Specification

German
Health Professional Card
and
Security Module Card

Part 2: HPC Applications and Functions

Version 2.3.2

05.08.2009

Bundesärztekammer

Kassenärztliche Bundesvereinigung

Bundeszahnärztekammer

Bundespsychotherapeutenkammer

Kassenzahnärztliche Bundesvereinigung

Bundesapothekerkammer

Deutsche Krankenhausgesellschaft

Editor: Ulrich Waldmann (Fraunhofer 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 F.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 D.3
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 improvements
21.02.2006	HPC V2.1.0	Changes: - Harmonization with eGK - some minor precisions and corrections
18.05.2007	HPC V2.1.1	Changes: - Some precisions for error cases - Some clarifications - Addition of profile 7: paramedic - Some adjustments to gematik SRQs No security relevant changes, no need to adjust the PP
04.07.2008	HPC V2.2.0- V2.3.0	Changes: - Adjustments for interoperability with the eGK generation 1 - Changes in respect to stack and comfort signatures - Adjustments of algorithms, key lengths etc. to BSI technical guidelines
09.04.2009	V2.3.1	SRQ-0010 "HPC-P2 Corrigenda1" SRQ-0014 "Introductionkeys in HBA und SMC" SRQ-0015 "Inhalt von EF.Version"
05.08.2009	V2.3.2	SRQ-0016 "Introductionkeys und Sicherheitszustand" SRQ-0017 "Festlegungen zum CAMS-Schlüssel"

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 an 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 [SigG01] and Signatur-Verordnung SigV [SigV01])
- 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 an HPC is at least 3 years. A health professional may have more than one HPC.

2 References

[ALGCAT]
Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen: Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen) vom 17. November 2008; see www.bundesnetzagentur.de

[COM-PKI-1]
T7, TeleTrusT: Common PKI Specification, Part 1: Certificate and CRL Profiles, Version 2.0, 20th January 2009, www.common-pki.org

[COM-PKI-9]
T7, TeleTrusT: Common PKI Specification, Part 9: SigG-Profile, Version 2.0, 20th January 2009, www.common-pki.org

[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]
gematik: Spezifikation der elektronischen Gesundheitskarte
Teil 1: Spezifikation der elektrischen Schnittstelle
Version 2.2.2, 16.09.2008

[eGK-P2]
gematik: Spezifikation der elektronischen Gesundheitskarte
Teil 2: Grundlegende Applikationen
Version 2.2.1, 19.06.2008

[EN14890-1]
EN 14890-1: 2008
Application Interface for smart cards used as secure signature creation devices
Part 1: Basic services

[EN14890-2]
EN 14890-2: 2008
Application Interface for smart cards used as Secure Signature Creation Devices
Part 2: Additional services

[EN1867]
prEN 1867: 1997
Machine readable cards - Health care applications - Numbering system and registration procedure for issuer identifiers

[GMG]
Gesetz zur Modernisierung der gesetzlichen Krankenversicherung (GKV-Modernisierungsgesetz - GMG), BGBl 2003 Teil I Nr. 55 S.2190, 19. November 2003

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

Part 1: Commands, Algorithms and Functions of the COS Platform
V2.3.2, 05.08.2009

[HPC-P3]
German Health Professional Card and Security Module Card
Part 3: SMC Applications and Functions
V2.3.2, 05.08.2009

[ISO3166]
ISO/IEC 3166-1: 2006
Codes for the representations of names of countries and their subdivisions – Part 1: Country codes

[ISO7812]
ISO/IEC 7812-1: 2006
Identification cards – Identification of issuers – Part 1: Numbering system

[ISO7816-1]
ISO/IEC 7816-1: 1998
Identification cards - Integrated circuit cards with contacts - Part 1: Physical characteristics

[ISO7816-2]
ISO/IEC 7816-2: 2007
Identification cards - Integrated circuit cards with contacts - Part 2: Dimensions and location of contacts

[ISO7816-3]
ISO/IEC 7816-3: 2006
Identification cards - Integrated circuit cards with contacts - Part 3: Electrical interface and transmission protocols

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

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

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

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

[ISO7816-9]
ISO/IEC 7816-9: 2004
Identification cards - Integrated circuit cards - Part 9: Commands for card management

[ISO7816-13]
ISO/IEC 7816-13: 2007
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: 2002
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: 2002
Banking – Personal Identification Number (PIN) management and security –
Part 1: Basic principles and requirements for online PIN handling in ATM and POS systems

