Specification

German
Health Professional Card
and
Security Module Card

Part 2: HPC Applications and Functions

Version 2.1.0

21.02.2006

BundesÄrzteKammer

Kassenärztliche Bundesvereinigung

BundesZahnÄrzteKammer

BundesPsychotherapeutenKammer

Kassenzahnärztliche Bundesvereinigung

Werbe- und Vertriebsgesellschaft Deutscher Apotheker mbH

Deutsche Krankenhaus-Gesellschaft

Zentralinstitut für die kassenärztliche Versorgung in der BRD

Editor: Bruno Struif, SIT

The HPC specification consists of the following parts:

Part 1: Commands, Algorithms and Functions of the COS Platform
Part 2: HPC Applications and Functions
Part 3: SMC Applications and Functions

Revision History

Date	Version	Modifications
31.07.2003	HPC V2.0	New version
22.05.2004	HPC V2.0 Rev. 1 (Draft)	- Precision and corrections - SE selection additionally at MF level
01.09.2004	HPC V2.0 Rev. 1 (Pre-Final)	- Precision and corrections - Card-to-card authentication only asymmetric
08.05.2005	HPC V2.05 Draft	Updated Version, some modifications still to be done
25.05.2005	HPC V2.06	Changes: - References updated - File structure updated - AID.DINSIG added - Access rules partially updated - Certificate hierarchy updated - Hashing enhanced
20.06.2005	HPC V2.07	Changes: - ES-Certificate content updated - ATR revised (greater card I/O buffer) - SMC types introduced - File structure updated - Key length adopted - New Figure H.1
30.06.2005	HPC V2.08	Changes: - Clause 2 completely revised (now Clause 4) - Downloading of applications added
19.09.2005	HPC V2.09	Changes: - Harmonization with eGK - Splitting of original HPA in HPA, QES & ESIGN - Notation completed - DF.CIA.ESIGN adopted to the functionality of DF.ESIGN (only AUT and ENC)
07.10.2005	HPC V2.09	Changes: - Role Id added in Table F.2
19.11.2005	HPC V2.1 Draft	Changes: - Figures with authentication proc. moved to Part 1 - Download with asym. Authentication - HPD file added - Move of PIN management to MF level - Addition of CVC.HPC.TCE - Reduction of Public Keys to PuK.RCA.CS
14.12.2005	HPC V2.1	Changes: - Harmonization CVC with eGK - Role identifiers encoded with profile ID - Editorial improvments
21.02.2006	HPC V2.1.0	Changes: - Harmonization with eGK - some minor precisions and corrections

Contents

1 Scope

This part of the specification defines the card interface to

- the Health Professional Card (HPC) for individuals from accredited health professions

and authentication procedures between a HPC and an electronic Health Card (eGK) for verifying the authenticity of an eGK and proving access rights.

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 eSIGN specification for electronic signatures
- the relevant ISO-Standards (especially ISO/IEC 7816 Part 1-4, 6, 8, 9 and 15)
- other sources (e.g. requirements from trust centers).

The lifetime of a HPC is at least 3 years. A health professional may have more than one HPC.

2 References

[ALGCAT]
Geeignete Algorithmen zur Erfüllung der Anforderungen nach §17 Abs. 1 bis 3 SigG vom 22. Mai 2001 in Verbindung mit Anlage 1 Abschnitt I Nr. 2 SigV vom 22. November 2001, 30. März 2005, Bundesanzeiger Nr. 59, S. 4695-4696
See also www.bundesnetzagentur.de

[BÄK-HBA-Ausgabe]
Bundesärztekammer
Ausgabe Heilberufsausweis (HBA)
Fachfeinkonzept, 200503

[BÄK-HBA-Lastenheft]
Bundesärztekammer
Lastenheft zur Spezifikation der HPC/SMC
Version 0.3.9 vom 05.09.2005

[CWA14890-1]
“eSIGN Specification”
Application Interface for smart cards used as Secure Signature Creation Devices
Part 1 – Basic Requierements
March 8th 2004

[CWA14890-2]
”eSIGN Specification”
Application Interface for smart cards used as Secure Signature Creation Devices
Part 2 – Additional services
March 12th 2004

[DIN66291-1]
DIN V66291-1: 2000
Chipkarten mit Digitaler Signatur-Anwendung/Funktion nach SigG und SigV
Teil 1: Anwendungsschnittstelle

[DIN66291-4]
DIN V66291-4: 2002
Chipkarten mit Digitaler Signatur-Anwendung/Funktion nach SigG und 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

[eGK-P1]
Die Spezifikation der elektronischen Gesundheitskarte
Teil 1 – Kommandos, Algorithmen und Funktionen der Betriebssystem-Plattform
V1.1.0, 07.02.2006

[eGK-P2]
Die Spezifikation der elektronischen Gesundheitskarte
Teil 2 – Anwendungen und anwendungsspezifische Strukturen
V1.1.0, 07.02.2006

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

[HPC-P1]
German Health Professional Card and Security Module Card

Part 1: Commands, Algorithms and Functions of the COS Platform
V2.1.0 21.02.2006

[HPC-P3]
German Health Professional Card and Security Module Card
Part 3: SMC Applications and Functions
V2.1.0 21.02.2006

[ISIS-MTT OP]
T7, TeleTrusT: ISIS-MTT Specification, Optional Profile "SigG-Profile", Version 1.1, 16th March 2004, www.teletrust.de

[ISIS-MTT P1]
T7, TeleTrusT: ISIS-MTT Specification, Part 1 "Certificate and CRL Profiles", Version 1.1, 16th March 2004, www.teletrust.de

[ISO3166]
ISO/IEC 3166: Codes for the representations of names of countries

[ISO7812]
ISO/IEC 7812-1:2000
Identification cards – Identification of issuers – Part 1: Numbering system

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

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

[ISO7816-3]
ISO/IEC 7816-3: FCD2 2005 (2nd edition)
Identification cards - Integrated circuit cards with contacts -
Part 3: Electrical interface and transmission protocols

[ISO7816-4]
ISO/IEC 7816-4: 2005 (2nd edition)
Identification cards - Integrated circuit cards -
Part 4: Organization, security and commands for interchange

[ISO7816-5]
ISO/IEC 7816-5: 2004 (2nd edition)
Identification cards - Integrated circuit cards -
Part 5: Registration of application providers

[ISO7816-6]
ISO/IEC 7816-6: 2004 (2nd edition)
Identification cards - Integrated circuit cards -
Part 6: Interindustry data elements for interchange

[ISO7816-8]
ISO/IEC 7816-8: 2004 (2nd edition)
Identification cards - Integrated circuit cards -
Part 8: Commands for security operations

[ISO7816-9]
ISO/IEC 7816-9: 2004 (2nd edition)
Identification cards - Integrated circuit cards -

Part 9: Commands for card management

[ISO7816-13]
ISO/IEC 7816-13: CD2 2005
Identification cards - Integrated circuit cards -
Part 13: Commands for application management in multi-application environment

[ISO7816-15]
ISO/IEC 7816-15: 2004
Identification cards - Integrated circuit cards -
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: 2002, Information technology – Security techniques – Digital signature schemes giving message recovery –
Part 2: Integer factorization based mechanisms

[ISO10118]
ISO 10118-2, Information technology – Security techniques – Hash functions, Part 2: Hash functions using an n-bit block cipher algorithm, 2000

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

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

[NIST-SHS]
NIST: FIPS Publication 180-2:
Secure Hash Standard (SHS-1),
01.08.2002

[PKCS#1]
PKCS #1 RSA Cryptography Standard
V2.1: June 14, 2002

[PP-HPC]
Common Criteria Protection Profile – Health Professional Card (HPC)
BSI-PP-018, 05.12.2005

[Resolution190]
Beschluss Nr. 190 der Europäischen Union vom 18. Juni 2003 betreffend die technischen Merkmale der europäischen Krankenversicherungskarte

[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

[SigG01]
Law Governing Framework Conditions for Electronic Signatures and Amending Other Regulations (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, Bundesgesetzblatt Nr. 509, 2001, S. 3074

[SSL]
Netscape:
SSL3.0 Specification

3 Abbreviations and Notations

3.1 Abbreviations

AC = Attribute Certificate
AID = Application Identifier
AM = Access Mode
AOD = Authentication Object Directory
ARR = Access Rule Reference
ASN.1 = Abstract Syntax Notation One
AT = Authentication Template
ATR = Answer-to-Reset
AUT = Authentication
AVS = Apothekenverwaltungssystem
BCD = Binary Coded Decimal
BÄK = Bundesärztekammer
BER = Basic Encoding Rules
BNA = BundesNetzAgentur
C = Certificate
CAMS = Card Application Management System
C2C = Card-to-Card
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)
CT = Confidentiality Template
CVC = Card Verifiable Certificate
CWA = CEN Working Agreement
D, DIR = 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
eGK = elektronische GesundheitsKarte (electronic Health Card)
EHIC = European Health Insurance Card
ENC = Encipherment
EOF = End-of-File
FCI = File Control Information
FI = Clock rate conversion factor
FID = File Identifier
FM = File Management
HB = Historical Bytes
HF2 = Hash Function ISO 10118-2
HI = Health Institution
HP = Health Professional
HPA = Health Professional Application
HPC = Health Professional Card

ICC = Integrated Circuit Card
ICCSN = ICC Serial Number
ICM = Integrated Circuit Manufacturer
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
KID = Key Identifier
KIS = Krankenhaus-Informations-System
LSB = Least Significant Byte(s)
MF = Master File
MII = Major Industry Identifier
MSE = MANAGE SECURITY ENVIRONMENT
OID = Object Identifier
P = Patient
PHAR = Pharmacist
PHYS = Physician
PK,PuK = Public Key
PI = Padding Indicator
PIN = Personal Identification Number
PIX = Proprietary Appl. Prov. Extension
PP = Protection Profile
PPS = Protocol Parameter Selection
PrK = Private Key
PRND = Padding Random Number
PSO = PERFORM SECURITY OPERATION
PUK = Personal Unblocking Key (= Resetting Code)
PVS = Praxis-Verwaltungssystem
QES = Qualified Electronic Signature
R = Role ID
RC = Retry Counter
RCA = Root CA
RD = Reference Data
RFC = Request for Comment
RID = Registered Application Provider ID
RND = Random Number
ROM = Read Only Memory
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
ZGW = CA for heath care (Zertifizierungsstelle Gesundheitswesen)

3.2 Notations

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

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

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

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

<private key>::= PrK (asym.)
<public key>::= PuK (asym.)
<secret key>::= SK (sym., not used)
<individual key>::= IK (sym., not used)

<keyholder>::= <health professional> | <card holder> | <certification authority> | <health professional card> | <electronic health card> | <security module card> | <server>

<health professional> ::= HP
<card holder> ::= CH
<certification authority>::= RCA | CA | CA_NN
<health professional card> ::= HPC
<electronic health card> ::= eGK (elektronische GesundheitsKarte)
<security module card> ::= SMC
<server> ::= S

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

<qualified electronic signature>::= QES
<encipherment>::= ENC
<authentication>::= AUT
<certsign>::= CS

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

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

<certificate>::= <X.509v3 certificate> | <card verifiable certificate>
<X.509v3 certificate>::= C
<card verifiable certificate>:: CVC

<certificate holder>::=
<health professional> | <certification authority> | <health professional card> | <security module card> | <server>

<certificate usage>::= <qualified electronic signature> | <encipherment> | <authentication> | <certsign>

<qualified electronic signature>::= QES | QES-ACx
<encipherment>::= ENC
<authentication>::= AUT
<certsign>::= CS

For subsequent data items the following notation is used:

|| = Concatenation of data

For simplification X.509v3 certificates are addressed without version number.

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

4.1 Technical Characteristics

HPCs are contact based smartcards 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.

A HPC is a normal size card (ID-1 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.

4.2 Answer-to-Reset

The coding for the ATR is shown in Annex A.

4.3 Transmission Protocols

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

5 The Health Professional Card

5.1 General Structure

The HPC contains

- some EFs at MF level for general data objects, CV certificates and global keys for authentication procedures (proving access rights to the eGK and verification of the authenticity of the eGK)

- the Health Professional Application (HPA) for the provision of health professional related data file(s)

- the Qualified Electronic Signature Application (QES) for signature computations

- the ESIGN application for
  - client/server authentication
  - document decipherment

- the Cryptographic Information Application (CIA) related to the ESIGN application as required by [CWA 14890-1].

The general structure is shown in Figure 1.

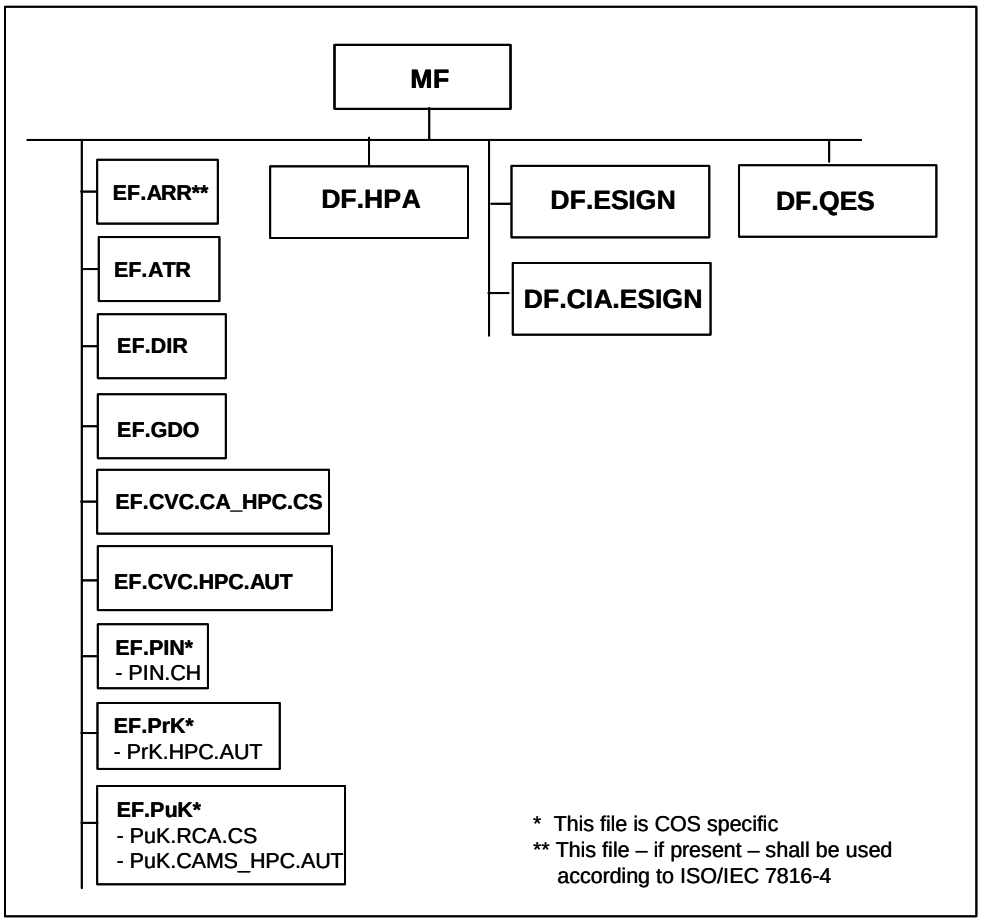

Figure 1 – General file structure of a HPC

NOTE – Currently no DF.CIA.QES is needed.

