Figure G.1 - Diagram of the authentication procedure with RSA according to ISO/IEC 9796-2 (description of exchange / verification of the certificates and the use of the MSE commands was omitted)

NOTE - The procedure described here with separated authentication (see ISO/IEC 9798-3) was (for implementation reasons) preferred to that described in ISO 11770-3 with mutual authentication and key transport mechanism 5.

Table G.1 shows the data fields in the INTERNAL AUTHENTICATE command/response. Before the command INTERNAL AUTHENTICATE is sent to the card, the keys PrK.HPC.AUT and PuK.IFD.AUT must be set via the command MSE.

NOTE - If PrK.HPC.AUT was not selected, the command INTERNAL AUTHENTICATE is not successful. If PuK.HPC.AUT instead of PuK.SMC.AUT was selected, the command EXTERNAL AUTHENTICATE is not successful.

Command data field	RND.SMC (8 B) ICCSN.SMC (8 LSB)
Response data field	E.PuK.SMC (SIGMIN) with SIGMIN = min (SIG, SIG*) SIG = SIG.PrK.HPC ('6A' PRND1 (x B) K1 (32 B) h(PRND1 (x B) K1 (32 B) RND.SMC (8 B) ICCSN.SMC (8 LSB)) 'BC') SIG* = N.HPC – SIG with N.HPC = modulus of HPC

Table G.1 - Data fields of the command INTERNAL AUTHENTICATE with RSA

PRND1, PRND2 = Padding random number produced by the card (length = key length – 20 byte hash value – 32 bytes K1 – 2 bytes header '6A' and trailer 'BC', i.e. length = 74 Bytes for key length of 1024 bits, 42 Bytes for key length of 768 Bits)

Preconditions:

- The length of the RSA modulus must be a multiple of 8 bit.
- Hash-function is SHA-1.
- The length of the two RSA moduli N.HPC and N.SMC must be identical. This together with the use of min (SIG, SIG*) according to ISO 9796-2 guarantees that the deciphering delivers a unique result.

Command procedure:

- Verify, that the key reference of PuK.SMC is identical to ICCSN.SMC
- Generate K1 (32 bit random number) and store temporarily for session key calculation
- Build hash input and calculate hash value
- Build DS input and calculate digital signature acc. to ISO/IEC 9796-2 with internal message mr = PRND1 (x B) || K1 (32 B) and external message RND.SMC (8 B) || ICCSN.SMC (8 LSB)
- Encrypt SIGMIN using PuK.SMC

Table G.2 shows the data fields in the EXTERNAL AUTHENTICATE command/response.

Command data field	E.PuK.HPC (SIGMIN) with SIGMIN = min (SIG, SIG*) SIG = SIG.PrK.SMC ('6A' PRND2 (x B) K2 (32 B) h (PRND2 (x B) K2 (32 B) RND.HPC (8 B) ICCSN.HPC (8 LSB)) 'BC') SIG* = N.SMC – SIG with N.SMC = modulus of SMC
Response data field	Empty

Table G.2 - Data fields of the command EXTERNAL AUTHENTICATE with RSA

Command procedure:

- Decipher the cryptogram
- Retransform the signature to build the DS input.
- Build the hash value.

- Compare the hash value with the hash value in the DS input.
- Build the session keys from K1 and K2.

G.4 Key Agreement

G.4.1 Key agreement using RSA

One part of the authentication procedure is the agreement of session keys for building cryptograms and cryptographic checksums with DES-3

In a first step the values XOR is calculated on K1 and K2.

$$K = K1 \oplus K2$$

K (32 bytes) is interpreted as concatenation of the 4 necessary keys (each 8 bytes).

$$K = K_a (\text{ENC}) \parallel K_b (\text{ENC}) \parallel K_a (\text{MAC}) \parallel K_b (\text{MAC})$$

The use of the keys K_a and K_b is depicted in figure G.3 and G.4.

Note: Setting and checking of the parity bits are optional.

G.5 Secure Messaging

G.5.1 SM-DOs

Table G.7 shows the DOs used within the scope of the HP application (these are a partial set of the SM-DOs described in ISO/IEC 7816-4).

Tag	Meaning
'81'	Plain Value (to be protected by CC)
'97'	Le (to be protected by CC)
'99'	Status-Info (to be protected by CC)
'8E'	Cryptographic Checksum
'87'	PI Cryptogram (to be protected by CC)

Table G.7 - SM Data objects

For cryptograms the padding indicator PI is always set to '01', i.e. padding acc. to ISO/IEC 7816-4 ('80 ...00').

Note: The plain value SM DOs are always set to Tag '81', because the structure of the data in the data field is irrelevant for the SM view (the card does not check whether the data in a file are of TLV structure or not).

G.5.2 Commands and Responses with SM

After the authentication procedure is completed, all commands and responses shall be transferred in the SM mode. Since the command header should be integrated into the CC calculation, the bits b4 and b3 of the CLA byte shall be set to 1. For simplification, the examples are outlined with channel #0. Thus there is the following structure for commands and responses:

Command:

'0C'	INS	P1-P2	Lc	TPV	LPV	PV	Te	'01'	Le	TCC	Lcc	CC
------	-----	-------	----	-----	-----	----	----	------	----	-----	-----	----

The DOs PV and Le are conditional, that is those DOs are only present, when they are present in the command without SM. At least the 4 most significant bytes of the cryptographic checksum (CC) are transferred.

Response with data:

TPV	LPV	PV	TCC	Lcc	CC	SW1-SW2
-----	-----	----	-----	-----	----	---------

Response without data:

Tsw	'02'	SW1-SW2	TCC	Lcc	CC	SW1-SW2
-----	------	---------	-----	-----	----	---------

The DO PV is conditional, i.e. the DO is only present, when response data occur. In this case the DO SW is not present, i.e. the status bytes are only protected in responses without data.

If the following commands are performed with SM, then the data in the data field shall be transferred as a cryptogram:

- READ BINARY (Display Message, Identification Data)
- UPDATE BINARY (Display Message, Identification Data)
- VERIFY
- CHANGE RD
- RESET RC

Thus there is the following structure for those commands and their responses:

Command without cryptogram (READ BINARY):

'0C'	INS	P1-P2	Lc	TLe	'01'	Le	TCC	Lcc	CC
------	-----	-------	----	-----	------	----	-----	-----	----

Response with cryptogram:

TCG	LCG	PI, CG	TCC	Lcc	CC	SW1-SW2
-----	-----	--------	-----	-----	----	---------

Command with cryptogram (VERIFY, CHANGE RD, RESET RC):

'0C'	INS	P1-P2	Lc	TCG	LCG	PI,CG	TLe	'01'	Le	TCC	Lcc	CC
------	-----	-------	----	-----	-----	-------	-----	------	----	-----	-----	----

Response:

Tsw	'02'	SW1-SW2	TCC	Lcc	CC	SW1-SW2
-----	------	---------	-----	-----	----	---------

NOTE – The status bytes of the command response shall be identical to the status bytes protected by CC.

G.5.3 Treatment of SM-Errors

When the HPC or the SMC recognizes an SM error while interpreting a command, then the status bytes must be returned without SM. In ISO/IEC 7816-4 the following status bytes are defined to indicate SM errors:

- '6987': Expected SM data objects missing
- '6988': SM data objects incorrect

NOTE - Further SM status bytes can occur in application specific contexts.

After occurring an SM error, the session keys shall be erased.

G.5.4 Padding for checksum calculation

The padding mechanism acc. to ISO/IEC 7816-4 ('80 ...00') is applied.

G.5.5 DES-Mode, Initial Value and Send Sequence Counter

G.5.5.1 Cryptograms

Cryptograms are built with DES-3 in CBC-Mode with the Null vector as initial value.

G.5.5.2 Cryptographic Checksums

Cryptographic checksums are built acc. to ISO/IEC 7816-4 (chapter 5.6.3.1) as follows (the basic mechanism is to build a retail MAC acc. to ANSI X9.19 with DES):

- Initial stage: The initial check block y_0 is $E(K_a, SSC)$.
- Sequential Stage: The check blocks y_1, \dots, y_n are calculated using K_a .
- Final Stage: The cryptographic checksum is calculated from the last check block y_n as follows: $E(K_a, D(K_b, y_n))$.

Here $E()$ means encryption with DES, respectively $D()$ decryption with DES.

The send sequence counter SSC must be increased (+1) each time before a MAC is calculated, i.e. if the starting value is x , in the next command the value of SSC is $x+1$. The SSC value of the first response is then $x+2$.

The starting value for the SSC is

$SSC = \text{RND.HPC (4 least significant bytes)} \parallel \text{RND.SMC (4 least significant bytes)}$.

G.6 Use of DES

The following figure shows the application of keys in DES-3 (see also ISO 11568-2).