[ISO9796-2]
ISO/IEC 9796-2: 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

[ISO10646]
ISO/IEC 10646:2003
Information technology -- Universal Multiple-Octet Coded Character Set (UCS)

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

[ISO11770]
ISO/IEC 11770-3: 2008
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

[PKI-Reg]
gematik: Registrierung einer CVC-CA der zweiten Ebene
Version 1.8.0

[PKI-Nota]
gematik: Festlegungen zu den Notationen von Schlüsseln und Zertifikaten kryptographischer Objekte in der TI, Version 1.1.0

[PP-HPC]
BSI: Common Criteria Protection Profile – Health Professional Card (PP-HPC) with SSCD Functionality, BSI-CC-PP-0018-V2-2009, Version 2.5, April 6th, 2009

[PP-SMC-A]
BSI: Common Criteria Protection Profile – Secure Module Card Type A (PP-SMC-A), BSI-PP-0019, Version 1.9.1, February 1st, 2008

[PP-SMC-B]
BSI: Common Criteria Protection Profile – Secure Module Card Type B (PP-SMC-B), BSI-PP-0019, Version 1.9.1, February 1st, 2008

[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

[RFC3629]
UTF-8, a transformation format of ISO 10646, November 2003

[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

[SMC-K]
gematik: Spezifikation der SMC-K
Version 1.2.0

[SSL]
Netscape: SSL3.0 Specification

[TID]
Spezifikation des Aufbaus der Telematik-ID für HBA und SMC, Version 1.0.0, 22.08.2008

[TR-03114]
BSI: TR-0311, Stapelsignatur mit dem Heilberufsausweis, Version 2.0, 19.10.2007,
www.bsi.de/literat/tr/tr03114/BSI-TR-03114.pdf

[TR-03115]
BSI: TR-03115, Komfortsignatur mit dem Heilberufsausweis, Version 2.0, 19.10.2007,
www.bsi.de/literat/tr/tr03115/BSI-TR-03115.pdf

[TR-03116]
BSI: TR-03116, Technische Richtlinie für die eCard-Projekte der Bundesregierung, Version: 3.0,
Datum: 08.04.2009, www.bsi.de/literat/tr/tr03116/BSI-TR-03116.pdf

3 Abbreviations and Notations

3.1 Abbreviations

AC	Attribute Certificate
AID	Application Identifier
AKS	Auslöser KomfortSignatur (Comfort Signature Trigger)
AOD	Authentication Object Directory
APDU	Application Protocol Data Unit [ISO7816-3]
ASN.1	Abstract Syntax Notation One
ASCII	American Standard Code for Information Interchange
ATR	Answer-to-Reset
AUT	Authentication
AUTD	CV-based device-authentication
AUTR	CV-based role-authentication
AUTO	Organization-specific Authentication
BA	Berufsausweis (professional card)
BCD	Binary Coded Decimal
BER	Basic Encoding Rules
BNA	Bundesnetzagentur (Federal Network Agengy)
C	Certificate
C2C	Card-to-Card
CA	Certification Authority
CAMS	Card Application Management System
CAR	Certification Authority Reference
CC	Cryptographic Checksum
CD	Certificate Directory
CER	Canonical Encoding Rules
CG	Cryptogram
CH	Cardholder
CHA	Certificate Holder Authorization
CHR	Certificate Holder Reference
CIA	Cryptographic Information Application
CIO	Cryptographic Information Objects
CLA	Class-Byte of a command APDU
COS	Card Operating System
CPI	Certificate Profile Identifier
CRL	Cerificate Revocation List

CS	CertSign (CertificateSigning)
CTA	Card Terminal Application
CV	Card Verifiable
CVC	Card Verifiable Certificate
D,DIR	Directory
DE	Data Element
DER	Distinguished Encoding Rules
DES	Daten Encryption Standard
DF	Dedicated File
DI	Baud rate adjustment factor
DM	Display Message
DO	Data Objekt
DS	Digital Signature
DSI	Digital Signature Input
DTBS	Data to be signed
ECDSA	Elliptic Curve Digital Signature Algorithm
EF	Elementary File
eGK	elektronische Gesundheitskarte (electronic Health Card) [eGK-P1] and [eGK-P2]
EHIC	European Health Insurance Card
ENC	Encryption
ES	Electronic Signature
FCI	File Control Information
FCP	File Control Parameter
FI	Clock rate conversion factor
FID	File Identifier
GDO	Global Data Object
GKV	Gesetzliche Krankenversicherung (compulsory health insurance)
GP	Global Platform
HB	Historical Bytes
HBA	Heilberufsausweis (Health Professional Card)
HCI	Health Care Institution
HP	Health Professional
HPA	Health Professional Application
HPC	Health Professional Card
HPD	Health Professional related Data
ICC	Integrated Circuit Card
ICCSN	ICC Serial Number