5.2 Elementary Files at MF-Level

5.2.1 EF.ARR

EF.ARR contains the access rules relevant at MF level.

5.2.2 EF.ATR

The transparent file EF.ATR contains a constructed data object for indication of I/O buffer sizes and the DO 'Pre-issuing data' relevant for CAMS services. The content of EF.ATR is specified in Table G.1.

5.2.3 EF.DIR

EF.DIR contains the application templates for DF.HPA, DF.QES, DF.ESIGN and DF.CIA.ESIGN according to ISO/IEC 7816-4. EF.DIR allows the addition of AIDs of further (downloaded) applications. The content of EF.DIR is specified in Table G.3.

5.2.4 EF.GDO

EF.GDO contains in compliance with [Resolution190] the DO ICC Serial Number (ICCSN, Tag '5A', see Figure 4).

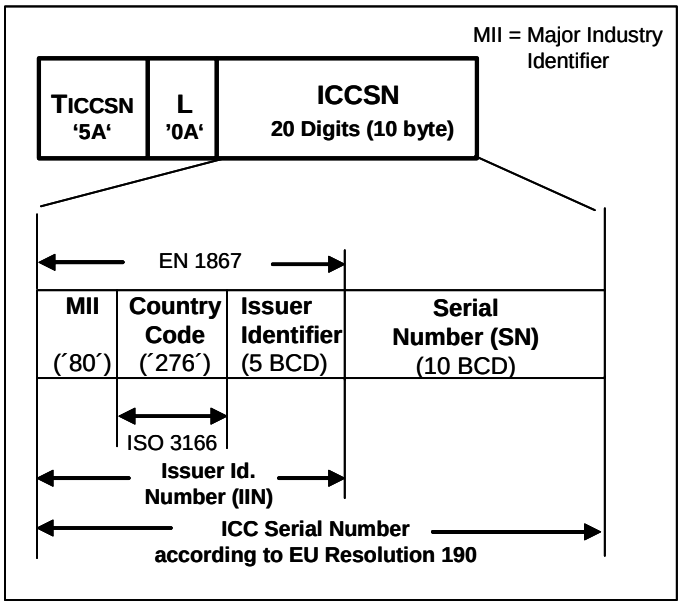

Figure 2 - ICC Serial No. for health cards

The Issuer Identification Numbers used for HPCs are outlined in Annex F.

In this specification, the ICCSN is used for

- card identification, e.g. by a CAMS
- key referencing in CVCs.

It is the responsibility of the card issuer to ensure that the serial number (SN) is unique (especially in the case that several card manufacturers are involved).

5.2.5 EF.CVC.CA_HPC.CS

EF.CVC.CA_HPC.CS contains the card verifiable certificate of the Certificate Service Provider, issued by the Root CA for Health Care for a CA_NN_HPC (NN e.g. CA for physicians or pharmacists). Structure and content of the CVC are outlined in [HPC-P1].

5.2.6 EF.CVC.HPC.AUT

EF.CVC.HPC.AUT contains the card verifiable certificate of the HPC for card-to-card authentication between HPC and eGK or HPC and SMC. Structure and content of the CVC are outlined in [HPC-P1].

5.2.7 EF.PIN

EF.PIN contains the global PIN of the cardholder (PIN.CH). The PIN consists of at least 5 digits and is changeable. The retry counter shall have the initial value 3.

The usage of the 8 digit resetting code is limited by a usage counter with an initial value of 10. The usage counter is decremented independent whether the resetting code was correct or not. The successful presentation resets the retry counter of the related PIN.CH.

The PIN characteristics are shown in the following table.

Table 1 – PIN characteristics

PIN Name	PIN Length	PIN Ref.	Initial Value of Retry Counter	Resetting Code (PUK)	Initial Value of the PUK Usage Counter
PIN.CH	5 - 8 digits	‘01’	3	8 digits	10

5.2.8 EF.PrK

EF.PrK contains

- the global private key PrK.HPC.AUT for C2C-authentications (HPC/eGK and HPC/SMC).

The key characteristics are shown in Table 2.

Table 2 – Key characteristics of PrK.HPC.AUT

Key Name	Key length	KeyID	SE #	Access Conditions
PrK.HPC.AUT (for HPC/eGK authentication without TC establishment and HPC authentication for SMC authorization)	1024 bit (RSA)	‘10’ see Note 1	'01'	PIN.CH
PrK.HPC.AUT (for HPC/SMC authentication with TC establishment)	1024 bit (RSA)	‘11’ see Note 1	'02'	Always, see Note 2

NOTES -
1. In cards supporting multi-KID-referencing of a key object (i.e. the same key has more than one KID), the key is only present once. In other cards the key may be stored twice. The purpose of the key is bound to the KID. Therefore the AlgID is implicit.
2. SE # 2 cannot be misused for HPC/eGK interactions, since SE # 2 uses SM. Since the HPC does not allow the production of secured commands to be sent to an eGK, no access to sensitive data in an eGK is possible, i.e.

access to an eGK is only possible in SE # 1, where a successful PIN.CH presentation by the health professional is required.

The public key associated with PrK.HPC.AUT is contained in CVC.HPC.AUT.

5.2.9 EF.PuK

EF.PuK contains

- the public key PuK.RCA.CS of the Root CA for Health Care for verification of CVCs issued by this RCA
- the public key PuK.CAMS_HPC.AUT for performing an HPC/CAMS authentication procedure with TC establishment.

The key identifier of PuK.RCA.CS can be retrieved from the HPC as described in Figure 3. The key identifier of PuK.CAMS_HPC.AUT is shown in the subsequent table.

Table 3 – Key characteristics of PuK.CAMS_HPC.AUT

Key Name	Key length	KeyID	AlgID
PuK.CAMS_HPC.AUT	1024 bit (RSA)	'13'	'1F'

PuK.CAMS_HPC.AUT is a global key and has always the same key reference at the interface to the HPC, but the key value itself may depend e.g. on the year of issuing the HPC, see Clause 12.2.

5.3 Security Environments at MF Level

At MF level the following SEs are used:

- SE # 1 is the general SE at MF level and used for all purposes except the one dedicated to SE # 2
- SE # 2 is dedicated to the usage of PrK.HPC.AUT for trusted channel establishment between HPC and SMC.

5.4 HPC Opening

5.4.1 Command Sequence after ATR

The command sequence to be performed after ATR 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, it may not be necessary to read the ICCSN and to identify with it e.g. a card profile containing e.g. the HPC related CV certificates. In third party environments, where a respective HPC is not known, it may be necessary first to read EF.DIR and EF.GDO and possibly also the Cryptographic Information Objects contained in the EFs of DF.CIA.ESIGN.

Subsequent the reading of EF.DIR and EF.GDO is described.

5.4.2 Reading of EF.ATR and EF.GDO

For reading EF.ATR and EF.GDO the ISO/IEC 7816-4 command READ BINARY is used.

Table 4 - READ BINARY command with SFID

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1	'xx' = b8-b6: 100 b5-b1: 11101 SFID of EF.ATR: 29 b5-b1: 00010 SFID of EF.GDO: 2
P2	'00' = Offset
Lc	Absent
Data field	Absent
Le	'00' = Read until end-of-file

Table 5 - READ BINARY response

Data field	Data
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

5.4.3 Reading EF.DIR

For reading EF.DIR, the ISO/IEC 7816-4 READ RECORD command is used.

Table 6 - READ RECORD Command

CLA	As defined in ISO/IEC 7816-4
INS	'B2' = READ RECORD
P1	'xx' = Record no.
P2	'F4' = b8-b4: 11110 SFID of EF.DIR: 30 b3-b1: 100 Read record P1
Lc	Absent
Data field	Absent
Le	'00' = Complete record

NOTE – In subsequent commands, the Short EFID may be set to zero (P2 = '04').

Table 7 – READ RECORD Response

Data field	Record
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

5.4.4 Reading HPC related CV Certificates

For reading the certificates CVC.CA_HPC.CS and CVC.HPC.AUT the READ BINARY command is used. The HPC related CV certificates should be stored in the health professional´s PC environment, so that this operation has to be performed only once for saving time.

Table 8 - READ BINARY command for reading a CV certificate

CLA	As defined in ISO/IEC 7816-4
INS	‘B0’ = READ BINARY
P1	‘8x’ = b8-b6:100 b5-b1: 00011 SFID of EF.CVC.HPC.AUT: 3 b5-b1: 00100 SFID of EF.CVC.CA_HPC.CS: 4
P2	‘00’ = Offset
Lc	Absent
Data field	Absent
Le	‘00’ = Read until end-of-file (max 256 B)

Table 9 - READ BINARY response

Data field	CV certificate
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

5.4.5 SE Selection at MF Level

If the usage of SE # 2 is necessary (see Clause 11.2), the ISO/IEC 7816-4 MSE command has to be sent to the HPC.

Table 10 - MSE command for setting SE # 2

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘F3’ = RESTORE
P2	‘02’ = SE # 2
Lc	Absent
Data field	Absent
Le	Absent

Table 11 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

5.5 PIN Management

5.5.1 PIN Verification

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

Table 12 - VERIFY command for PIN verification

CLA	As defined in ISO/IEC 7816-4
INS	‘20’ = VERIFY
P1	‘00’
P2	‘01’ = PIN.CH reference
Lc	‘08’ = Length of subsequent data field
Data field	PIN (PIN format: see [HPC-P1])
Le	Absent

Table 13 - VERIFY response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

In case, the PIN was wrong and retries are left, the number of further allowed retries shall be indicated in the status code as specified in [HPC-P1].

5.5.2 PIN Change

For PIN change the ISO/IEC 7816-4 command CHANGE RD may be used at any time convenient for the HP. If the command is successfully completed, then the security status “PIN with reference '01' successfully presented” shall be set.

Table 14 - CHANGE RD command

CLA	As defined in ISO/IEC 7816-4
INS	‘24’ = CHANGE REFERENCE DATA
P1	‘00’ = Exchange reference data
P2	‘01’ = PIN.CH reference
Lc	‘10’ = Length of subsequent data field
Data field	PIN_old || PIN_new (PIN format: see [HPC-P1])
Le	Absent

NOTE – Whether a security status is set, if the transport PIN is presented, depends on the transport PIN mechanism of the respective HPC.

Table 15 - CHANGE RD response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

5.5.3 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.CH, the RESET RETRY COUNTER command is used.
The command does not set or influence the security status, i.e. a VERIFY command is still needed.

Table 16 – RESET RC command for RC reset and possibly setting a new PIN

CLA	As defined in ISO/IEC 7816-4
INS	'2C' = RESET RETRY COUNTER
P1	'00' or '01'
P2	'01' = PIN.CH reference
Lc	'10' or '08' = Length of subsequent data field
Data field	- If P1 = '00' and P2 = '01': Resetting code (8 byte) followed by a new PIN (8 byte) - If P1 = '01': Resetting code (8 byte) (RC and PIN format: see [HPC-P1])
Le	Absent

Table 17 - RESET RC response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

5.6 HPC/eGK Authentication

5.6.1 General

For the HPC/eGK interaction two actions are required:

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

Therefore, a CV based authentication procedure has to be performed, so that in the eGK the related security status can be set, i.e. "certificate holder authorization y (see Annex F) has been successfully presented".
In the authentication procedure, no SM keys are established, since in the subsequent communication with the eGK (reading/writing data) a HPC is not involved.

The HPC has to ensure that the HPC/eGK-authentication procedure can only be performed after successful PIN.CH presentation.

5.6.2 CVC Key Management Authorities

The general model for CV Certificates and CVC Key Management Authorities is shown in Figure 3. The figure denotes especially, where the key identifier are found to be set for CVC verification and the subsequent authentication procedure.

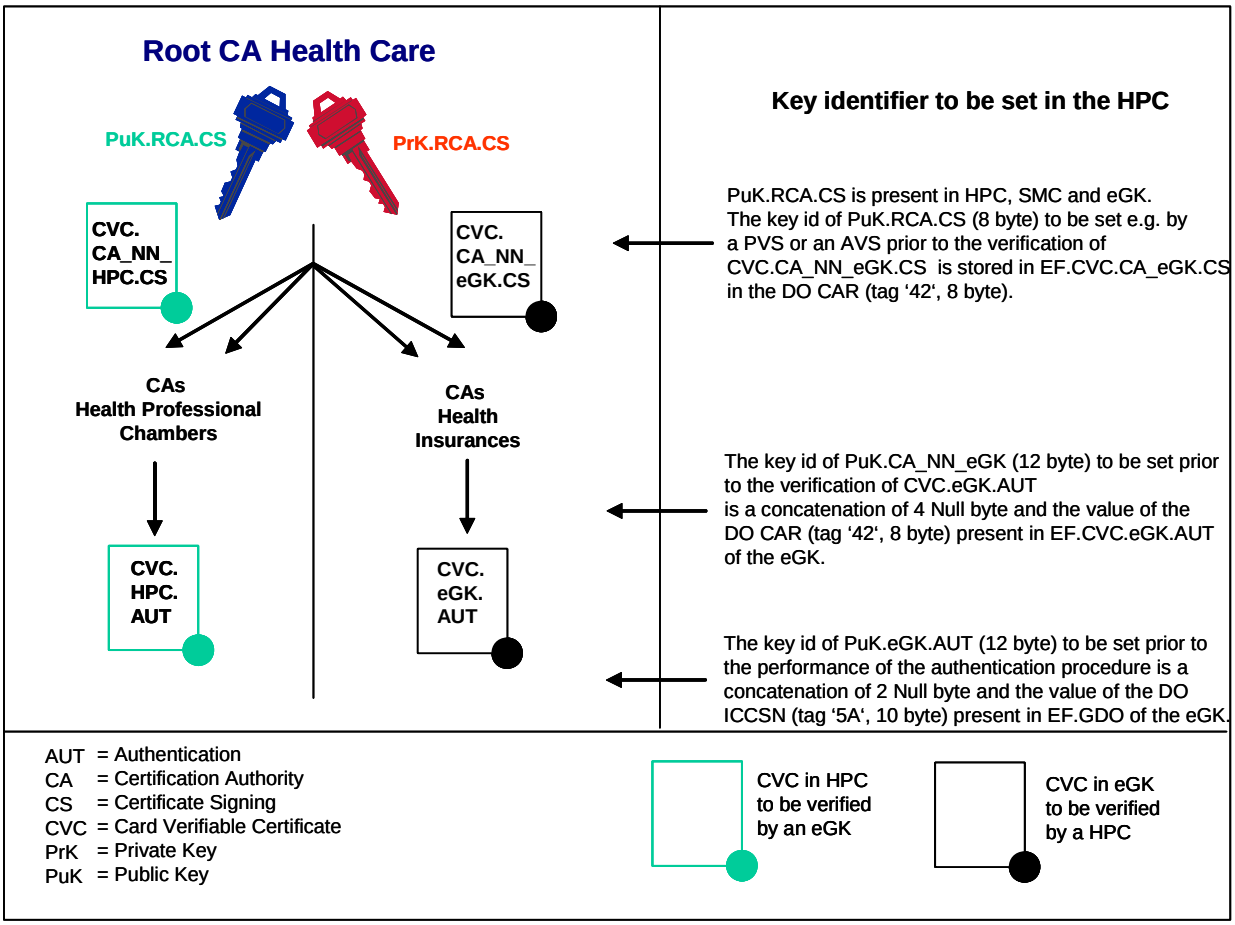

Figure 3 – CVC Key Management Authorities, CV Certificates and Key Selection in a HPC

In case that an eGK is presented with a new Public Root Key, cross certificates are needed verifiable with the Public Root Key present in the HPC (a cross certificate contains the new PuK.RCA.CS and is verifiable with the PuK.RCA.CS in the HPC). Cross certificates are provided by the Root CA and are retrievable e.g. by the connector of a PVS/AVS/KIS.
Since the CVC based authentication procedure allows cross border usage, cross certificates would be also applicable for dynamically importing the Public Root Key of another country. With this imported key the verification of the authenticity of an intelligent European Health Insurance Card EHIC would be possible without any change or addition in the HPC (the verification of the authenticity of an EHIC may become relevant in short time, since an EHIC with just a printed image can be faked easily and may cause financial damage for health care service providers).