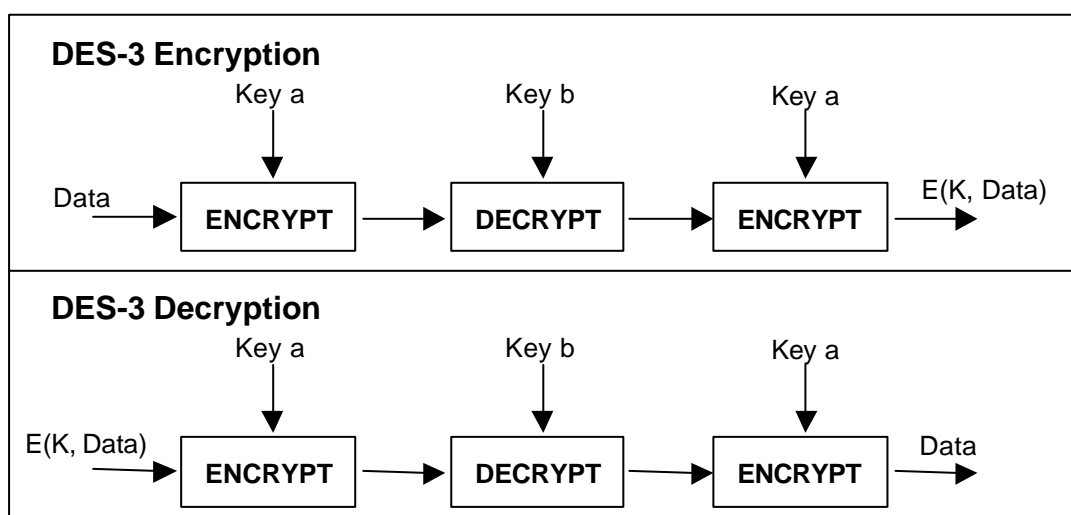


Figure G.3 - DES-3-Encryption/Decryption

The retail MAC is calculated as depicted in figure G.4.

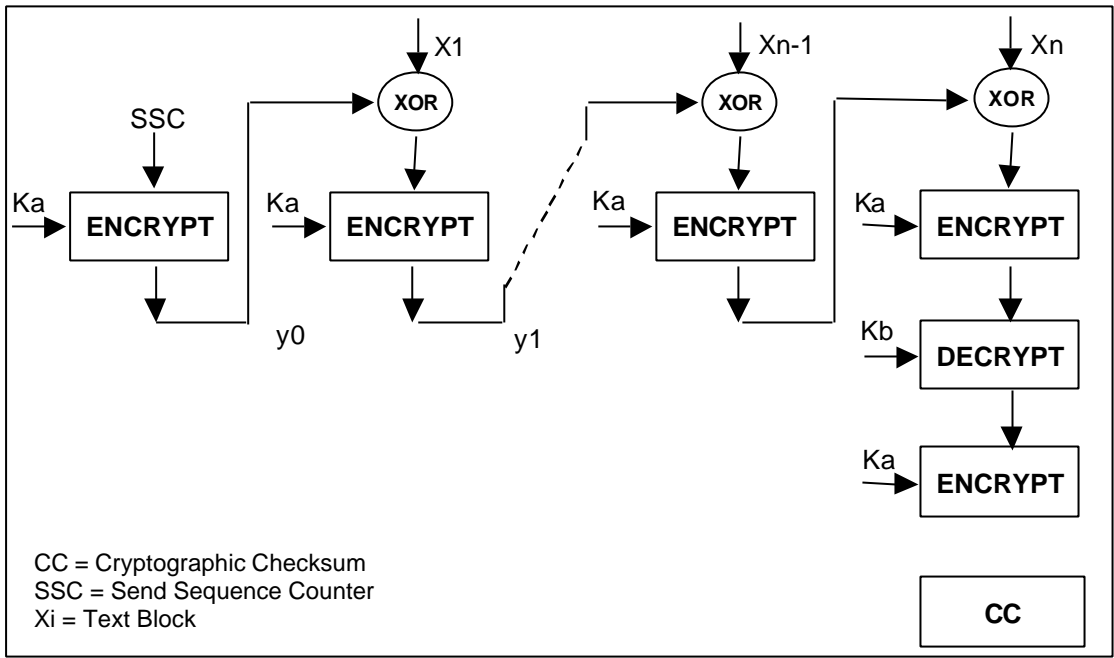


Figure G.4 - Calculation of the retail MAC

Annex H

(informative)

Cryptographic Information Objects

This annex shows a detailed example of the CIOs present in the EFs belonging to DF.CIA of an HPC.

H.1 EF.CIAInfo

The “CIAInfo” EF contains the version indication of the CIO description and the label of the application to which the CIO description belongs. The “cardflags” settings are “authRequired” (there are cryptographic functions requiring user authentication) and “prnGeneration” (card has the ability to generate pseudo-random numbers). Additionally, there are several references to the hardware-supported algorithm of the HPC, which get referenced by the keys to indicate the allowed algorithm-IDs.

H.1.1 ASN.1 Value notation

```
1      cialInfoExample CIAInfo ::= {
2          version v2,
3          label "Health Professional Application v2.0",
4          cardflags { authRequired, prnGeneration }
5          selInfo {
6              {
7                  se 2,
8                  aid ' D27600004002'H
9                  },
10             {
11                 se 3,
12                 aid ' D27600004002'H
13             }
14         },
15         supportedAlgorithms {
16             {
17                 reference 1,
18                 algorithm -2147483648, -- acc. to PKCS#11 "vendor defined"
19                 parameters NULL : NULL,
20                 supportedOperations { compute-signature, verify-signature, hash },
21                 objId { 1 3 36 3 4 3 2 1 }, -- OID for RSA-signature with DSI acc. to
22                 -- ISO/IEC 9796-2 with RND and SHA-1
23                 algRef 11 -- used in MSE-SET-command
24             },
25             {
26                 reference 2,
27                 algorithm 6, -- equals PKCS#11 Mechanism type
28                 -- "CKM_SHA1_RSA_PKCS"
29                 parameters NULL : NULL,
30                 supportedOperations { compute-signature, verify-signature, hash },
31                 objId { 1 2 840 113549 1 1 5 }, -- OID for RSA-signature with DSI acc. to
32                 -- PKCS#1 and SHA-1
33                 algRef 12 -- used in MSE-SET-command
34             },
35             {
36                 reference 3,
37                 algorithm 1, -- equals PKCS#11 Mechanism type
38                 -- "CKM_RSA_PKCS"
39                 parameters NULL : NULL,
40                 supportedOperations { compute-signature, verify-signature, encipher,
41                                     decipher },
42             }
43         }
44     }
```

```

32         objId { 1 2 840 113549 1 1 1 },      -- OID for RSA-encryption acc. to
33         algRef 5                               -- PKCS#1 (here without OID)
34     },                                         -- used in MSE-SET-command
35     {
36         reference 4,
37         algorithm -2147483647, -- acc. to PKCS#11 "vendor defined"
38         parameters NULL : NULL,
39         supportedOperations { compute-signature, verify-signature },
40         algRef 30                          -- used in MSE-SET-command
41     },
42     {
43         reference 5,
44         algorithm -2147483646, -- acc. to PKCS#11 "vendor defined"
45         parameters NULL : NULL,
46         supportedOperations { compute-signature, verify-signature },
47         algRef 31                          -- used in MSE-SET-command
48     },
49     {
50         reference 6,
51         algorithm 307,                      -- equals PKCS#11 Mechanism type
52         parameters NULL : NULL,            -- "CKM_DES3_CBC"
53         supportedOperations { encipher, decipher }
54         objId { 1 3 36 3 1 }              -- OID for DES-3 cryptography
55     }
56 }

```

H.1.2 ASN.1 Description, tags, lengths and values

```

1  CIAInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 215
2  version INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
3  label Label UTF8String: tag = [0] primitive; length = 29
4  0x4865616C74682050726F66657373696F6E616C20436172642076322E30
5  cardflags CardFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
6  0x0560
7  seInfo SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 26
8  SecurityEnvironmentInfo SEQUENCE: tag = [UNIVERSAL 16] constructed;
9  length = 11
10 se INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
11 0x02
12 aid OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 6
13 0xD27600004002
14 SecurityEnvironmentInfo SEQUENCE: tag = [UNIVERSAL 16] constructed;
15 length = 11
16 se INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
17 0x03
18 aid OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 6
19 0xD27600004002
20 supportedAlgorithms SEQUENCE OF: tag = [2] constructed; length = 146
21 AlgorithmInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 27
22 reference Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
23 0x01
24 algorithm INTEGER: tag = [UNIVERSAL 2] primitive; length = 4
25 0x80000000
26 parameters NULL: tag = [UNIVERSAL 5] primitive; length = 0
27 supportedOperations Operations BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
28 0x0152

```

```

18     objId OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 7
        0x2B240304030201
19     algRef Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
        0x11
20     AlgorithmInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 26
21         reference Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x02
22         algorithm INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x06
23         parameters NULL: tag = [UNIVERSAL 5] primitive; length = 0
24         supportedOperations Operations BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x0152
25         objId OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 9
            0x2A864886F70D010105
26         algRef Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x12
27     AlgorithmInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 26
28         reference Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x03
29         algorithm INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x01
30         parameters NULL: tag = [UNIVERSAL 5] primitive; length = 0
31         supportedOperations Operations BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x025C
32         objId OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 9
            0x2A864886F70D010101
33         algRef Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x05
34     AlgorithmInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 18
35         reference Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x04
36         algorithm INTEGER: tag = [UNIVERSAL 2] primitive; length = 4
            0x80000001
37         parameters NULL: tag = [UNIVERSAL 5] primitive; length = 0
38         supportedOperations Operations BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x0450
39         algRef Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x1E
40     AlgorithmInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 18
41         reference Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x05
42         algorithm INTEGER: tag = [UNIVERSAL 2] primitive; length = 4
            0x80000002
43         parameters NULL: tag = [UNIVERSAL 5] primitive; length = 0
44         supportedOperations Operations BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x0450
45         algRef Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x1F
46     AlgorithmInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 19
47         reference Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x06
48         algorithm INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
            0x0133
49         parameters NULL: tag = [UNIVERSAL 5] primitive; length = 0
50         supportedOperations Operations BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x020C
51         objId OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 4
            0x2B240301