ICM	IC Manufacturer
ID	Identifier
IETF	The Internet Engineering Task Force
IFSC	Information Field Size Card
IIN	Issuer Identification Number
INS	Instruction-Byte of a command APDU
KeyRef	Key Reference
KM	Komfortmerkmal (comfort feature)
KT	Karten-Terminal (card terminal)
LCS	Life Cicle Status
LSB	Least Significant Byte(s)
MAC	Message Authentication Code
MF	Master File
MII	Major Industry Identifier
MSE	Manage Security Environment
OCSP	Online Certicate Status Protocol
OD	Object Directory
OID	Object Identifier
OSIG	Organisation Signature
PIN	Personal Identification Number
PIX	Proprietary Application Provider Extension
PK,PuK	Public Key
PKCS	Public Key Cryptography Standard (here [PKCS#1])
PKI	Public Key Infrastructure
PKIX	Public Key Infrastructure for X.509 Certificates (IETF)
PP	Protection Profile
PrK	Private Key
PSO	Perform Security Operation
PUK	Personal Unblocking Key (Resetting Code)
PV	Plain Value
P1	Parameter P1 of a command APDU
P2	Parameter P2 of a command APDU
QES	Qualified Electronic Signature
RA	Registration Authority
RAM	Random Access Memory
RC	Retry Counter
RCA	Root CA
RD	Reference Data

RF	Radio Frequency
RFC	Request für Comment
RFID	Radio Frequency Identification
RFU	Reserved for future use
RID	Registered Application Provider Identifier
RND	Random Number
ROM	Read Only Memory
RPE	Remote PIN-Receiver
RPS	Remote PIN-Sender
RSA	Algorithmus of Rivest, Shamir, Adleman [RSA]
SAK	Signaturanwendungskomponente (signature creation component)
SE	Security Environment
SFID	Short EF Identifier
SIG	Signature
SigG	Signaturgesetz (Signature Law) [SigG01]
SigV	Signaturverordnung (Signature Ordinance) [SigV01]
SK	Secret Key
SM	Secure Messaging
SMA	Security Module Application
SMC	Security Module Card
SMD	Security Module Data
SMKT	Sicherheitsmodul Kartenterminal (Security Module Card Terminal)
SN	Serial Number
SO	Security Officer (administrator)
SSCD	Secure Signature Creation Device
SSEC	Security Status Evaluation Counter
SSEE	Sichere Signaturerstellungseinheit (secure signature creation device)
SSL	Security Sockets Layer [SSL]
SUK	Stapel- und Komfortsignatur (stack and comfort signature)
TLV	Tag Length Value
TC	Trusted Channel
TLS	Transport Layer Security
UID	User Identification
UTF8	8-bit Unicode Transformation Format
WTLS	Wireless Transport Layer Security
ZDA	Zertifizierungsdiensteanbieter (Certification Authority)
3TDES	3-Key-Triple-DES

3.2 Notations

For keys and certificates the following simplified Backus-Naur notation applies (see [PKI-Nota] for definitions):

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

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

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

<private key>::= PrK (asym.)
<public key>::= PuK (asym.)
<secret key>::= SK (sym.)

<keyholder>::= <health professional> | <card holder> | <certification authority> |
<health professional card> | <card application management system> |
<health care institution> | <security module card> | <signature application component> |
<security module card terminal> |
<electronic health card>

<health professional>::= HP
<card holder>::= CH
<certification authority>::= <root certification authority> |
<certification authority for CAMS of HPC> | <certification authority for HPC> |
<certification authority for SMC> | <certification authority for eGK> |
<certification authority for comfort signature trigger>
<root certification authority>::= RCA
<certification authority for card application management system of health professional card>::=
CA_CAMS_HPC
<certification authority for health professional card>::= CA_HPC
<certification authority for security module card>::= CA_SMC
<certification authority for electronic health card>::=
CA_eGK (CA elektronische Gesundheitskarte)
<certification authority for comfort signature trigger>::= CA_KM (CA Komfortmerkmal)