5.6.3 Verification of eGK related CV Certificates

At first, the public key of the common Root CA has to be selected with the MSE command. The key reference can be found by the software (e.g. a PVS) as shown in Figure 3.

Table 18 - MSE command for selecting the Root PuK

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.RCA.CS (for retrieval of the KeyRef see Figure 3)
Le	Absent

Table 19 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After the Root PuK for verification of the CVC.CA_eGK.CS is set, then the PSO: VERIFY CERTIFICATE command is sent to the HPC. The data field contains the signature (covering the first part of the CV certificate, the plain value of which is recovered after processing the signature) and the PK remainder (presenting the final part of the CVC, see [HPC-P1].

Table 20 - PSO: VERIFY CERTIFICATE command for verifying the higher level eGK CVC

CLA	As defined in ISO/IEC 7816-4
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.RCA || ‘5F38’-L-PK_remainder
Le	Absent

Table 21 - PSO: VERIFY CERTIFICATE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

In the HPC the public key for verifying the CVC.eGK.AUT is now available and has to be selected.

Table 22 - MSE command for selecting the PuK of CA_NN_eGK

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for verification
P2	‘B6’ = DST
Lc	‘0E’= Length of subsequent data field
Data field	‘83 0C ...’ = DO for KeyRef of PuK.CA_NN_eGK.CS (for retrieval of the KeyRef see Figure 3)
Le	Absent

Table 23 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

Now the CVC.eGK.AUT can be verified.

Table 24 - PSO: VERIFY CERTIFICATE command for verifying the CVC.eGK.AUT

CLA	As defined in ISO/IEC 7816-4
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 || ‘5F38’-L-PK_remainder
Le	Absent

Table 25 - PSO: VERIFY CERTIFICATE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

5.6.4 Performing the Authentication Procedure

Before the authentication procedure is performed, PIN.CH must have been successfully presented.

In a first step, the key references for PuK.eGK.AUT and PrK.HPC.AUT are set.

Table 26 - MSE command for key selection

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 || 84 01 10’ = DO for KeyRef of PuK.eGK.AUT (for retrieval of the KeyRef see Figure 3) || DO for KeyRef of PrK.HPC.AUT
Le	Absent

NOTE – The key reference has a length of 12 byte: '0000' (2 byte) || ICCSN.eGK (10 byte)

Table 27 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After that a challenge is retrieved from the HPC.

Table 28 - GET CHALLENGE command

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

Table 29 - GET CHALLENGE response

Data field	RND.HPC (8 byte)
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After the GET CHALLENGE command follows an EXTERNAL AUTHENTICATE command, which delivers the digital signature of the eGK to the HPC. To receive this digital signature, an INTERNAL AUTHENTICATE command with RND.HPC || ICCSN8.HPC in the data field (see [HPC-P1], Annex E.2) has to be sent to the eGK. The HPC has to verify the eGK signature.

Table 30 - EXT. AUTHENTICATE command for eGK authentication

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, see [HPC-P1], Annex E.2
Le	Absent

Table 31 - EXT. AUTHENTICATE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After the eGK has been authenticated, the HPC has to prove its access rights to the eGK. The software system will require a challenge from the eGK prior to sending the subsequent commands.

Table 32 - INT. AUTHENTICATE command

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	Authentication related data, see [HPC-P1], Annex E.2
Le	‘00’ or ‘xx’ = length of expected signature

Table 33 - INT. AUTHENTICATE response

Data field	Authentication related data, see [HPC-P1], Annex E.2
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

The authentication token (content of the response data field) will be verified in the related eGK and sets in the eGK the security status “CHA with role/profile ID 'xx' successfully presented”.

5.7 Authorization of a SMC to interact with an eGK

According to [GMG], a health professional can authorize other dedicated persons to access an eGK. This is achieved by using a SMC. The usage of the PrK.SMC.AUT for an SMC/eGK authentication procedure requires in the access rule of PrK.SMC.AUT the successful authentication of an HP (for further details see [HPC-P3]).

In a first step, the PrK.HPC.AUT has to be selected.

Table 34 - MSE command

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 10’ = DO for KeyID of PrK.HPC.AUT
Le	Absent

Table 35 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After that the INTERNAL AUTHENTICATE command is send to the HPC. Before this step, a challenge (RND.SMC) is requested from the SMC.

Table 36 - INT. AUTHENTICATE command for proving the authenticity of a HPC for authorization of a SMC

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	Authentication related data, see [HPC-P1], Annex E.2
Le	'00' or 'xx' = length of expected signature

Table 37 - INT. AUTHENTICATE response

Data field	Authentication related data, see [HPC-P1], Annex E.2
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

The digital signature produced by the HPC for authentication will be verified by the SMC, which sets – if authentication successful – the security status required for the usage of the PrK.SMC.AUT for interactions with an eGK.

6 Channel Management

6.1 General Aspects

The support of several logical channels by a HPC is mandatory, see [HPC-P1]. If the HPC is used in an environment, where only one channel is used, then the basic channel with no. 0 shall be taken for the processing of all commands.
Each channel has its own independent security status, i.e. the presentation of e.g. a global PIN in one channel does not set a security status in any other channel.

6.2 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.

Table 38 – MANAGE CHANNEL command for logical channel selection

CLA	As defined in ISO/IEC 7816-4
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 39 - MANAGE CHANNEL response

Data field	- Logical channel no. assigned by the HPC (1 byte) - Absent in case that no further channel is available
SW1-SW2	- ‘9000’, if channel no. assigned - ‘6881’ Further logical channel not available or specific status bytes, see [HPC-P1]

6.3 Closing a Logical Channel

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

Table 40 – MANAGE CHANNEL command for closing a logical channel

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 41 – MANAGE CHANNEL response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

7 The Health Professional Application HPA

7.1 File Structure and File Content

The file structure of DF.HPA is shown in Figure 4.

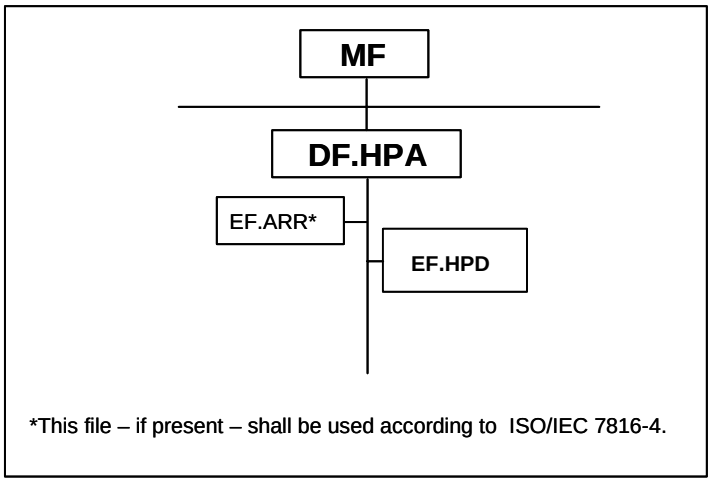

Figure 4 - File structure of DF.HPA

The keys and CVCs for the authentication procedure are placed at the MF level. The HPA allows the installation of further files due to possibly upcoming needs, see Clause 12.

7.1.1 EF.ARR

EF.ARR contains the access rules related to DF.HPA.

7.1.2 EF.HPD (Health Professional related Data)

The transparent file EF.HPD is intended to be used for storing HP related information, e.g. training confirmation information. The file can be read always, but update is only possible after successful presentation of PIN.CH.

7.2 Security Environments

In DF.HPA, only the default SE # 1 is used.

7.3 Application Selection

The application selection is performed with the ISO/IEC 7816-4 SELECT command as shown in the subsequent two tables.

Table 42 - SELECT command for DF.HPA

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

NOTE – In HPC version 1.0, the AID of the HPA was ‘D27600004001’. The RID ‘D276000040’ is assigned to the German Health Care.

Table 43 - SELECT response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

7.4 Reading and Updating of EF.HPD

For reading EF.HPD the READ BINARY command is used.

Table 44 - READ BINARY command for reading EF.HPD

CLA	As defined in ISO/IEC 7816-4
INS	‘B0’ = READ BINARY
P1	'81' = b8-b6:100 b5-b1: 00001 SFID of EF.HPD: 1
P2	‘00’ = Offset
Lc	Absent
Data field	Absent
Le	'08'

Table 45- READ BINARY response

Data field	Data
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

Update of EF.HPD is possible with the UPDATE BINARY command after successful presentation of PIN.CH.

Table 46 - UPDATE BINARY command for updating EF.HPD

CLA	As defined in ISO/IEC 7816-4
INS	‘D6’ = UPDATE BINARY
P1	'81' = b8-b6:100 b5-b1: 00001 SFID of EF.HPD: 1
P2	‘00’ = Offset
Lc	‘xx’ = Length of subsequent data field
Data field	Data
Le	Absent

Table 47 - UPDATE BINARY response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

8 The Qualified Electronic Signature Application

8.1 File Structure and File Content

The general structure of the QES application, which is in accordance with [DINSIG], is shown in Fig. 5.

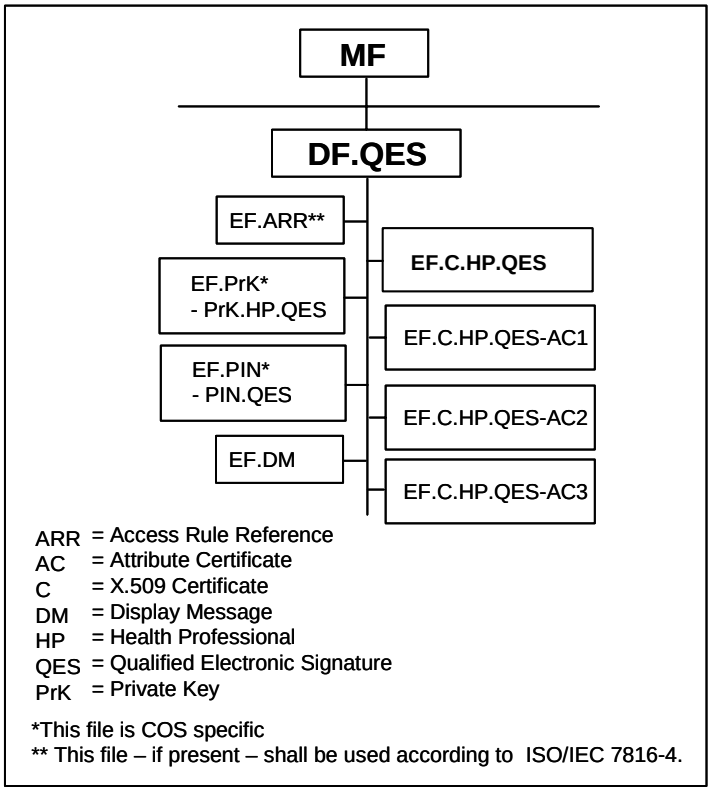

Figure 5 – General structure of the QES application

There are EFs for the X.509-QES certificate and up to 3 attribute certificates and an EF for a display message relevant to the remote usage of a HPC.

8.1.1 EF.ARR

EF.ARR contains the access rules related to DF.QES.

8.1.2 EF.PrK

EF.PrK contains the private key PrK.HP.QES for the computation of qualified electronic signatures. The key identifier and the key access conditions are shown in the subsequent table. The PIN characteristics are outlined in Clause 8.1.3.

Table 48 – Key characteristics

Key Name	Key length	KeyID	Access Conditions	Security Status Evaluation Counter of PrK.HP.QES
PrK.HP.QES	According to [ALGCAT]	'84'	- SE # 1: PIN.QES - SE # 2: PIN.QES and SM	Unrestricted number of signatures after successful presentation of PIN.QES possible (value COS specific), see note

NOTE – If required by BNA, a concrete value for the security status evaluation counter will be used.

The generally allowed padding schemes are presented in Annex E, but only the padding scheme indicated in the X.509 QES certificate will be used.

8.1.3 EF.PIN

EF.PIN contains the DF-specific PIN.QES, which is only used for the protection of the SigG/SigV-related private electronic signature key of the health professional (PrK.HP.QES).

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

Table 49 – PIN characteristics

PIN Name	PIN Length	PIN Ref.	Initial value of Retry Counter	Resetting Code (PUK)	PUK Usage Counter
PIN.QES	6 - 8 digits	'81'	3	8 digits	10

The initialization of PIN.QES e.g. by using a transport PIN is COS dependent and has to comply with BNA regulations.

8.1.4 EF.DM

The transparent file 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 after successful presentation of PIN.CH.

8.1.5 EF.C.HP.QES

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

8.1.6 EF.C.HP.QES-AC1, -AC2 and -AC3

The transparent files EF.C.HP.QES-AC1, -AC2 and -AC3 may contain an X.509v3 attribute certificate e.g. from a profession chamber (e.g. physicians chamber, pharmacists chamber) or professional organizations (e.g. doctors association).

8.2 Creation of a Qualified Electronic Signature

An electronic signature process consists of several steps as shown in Figure 6.