```

H.1.3 Hexadecimal DER-encoding

```

1  30 81 D7
2      02 01
      01
3      80 1D
      48 65 61 6C 74 68 20 50 72 6F 66 65 73 73 69 6F 6E 61 6C 20 43
      61 72 64 20 76 32 2E 30
4      03 02
      05 60
5      30 1A
6      30 0B
7          02 01
          02
8          04 06
          D2 76 00 00 40 02
9      30 0B
10         02 01
11         03
11         04 06
11         D2 76 00 00 40 02
12      A2 81 92
13      30 1B
14         02 01
14         01
15         02 04
15         80 00 00 00
16         05 00
17         03 02
17         01 52
18         06 07
18         2B 24 03 04 03 02 01
19         02 01
19         11
20      30 1A
21         02 01
21         02
22         02 01
22         06
23         05 00
24         03 02
24         01 52
25         06 09
25         2A 86 48 86 F7 0D 01 01 05
26         02 01
26         12
27      30 1A
28         02 01
28         03
29         02 01
29         01
30         05 00
31         03 02
31         02 5C
32         06 09
32         2A 86 48 86 F7 0D 01 01 01
33         02 01
33         05
34      30 12
35         02 01
35         04
36         02 04
36         80 00 00 01
37         05 00
38         03 02
38         04 50
39         02 01
39         1E
40      30 12
41         02 01
41         05
42         02 04
42         80 00 00 02
43         05 00
44         03 02
44         04 50
45         02 01
45         1F

```

```

46          30 13
47          02 01
48          02 02
49          05 00
50          03 02
51          06 04
           2B 24 03 01

```

H.2 EF.OD

The “Object Directory” EF contains information which CIO-EFs are present and how they are referenced by SFID.

H.2.1 ASN.1 Value notation

```

1  authObjects :
2    path : {
3      efidOrPath '14'H -- SFID of EF.AOD
4    },
4  privateKeys :
5    path : {
6      efidOrPath '15'H -- SFID of EF.PrKD
7    },
7  trustedPublicKeys :
8    path : {
9      efidOrPath '16'H -- SFID of EF.PuKD-CSP
10   },
10 secretKeys :
11   path : {
12     efidOrPath '17'H -- SFID of EF.SKD
13   },
13 certificates :
14   path : {
15     efidOrPath '18'H -- SFID of EF.CD
16   },
16 trustedCertificates :
17   path : {
18     efidOrPath '19'H -- SFID of EF.CD-CSP
19   },
19 dataContainerObjects :
20   path : {
21     efidOrPath '1A'H -- SFID of EF.DCOD
22   }

```

H.2.2 ASN.1 Description, tags, lengths and values

```

1  CIOChoice CHOICE
   authObjects : tag = [8] constructed; length = 5
2  AuthObjects CHOICE
   path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
3  efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x14
4  CIOChoice CHOICE
   privateKeys : tag = [0] constructed; length = 5
5  PrivateKeys CHOICE
   path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
6  efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x15

```

```

7 CIOChoice CHOICE
  trustedPublicKeys : tag = [2] constructed; length = 5
8   PublicKeys CHOICE
  path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
9   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
    0x16
10 CIOChoice CHOICE
  secretKeys : tag = [3] constructed; length = 5
11   SecretKeys CHOICE
  path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
12   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
    0x17
13 CIOChoice CHOICE
  certificates : tag = [4] constructed; length = 5
14   Certificates CHOICE
  path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
15   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
    0x18
16 CIOChoice CHOICE
  trustedCertificates : tag = [5] constructed; length = 5
17   Certificates CHOICE
  path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
18   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
    0x19
19 CIOChoice CHOICE
  dataContainerObjects : tag = [7] constructed; length = 5
20   DataContainerObjects CHOICE
  path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
21   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
    0x1A

```

H.2.3 Hexadecimal DER-encoding

```

1 A8 05
2   30 03
3     04 01
4       14
5 A0 05
6   30 03
7     04 01
8       15
9 A2 05
10  30 03
11   04 01
12     16
13 A3 05
14  30 03
15   04 01
16     17
17 A4 05
18  30 03
19   04 01
20     18
21 A5 05
22  30 03
23   04 01
24     19

```

```

19 A7 05
20      30 03
21          04 01
                1A

```

H.3 EF.AOD

The “authentication object directory” EF holds the information about the two PINs and the external / certificate authentication scheme used for gaining reading-access to “EF.DM”.

H.3.1 ASN.1 Value notation

```

1  pwd : {
2      commonObjectAttributes {
3          label "PIN.HP.SIG",
4          flags { private }
5      },
6      classAttributes {
7          authId '01'H
8      },
9      typeAttributes {
10         pwdFlags { local, initialized, exchangeRefData },
11         pwdType iso9564-1,
12         minLength 6,
13         storedLength 0,
14         maxLength 12,
15         pwdReference -127 -- P2 value of the VERIFY / CHANGE RD command
16     }
17 },
18 pwd : {
19     commonObjectAttributes {
20         label "PIN.HP.ASS",
21         flags { private }
22     },
23     classAttributes {
24         authId '02'H
25     },
26     typeAttributes {
27         pwdFlags { initialized, exchangeRefData },
28         pwdType iso9564-1,
29         minLength 5,
30         storedLength 0,
31         maxLength 12,
32         pwdReference 1 -- P2 value of the VERIFY / CHANGE RD command
33     }
34 },
35 external : {
36     commonObjectAttributes {
37         label "C.SMC.AUT",
38         flags { private }
39     },
40     classAttributes {
41         authId '03'H
42     },
43     typeAttributes
44     certBasedAttributes : {
45         cha 'D2760000400004'H -- CHA of the health professionals SMC-AUT-Certificate
46     }
47 }

```

H.3.2 ASN.1 Description, tags, lengths and values

```
1  AuthenticationObjectChoice CHOICE
   pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 47
2   commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 16
3   label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
   0x50494E2E48502E534947
4   flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
   0x0780
5   classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
   tag = [UNIVERSAL 16] constructed; length = 3
6   authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x01
7   typeAttributes : tag = [1] constructed; length = 22
   PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 20
8   pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
   length = 3
   0x044810
9   pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
   4
10  minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   6
11  storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   0
12  maxLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   12
13  pwdReference Reference INTEGER: [0] primitive; length = 1
   0x81

14 AuthenticationObjectChoice CHOICE
   pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 47
15  commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 16
16  label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
   0x50494E2E48502E415353
17  flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
   0x0780
18  classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
   tag = [UNIVERSAL 16] constructed; length = 3
19  authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x02
20  typeAttributes : tag = [1] constructed; length = 22
   PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 20
21  pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
   length = 3
   0x040810
22  pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
   4
23  minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   5
24  storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   0
25  maxLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   12
26  pwdReference Reference INTEGER: [0] primitive; length = 1
   0x01
```

```

27 AuthenticationObjectChoice CHOICE
    external SEQUENCE: tag = [2] constructed; length = 35
28     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 15
29         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 9
            0x432E534D432E415554
30         flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
            0x0780
31         classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
            tag = [UNIVERSAL 16] constructed; length = 3
32             authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                0x03
33         typeAttributes : tag = [1] constructed; length = 11
            ExternalAuthObjectAttributes CHOICE
34             CertBasedAttributes SEQUENCE: tag = [0] constructed; length = 9
35                 cha OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 7
                    0xD2760000400004

```

H.3.3 Hexadecimal DER-encoding

```

1  30 2F
2      30 10
3          0C 0A
4              50 49 4E 2E 48 50 2E 53 49 47
5                  03 02
6                      07 80
7                          30 03
8                              04 01
9                                  01
10
11      A1 16
12          30 14
13              03 03
14                  04 48 10
15                      0A 01
16                          04
17                              02 01
18                                  06
19                                      02 01
20                                          00
21                                              02 01
22                                                  0C
23                                                      80 01
24                                                          81
25
26      30 2F
27          30 10
28              0C 0A
29                  50 49 4E 2E 48 50 2E 41 53 53
30                      03 02
31                          07 80
32                              30 03
33                                  04 01
34                                      02
35                                          A1 16
36                          30 14
37                              03 03
38                                  04 08 10
39                                      0A 01
40                                          04
41                                              02 01
42                                                  05
43                                                      02 01
44                                                          00
45                                                              02 01
46                                                                  0C
47                                                                      80 01
48                                                                          01

```

```

27 A2 23
28     30 0F
29         0C 09
30             43 2E 53 4D 43 2E 41 55 54
31                 03 02
32                     07 80
33                         30 03
34                             04 01
35                                 03
                                     A1 0B
                                         A0 09
                                             04 07
                                                 D2 76 00 00 40 00 04

```

H.4 EF.PrKD

The "private key directory" EF provides necessary information about the four private keys of the HP application. "PrK.HPC.AUT" is mentioned twice because of the two different usages of this key: for HPC-SMC- or HPC-PDC-authentication (which needs former user-authentication with „PIN.HPC.ASS“) and for establishing a „trusted channel“ (without user-authentication).