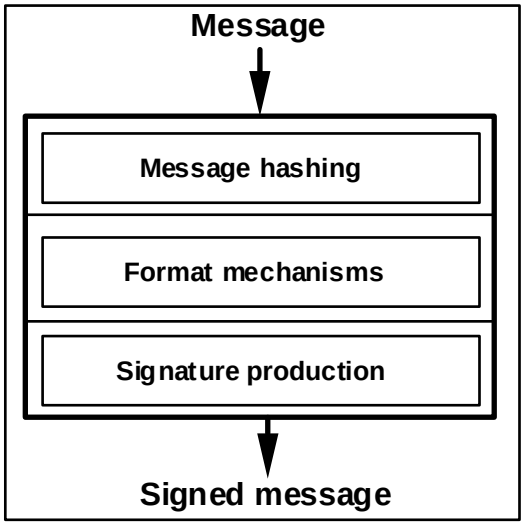

Figure 6 - Signature process according to ISO/IEC 9796-2

If a hash algorithm present in the HPC is used, 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.

If a hash algorithm not present in the HPC is used, then complete hashing is performed outside the HPC and only the padding is done by the HPC.

8.3 Security Environments

In DF.QES, there are 2 SEs:

- SE # 1: Default SE. No use of a trusted channel.
- SE # 2: SE for use of the signature function with a trusted channel to support remote HPC configurations.

8.4 QES Application Selection

The QES application selection is performed with the SELECT command.

Table 50 - SELECT Command for DF.QES

CLA	As defined in ISO/IEC 7816-4
INS	‘A4’ = SELECT
P1	‘04’ = DF Selection with AID
P2	'0C' = No FCI to return
Lc	‘06’ = Length of subsequent data field
Data field	‘D27600006601’ = AID of DF.QES
Le	Absent

Table 51 - SELECT Response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After DF selection, SE # 1 is implicitly selected. For usage of SE # 2 see Clause 11.

8.5 PIN Management

8.5.1 PIN Verification

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

Table 52 - VERIFY command for PIN verification

CLA	As defined in ISO/IEC 7816-4
INS	‘20’ = VERIFY
P1	‘00’
P2	‘81’ = PIN.QES reference
Lc	‘08’ = Length of subsequent data field
Data field	PIN (Format: see [HPC-P1])
Le	Absent

Table 53 - VERIFY response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

In case, the PIN was wrong and retries are left, the number of further allowed retries shall be indicated in the status code as specified in [HPC-P1].

8.5.2 PIN Change

For PIN change the ISO/IEC 7816-4 command CHANGE RD may be used at any time convenient for the HP. If the command is successfully completed, then the security status “PIN with reference '81' successfully presented” shall be set.

Table 54 - CHANGE RD command

CLA	As defined in ISO/IEC 7816-4
INS	‘24’ = CHANGE REFERENCE DATA
P1	‘00’ = Exchange reference data
P2	‘81’ = PIN.QES reference
Lc	‘10’ = Length of subsequent data field
Data field	PIN_old || PIN_new (PIN format: see [HPC-P1])
Le	Absent

NOTE – Whether a security status is set, if the transport PIN is presented, depends on the transport PIN mechanism of the respective HPC.

Table 55 - CHANGE RD response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

8.5.3 Reset of Retry Counter

For resetting the retry counter to its initial value, the ISO/IEC 7816-4 RESET RETRY COUNTER command is used.
The command does not set or influence the security status, i.e. the successful presentation of PIN.QES is still needed prior to the usage of PrK.HP.QES.

Table 56 – RESET RC command for RC reset

CLA	As defined in ISO/IEC 7816-4
INS	‘2C’ = RESET RETRY COUNTER
P1	‘01’
P2	‘81’ = Reference of PIN.QES, to which the resetting code (“PUK”) according to ISO/IEC 7816-4 belongs
Lc	‘08’ = Length of subsequent data field
Data field	Resetting code ("PUK", Format: see [HPC-P1])
Le	Absent

Table 57 - RESET RC response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

8.6 Computation of a QES

8.6.1 Signing with final Hashing in the HPC

In a first step, the hash algorithm shall be selected using the MSE command.

Table 58– MSE command for setting the hash algorithm

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for Hash computation
P2	‘AA’ = Hash-Template
Lc	‘03’ = Length of subsequent data field
Data field	80 01 xx’ = DO für AlgID, see Tab. 10 in [HPC-P1]
Le	Absent

Table 59 – MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

In a second step, PrK.HP.QES and the signature algorithm in combination with the padding scheme has to be selected.

Table 60 - MSE command

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for computation
P2	'B6' = DST
Lc	'06' = Length of subsequent data field
Data field	'84 01 84 || 80 01 xx' = DO for KeyID of PrK.HP.QES || DO for AlgID, see Table 10 in [HPC-P1]
Le	Absent

Table 61 - MSE response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

In a third step, the PSO: HASH command shall be sent.

Table 62 - PSO: HASH command

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 (without padding) Intermediate hash value: hash value (20 byte for SHA-1) || counter with number of bits already hashed (8 B for SHA-1); other values see [HPC-P1]
Le	Absent

If the message to be hashed is shorter than the block length, then the length of the DO with tag '90' shall be set to zero (i.e. there is no intermediate hash value) followed by the DO with tag '80' with the message to be hashed.

Table 63 - PSO: HASH response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

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

Table 64 - PSO: COMPUTE DS command

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 65 - PSO: COMPUTE DS response

Data field	Digital signature
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

8.6.2 Signing with complete Hashing outside the HPC

In this case, the PSO: COMPUTE DS command without a preceding PSO: HASH command is sent to the HPC. Before the private signature key and the algorithm identifier have to be selected.

Table 66 - MSE command

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for computation
P2	'B6' = DST
Lc	'06' = Length of subsequent data field
Data field	'84 01 84 || 80 01 xx' = DO for KeyID of PrK.HP.QES || DO for AlgID, see Table 10 in [HPC-P1]
Le	Absent

Table 67 - MSE response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

Table 68 - PSO: COMPUTE DS command

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	'xx' = Length of subsequent data field
Data field	- Digestinfo in case padding according to PKCS#1 Clause 9.2 - Hash value in case of ISO 9796 padding with RND
Le	'00' or 'xx' = length of expected digtal signature

Table 69 - PSO: COMPUTE DS response

Data field	Digital signature
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

8.7 Reading of QES related Certificates

For reading the X.509-QES certificate or an attribute certificate the READ BINARY command is used.

Table 70 - READ BINARY command for reading QES certificates

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1, P2	- P1 = b8-b6: 100 b5-b1: 10000 SFID of EF.C.HP.QES: 16 or 00001 SFID of EF.C.HP.QES-AC1: 1 or 00010 SFID of EF.C.HP.QES-AC2: 2 or 00011 SFID of EF.C.HP.QES-AC3: 3 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	'00' or '000000'

Table 71 - READ BINARY response

Data field	Ceritifcate data
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

If the card does not support extended length, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

8.8 Writing of Attribute Certificates

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

Table 72 - UPDATE BINARY command for writing a new attribute certificate in the HPC

CLA	As defined in ISO/IEC 7816-4
INS	'D6' = UPDATE BINARY
P1, P2	- P1 = b8-b6: 100 b5-b1: 00001 SFID of EF.C.HP.QES-AC1: 1 or 00010 SFID of EF.C.HP.QES-AC2: 2 or 00011 SFID of EF.C.HP.QES-AC3: 3 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	'xx' = Length of subsequent data field
Data field	Data
Le	Absent

Table 73 - UPDATE BINARY response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

If the certificate is longer than 256 byte, then the UPDATE BINARY command shall be repeated specifying the respective offset in P1-P2.

8.9 Remote QES Usage

The remote usage of this application is outlined in Clause 11. The mechanisms for controlling the usage of the TC are out of scope of this specification.

9 The ESIGN Application

9.1 File Structure and File Content

The general structure of the ESIGN application, which is in accordance with [CWA14890], is shown in Figure 7.

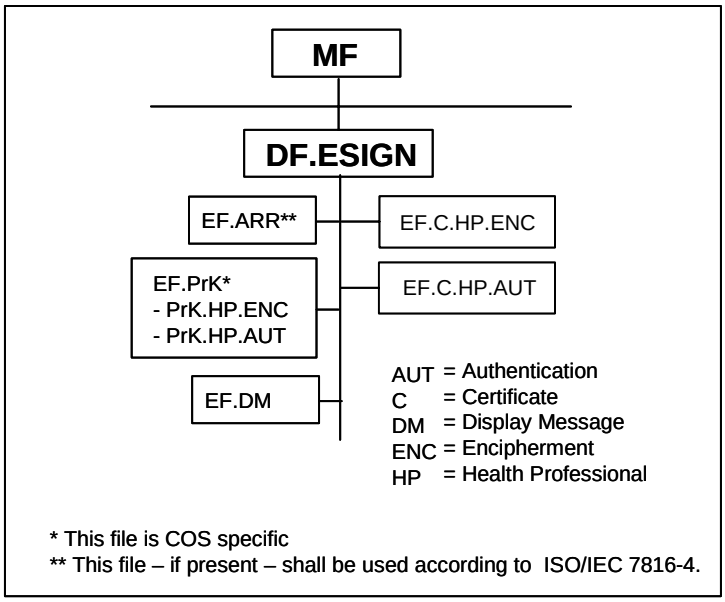

Figure 7 – General structure of DF.ESIGN

9.1.1 EF.ARR

EF.ARR contains the access rules related to DF.ESIGN.

9.1.2 EF.PrK

EF.PrK contains the following private keys:

- PrK.HP.AUT for client/server authentication
- PrK.HP.ENC for document cipher key decipherment.

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

Table 74 – Key identifiers and protection of HP-related private keys

Key Name	KeyID	Access Conditions
PrK.HP.AUT	'82'	- SE # 1: PIN.CH - SE # 2: PIN.CH and SM
PrK.HP.ENC	'83'	- SE # 1: PIN.CH - SE # 2: PIN.CH and SM

With respect to key length the same conventions shall be used as for legally binding electronic signature keys, see [ALGCAT].

The allowed padding schemes are presented in Annex E.

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

9.1.3 EF.DM

The transparent file EF.DM contains the same display message as described in clause 8.1.4.

9.1.4 EF.C.HP.AUT

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

9.1.5 EF.C.HP.ENC

EF.C.HP.ENC contains the X.509 ENC certificate of the health professional.

9.2 Security Environments

In DF.ESIGN, there are 2 SEs:

- SE # 1: Default SE. No use of a trusted channel.
- SE # 2: SE for use of the ESIGN functions with a trusted channel to support remote HPC configurations.

9.3 ESIGN Application Selection

The ESIGN application selection is performed with the SELECT command.

Table 75 - SELECT Command for DF.ESIGN

CLA	As defined in ISO/IEC 7816-4
INS	‘A4’ = SELECT
P1	‘04’ = DF Selection with AID
P2	'0C' = No FCI to return
Lc	‘0A’ = Length of subsequent data field
Data field	'A000000167 455349474E’ = AID of DF.ESIGN
Le	Absent

Table 76 - SELECT Response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

After DF selection, SE # 1 is implitly selected. For usage of SE # 2 see Clause 11.

9.4 Reading an X.509 Certificate

For reading the certificates C.HP.AUT and C.HP.ENC, the READ BINARY command is used.

Table 77 - READ BINARY command for reading an X.509 certificate

CLA	As defined in ISO/IEC 7816-4
INS	'B0' = READ BINARY
P1-P2	- P1 = b8-b6: 100 b5-b1: 00001 SFID of EF.C.HP.AUT: 1 or 00010 SFID of EF.C.HP.ENC: 2 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	'00' or '000000'

Table 78 - READ BINARY response

Data field	CV certificate
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

If the card does not support extended length, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

9.5 PIN Management

Prior to the usage of PrK.HP.AUT or PrK.HP.ENC, PIN.CH has to be successfully presented, see Clause 5.5.1.

9.6 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.

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 identifier denoting the type of authentication protocol may be set additionally.

Table 79 – MSE command

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 82' || '80 01 05' = DO for KeyID of PrK.HP.AUT || DO AlgID, see Table E.2
Le	Absent

Table 80 - MSE response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

Table 81 - INT. AUTHENTICATE command

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	- DigestInfo, used for KERBEROS - H_MD5 || H_SHA1, used for SSL/TLS - H_SHA1, used for WTLS and NETSCAPE The formatting of the DSI for authentication is specified in [HPC-P1], Annex E.6
Le	'00' or 'xx' = length of expected digital signature

Table 82 - INT. AUTHENTICATE response

Data field	Digital signature
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

9.7 Document Cipher Key Decipherment

For confidential document exchange, the following scheme is applied:

- key transport is organized by enciphering the document cipher key with the receiver's PuK.HP.ENC
- 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 ENC certificate retrieved e.g. from a certificate server.
Before on the receiver side key decipherment can be performed, the PIN.CH has to be presented successfully. In addition, the private key PrK.HP.ENC has to be selected with the ISO/IEC 7816-4 command MSE.

Table 83 - MSE command

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 83’ = DO for KeyID of PrK.HP.ENC
Le	Absent

Table 84 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

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

Table 85 – PSO: DECIPHER command

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, see [HPC-P1], Table E.2
Le	‘00’ or ‘xx’ = Length of document cipher key

Table 86 – PSO: DECIPHER response

Data field	Document cipher key
SW1-SW2	‘9000’or specific status bytes, see [HPC-P1]

9.8 Remote ESIGN Usage

The remote usage of this application is outlined in Clause 11. The mechanisms for controlling the usage of the TC are out of scope of this specification.

10 Cryptographic Information Application

10.1 General Structure

The general structure of the Cryptographic Information Application (CIA) associated to the ESIGN application 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.

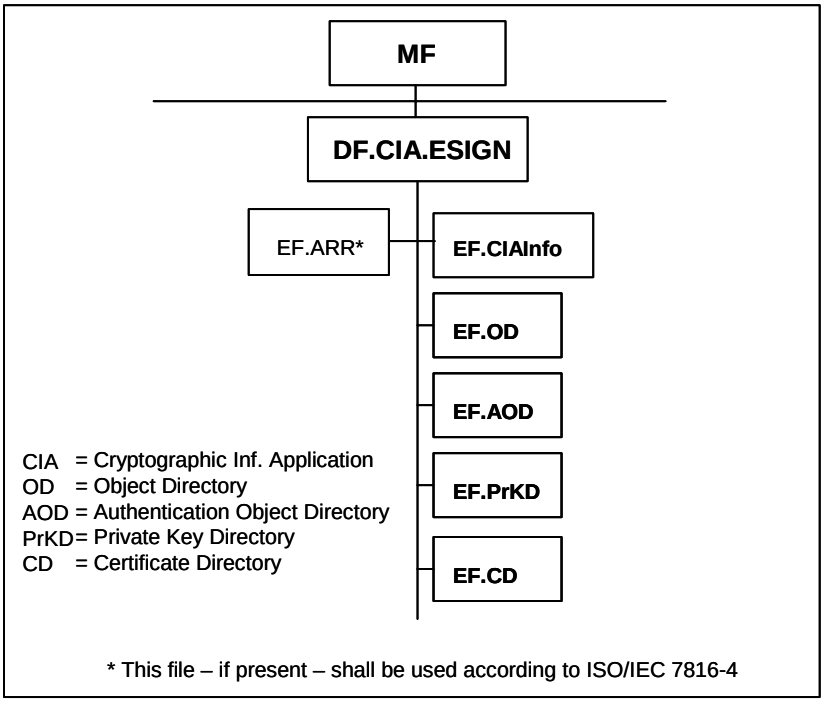

Figure 8 – DF.CIA and substructure

10.2 Application Selection

For application selection, ISO/IEC 7816-4 SELECT command is used as shown in the subsequent tables.

Table 87 – SELECT command for DF.CIA.ESIGN

CLA	'00'
INS	'A4' = SELECT
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'0F' = Length of subsequent data field
Data field	'E828BD080F A000000167 455349474E' = AID of DF.CIA.ESIGN
Le	Absent

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, i.e. AID.ESIGN.

Table 88 – SELECT response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

10.3 Reading the CIA Files

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

Table 89 – READ BINARY command for reading the CIA files

CLA	As defined in ISO/IEC 7816-4
INS	‘B0’ = READ BINARY
P1	‘xx’ = b8-b6: 100 b5-b1: 10010 SFID of EF.CIAInfo: 18 or b5-b1: 10001 SFID of EF. OD: 17 or b5-b1: 10100 SFID of EF. AOD: 20 or b5-b1: 10101 SFID of EF. PrKD: 21 or b5-b1: 10110 SFID of EF. CD: 22
P2	‘00’ = Offset
Lc	Absent
Data field	Absent
Le	‘00’ = Read until end-of-file (max 256 B)

NOTE – SFID 19 cannot be used, see ISO/IEC 7816-15.

Table 90- SELECT response

Data field	CIOs
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

11 Interactions between HPC and SMC

11.1 Reading CV certificates

In Clause 5.4.4 is specified, how the HPC CVC certificates – located at MF level – can be retrieved. It is assumed, that reading is performed only once by the software environment, e.g. PVS or AVS, which stores the CVCs e.g. associated with the respective ICCSN.

11.2 SE selection at MF Level

For the HPC/SMC interaction, SE # 2 has to be selected at MF level, see Clause 5.3.

11.3 Selection of the Application to be used with SM

Before establishing a TC, the respective application, DF.ESIGN or DF.QES, has to be selected, see Clause 8.4 or 9.3.
Since the SELECT command for application selection is always sent without SM, a TC is lost, if subsequently a further application is selected, i.e. TC establishing has to be performed again.

11.4 SE Selection at DF Level

Prior to any command processing after application selection, the SE # 2 shall be selected with the MSE command using the RESTORE function.

Table 91 - MSE command for selecting the SE relevant for TC establishing

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'F3' = RESTORE
P2	'02' = SE # 2
Lc	Absent
Data field	Absent
Le	Absent

Table 92 - MSE response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

11.5 CVC Verification

The following CVCs have to be verified:

- CVC.CA_SMC.CS
- CVC.SMC.AUT

The procedure is equivalent to the procedure described in Clause 5.6.3.

11.6 Authentication Procedure with TC Establishment

The command sequence at the HPC side starts with selecting the involved keys.

Table 93 - MSE command for key selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'C1' = SET for int./ext. authentication
P2	'A4' = AT
Lc	'14' = Length of subsequent data field
Data field	'83 0C xx ... xx' || '84 01 11' = DO for KeyRef of PuK.SMC.AUT (for retrieval of the KeyRef see Figure 3) || DO for KeyRef of PrK.HPC.AUT
Le	Absent

Table 94 - MSE response

Data field	Absent
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

Now the authentication procedure is performed as described in [HPC-P1], Annex E.3, whereby the procedural steps from the viewpoint of the HPC shall be executed in the order shown in [HPC-P1], Annex E.2., i.e. in a first step a challenge has to be retrieved from the HPC.

Table 95 - GET CHALLENGE command

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

Table 96 - GET CHALLENGE response

Data field	RND.HPC (8 byte)
SW1-SW2	'9000' or specific status bytes, see [HPC-P1]

The challenge is presented to the SMC together with the 8 LSB of HPC's ICC Serial number. The result of the INTERNAL AUTHENTICATE performed by the SMC is presented to the HPC in the EXTERNAL AUTHENTICATE command.

Table 97 - EXT. AUTHENTICATE command

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, see [HPC-P1], Annex E.3
Le	Absent

NOTE – The authentication related data contain data elements for SM key computation.

Table 98 - EXT. AUTHENTICATE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

In a third step the HPC has to prove its authenticity.

Table 99 - INT. AUTHENTICATE command

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	Authentication related data, see [HPC-P1], Annex E.3
Le	‘00’ or ‘xx’ = Length of expected signature

Table 100 - INT. AUTHENTICATE response

Data field	Authentication related data, see [HPC-P1], Annex E.3
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

The SM keys are computed as described in [HPC-P1], Annex E.3 and the initial value of the Send Sequence Counter as specified in [HPC-P1], Clause C.5.2.

11.7 Reading and Updating of the Display Message

After TC establishing, the display message shall be read and shown to the health professional as feedback, that a TC has been successfully established. The reading, however, is not controlled by the HPC.
For reading the display message, the READ BINARY command is used.

Table 101 - READ BINARY command for reading the display message

CLA	As defined in ISO/IEC 7816-4
INS	‘B0’ = READ BINARY
P1	'84' = b8-b6:100 b5-b1: 00100 SFID of EF.DM: 4
P2	‘00’ = Offset
Lc	Absent
Data field	Absent
Le	'08'

NOTE – Each TC relevant application has its own EF.DM with SFID 4.

Table 102 - READ BINARY response

Data field	Display message
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

The display message can be modified with the UPDATE BINARY command after successful presentation of PIN.CH.

Table 103 - UPDATE BINARY command for changing the display message

CLA	As defined in ISO/IEC 7816-4
INS	‘D6’ = UPDATE BINARY
P1	b8-b6:100 b5-b1: 00100 SFID von EF.DM: 4
P2	‘00’ = Offset
Lc	‘08’ = Length of subsequent data field
Data field	New display message (8 byte)
Le	Absent

Table 104 - UPDATE BINARY response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

12 Loading a new Application or Creation of an EF after Issuing

12.1 Identification of the HPC COS Platform

It is assumed that the loading of a new application or the creation of a new EF in DF.HPA after issuing of the HPC is performed by a Card Application Management System (CAMS). A CAMS may have already information about the respective HPC, e.g. ICCSN, the related COS platform and the current set of applications in the respective HPC, but may also retrieve information from the respective HPC e.g. by reading EF.ATR, EF.GDO and EF.DIR.

12.2 Establishing a Trusted Channel between CAMS and HPC

A trusted channel may be achieved with a symmetric or asymmetric authentication procedure depending on the involved CAMS. Subsequently the asymmetric authentication procedure is described, which is also part of the protection profile [PP-HPC].

For establishing a TC between the related CAMS and a HPC, an asymmetric authentication procedure with SM key computation has to be performed as specified in [HPC-P1], Annex E 3. Since the PuK.CAMS_HPC.AUT is already present in the HPC, no verification of CVCs of the CAMS is necessary. To the CAMS however, the CVCs of the HPC have to be presented, see Figure 9.

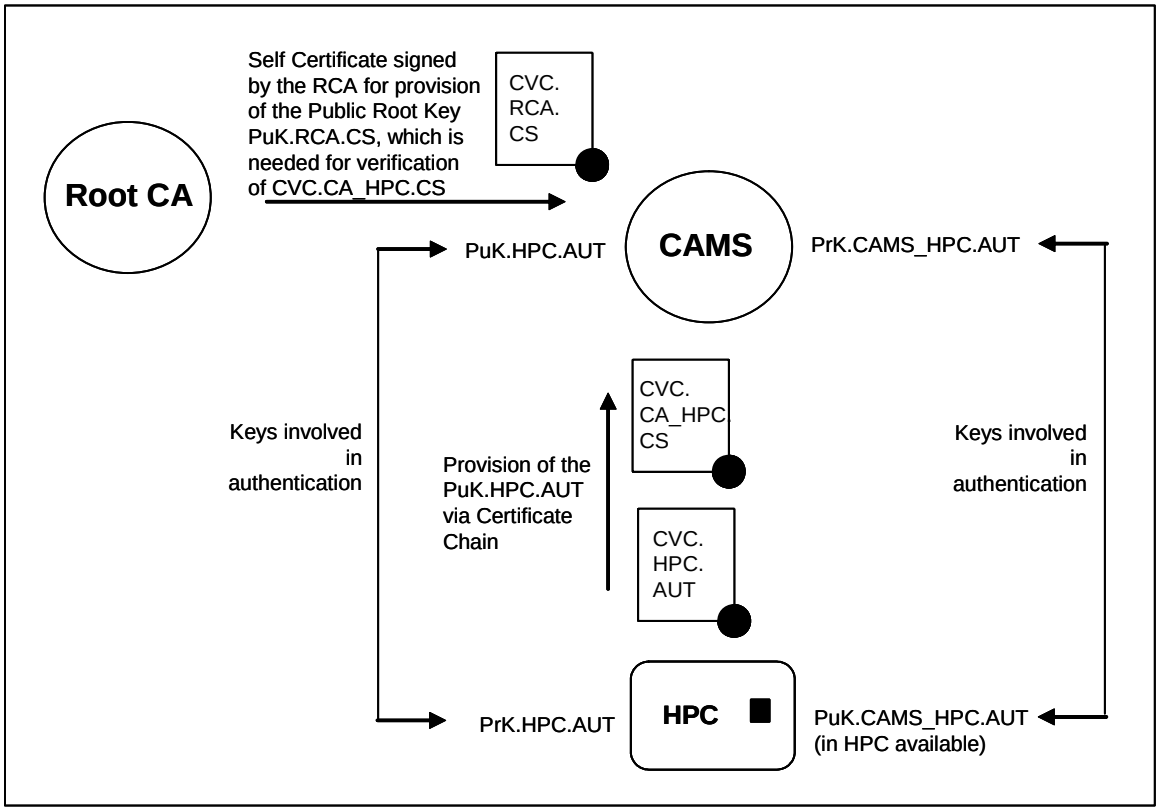

Figure 9 – Keys and key import for HPC/CAMS authentication with TC establishment

In a first step, the public key of the CAMS has to be set with the MSE command.

Table 105 - MSE command for selecting PuK.CAMS_HPC.AUT

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘C1’ = SET for int./ext. authentication
P2	‘A4’ = AT
Lc	‘03’ = Length of subsequent data field
Data field	‘840113' = DO KeyID for PuK.CAMS_HPC.AUT
Le	Absent

Table 106 - MSE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

12.3 Loading of an Application

The command sequence for loading of a new application is manufacturer specific. Commands relevant for the loading process are specified in [HPC-P1].

12.4 Registering the new Application in EF.DIR

In EF.DIR, a DO application template with the Application Identifier of the new application has to be inserted. For this pupose the APPEND RECORD command is used.

Table 107 - APPEND RECORD command

CLA	As defined in ISO/IEC 7816-4
INS	‘E2’ = APPEND RECORD
P1	‘00’
P2	‘F0’ = b8-b4: 11110 SFID of EF.DIR: 30 b3-b1: 000
Lc	’xx’ = Length of subsequent data field
Data field	DO Application template, see Table G.3
Le	Absent

Table 108 – APPEND RECORD response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

12.5 Creation of an EF

For creation of an EF at MF level or under DF.HPA, the ISO/IEC 7816-8 CREATE FILE command is used.

Table 109 – CREATE FILE command

CLA	As defined in ISO/IEC 7816-4
INS	‘E0’ = CREATE FILE
P1	‘00’ = EF creation
P2	‘00’
Lc	’xx’ = Length of subsequent data field
Data field	FCP of EF to be created (COS specific)
Le	Absent

Table 110 – CREATE FILE response

Data field	Absent
SW1-SW2	‘9000’ or specific status bytes, see [HPC-P1]

If a new access rule for this EF applies, then this access rule has be to integrated in the related EF.ARR with the APPEND RECORD command.

Annex A
(normative)

ATR

A.1 ATR Coding

Table A.1 shows the coding of the ATR for HPCs (T = 1 cards).

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

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	'FE'	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, i = max. 15)
TCK	XOR	Check Character (exclusive OR)

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 Identifier

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'

For performance acceleration, it is allowed to send a TC1 with the value 'FF'.

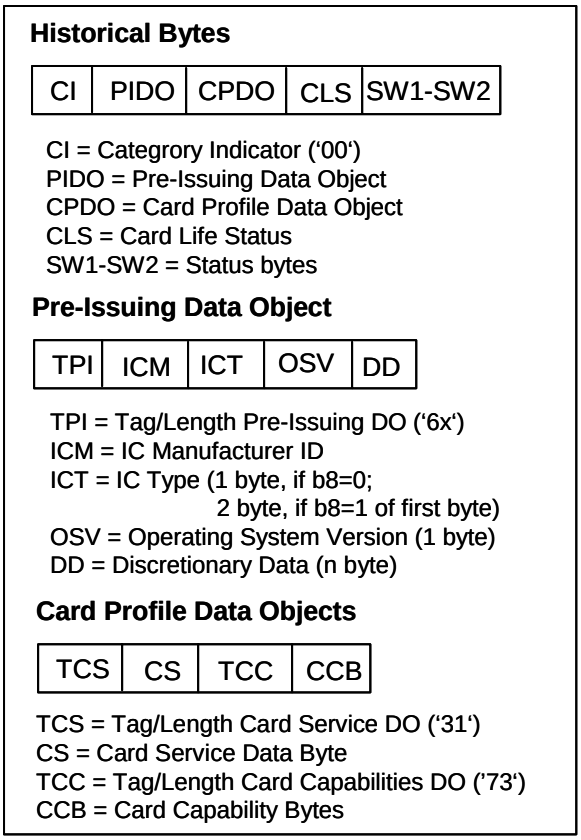

Figure A.1 – Structure of the Historical Bytes

The actual value of the ICM data element shall be the identifier assigned by ISO for this IC manufacturer (see www.sc17.com).

Annex B
(normative)

File Attributes, Access Conditions and Security Environments

B.1 Access Rules

If a COS does not support EF.ARR, the functionality has to be ensured by other means. SMCs with EF.ARR may allow read access to the stored access rules. Therefore the security condition for the READ RECORD command related to EF.ARR is set to "always". An EF.ARR may be read e.g. by a CAMS.

The specified access rules below represent coding examples. They may be supported as such or by other codings, but he same functionality has to be achieved.

If a dual-interface chip with a contact-based interface and a RF interface is used, then there shall be an access rule restricting the usage of the subsequent applications to the contact-based interface.

B.2 MF Level

At MF level, SE # 1 and SE # 2 are used. The access rules are SE independent, if not specified otherwise.