H.4.1 ASN.1 Value notation

```

1  privateRSAKey : {
2      commonObjectAttributes {
3          label "PrK.HP.ES",
4          flags { private },
5          authId '01'H          -- pointer to the AOD-entry of PIN.HP.SIG
6          accessControlRules {
7              {
8                  accessMode { execute },
9                  securityCondition and : {
10                     authId : '01'H, -- pointer to the AOD-entry of PIN.HP.SIG
11                     authReference : {
12                         authMethod { userAuthentication },
13                         seldentifier 2          -- SE #2
14                     }
15                 }
16             }
17         }
18     },
19     classAttributes {
20         id '91'H,          -- cross-reference to the CD-entry of C.HP.ES
21         usage { nonRepudiation },
22         keyReference -111 -- used in MSE-SET-command
23         algReference {
24             1,          -- pointer to the supportedAlgorithms-entry in CIAInfo
25             2
26         }
27     },
28     typeAttributes {
29         value {
30             efidOrPath "H"
31         },
32         modulusLength 1536 -- for keys usable until 2007
33     }
34 },
35 privateRSAKey : {
36     commonObjectAttributes {
37         label "PrK.HP.AUT",
38         flags { private },
39         authId '02'H          -- pointer to the AOD-entry of PIN.HP.ASS

```

```

29     accessControlRules {
30         {
31             accessMode { execute },
32             securityCondition and : {
33                 authId : '02'H, -- pointer to the AOD-entry of PIN.HP.ASS
34                 authReference : {
35                     authMethod { userAuthentication },
36                     seldentifier 2 -- SE #2
37                 }
38             }
39         }
40     },
41     classAttributes {
42         id '92'H, -- cross-reference to the CD-entry of C.HP.AUTH
43         usage { signRecover },
44         keyReference -110 -- used in MSE-SET-command
45         algReference {
46             3 -- pointer to the supportedAlgorithms-entry in CIAInfo
47         }
48     },
49     typeAttributes {
50         value {
51             efidOrPath "H"
52         },
53         modulusLength 1024
54     }
55 },
56 privateRSAKey : {
57     commonObjectAttributes {
58         label "PrK.HP.KE",
59         flags { private },
60         authId '02'H -- pointer to the AOD-entry of PIN.HP.ASS
61         accessControlRules {
62             {
63                 accessMode { execute },
64                 securityCondition and : {
65                     authId : '02'H, -- pointer to the AOD-entry of PIN.HP.ASS
66                     authReference : {
67                         authMethod { userAuthentication },
68                         seldentifier 2 -- SE #2
69                     }
70                 }
71             }
72         }
73     },
74     classAttributes {
75         id '93'H, -- in this specification unused (but mandatory)
76         usage { keyDecipher },
77         keyReference -109 -- used in MSE-SET-command
78         algReference {
79             3 -- pointer to the supportedAlgorithms-entry in CIAInfo
80         }
81     },
82     typeAttributes {
83         value {
84             efidOrPath "H"
85         },
86         modulusLength 1024
87     }
88 },

```

```

68 privateRSAKey : {
69     commonObjectAttributes {
70         label "PrK.HPC.AUT for HPC-SMC-/HPC-PDC-AUT",
71         flags { private },
72         authId '02'H,           -- pointer to the AOD-entry of PIN.HP.ASS
73         accessControlRules {
74             {
75                 accessMode { execute },
76                 securityCondition and : {
77                     authId : '02'H, -- pointer to the AOD-entry of PIN.HP.ASS
78                     authReference : {
79                         authMethod { userAuthentication },
80                         seldentifier 2           -- SE #2
81                     }
82                 }
83             }
84         },
85     classAttributes {
86         id '11'H,           -- cross-reference to the CD-entry of C.HPC.AUT
87         usage { signRecover },
88         keyReference 17,   -- used in MSE-SET-command
89         algReference {
90             4           -- pointer to the supportedAlgorithms-entry in CIAInfo
91         },
92     typeAttributes {
93         value {
94             efidOrPath "H
95         },
96         modulusLength 1024
97     }
98 }
99 privateRSAKey : {
100     commonObjectAttributes {
101         label "PrK.HPC.AUT for TC",
102         flags { private },
103         accessControlRules {
104             {
105                 accessMode { execute },
106                 securityCondition authReference : {
107                     authMethod {},
108                     seldentifier 3           -- SE #3
109                 }
110             }
111         },
112     classAttributes {
113         id '11'H,           -- cross-reference to the CD-entry of C.HPC.AUT
114         usage { signRecover },
115         keyReference 17,   -- used in MSE-SET-command
116         algReference {
117             5           -- pointer to the supportedAlgorithms-entry in CIAInfo
118         },
119     typeAttributes {
120         value {
121             efidOrPath "H
122         },
123         modulusLength 1024
124     }
125 }

```

H.4.2 ASN.1 Description, tags, lengths and values

```
1 PrivateKeyChoice CHOICE
  privateKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 75
2   commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 40
3     label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 9
       0x50724B2E48502E4553
4     flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
       length = 2
       0x0780
5     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
       0x01
6     accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 20
7       AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 18
8         accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
           0x0520
9         securityCondition SecurityCondition CHOICE
10          and SEQUENCE OF: tag = [1] constructed; length = 12
11           authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
             length = 1
             0x01
12           authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
             structed; length = 7
13             authMethod AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
               length = 2
               0x0520
14             seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
               0x02
15         classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
           structed; length = 19
16         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
           0x91
17         usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 3
           0x060040
18         keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
           length = 1
           0x91
19         algReference SEQUENCE OF: tag = [1] constructed; length = 6
20         Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
           0x01
21         Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
           0x02
22         typeAttributes : tag = [1] constructed; length = 10
           PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
           length = 8
23         value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
           efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
24         modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
           1536

24 PrivateKeyChoice CHOICE
  privateKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 72
25   commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 41
26     label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
       0x50724B2E48502E415554
27     flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
       length = 2
       0x0780
```

```

28     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x02
29     accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 20
30     AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 18
31     accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
        0x0520
32     securityCondition SecurityCondition CHOICE
        and SEQUENCE OF: tag = [1] constructed; length = 12
33     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
        length = 1
        0x02
34     authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
        structed; length = 7
35     authMethod AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
        length = 2
        0x0520
36     seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
        0x02
37     classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
        structed; length = 15
38     id Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x92
39     usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
        0x0410
40     keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
        length = 1
        0x92
41     algReference SEQUENCE OF: tag = [1] constructed; length = 3
42     Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
        0x03
43     typeAttributes : tag = [1] constructed; length = 10
        PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 8
44     value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
        efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
45     modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
        1024

46 PrivateKeyChoice CHOICE
    privateKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 71
47     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 40
48     label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 9
        0x50724B2E48502E4B45
49     flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
        length = 2
        0x0780
50     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x02
51     accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 20
52     AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 18
53     accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
        0x0520
54     securityCondition SecurityCondition CHOICE
        and SEQUENCE OF: tag = [1] constructed; length = 12
55     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
        length = 1
        0x02

```

```

56         authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
          structed; length = 7
57         authDomain AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
          length = 2
          0x0520
58         seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          0x02
59     classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
        structed; length = 15
60         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x93
61         usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
          0x0204
62         keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
          length = 1
          0x93
63         algReference SEQUENCE OF: tag = [1] constructed; length = 3
64             Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              0x03
65     typeAttributes : tag = [1] constructed; length = 10
        PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 8
66         value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
            efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
67         modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
            1024

68 PrivateKeyChoice CHOICE
    privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 98
69     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 67
70         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 36
          0x50724b2e4850432e41555420666f72204850432d534d432d2f4850432d5044432d415554
71         flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
          length = 2
          0x0780
72         authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x02
73         accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 20
74             AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 18
75                 accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                    0x0520
76                 securityCondition SecurityCondition CHOICE
                    and SEQUENCE OF: tag = [1] constructed; length = 12
77                     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
                        length = 1
                        0x02
78                     authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
                        structed; length = 7
79                     authDomain AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
                        length = 2
                        0x0520
80                     seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
                        0x02
81                 classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
                    structed; length = 15
82                     iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                        0x11
83                     usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                        0x0410

```

```

84     keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
      length = 1
      0x11
85     algReference SEQUENCE OF: tag = [1] constructed; length = 3
86     Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
      0x04
87     typeAttributes : tag = [1] constructed; length = 10
      PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
      length = 8
88     value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
      efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
89     modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
      1024

90 PrivateKeyChoice CHOICE
      privateKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 71
91     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 40
92     label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 18
      0x50724b2e4850432e41555420666f72205443
93     flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
      length = 2
      0x0780
94     accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 14
95     AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 12
96     accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
      0x0520
97     securityCondition SecurityCondition CHOICE
      authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
      structed; length = 6
98     authMethod AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
      length = 1
      0x00
99     seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
      0x03
100    classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
      structed; length = 15
101    id Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x11
102    usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
      0x0410
103    keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
      length = 1
      0x11
104    algReference SEQUENCE OF: tag = [1] constructed; length = 3
105    Reference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
      0x05
106    typeAttributes : tag = [1] constructed; length = 10
      PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
      length = 8
107    value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
      efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
108    modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
      1024

```

H.4.3 Hexadecimal DER-encoding

```

1  30 4B
2      30 28
3          0C 09
4              50 72 4B 2E 48 50 2E 45 53
5                  03 02
6                      07 80
7                          04 01
8                              01
9                                  30 14
10                                      30 12
11                                          03 02
12                                              05 20
13                                                  A1 0C
14                                                      04 01
15                                                          01
16                                                              30 07
17                                                                  03 02
18                                                                      05 20
19                                                                          02 01
20                                                                              02
21
22
23
24      30 13
25          04 01
26              91
27                  03 03
28                      06 00 40
29                          02 01
30                              91
31                                  A1 06
32                                      02 01
33                                          01
34                                              02 01
35                                                  02
36
37      A1 0A
38          30 08
39              30 02
40                  04 00
41                      02 02
42                          06 00
43
44      30 48
45          30 29
46              0C 0A
47                  50 72 4B 2E 48 50 2E 41 55 54
48                      03 02
49                          07 80
50                              04 01
51                                  02
52                                      30 14
53                                          30 12
54                                              03 02
55                                                  05 20
56                                                      A1 0C
57                                                          04 01
58                                                              02
59                                                                  30 07
60                                                                      03 02
61                                                                          05 20
62                                                                              02 01
63                                                                                  02
64
65      30 0F
66          04 01
67              92
68                  03 02
69                      04 10
70                          02 01
71                              92
72                                  A1 03
73                                      02 01
74                                          03
75
76      A1 0A
77          30 08
78              30 02
79                  04 00
80                      02 02
81                          04 00

```

46 30 47
47 30 28
48 0C 09 50 72 4B 2E 48 50 2E 4B 45
49 03 02 07 80
50 04 01 02
51 30 14
52 30 12
53 03 02 05 20
54 A1 0C
55 04 01 02
56 30 07
57 03 02 05 20
58 02 01 02
59 30 0F
60 04 01 93
61 03 02 02 04
62 02 01 93
63 A1 03
64 02 01 03
65 A1 0A 30 08
66 30 02 04 00
67 02 02 04 00
68 30 62
69 30 43
70 0C 24 50 72 4b 2e 48 50 43 2e 41 55 54 20 66 6f 72 20 48 50 43 2d 53
4d 43 2d 2f 48 50 43 2d 50 44 43 2d 41 55 54
71 03 02 07 80
72 04 01 02
73 30 14
74 30 12
75 03 02 05 20
76 A1 0C
77 04 01 02
78 30 07
79 03 02 05 20
80 02 01 02
81 30 0F
82 04 01 11
83 03 02 04 10
84 02 01 11
85 A1 03
86 02 01 04
87 A1 0A 30 08
88 30 02 04 00
89 02 02 04 00

```

90 30 47
91   30 28
92     0C 12
93       50 72 4b 2e 48 50 43 2e 41 55 54 20 66 6f 72 20 54 43
94       03 02
95       07 80
96       30 0E
97         30 0C
98           03 02
99             05 20
100             30 06
101               03 01
102                 00
103                 02 01
104                   03
105     30 0F
106       04 01
107         11
108       03 02
109         04 10
110       02 01
111         11
112       A1 03
113         02 01
114           05
115     A1 0A
116       30 08
117         30 02
118           04 00
119       02 02
120         04 00

```

H.5 EF.PuKD-CSP

The “public key directory” EF for the CSP’s trusted public key gives information about type of the key and the key-reference needed for the “MSE SET” command.

H.5.1 ASN.1 Value notation

```

1  publicRSAKey : {
2    commonObjectAttributes {
3      label "PuK.CSP.CS-CV"
4    },
5    classAttributes {
6      iD '44455A4757000003'H,           -- has no specific meaning/not used as pointer
7      usage { verifyRecover },
8      keyReference 4919437430420930563 -- used in MSE-SET-command; has to represent
9                                         -- the CV's CAR-entry.
10                                        -- As it is only an example, it has to be
11                                        -- modified acc. to the actually used certificate !
12    },
13    typeAttributes {
14      value indirect :
15        path : {
16          efidOrPath "H
17        },
18      modulusLength 1024 -- for keys usable until 2006
19    }
20  }

```

H.5.2 ASN.1 Description, tags, lengths and values

```
1  PublicKeyChoice CHOICE
   publicRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 55
2   commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 15
3   label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 13
   0x50754B2E4353502E43532D4356
4   classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
   structed; length = 24
5   iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 8
   0x44455A4757000003
6   usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
   0x0001
7   keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
   length = 8
   0x44455A4757000003
8   typeAttributes : tag = [1] constructed; length = 10
   PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
   length = 8
9   value CHOICE
   indirect ReferencedValue CHOICE
   path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
10  modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
   1024
```

H.5.3 Hexadecimal DER-encoding

```
1  30 37
2    30 0F
3      0C 0D
4        50 75 4B 2E 43 53 50 2E 43 53 2D 43 56
5      30 18
6        04 08
7          44 45 5A 47 57 00 00 03
8        03 02
9          00 01
10       02 08
11         44 45 5A 47 57 00 00 03
12     A1 0A
13       30 08
14         30 02
15           04 00
16         02 02
17           04 00
```

H.6 EF.SKD

This file contains the information about the “group key” used by the HPC for the symmetrical authentication. In this specification there are keys for different professions/specialists, differentiated by their keyReference. The following example defines the SK-DO for “GK.HP.AUT-PHYS1”, a physicians key. *Only one GK is stored on an HPC!*

H.6.1 ASN.1 Value notation

```
1  genericSecretKey : {
2    commonObjectAttributes {
3      label "GK.HPC.AUT-PHYS1",           -- "GK.HPC.AUT-PHYS1" or "...-PHYS2" for
                                           -- physicians, "GK.HPC.AUT-PHAR1" or
                                           -- "-PHAR2" for pharmacists
4      flags { private },
5      authId '02'H                       -- pointer to the AOD-entry of PIN.HP.ASS
    },
```

```

6      classAttributes {
7          id '9A'H,                -- has no specific meaning/not used as pointer
8          usage { encipher, decipher, keyEncipher, keyDecipher },
9          keyReference -102,      -- -102 ('9A'H) / -101 ('9B'H) for physicians,
                                -- -99 ('9D'H) / -98 ('9E'H) for pharmacists !
10         algReference {
11             6                    -- pointer to the supportedAlgorithms-entry in
                                -- CIAInfo
12         },
13     typeAttributes {
14         keyType { 1 3 36 3 1 }, -- OID of DES-3
15         keyAttr NULL : NULL
16     }
17 }

```

H.6.2 ASN.1 Description, tags, lengths and values

```

1  SecretKeyChoice CHOICE
   genericSecretKey SEQUENCE: tag = [15] constructed; length = 56
2  commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 25
3  label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 16
   0x474B2E4850432E4155542D5048595331
4  flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
   length = 2
   0x0780
5  authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x02
6  classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
   structed; length = 15
7  id Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x9A
8  usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
   0x02CC
9  keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
   length = 1
   0x9A
10 algReference SEQUENCE OF: tag = [1] constructed; length = 3
11   INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
   0x06
12 typeAttributes : tag = [1] constructed; length = 10
   GenericKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
   length = 8
13   keyType OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 4
   0x2B240301
14   keyAttr NULL: tag = [UNIVERSAL 5] primitive; length = 0

```

H.6.3 Hexadecimal DER-encoding

```

1  AF 38
2     30 19
3         0C 10
4             47 4B 2E 48 50 43 2E 41 55 54 2D 50 48 59 53 31
5                 03 02
6                     07 80
7                         04 01
8                             02

```

```

6      30 0F
7          04 01
8          9A
9          03 02
10         02 CC
11         02 01
12         9A
13         A1 03
14         02 01
15         06
16         A1 0A
17         30 08
18         06 04
19         2B 24 03 01
20         05 00

```

H.7 EF. CD

The “certificate directory” EF contains the file-references to the user’s and card’s certificates, connected to the descriptions of the private keys in “EF.PrKD” with the “iD”-attribute.

H.7.1 ASN.1 Value notation

```

1  x509Certificate : {
2      commonObjectAttributes {
3          label "C.HP.ES",
4          flags { private },
5          authId '02'H      -- pointer to the AOD-entry of PIN.HP.ASS
6      },
7      classAttributes {
8          iD '91'H          -- cross-reference to the PrKD-entry of PrK.HP.ES
9      },
10     typeAttributes {
11         value indirect :
12             path : {
13                 efidOrPath '04'H  -- SFID of C.HP.ES
14             }
15     },
16     x509AttributeCertificate : {
17         commonObjectAttributes {
18             label "C.HP.ES-AC1",
19             flags { private, modifiable },
20             authId '02'H      -- pointer to the AOD-entry of PIN.HP.ASS
21         },
22         classAttributes {
23             iD '91'H          -- cross-reference to the PrKD-entry of PrK.HP.ES
24         },
25         typeAttributes {
26             value indirect :
27                 path : {
28                     efidOrPath '05'H -- SFID of C.HP.ES-AC1
29                 }
30     },
31     x509AttributeCertificate : {
32         commonObjectAttributes {
33             label "C.HP.ES-AC2",
34             flags { private, modifiable },
35             authId '02'H      -- pointer to the AOD-entry of PIN.HP.ASS
36         },
37         classAttributes {
38             iD '91'H          -- cross-reference to the PrKD-entry of PrK.HP.ES
39         },

```

```

28     typeAttributes {
29         value indirect :
30             path : {
31                 efidOrPath '06'H -- SFID of C.HP.ES-AC2
32             }
33     },
34 x509AttributeCertificate : {
35     commonObjectAttributes {
36         label "C.HP.ES-AC3",
37         flags { private, modifiable },
38         authId '02'H      -- pointer to the AOD-entry of PIN.HP.ASS
39     },
40     classAttributes {
41         id '91'H          -- cross-reference to the PrKD-entry of PrK.HP.ES
42     },
43     typeAttributes {
44         value indirect :
45             path : {
46                 efidOrPath '07'H -- SFID of C.HP.ES-AC3
47             }
48     },
49 x509Certificate : {
50     commonObjectAttributes {
51         label "C.HP.AUT",
52         flags { private },
53         authId '02'H      -- pointer to the AOD-entry of PIN.HP.ASS
54     },
55     classAttributes {
56         id '92'H          -- cross-reference to the PrKD-entry of PrK.HP.AUT
57     },
58     typeAttributes {
59         value indirect :
60             path : {
61                 efidOrPath '08'H --SFID of C.HP.AUT
62             }
63     },
64 cvCertificate : {
65     commonObjectAttributes {
66         label "C.HPC.AUT"
67     },
68     classAttributes {
69         id '11'H          -- cross-reference to the PrKD-entry of PrK.HPC.AUT
70     },
71     typeAttributes {
72         value indirect :
73             path : {
74                 efidOrPath '01'H -- SFID of C.HPC.AUT
75             }
76     }
77 }