Table B.1 – EFs at MF level and their characteristics

File	**FID / SFID**	**File structure**	**File size** (length of data)	**Access condition**
EF.ARR (Access Rules)	COS specific	linear variable	8 records with x byte	ARR # 1
EF.ATR (ATR Extension Data)	‘2F01’ / 29	transparent	32 Byte	ARR # 1
EF.DIR (Application Directory)	‘2F00’ / 30	linear variable	6 records with x byte	ARR # 2
EF.GDO (Global Data Objects)	‘2F02’ / 2	transparent	12 byte	ARR # 1
EF.CVC.HPC.AUT (Card Verifiable Certificate)	‘2F03’ / 3	transparent	209 byte	ARR # 1
EF.CVC.CA_HPC.CS (Card Verifiable Certificate)	‘2F04’ / 4	transparent	210 byte	ARR # 1

Table B.2 – Access Rules in EF.ARR at MF level

Rec No.	**Value**	**Meaning**
1	’80 01 01 9000’	AM: READ BINARY / READ RECORD SC: Always Referenced in EF.ATR, EF.GDO, EF.CVC.CA_HPC.CS, EF.CVC.HPC.AUT and EF.ARR
2	’80 01 01 9000	AM: READ RECORD SC: Always

	80 01 04 AF 0D A406 950180 830113 B403 950130’	AM: APPEND RECORD SC: AND template {AT (UQ = Ext. authentication, Key Ref = PuK.CAMS_HPC.AUT) || CCT (UQ = CC in SM command/response)} Referenced in EF.DIR
3	'86 08 2000 2400 2C00 2C01 9000’	AM: VERIFY, CHANGE RD (Option ‘00’), RESET RC (Option ‘00’ and ‘01’) SC: Always Referenced by PIN.CH in EF.PIN, SE # 1
4	'86 08 2000 2400 2C00 2C01 AF 0A B4 03 95 01 30 B8 03 95 01 30'	AM: VERIFY, CHANGE RD (Option ‘00’), RESET RC (Option ‘00’ and ‘01’) SC: AND template {CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} Referenced by PIN.CH in EF.PIN, SE # 2
5	'84 01 88 A4 06 950108 830101'	AM: INTERNAL AUTHENTICATE SC: AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) Referenced by PrK.HPC.AUT in EF.PrK, SE # 1
6	'84 02 8882 9000’	AM: INT. / EXT. AUTHENTICATE SC: Always Referenced by PrK.HPC.AUT in EF.PrK SE # 2
7	’87 03 2A00AE 9000’	AM: VERIFY CERTIFICATE SC: Always Referenced by PuK.RCA.CS in EF.PuK
8	'84 02 EA E0 AF 12 A4 06 95 01 80 83 01 13 B4 03 95 01 30 B8 03 95 01 30'	AM: LOAD APPLICATION, CREATE FILE DF, CREATE FILE EF (after HPC issuing) SC: AND template {AT (UQ = ext. authentication, Key Ref = PuK.CAMS_HPC.AUT) || CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} Referenced in MF

B.3 DF.HPA

In DF.HPA, only SE # 1 (default SE) is used.

Table B.3 – EFs in DF.HPA and their characteristics

File	FID / SFID	File structure	File size (length of data)	Access condition
EF.ARR (Access Rules)	COS specific	linear variable	3 records with x byte	ARR # 1
EF.HPD (Health Professional related Data)	'D0001' / 1	transparent	2 KB	ARR # 2

Table B.4 – Access Rules in EF.ARR in DF.HPA

Rec-No.	Value	Meaning
1	'80 01 01 9000 80 01 04 AF0D A406 950180 830113 B403 950130'	AM: READ RECORD SC: Always AM: APPEND RECORD SC: AND template {AT (UQ = ext. authentication, Key Ref = PuK.CAMS_HPC.AUT) || CCT (UQ = CC in SM command/response)} Referenced in EF.ARR
2	'80 01 01 9000 80 01 02 A406 950108 830101'	AM: READ BINARY SC: Always AM: UPDATE BINARY SC AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) Referenced in EF.HPD
3	'80 01 02 AF 12 A4 06 95 01 80 83 01 13 B4 03 95 01 30 B8 03 95 01 30'	AM: CREATE FILE (EF) SC: AND template {AT (UQ = ext. authentication, Key Ref = PuK.CAMS_HPC.AUT) || CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} Referenced in FCP of DF.HPA

B.4 DF.QES

In DF.QES, SE # 1 and SE # 2 are used. The access rules are SE independent, if not specified otherwise.

Table B.5 – EFs in DF.QES and their characteristics

File	FID / SFID	File structure	File size (length of data)	Access condition
EF.ARR (Access Rules)	COS specific	linear variable	7 records with x records	ARR # 1
EF.C.HP.QES (Qualified Electronic Signature Certificate)	'C000' / 16	transparent	1.5 k byte or length of certificate	ARR # 1
EF.C.HP.QES-AC1 (QES Attribute Certificate 1)	'C001' / 1	transparent	1k byte or length of certificate	ARR # 2
EF.C.HP.QES-AC2 (QES Attribute Certificate 2)	'C002' / 2	transparent	1k byte or length of certificate	ARR # 2
EF.C.HP.QES-AC3 (QES Attribute Certificate 3)	'C003' / 3	transparent	1k byte or length of certificate	ARR # 2
EF.DM (Display Message)	'D004' / 4	transparent	8 byte	ARR # 3

NOTE – The HPC contains all certificates, when delivered to the HP. This does not exclude personalisation in several steps, e.g. interactions with certificate service providers.

Table B.6 – Access Rules in EF.ARR in DF.QES

Rec-No.	Value	Meaning
1	'80 01 01 9000'	AM: READ RECORD / READ BINARY SC: Always Referenced in EF.ARR and in EF.C.HP.QES
2	'80 01 01 9000 80 01 02 A406 950108 830101'	AM: READ BINARY SC: Always AM: UPDATE BINARY SC: AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) Referenced in EF.CH.QES-AC1 / AC2 / AC3
3	'80 01 01 AF 0A B4 03 95 01 30 B8 03 95 01 30 80 01 02 A406 950108 830101'	AM: READ BINARY SC: AND template {CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} AM: UPDATE BINARY AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) Referenced in EF.DM
4	'86 06 2000 2400 2C01 9000'	AM: VERIFY, CHANGE RD (Option '00'), RESET RC (Option '01') SC: Always Referenced by PIN.QES in EF.PIN, SE # 1
5	'86 06 2000 2400 2C01 AF 0A B4 03 95 01 30 B8 03 95 01 30'	AM: VERIFY, CHANGE RD (Option '00'), RESET RC (Option '01') SC: AND template {CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} Referenced by PIN.QES in EF.PIN, SE # 2

6	'87 03 2A9E9A A406 950108 830181'	PSO: COMPUTE DS SC: AT (UQ = Knowledge based user authentication, KeyRef = PIN.QES) Referenced in EF.PrK by PrK.HP.QES, SE # 1
7	'87032A9E9A AF 12 A406 950108 830181 B4 03 95 01 30 B8 03 95 01 30'	PSO: COMPUTE DS SC: AND template {AT (UQ = Knowledge based user authentication, KeyRef = PIN.QES) || CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} Referenced in EF.PrK by PrK.HP.QES, SE # 2

B.5 DF.ESIGN

In DF.ESIGN, SE # 1 and SE # 2 are used. The access rules are SE independent, if not specified otherwise.

Table B.7 – EFs in DF.ESIGN and their characteristics

File	**FID / SFID**	**File structure**	**File size** (length of data)	**Access condition**
EF.ARR (Access Rules)	COS specific	linear variable	4 records with x byte	ARR # 1
EF.C.HP.AUT (Authentication Certificate)	'C500' / 1	transparent	1.5 k byte or length of certificate	ARR # 1
EF.C.HP.ENC (Encipherment Certificate)	'C200' / 2	transparent	1k byte or length of certificate	ARR # 1
EF.DM (Display Message)	'D004' / 4	transparent	8 byte	ARR # 2

Table B.8 – Access Rules in EF.ARR in DF.ESIGN

Rec-No.	**Value**	**Meaning**
1	'80 01 01 9000'	AM: READ RECORD / READ BINARY SC: Always Referenced in EF.ARR, EF.C.HP.AUT and in EF.C.HP.ENC
2	'80 01 01 AF 0A B4 03 95 01 30 B8 03 95 01 30 80 01 02 A406 950108 830101'	AM: READ BINARY SC: AND template {CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} AM: UPDATE BINARY AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) Referenced in EF.DM
3	'87 06 880000 2A8086 A4 06 950108 830101'	AM: INT. AUTHENTICATE / PSO: DECIPHER SC: AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) Referenced by PrK.HP.AUT and PrK.HP.ENC in EF.PrK, SE # 1
4	'87 06 880000 2A8086 AF 12 A406 950108 830101	AM: INT. AUTHENTICATE / PSO: DECIPHER SC: AND template {AT (UQ = Knowledge based user authentication, KeyRef = PIN.CH) ||

	B4 03 95 01 30 B8 03 95 01 30'	CCT (UQ = CC in SM command/response) || CT (UQ = CG in SM command/response)} Referenced by PrK.HP.AUT and PrK.HP.ENC in EF.PrK, SE # 2

NOTE – The access rules for PIN.CH are specified at MF level.

B.6 DF.CIA.ESIGN

In DF.CIA.ESIGN, only SE # 1 (default SE) is used.

Table B.9 – CIA files and their characteristics

File	**FID / SFID**	**File structure**	**File size** (length of data)	**Access condition**
EF.ARR (Access Rules)	COS specific	linear variable	1 record with x byte	ARR # 1
EF.CIAInfo (CIA Information)	'5032' / 18	transparent	256 byte or length of CIOs	ARR # 1
EF.OD (Object Directory)	'5031' / 17	transparent	64 byte or length of CIOs	ARR # 1
EF.AOD (Authent. Object Directory)	'5034' / 20	transparent	128 byte or length of CIOs	ARR # 1
EF.PrKD (Private Key Directory)	'5035' / 21	transparent	128 byte or length of CIOs	ARR # 1
EF.CD (HP Certificate Directory)	'5038' / 22	transparent	128 byte or length of CIOs	ARR # 1

NOTE: SFID 19 not used (see ISO/IEC 7816-15)

Table B.10 – Access Rules in EF.ARR in DF.CIA.ESIGN

Rec-No.	**Value**	**Meaning**
1	'80 01 01 9000'	AM: READ RECORD / READ BINARY SC: Always Referenced in EF.ARR, EF.CIAInfo, EF.OD, EF.AOD, EF.PrKD and EF.CD

Annex C
(normative)

Structure and Content of QES Certificates

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

C.1.1 General aspects

The coding scheme of HP QES-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 a QES certificate for health professionals. In addition there are further mandatory certificate fields due to the intended usage of the HP QES-certificates in national and international environments.

The QES certificates consist of

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

The QES 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 {
```

```
    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 QES-certificates, some extensions are mandatory. The optional extensions *issuerUniqueID* and *subjectUniqueID* that are contained in the ASN.1 definition shall not be used.

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 QES-certificates, only X.509v3 is relevant.

C.2.3 Serial Number

The *serialNumber* in combination with the *issuer* name fields of QES 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 be taken into account for HP QES-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 QES-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))
```

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 SHALL contain at least the attributes *commonName* and *countryName*.

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

- countryName
- commonName.

The use of other attribute types conforming to the ISIS-MTT-specification or relevant RFCs is optional.

The coding scheme for these *subject* attribute types in HP QES-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 SHALL 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 *subjectDirectory-Attributes* certificate extension (in the *gender* attribute, see C.2.9.5).
If the *commonName* contains a pseudonym, then it must end with the suffix *":PN*". In order to provide unique subject distinguished names CAs must not use the same pseudonym for different persons and may choose to place an integer before *: PN*, e.g. *CN = "givenName surname 1:PN"*. A pseudonym shall be used only for technical and/or legal reasons. It shall contain the *surname* and at least one *givenName* of the certificate holder.

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 are

- the signature algorithm RSA or – in the future – also the ECDSA,
- the hash algorithm SHA-1 and hash algorithms of the SHA-2 family (especially SHA-256)
- 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
- 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 (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]
- private X.509 extensions from [ISIS-MTT OP]
- private X.509 extensions from BNA.

X.509 basic extensions include:

- basic constraints
- key usage
- certificate policies
- subject alternative name
- authority key identifier
- subject key identifier
- subject directory attributes
- CRL distribution points.

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
- authority information access.

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
- restriction
- additional information

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 }
```

BNA Private X.509 Extensions:

- validity model.

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 extension, if used in an HP certificate, SHALL be marked critical, and the value of the *cA* component if present 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 QES-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 SHALL 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-accredited 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 unique e-mail addresses 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] IMPLICT 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 Authority Information Access

The Authority Information Access extension contains the URL of the OCSP-Server responsible for providing status information about the certificate. The extension MUST NOT be marked critical.

ASN.1 definition of type *AuthorityInfoAccess*:

```
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }
AuthorityInfoAccessSyntax ::= SEQUENCE (1..MAX) OF AccessDescription
AccessDescription ::= SEQUENCE {
```

```
      accessMethod         OBJECT IDENTIFIER
      accessLocation       GeneralName }
```

The OID value *id-ad-ocsp {id-ad 1}* is used the *accessMethod* component to indicate the OCSP-service. The OCSP-URL is specified in the *accessLocation* field.

C.2.9.9 Validity Model

The private BNA extension Validity Model identifies the underlying algorithms that must be used in order to determine the validity of the certificate. The extension MUST NOT be marked critical. Possible values are id-validityModel-chain {1.3.6.1.4.1.8301.3.5.1} for the “chain model” used by the BNA and id-validityModel-shell {1.3.6.1.4.1.8301.3.5.2} for the “shell model” used by PKIX.
Further validity models, e.g. hybrid models are also possible. Information about the validity models can be found at the internet:
- BNA: http://www.bundesnetzagentur.de/media/archive/1343.pps
- OIDs: http://www.informatik.tu-darmstadt.de/TI/Forschung/FlexiPKI/validitymodel/index.html

ASN.1 definition of type *ValidityModel*:

```
id-validityModel OBJECT IDENTIFIER ::= { 1 3 6 1 4 1 8301 3 5 }
id-validityModel-chain {1 3 6 1 4 1 8301 3 5 1} -- yet without validityModelInfo.
id-validityModel-shell {1.3.6.1.4.1.8301.3.5.2} -- yet without validityModelInfo.
ValidityModel::= SEQUENCE {
      validityModelId      OBJECT IDENTIFIER
      validityModelInfo    ANY DEFINED BY validityModelID OPTIONAL }
```

C.2.9.10 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 IDENTIIFER ::= {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 a transaction is given by amount * $10^{exponent}$

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 IDENTIIFER ::= {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.2.9.11 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),
    priveledgeWithdrawn         (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.2.9.12 Professional Admission

Professional admission information can be expressed either in the base QES 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] EXPLICT 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] EXPLICT 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 QES certificate it shall be expressed that the professional is a physician. The minimum information to be provided in the *admission* extension of the base QES certificate is "PHYSICIAN" in the component *professionItems* of *AdmissionSyntax*.

```
{ ..., professionItems {"Ärztin/Arzt"}, ... }
```

Note: This and further professional information may 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 QES 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 QES 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 QES 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 QES 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.2.9.13 Restriction

This optional extension indicates some restrictions regarding the usage of the certificate. Professional group authorities responsible for the certificate contents may choose to limit the usage of the certificate to the health sector only. If set, the extension must not be marked critical.