```

H.7.2 ASN.1 Description, tags, lengths and values

```

1  CertificateChoice CHOICE
   x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 32
2  commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 16
3  label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 7
   0x432E48502E4553
4  flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
   0x0780

```

```

5     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length=2
      0x02
6     classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 3
7     iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x91
8     typeAttributes : tag = [1] constructed; length = 7
      X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
      length = 5
9     value CHOICE
      indirect ReferencedValue CHOICE
        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
10    efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x04

11 CertificateChoice CHOICE
    x509AttributeCertificate SEQUENCE: tag = [0] constructed; length = 36
12    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 20
13    label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 11
      0x432E48502E45532D414331
14    flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
      0x06C0
15    authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length=2
      0x02
16    classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 3
17    iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x91
18    typeAttributes : tag = [1] constructed; length = 7
      X509AttributeCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 5
19    value CHOICE
      indirect ReferencedValue CHOICE
        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
20    efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x05

21 CertificateChoice CHOICE
    x509AttributeCertificate SEQUENCE: tag = [0] constructed; length = 36
22    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 20
23    label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 11
      0x432E48502E45532D414332
24    flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
      0x06C0
25    authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length=2
      0x02
26    classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 3
27    iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x91
28    typeAttributes : tag = [1] constructed; length = 7
      X509AttributeCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 5
29    value CHOICE
      indirect ReferencedValue CHOICE
        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
30    efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x06

```

```

31 CertificateChoice CHOICE
    x509AttributeCertificate SEQUENCE: tag = [0] constructed; length = 36
32     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 20
33         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 11
            0x432E48502E45532D414333
34         flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
            0x06C0
35         authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length=2
            0x02
36         classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
            constructed; length = 3
37         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
            0x91
38         typeAttributes : tag = [1] constructed; length = 7
            X509AttributeCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
                constructed; length = 5
39             value CHOICE
                indirect ReferencedValue CHOICE
                    path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
40                     efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                        0x07

41 CertificateChoice CHOICE
    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 33
42     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 17
43         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 8
            0x432E48502E415554
44         flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
            0x0780
45         authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length=2
            0x02
46         classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
            constructed; length = 3
47         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
            0x92
48         typeAttributes : tag = [1] constructed; length = 7
            X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
            length = 5
49             value CHOICE
                indirect ReferencedValue CHOICE
                    path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
50                     efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                        0x08

51 CertificateChoice CHOICE
    cvCertificate SEQUENCE: tag = [5] constructed; length = 27
52     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 11
53         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 9
            0x432E4850432E415554
54         classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
            constructed; length = 3
55         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
            0x11

```

```

56 typeAttributes : tag = [1] constructed; length = 7
   CVCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
   length = 5
57   value CHOICE
     indirect ReferencedValue CHOICE
       path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
58       efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
         0x01

```

H.7.3 Hexadecimal DER-encoding

```

1  30 20
2    30 10
3      0C 07
4        43 2E 48 50 2E 45 53
5          03 02
6            07 80
7              04 01
8                02
9          30 03
10            04 01
11              91
12            A1 07
13              30 05
14                30 03
15                  04 01
16                    04
17
18
19
20
21  A0 24
22    30 14
23      0C 0B
24        43 2E 48 50 2E 45 53 2D 41 43 31
25          03 02
26            06 C0
27              04 01
28                02
29          30 03
30            04 01
31              91
32            A1 07
33              30 05
34                30 03
35                  04 01
36                    05
37
38
39
40
41  A0 24
42    30 14
43      0C 0B
44        43 2E 48 50 2E 45 53 2D 41 43 32
45          03 02
46            06 C0
47              04 01
48                02
49          30 03
50            04 01
51              91
52            A1 07
53              30 05
54                30 03
55                  04 01
56                    06
57
58
59
60
61  A0 24
62    30 14
63      0C 0B
64        43 2E 48 50 2E 45 53 2D 41 43 33
65          03 02
66            06 C0
67              04 01
68                02
69          30 03
70            04 01
71              91

```

```

38      A1 07
39          30 05
40              30 03
41                  04 01
42                      07
41 30 21
42      30 11
43          0C 08
44              43 2E 48 50 2E 41 55 54
45          03 02
46              07 80
47          04 01
48              02
49      30 03
50          04 01
51              92
52      A1 07
53          30 05
54              30 03
55                  04 01
56                      08
57  A5 1B
58      30 0B
59          0C 09
60              43 2E 48 50 43 2E 41 55 54
61      30 03
62          04 01
63              11
64      A1 07
65          30 05
66              30 03
67                  04 01
68                      01

```

H.8 EF. CD-CSP

The "certificate directory" for the certificate service providers contains references to the trusted CA-certificates.

H.8.1 ASN.1 Value notation

```

1  x509Certificate : {
2      commonObjectAttributes {
3          label "C.CSP.CS"
4      },
5      classAttributes {
6          id 'F1'H, -- has no specific meaning/not used as pointer
7          authority TRUE,
8          trustedUsage {
9              keyUsage { keyCertSign }
10         }
11     },
12     typeAttributes {
13         value indirect :
14         path : {
15             efidOrPath '09'H
16         }
17     }
18 },
19 x509Certificate : {
20     commonObjectAttributes {
21         label "C.RCSP.ES-SELF"
22     },

```

```

15     classAttributes {
16         id 'F2'H,                -- has no specific meaning/not used as pointer
17         authority TRUE,
18         trustedUsage {
19             keyUsage { keyCertSign }
20         },
21     typeAttributes {
22         value indirect :
23             path : {
24                 efidOrPath '0A'H
25             }
26     },
27 x509Certificate : {
28     commonObjectAttributes {
29         label "C.RCSP.AUT-SELF"
30     },
31     classAttributes {
32         id 'F3'H,                -- has no specific meaning/not used as pointer
33         authority TRUE,
34         trustedUsage {
35             keyUsage { keyCertSign }
36         },
37     typeAttributes {
38         value indirect :
39             path : {
40                 efidOrPath '0B'H
41             }
42     }
43 }

```

H.8.2 ASN.1 Description, tags, lengths and values

```

1  CertificateChoice CHOICE
2    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 35
3    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
4    constructed; length = 13
5    label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 8
6    0x432E4353502E4353
7    classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
8    constructed; length = 12
9    id Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
10   0xF1
11   authority BOOLEAN: tag = [UNIVERSAL 1] primitive; length = 1
12   0xFF
13   trustedUsage Usage SEQUENCE: tag = [1] constructed; length = 4
14   keyUsage KeyUsage BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
15   0x0204
16   typeAttributes : tag = [1] constructed; length = 7
17   X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
18   length = 5
19   value CHOICE
20     indirect ReferencedValue CHOICE
21       path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
22       efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
23       0x09

```

```

12 CertificateChoice CHOICE
    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 41
13     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 16
14         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 14
            0x432E524353502E45532D53454C46

15         classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
            constructed; length = 12
16             iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                0xF2
17             authority BOOLEAN: tag = [UNIVERSAL 1] primitive; length = 1
                0xFF
18             trustedUsage Usage SEQUENCE: tag = [1] constructed; length = 4
19                 keyUsage KeyUsage BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                    0x0204
20             typeAttributes : tag = [1] constructed; length = 7
                X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
                length = 5
21                 value CHOICE
                    indirect ReferencedValue CHOICE
                        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
22                             efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                                0x0A

23 CertificateChoice CHOICE
    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 42
24     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 17
25         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 15
            0x432E524353502E4155542D53454C46
26         classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
            constructed; length = 12
27             iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                0xF3
28             authority BOOLEAN: tag = [UNIVERSAL 1] primitive; length = 1
                0xFF
29             trustedUsage Usage SEQUENCE: tag = [1] constructed; length = 4
30                 keyUsage KeyUsage BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                    0x0204
31             typeAttributes : tag = [1] constructed; length = 7
                X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
                length = 5
32                 value CHOICE
                    indirect ReferencedValue CHOICE
                        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
33                             efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                                0x0B

```

H.8.3 Hexadecimal DER-encoding

```

1  30 23
2      30 0A
3          0C 08
4              43 2E 43 53 50 2E 43 53
5          30 0C
6              04 01
7                  F1
8                      01 01
9                          FF
10                             A1 04
11                                 03 02
12                                     02 04

```

```

9      A1 07
      30 05
10     30 03
11     04 01
      09

12 30 29
13 30 10
14 0C 0E
      43 2E 52 43 53 50 2E 45 53 2D 53 45 4C 46
15 30 0C
16 04 01
      F2
17 01 01
      FF
18 A1 04
19 03 02
      02 04
20 A1 07
      30 05
21 30 03
22 04 01
      0A

23 30 2A
24 30 11
25 0C 0F
      43 2E 52 43 53 50 2E 41 55 54 2D 53 45 4C 46
26 30 0C
27 04 01
      F3
28 01 01
      FF
29 A1 04
30 03 02
      02 04
31 A1 07
      30 05
32 30 03
33 04 01
      0B

```

H.8 EF.DCOD

The “data container object directory” holds the references to the additional information files in HPA (“EF.HPD” with the health professional's personal data and “EF.DM” with the message to be displayed upon successful trusted channel establishment).

F.8.1 ASN.1 Value notation

```

1  iso7816DO : {
2      commonObjectAttributes {
3          label "EF.HPD"
4      },
5      classAttributes {
6          applicationName "Health Professional Application v2.0"
7      },
8      typeAttributes indirect :
9          path : {
10             efidOrPath '02'H
11         }
12     }
13
14 opaqueDO : {
15     commonObjectAttributes {
16         label "EF.DM",
17         flags { private, modifiable },
18         authid '02'H, -- pointer to the AOD-entry of PIN.HP.ASS
19     }
20 }

```

```

14     accessControlRules {
15         {
16             accessMode { read },
17             securityCondition or : {
18                 authId '02'H,      -- pointer to the AOD-entry of PIN.HP.ASS
19                 and : {
20                     authId : '03'H, -- pointer to the AOD-entry of the
                                     -- external/certificate authentication
                                     -- scheme
21                     authReference : {
22                         authMethod { secureMessaging, extAuthentication }
23                     }
24                 }
25             },
26         {
27             accessMode { update },
28             securityCondition authId : '02'H      -- pointer to the AOD-entry
29                                                     -- of PIN.HP.ASS
30         }
31     },
32     classAttributes {
33         applicationName "Health Professional Application v2.0"
34     },
35     typeAttributes indirect :
36     path : {
37         efidOrPath '03'H
38     }
39 }

```

H.8.2 ASN.1 Description, tags, lengths and values

```

1  DataContainerObjectChoice CHOICE
   iso7816DO SEQUENCE: tag = [0] constructed; length = 57
2   commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 8
3   label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 6
   0x45462E485044
4   classAttributes CommonDataContainerObjectAttributes SEQUENCE:
   tag = [UNIVERSAL 16] constructed; length = 38
5   applicationName Label UTF8String: tag = [UNIVERSAL 12] primitive;
   length = 36
   0x4865616c74682050726666657373696665616c204170706c696361746966652076322e30
6   typeAttributes : tag = [1] constructed; length = 5
   ISO7816DOAttributes CHOICE
   indirect ReferencedValue CHOICE
7   path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
8   0x02

9  DataContainerObjectChoice CHOICE
   opaqueDO SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 96
10  commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
   constructed; length = 47
11  label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 5
   0x45462E444D
12  flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
   0x06C0
13  authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
   0x02

```

```

14     accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed;
      length = 31
15     AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 20
16     accessMode AccessMode BIT STRING: tag = [UNIVERSAL 3] primitive;
      length = 2
      0x0780
17     securityCondition SecurityCondition CHOICE
      or SEQUENCE OF: tag = [2] constructed; length = 15
18     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
      length = 1
      0x02
19     and SEQUENCE OF: tag = [1] constructed; length = 9
20     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
      length = 1
      0x03
21     authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
      structed; length = 4
22     authMethod AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
      length = 2
      0x06C0
23     AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 7
24     accessMode AccessMode BIT STRING: tag = [UNIVERSAL 3] primitive;
      length = 2
      0x0640
25     securityCondition SecurityCondition CHOICE
      authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
      length = 1
      0x02
26     classAttributes CommonDataContainerObjectAttributes SEQUENCE:
      tag = [UNIVERSAL 16] constructed; length = 38
27     applicationName Label UTF8String: tag = [UNIVERSAL 12] primitive;
      length = 36
      0x4865616c74682050726f666657373696f6e616c204170706c6963617469666e2076322e30
28     typeAttributes : tag = [1] constructed; length = 5
      opaqueDOAttributes CHOICE
      indirect ReferencedValue CHOICE
29     path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
30     efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
      0x03

```

H.8.3 Hexadecimal DER-encoding

```

1  A0 39
2      30 08
3          0C 06
4              45 46 2E 48 50 44
5      30 26
6          0C 24
7              48 65 61 6c 74 68 20 50 72 6f 66 65 73 73 69 6f 6e 61 6c 20 41 70 70 6c
8              69 63 61 74 69 6f 6e 20 76 32 2e 30
          A1 05
              30 03
                  04 01
                      02

```

9 30 60
 10 30 2F
 11 0C 05
 45 46 2E 44 4D
 12 03 02
 06 C0
 13 04 01
 02
 14 30 1F
 15 30 14
 16 03 02
 07 80
 17 A2 0E
 18 04 01
 02
 19 A1 09
 20 04 01
 03
 21 30 04
 22 03 02
 06 C0
 23 30 07
 24 03 02
 06 40
 25 04 01
 02
 26 30 26
 27 0C 24
 48 65 61 6c 74 68 20 50 72 6f 66 65 73 73 69 6f 6e 61 6c 20 41 70 70 6c
 69 63 61 74 69 6f 6e 20 76 32 2e 30
 28 A1 05
 29 30 03
 30 04 01
 03

Annex I

(informative)

SMC/HPC Interaction

It is assumed, that an authentication procedure between the HPC and a security module card SMC in the IFD has been successfully completed. The authentication procedure is based e.g. on card verifiable certificates and has an inherit key transport or key agreement mechanism so that after this procedure

- a symmetric SM key for the computation of cryptographic checksums SK.CC,
- a symmetric SM key for the computation of cryptograms SK.CG and
- a send sequence counter SSC with its initial value is available in the USC and the SMC.

All commands to the HPC are sent in SM mode with bit b4 = 1 and b3 = 1 in the CLA byte, i.e. the command header will be integrated in the CC computation.

All commands sent to the SMC are PSO related commands, not in SM mode but using SM-DOs and the secure messaging keys set with an MSE command. The send sequence counter SSC is incremented each time before usage.

The general construction principle for secured command production and secured response processing is shown in Figure I.1 and I.2.

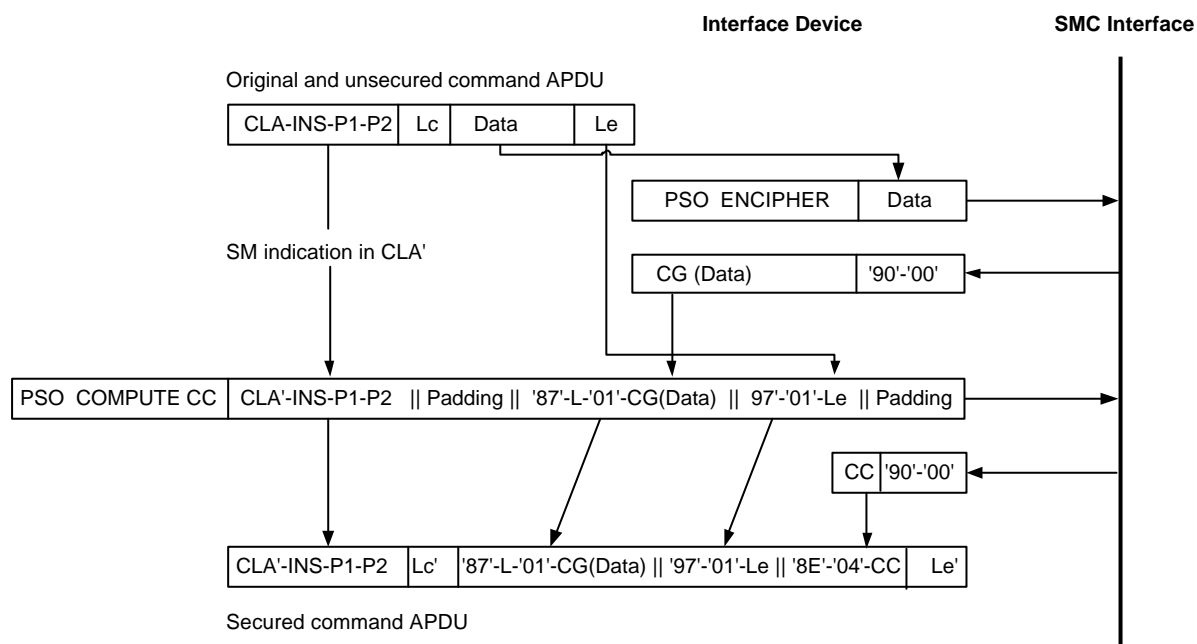


Figure I.1 – Example of secured command production

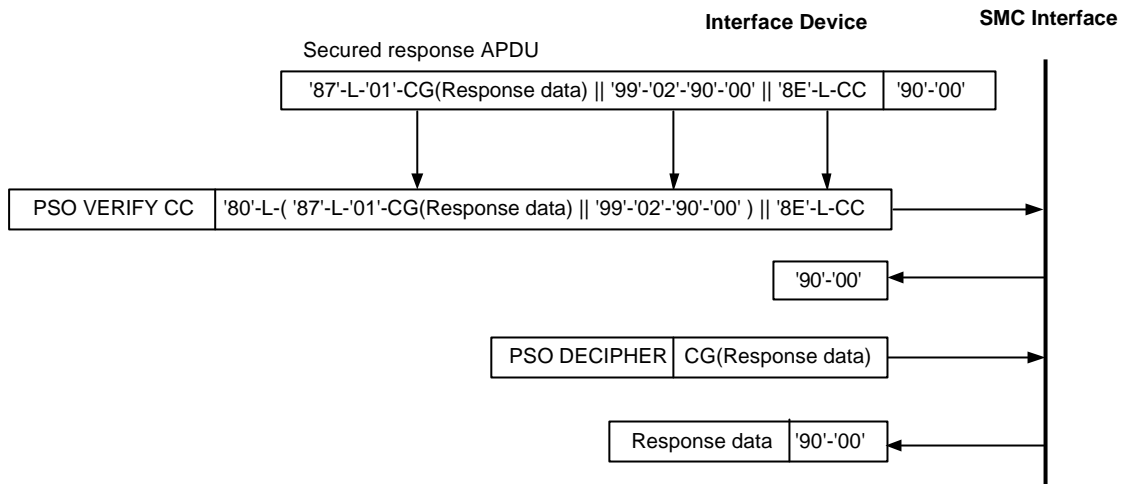


Figure I.2 – Example of secured response processing

The subsequent example presents the computation of a digital signature (DS) whereby the usage of the signature key requires the successful presentation of reference data (password).