ASN.1 definition of type *Restriction*:

```
id-isismtt-at-restriction OBJECT IDENTIFIER ::= {isismtt-at 8}
RestrictionSyntax ::= DirectoryString (SIZE(1..1024))
```

The encoding of the DirectoryString shall be UTF-8.

C.2.9.14 Additional Information

This optional extension indicates some other information of non-restrictive nature regarding the usage of the certificate. Professional group authorities responsible for the certificate contents may choose to include information such as “This certificate belongs to an approved medical id card issued by the responsible local medical chamber” or “Zertifikat als Teil eines elektronischen Arztausweises, herausgegeben durch die zuständige Landesärztekammer”. If set, the extension must not be marked critical.

ASN.1 definition of type *AdditionalInformation*:

```
id-isismtt-at-additionalInformation OBJECT IDENTIFIER ::= {isismtt-at 15}
AdditionalInformationSyntax ::= DirectoryString (SIZE(1..2048))
```

The encoding of the DirectoryString shall be UTF-8.

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 QES certificate in table form and in complete syntax form

The following table shows the QES certificate in table form. Only fields relevant to a QES 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-QES 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	e.g. 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 I	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
authorityInfoAccess	Identification of how and where information about the status of the certificate can be obtained	X	X
validityModel	Information about the mechanisms that must be used in order to prove the validity of the certificate		X
qcStatements	Indication that the certificate is a qualified certificate, and where applicable monetary limit of transactions	X	X
cRLDistributionPoints	Identification of how CRL information can be obtained	X	X
admission	Professional admission information	X	X
restriction	Other restrictions regarding the usage of the certificate	X	X
additionalInformation	Non-restrictive information regarding the usage of the certificate	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 QES-certificate of the health professional

The complete syntax form of the QES 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 { RDNSequence }

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

-- key usage extenion
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] IMPLICT 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),
    priveledgeWithdrawn        (7),
    aACompromise               (8) }

AuthorityInfoAccessSyntax ::= SEQUENCE (1..MAX) OF AccessDescription
AccessDescription ::= SEQUENCE {
                           accessMethod   OBJECT IDENTIFIER
                           accessLocation GeneralName }

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

Admissions                ::= SEQUENCE {
```

```
    admissionAuthority          [0] EXPLICT 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]  EXPLICT 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 }


ValidityModel              ::= SEQUENCE {
      validityModelId             OBJECT IDENTIFIER
      validityModelInfo           ANY DEFINED BY validityModelID OPTIONAL }

RestrictionSyntax          ::= DirectoryString (SIZE(1..1024))

AdditionalInformationSyntax ::= DirectoryString (SIZE(1..2048))
```

C.6 Object Identifiers Used in this Specification

The subsequent table shows the relevant object identifiers.

Table C.2 – Overview of OIDs

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 PuK
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 card holder is not allowed to perform 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-restriction	{ id-isismtt-at 8 }	Indicates other (non-monetary) restriction on certificate usage
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-at-additionalInformation	{ id-isismtt-at 15 }	Indicates non-restricitve information about certificate usage
id-isismtt-cp	{ id-isismtt 1 }	arc for ISIS-MTT certificate policies
id-isismtt-cp-accredited	{ 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-validityModel	{ 1 3 6 1 4 1 8301 3 5 }	arc for BNA extension to indicate validity model
id-validityModel-chain	{ id-validityModel 1 }	Indicates the chain model as the underlying validity model
id-validityModel-shell	{ id-validityModel 2 }	Indicates the shell model as the underlying validity model
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 is in accordance with [ECDIR]
id-etsi-qcs-QcLimitValue	{ id-etsi-qcs 2 }	indicates a limitation of the transaction value in acc. with [ECDIR]
id-etsi-qcs-QcRetentionPeriod	{ id-etsi-qcs 3 }	indicates how many years after expiry date of certificate external information about the card holder of the QC will be retained

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 QES 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 {
    acinfo                     AttributeCertificateInfo,
    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 *issuerUniqueID* Must NOT be used. The reference to the QES base certificate to which the attribute certificate belongs is made by specifying the issuer and the certificate serial number of the QES base certificate.

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

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

The *issuerUID* Must NOT be used. 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*. Attribute certificates must include the following exten-

sions with appropriate values as in the corresponding base certificates: *AuthorityKeyIdentifier*, *CertificatePolicies*, *CRLDistributionPoints*, *AuthorityInfoAccess* and *QCStatements*.
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 "KASSENAERZTLICHE VEREINIGUNG BAYERNS",
  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" } } } } }
```

C. 8 Certification Authorities for X.509 Certificates

The HPC security mechanisms require different types of end user X.509 certificates:

- QES certificate (X.509 qualified electronic signature certificate) and up to 3 attribute certificates
- AUT certificate (X.509 authentication certificate)
- ENC certificate (X.509 encipherment certificate).

End user certificates are produced by one or more Certification Authorities ("Zertifizierungsdienstean-bieter ZDAs"). The root CA for qualified electronic signatures produces a certificate for CA Health Care, whereby the certified key pair is only used for the production of end user certificates for qualified electronic signatures. The CA Health Care produces also AUT and ENC X.509 certificates, whereby the public key of the related key pair is signed by the HP Sector Root CA.

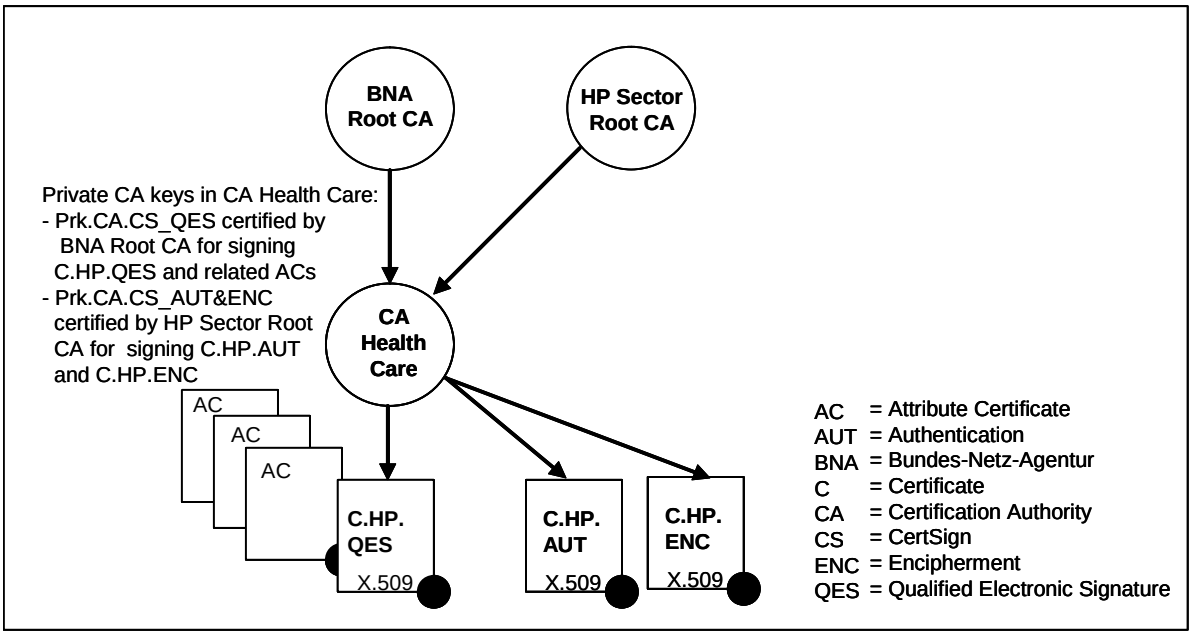

Figure C.1 – X.509 Certificates and Certification Authorities

NOTE – The role of a CA Health Care can be performed by several institutions (akkreditierte ZDAs).

Annex D
(normative)

Certificates for Authentication and Key Encipherment

D.1 Structure and content

The AUT-certificate and the ENC-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 ENC-certificate contains information about the certificate holder, i.e. the receiver of a confidential document.

D.2 Coding

The coding is structurally equivalent to the QES public key certificate with changes such as:

- 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. For the purpose of web-based client authentication with the TLS protocol, the "key encipherment" bit may optionally be included in the AUT certificate according to the ISIS-MTT Authentication Certificate Profile.
- key usage is set to "key encipherment" and "data encipherment" in the ENC certificate.
- the algorithm OID of the subject public key info field has to be set according to the use of the PK certified.

For the purpose of web-based client authentication with the TLS protocol, the non-critical extendedKeyUsage extension may optionally be included in AUT certificates accordingly to the ISIS-MTT Authentication Certificate Profile. If this extension is used clientAuth { 1 3 6 1 5 5 7 3 2 } SHALL be set as KeyPurposeID.

ASN.1-Definition of Type *ExtendedKeyUsage*

```
ExtendedKeyUsage          ::=  SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPurposeId              ::=  OBJECT IDENTIFIER
```

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, Clause 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] Clause 9.2 (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 PSO: HASH and PSO: COMPUTE DS

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

Table E.1 – Algorithm Identifier for QES

AlgID	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)
'41'	SHA-256 and RSA with DSI according to ISO/IEC 9796-2 with RND	{1 3 36 3 4 3 2 4} (TTT)
'42'	SHA-256 and RSA with DSI according to PKCS #1	{1 2 840 113549 1 1 11} (RSADSI)

The OIDs of the hash algorithms relevant for the construction of the digestinfo are shown in Table 6.a of [HPC-P1].

E.2 Algorithm Identifier for INTERNAL AUTHENTICATE

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

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

AlgID	Meaning
'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

E.3 Decryption of document cipher key

Table E.3 shows the algorithm identifier relevant for decipherment. The encipherment input format is specified in [HPC-P1], Table E.2.

Table E.3 - Algorithm Identifier for decipherment

AlgID	Meaning
'1A'	RSA , PI $\neq$ '00', see Table E.4

Annex F
(normative)

Issuer Identifier and Certificate Holder Authorizations

F.1 Issuer Identifier

The issuer identifier in combination with the country code allows a worldwide unique identification of the card issuer. The BCD coded country code for Germany has the value ‘276’ according to [ISO 3166]. The issuer identifier (“Kartenausgeberschlüssel”) is assigned in Germany on behalf of DIN by GS1-Germany GmbH, Köln. The card issuer is usually the legal owner of the issued card especially in the case that the respective card is an identification card, since there may be reasons for recalling a card, e.g. termination of work or loss of approbation. Therefore a hint is usually printed on the back-side of the card, that the card issuer (e.g. a physician chamber) has the right to recall the respective card.

The issuer identifier is part of the ICCSN (Integrated Circuit Card Serial Number), which allows a worldwide unique identification of the respective card. For health care cards, such an identifier is required in the European standard [EN 1867]. In this standard, there is also specified the so called Major Industry Identifier (MII) for card issuers in the health care sector, which has the vale ‘80’.

Technically, the ICCSN or parts of it are usable e.g. for

- card revocation
- card individual key derivation
- key identification (e.g. the public key in a card verifiable certificate)
- constructing authentication related data
- binding objects or operations to a specific card.

The following table shows issuer identifiers used in German health care for Health Professional Cards (HPC) and Security Module Cards (SMC).

Table F.1 - Issuer Identification Numbers (incomplete list)

MII for Health care	Country Code Germany	Issuer Identifier
‘80’	‘276’	‘00101’ Werbe- und Vertriebsgesellschaft Deutscher Apotheker mbH
		’00108’ Bundesärztekammer
		’00109’ Bundespsychotherapeutenkammer
		’00110’ Bundeszahärztekammer
		'00xxx' Landeskrankenhausgesellschaften (DKTIG)

F.2 Role Identifier

A role identifier is a one-byte field in a card verifiable certificate used in an asymmetric card-to-card authentication procedure. Roles are assigned to entities, e.g.

- professional groups, e.g. physicians and pharmacists
- certificate service provider
- card management system.

The concept of role identifiers is used in several application sectors. Therefore it is essential, that the usage area of role identifiers can be clearly distinguished. For this purpose the role identifier is preceded by a prefix denoting

- the application area identified by an application identifier (AID) or part of it or

- the respective entity in charge of assigning the role identifier.

For German health care, the application identifier ‘D27600004000’ is used, which is worldwide unique. The Registered Application Provider Identifier (RID, first 5 bytes of an AID) has been assigned by the RID German National Registration Authority (now SIT on behalf of DIN) to ZI (Zentralinstitut für die kassenärztliche Versorgung in der Bundesrepublik Deutschland) for usage in German health care. The final byte of the above mentioned AID is the PIX (Proprietary Application Provider Extension) and set to ‘00’. This AID in combination with the respective role id represents the so called Certificate Holder Authorization (CHA) present in a card verifiable certificate, see [ISO 7816-8]. The CHA is used especially in security attributes related to data files so that for a given access mode (e.g. read or update) the card can verify whether the respective command can be executed or has to be rejected. Such a CHA is set as valid only in case an authentication procedure has been performed successfully. In the authentication procedure the external entity (e.g. a HPC of a health professional of group x) has to prove that it has the private key related to the public key presented in the card verifiable certificate. The usage of this private key is bound to the successful presentation of a PIN, so that the performance of an authentication procedure is protected.

If several professional groups have the same access rights to an eGK, then they have the same "profile". Therefore a profile identifier is coded in the CHA field "role identifier". This leads to a reduction of access rules in the eGK. The mapping of professional groups to a dedicated profile is described in another document.

The following tables show the CHA coding as used in CVC of HPC, SMC and eGK. The usage of CVC.HPC.AUT and CVC.SMC.AUT is outlined in [eGK-P2].

Table F.2 - CHA coding for HPCs and SMCs

CHA Prefix	CHA Role ID / Profile ID	CV Certificate holder
‘D27600004000’	‘1A’	Profile 1 (SMC eKiosk)
‘D27600004000’	‘2A’	Profile 2 (HPC, SMC)
‘D27600004000’	‘3A’	Profile 3 (HPC, SMC)
‘D27600004000’	‘4A’	Profile 4 (HPC, SMC)
‘D27600004000’	‘5A’	Profile 5 (HPC, SMC)
‘D27600004000’	‘6A’	Profile 6 (SMC for internet pharmacies)

Table F.3 - CHA in CVCs issued by RCA for a CA (CA-eGK and CA-HPC)

CHA Prefix	CHA Role ID	CV Certificate holder
… (5 B) || ‘00’	‘00’	Certification Authority (CA)

NOTE – As CA name for the Root CA in health care the acryonym “DEZGW” may be used, which has been registered 1999 for a (virtual) „Deutsche Zertifizierungsstelle Gesundheitswesen (DEZGW)“.