Notation:

CLA' = Header with SM indication (b4 = 1, b3 = 1)

Step 1: Verification of reference data (password)

Command to SMC: MSE SET <CT, '83'-01-'F1'> -- In this example SK.CG has the key reference 'F1'
 SMC response: OK

Command to SMC: MSE SET <CCT, '83'-01-'F2'> -- In this example SK.CC has the key reference 'F2'
 SMC response: OK

Command to SMC: PSO ENCIPHER <RD>
 SMC response: <CG(RD)>

Command to SMC: PSO COMPUTE CC <CLA'-INS-P1-P2 - Padding - '87'-L-PI-CG(RD) - '97'-01'-Le - Padding>
 SMC response: <CC>

Now the IFD is able to construct the secured VERIFY command.

Command to USC: VERIFY <'87'-L-'01'-CG(RD) - '97'-01'-Le - '8E'-L-CC>
 USC response : <'99'-02'-SW - '8E'-L-CC>

Command to SMC: PSO VERIFY CC <'80'-L-('99'-02'-SW) - '8E'-L-CC>
 SMC response: OK

Step 2: Computation of a hash value

Command to SMC: PSO COMPUTE CC <CLA'-INS-P1-P2 - Padding - '81'-L-('90'-L-Intermediate Hash - '80'-L-Last block) - '97'-01'-Le - Padding>

SMC response: <CC>

Command to USC: PSO HASH <'81'-L-('90'-L- Intermediate Hash - '80'-L-Last block)> - '8E'-L-CC>

USC response : < '99'-'02'-SW - '8E'-L-CC>

Command to SMC: PSO VERIFY CC <'80'-L-('99'-'02'-SW) - '8E'-L-CC>

SMC response: OK

Step 3: Computation of a digital signature

Command to SMC: PSO COMPUTE CC <CLA'-INS-P1-P2 - Padding - '97'-'01'-'00'>

SMC response: <CC>

Command to USC: PSO: COMPUTE DS <'97'-'01'-'00' - '8E'-L-CC >

USC response : <'81'-L-DS - '8E'-L-CC>

Command to SMC: PSO VERIFY CC <'80'-L('81'-L-DS) - '8E'-L-CC>

SMC response: OK

Annex J

(informative)

Examples of Card-to-Card Communications

J.1 General

All types of considered cards can have different communication links. Several communication links may be coexistent and associated to different channels (#0-3). At a time only one command can be processed by a card.

Channel #0 is used for

- authentication for proving access rights to a PDC or HPC (access to EF.HPD)
- channel management
- usual HPA or SMA processing.

Channels #1-3 are used for

- remote access to HPCs via a trusted channel.

An HPC can always be used also in a stand-alone mode.

J. 2 Card-to-Card Communications for HPCs and SMCs

The subsequent figures show the card-to-card communications for the respective cards valid for different communication situations with usage of up to 4 channels.

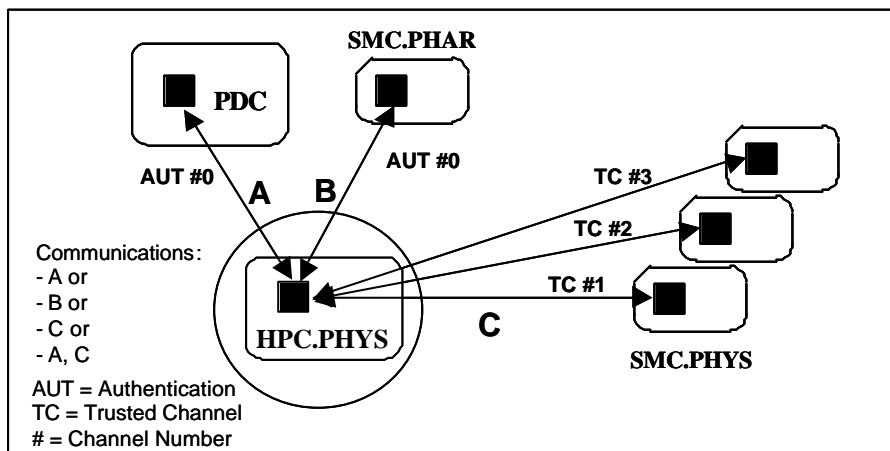


Figure J.1 - HPC.PHYS and card-to-card communications

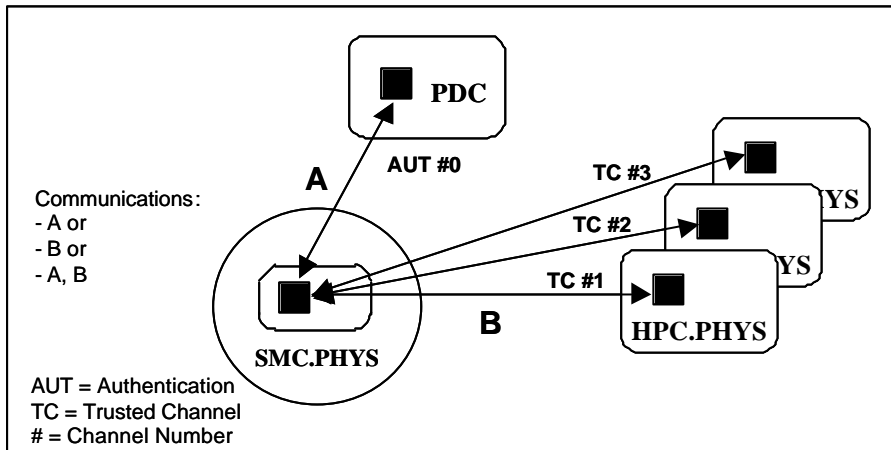


Figure J.2 - SMC.PHYS and card-to-card communications

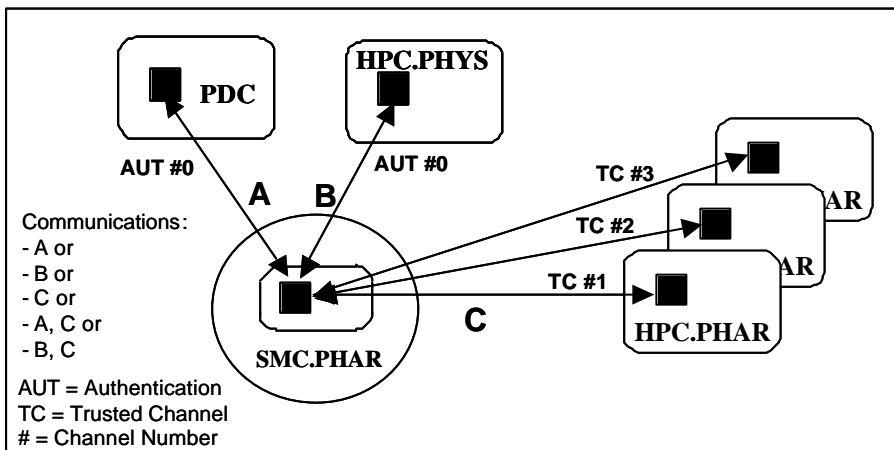


Figure J.3 - HPC.PHAR and card-to-card communications

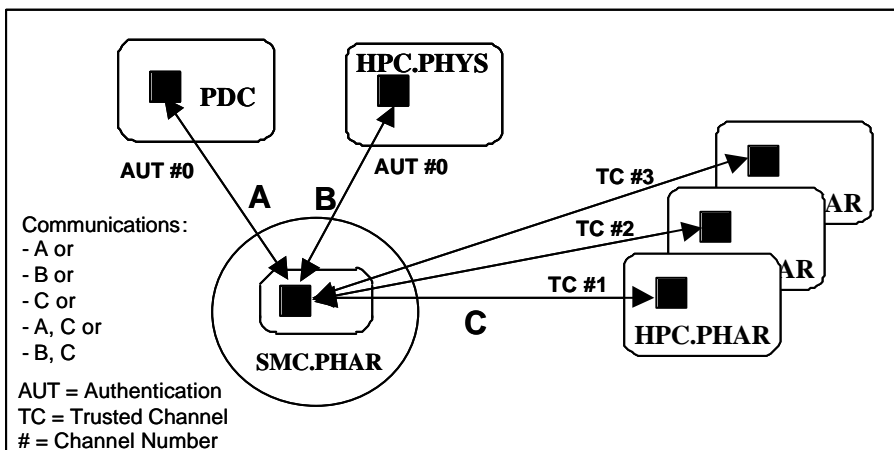


Figure J.4 - SMC.PHAR and card-to-card communications