Table F.4 - CHA of the eGK (informative)

CHA Prefix	CHA Role ID	CV Certificate holder
‘D27600000100’	‘00’	eGK

Annex G
(normative)

Content of EFs for Personalization

G.1 EF.ATR

Table G.1 – Content of EF.ATR

Tag	L	Value	Meaning
'E0'	'xx'	'02 xx xxxx 02 xx xxxx 02 xx xxxx 02 xx xxxx'	I/O buffer sizes, see [HPC-P1]
'66'	'xx'	'46 xx …' see Table G.2	DO Card Data

Table G.2 – Value of DO Pre-issuing Data (Tag '46')

L (byte)	Meaning of concatenated data elements (most significant byte: ICM)
1	IC manufacturer ID ICM (see www.sc17.com)
5	Card manufacturer ID (see DIN-RA: www.sit.fraunhofer.de)
x	IC-ID (card manufacturer specific)
x	COS version (card manufacturer specific)
x	ROM mask (card manufacturer specific)

G.2 EF.DIR

Table G.3 – Application Templates in EF.DIR

Tag	L	Application Template	Meaning
'61'	'08'	'4F 06 D276 00004002'	Application Template with AID.HPA
'61'	'08'	'4F 06 D276 00006601'	Application Template with AID.QES (DINSIG)
'61'	'0C'	'4F 0A A000000167 455349474E'	Application Template with AID.ESIGN
'61'	'11'	'4F 0F E828BD080F A000000167 455349474E'	Application Template with AID.CIA.ESIGN

G.3 EF.GDO

Table G.4– DO ICCSN in EF.GDO

Tag	L	Value	Meaning
'5A'	'0A'	'80276 …'	ICCSN

G.4 EF.CVC.CA_HPC.CS and EF.CVC.HPC.AUT

In the CVC related EFs the respective CVC has to be stored (general coding: see [HPC-P1]).

G.5 EF.C.HP.QES, EF.C.HP.AUT and EF.C.HP.ENC

In the X.509 certificate related EFs the respective certificate has to be stored (see Annex C and D).

G.6 EF.CIAInfo

The content of EF.CIAInfo is shown in H.1.3.

G.7 EF.OD

The content of EF.OD is shown in H.2.3.

G.8 EF.AOD

The content of EF.AOD is shown in H.3.3.

G.9 EF.PrKD

The content of EF.PrKD is shown in H.4.3.

G.10 EF.CD

The content of EF.CD is shown in H.5.3.

Annex H
(informative)

Cryptographic Information Objects

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

H.1.1 ASN.1 Value notation

```
1       ciaInfoExample CIAInfo ::= {
2           version v2,
3           label  “CIA of ESIGN Application”,
4           cardflags { authRequired, prnGeneration }
5           seInfo {
7                       se 1,
8                       aid 'A000000167 455349474E’ H
                }}
```

H.1.2 ASN.1 Description, tags, lengths and values

```
1   CIAInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 193
2     version INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
        1
3     label Label UTF8String: tag = [0] primitive; length = 29
        0x4865616C74682050726F66657373696F6E616C20436172642076322E30
4     cardflags CardFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
        0x0560
5     seInfo SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 26
6       SecurityEnvironmentInfo SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 11
7         se INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            0x01
8         aid OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 10
            0xA000000167455349474E
```

H.1.3 Hexadecimal DER-encoding

```
1   30 34
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 10
6               30 0B
7                     02 01
                            01
8                     04 0A
                            A000000167455349474E
```

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

```
authObjects :
        path : {
                efidOrPath '14'H -- SFID of EF.AOD
                },
privateKeys :
        path : {
                efidOrPath '15'H -- SFID of EF.PrKD
                },
certificates :
        path : {
                efidOrPath '16'H -- SFID of EF.CD
                }
```

H.2.2 ASN.1 Description, tags, lengths and values

```
CIOChoice CHOICE
  authObjects : tag = [8] constructed; length = 5
    AuthObjects CHOICE
      path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
        efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x14

CIOChoice CHOICE
  privateKeys : tag = [0] constructed; length = 5
    PrivateKeys CHOICE
      path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
        efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x15

CIOChoice CHOICE
  certificates : tag = [4] constructed; length = 5
   Certificates CHOICE
      path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
       efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x16
```

H.2.3 Hexadecimal DER-encoding

```
A8 05
      30 03
            04 01
                  14
A0 05
      30 03
            04 01
                  15
A4 05
     30 03
          04 01
                  16
```

H.3 EF.AOD

The “authentication object directory” EF holds the information about the PIN.CH.

H.3.1 ASN.1 Value notation

```
pwd : {
```

```
2           commonObjectAttributes {
3                   label "PIN.CH",
4                   flags { private },
5                   authId '12'H
                    },
6           classAttributes {
7                   authId '02'H
                    },
8           typeAttributes {
9                   pwdFlags { initialized, exchangeRefData },
10                  pwdType iso9564-1,
11                  minLength 5,
12                  storedLength 0,
13                  maxLength 8,
14                  pwdReference 1          -- P2 value of the VERIFY command
                    }
            },
15      pwd : {
16          commonObjectAttributes {
17                  label "PUK.HP",
18                  flags { private }
                    },
19          classAttributes {
20                  authId '12'H
                    },
21          typeAttributes {
22                  pwdFlags {  global, change-disabled, unblock-disabled, initialized, unblockingPassword },
23                  pwdType iso9564-1,
24                  minLength 8,
25                  storedLength 8,
26                  maxLength 8,
27                  pwdReference 1          -- P2 value of the RESET RC command
                    }
            }
```

H.3.2 ASN.1 Description, tags, lengths and values

```
1   AuthenticationObjectChoice CHOICE
        pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 50
2        commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
          constructed; length = 19
3          label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
              0x50494E2E48502E415353
4          flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
              0x0780
5          authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
              0x12
6        classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
          tag = [UNIVERSAL 16] constructed; length = 3
7          authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
              0x02
8        typeAttributes : tag = [1] constructed; length = 22
            PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 20
9            pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
             length = 3
               0x040810
10           pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
               4
11           minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
               5
12           storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
```

```
            0
13          maxLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              8
14          pwdReference Reference INTEGER: [0] primitive; length = 1
              0x01

15  AuthenticationObjectChoice CHOICE
      pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 46
16      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 16
17        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
            0x50554B2E48502E415353
18        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
            0x0780
19      classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
        tag = [UNIVERSAL 16] constructed; length = 3
20        authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
            0x12
21      typeAttributes : tag = [1] constructed; length = 21
          PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 19
22          pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
            length = 2
              0x013A
23          pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
              4
24          minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              8
25          storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              8
26          maxLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              8
27          pwdReference Reference INTEGER: [0] primitive; length = 1
              0x01
```

H.3.3 Hexadecimal DER-encoding

```
1  30 32
2        30 13
3              0C 0A
                      50 49 4E 2E 48 50 2E 41 53 53
4              03 02
                      07 80
5              04 01
                      12
6        30 03
7              04 01
                      02
8        A1 16
                30 14
9                    03 03
                            04 08 10
10                   0A 01
                            04
11                   02 01
                            05
12                   02 01
                            00
13                   02 01
                            08
14                   80 01
                            01

15  30 2E
16        30 10
17              0C 0A
                      50 55 4B 2E 48 50 2E 41 53 53
```

```
18              03 02
                      07 80
19        30 03
20              04 01
                      12
21        A1 15
                30 13
22                    03 02
                            01 3A
23                    0A 01
                            04
24                    02 01
                            08
25                    02 01
                            08
26                    02 01
                            08
27                    80 01
                            01
```

H.4 EF.PrKD

The “private key directory” EF provides necessary information about the two private keys PrK.HP.AUT and PrK.HP.ENC.

H.4.1 ASN.1 Value notation

```
1     privateRSAKey : {
2           commonObjectAttributes {
3                 label "PrK.HP.AUT",
4                 flags { private },
5                 authId '02'H              -- pointer to the AOD-entry of PIN.CH
6                 accessControlRules {
7                       {
8                             accessMode { execute },
9                             securityCondition and : {
10                                  authId : '02'H,    -- pointer to the AOD-entry of PIN.CH
11                                  authReference : {
12                                        authMethod { userAuthentication }
                                          }
                                    }
                              }
                        }
                  },
13          classAttributes {
14                iD '82'H,                 -- cross-reference to the CD-entry of C.HP.AUT
15                usage { signRecover },
16                keyReference –130         -- used in MSE-SET-command
                  },
17          typeAttributes {
18                value {
                        efidOrPath "H
                        },
19                modulusLength 1728
                  }
          },

20    privateRSAKey : {
21          commonObjectAttributes {
22                label "PrK.HP.ENC",
23                flags { private },
24                authId '02'H              -- pointer to the AOD-entry of PIN.CH
25                accessControlRules {
26                      {
27                            accessMode { execute },
28                            securityCondition and : {
29                                  authId : '02'H,    -- pointer to the AOD-entry of PIN.CH
```

```
30                                      authReference : {
31                                          authMethod { userAuthentication }
                                            }
                                        }
                                    }
                                }
                            }
                        },
32                  classAttributes {
33                      iD '83'H,               -- in this specification unused (but mandatory)
34                      usage { keyDecipher },
35                      keyReference –131       -- used in MSE-SET-command
                        },
36                  typeAttributes {
37                      value {
                            efidOrPath "H
                            },
38                      modulusLength 1728
                        }
                },
```

H.4.2 ASN.1 Description, tags, lengths and values

```
1  PrivateKeyChoice CHOICE
       privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 61
2       commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
         constructed; length = 38
3          label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
             0x50724B2E48502E415554
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          accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 17
7            AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 15
8              accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                 0x0520
9              securityCondition SecurityCondition CHOICE
                 and SEQUENCE OF: tag = [1] constructed; length = 9
10                 authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
                   length = 1
                     0x02
11                 authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
                   structed; length = 4
12                   authMethod AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
                     length = 2
                       0x0520
13       classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
         structed; length = 10
14         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
             0x82
15         usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
             0x0410
16         keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
           length = 1
             0x82
17       typeAttributes : tag = [1] constructed; length = 10
           PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
           length = 8
18           value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
                   efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
```

```
19          modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
              1728

20  PrivateKeyChoice CHOICE
      privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 60
21      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
        constructed; length = 37
22        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 9
            0x50724B2E48502E4B45
23        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
          length = 2
            0x0780
24        authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
            0x02
25        accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 17
26          AccessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 15
27            accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                0x0520
28            securityCondition SecurityCondition CHOICE
                and SEQUENCE OF: tag = [1] constructed; length = 9
29                authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive;
                  length = 1
                    0x02
30                authReference AuthReference SEQUENCE: tag = [UNIVERSAL 16] con-
                  structed; length = 4
31                  authMethod AuthMethod BIT STRING: tag = [UNIVERSAL 3] primitive;
                    length = 2
                      0x0520
32      classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
        structed; length = 10
33        iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
            0x83
34        usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x0204
35        keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive;
          length = 1
            0x83
36      typeAttributes : tag = [1] constructed; length = 10
          PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
          length = 8
37          value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
                  efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
38          modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
              1728
```

H.4.3 Hexadecimal DER-encoding

```
1  30 3D
2        30 26
3              0C 0A
                      50 72 4B 2E 48 50 2E 41 55 54
4              03 02
                      07 80
5              04 01
                      02
6              30 11
7                    30 0F
8                          03 02
                                 05 20
9                          A1 09
10                               04 01
                                       02
11                               30 04
```

```
12                                          03 02
                                                  05 20
13          30 0A
14                04 01
                      82
15                03 02
                      04 10
16                02 01
                      82
17          A1 0A
                30 08
18                    30 02
                            04 00
19                    02 02
                            04 00

20    30 3C
21          30 25
22                0C 09
                      50 72 4B 2E 48 50 2E 4B 45
23                03 02
                      07 80
24                04 01
                      02
25                30 11
26                    30 0F
27                          03 02
                                  05 20
28                          A1 09
29                                04 01
                                        02
30                                30 04
31                                      03 02
                                              05 20
32          30 0A
33                04 01
                      83
34                03 02
                      02 04
35                02 01
                      83
36          A1 0A
                30 08
37                    30 02
                            04 00
38                    02 02
                            04 00
```

H.5 EF.CD

The “certificate directory” EF contains the file references to the X.509 certificates C.HP.AUT and C.HP.ENC.

H.5.1 ASN.1 Value notation

```
1       x509Certificate : {
2               commonObjectAttributes {
3                       label "C.HP.AUT",
4                       flags { private },
5               classAttributes {
6                       iD '82'H                -- cross-reference to the PrKD-entry of PrK.HP.AUT
                        },
7               typeAttributes {
8                       value indirect :
                                path : {
9                                       efidOrPath '01'H --SFID of C.HP.AUT
                                        }
                        }
                },
10      x509Certificate : {
11              commonObjectAttributes {
```

```
12              label "C.HP.ENC",
13              flags { private },
14          classAttributes {
15              iD '83'H            -- cross-reference to the PrKD-entry of PrK.HP.ENC
                },
16          typeAttributes {
17              value indirect :
                    path : {
18                      efidOrPath '02'H --SFID of C.HP.ENC
                        }
                }
```

H.5.2 ASN.1 Description, tags, lengths and values

```
1  CertificateChoice CHOICE
       x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 33
2       commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
         constructed; length = 17
3         label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 8
             0x432E48502E415554
4         flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
             0x0780
5       classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
         constructed; length = 3
6         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
             0x82
7       typeAttributes : tag = [1] constructed; length = 7
           X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
           length = 5
8           value CHOICE
               indirect ReferencedValue CHOICE
                 path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
9                 efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                     0x01
10 CertificateChoice CHOICE
       x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 33
11       commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
         constructed; length = 17
12        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 8
             0x432E48502E415554
13        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
             0x0780
14       classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
         constructed; length = 3
15         iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
             0x83
16       typeAttributes : tag = [1] constructed; length = 7
           X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
           length = 5
17           value CHOICE
               indirect ReferencedValue CHOICE
                 path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
18                 efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                     0x02
```

H.5.3 Hexadecimal DER-encoding

```
1  30 1E
2        30 11
3              0C 08
```

```
                    43 2E 48 50 2E 41 55 54
4             03 02
                    07 80
5       30 03
6             04 01
                    82
7       A1 07
                30 05
8                    30 03
9                          04 01
                                 01
10  30 1E
11       30 11
12            0C 08
                    43 2E 48 50 2E 41 55 54
13            03 02
                    07 80
14       30 03
15            04 01
                    83
16       A1 07
                30 05
17                  30 03
18                         04 01
                                 02
```