<health professional card>::= HPC
<card application management system> ::= CAMS
<health care institution>::= HCI
<security module card>::= SMC
<signature application component>::= SAK
<security module card terminal>::= SMKT
<electronic health card>::= eGK (elektronische Gesundheitskarte eGK)

<key usage>::= <organizational signature> | <encipherment> | <authentication> |
<certsign cvc> | <certsign x509>

<organizational signature>::= OSIG
<encipherment>::= ENC
<certsign cvc>::= CS
<certsign x509>::= CA

<authentication>::= AUT | <cv based authentication>

<cv based authentication>::= <role authentication> | <device authentication>
<role authentication> ::= AUTR_CVC
<device authentication> ::= AUTD_CVC | <remote pin sender> | <remote pin receiver> |
<stack and comfort signature card> | <comfort signature trigger> |
<signature application component>

<remote pin Sender>:: = AUTD_RPS_CVC (Remote PIN Sender)
<remote pin Receiver>::= AUTD_RPE_CVC (Remote PIN Empfänger)
<stack and comfort signature card>::= AUTD_SUK_CVC (Stapel- und Komfortsignatur)
<comfort signature trigger>::= AUTD_AKS_CVC (Auslöser Komfort Signatur)

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

<certificate>::= C

<certificate holder>::=
<health professional> | <certification authority> | <health professional card> |
<card application management system> | <security module card> |
<security module card terminal> | <electronic health card>

<certificate usage>::=
<organizational signature> | <encipherment> | <authentication> | <certsign cvc> |
<certsign x509>

For subsequent data items the following notation is used:

|| = Concatenation of data

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

4 The Health Professional Card

4.1 ATR coding

Table 1 – (N2001.00) ATR coding (sequence from top to down)

Parameter	Value	Meaning
TS	‘3B’	Initial Character (direct convention)
T0	'Dx' or ‘9x’	Format Character (TA1 / TD1 indication), 'Dx' means TC1 is available, '9x' means TC1 is absent, x = number of Historical Bytes; x = 0 recommended, i.e. no Historical Bytes in ATR, but available only in EF.ATR
TA1	‘xx’	Interface Character (FI / DI value), 'xx' = '18', '95', '96' or '97'
TC1	'FF'	Extra Guard Time Integer; allowed options are: a) TC1 available with value 'FF' (strongly recommended), b) TC1 absent
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	xx000x11	Interface Character (XI / UI coding), x = manufacturer-specific
CI	'00' or '80'	Category Indicator Byte (first byte of max. 15 Historical Bytes) '00' means a status indicator is present as the last three historical bytes, '80' means a status indicator is present in a data object (one, two or three bytes)
Tag/Length	'6x'	Compact header of Pre-issuing Data Object (PIDO) with x = number of subsequent PIDO bytes
ICM	'xx'	IC Manufacturer Identifier (part of PIDO) Assigned by ISO for this IC manufacturer, see www.sc17.com
ICT	'xx' or 'xxxx'	IC Type (part of PIDO), 1 byte if b8 = 0 or 2 bytes if b8 = 1 in the first byte; Coding manufacturer-specific
OSV	'xx'	Operating System Version (part of PIDO) Coding manufacturer-specific
DD	'xx...'	Discretionary Data (part of PIDO), n bytes Coding manufacturer-specific
Tag/Length	'73'	Compact header of Card Capabilities Bytes (CCB) The Card Capabilities are also contained in EF.ATR, see Table 4 (N2005.00)
1. CCB	1xx10110 = 'x6'	Card Capabilities Byte of selection methods b8 b7 b6 b5 b4 = 1xx10: DF selection by full DF name and by file identifier, b3 = 1: support of Short EF identifier, b2 = 1: support of record number, b1 = 0: no support of record identifier
2. CCB	00100001 = '21'	Card Capabilities Byte of data coding b8 = 0: no support of EFs of TLV structure, b7 b6 = 01: proprietary behavior of write functions, b5 = 0 : value 'FF' for the first byte of TLV tag fields invalid, b4 b3 b2 b1 = 0001: data unit size in quartets (power of 2), i.e. one byte
3. CCB	11010yzt = 'Dx'	Card Capabilities Byte of command chaining, length fields and logical channels b8 = 1: support of command chaining for LOAD APPLICATION (other commands are not specified for command chaining), b7 = 1: support of Extended Lc and Le fields, b6 is RFU (b6 = 0 recommended), b5 b4 = 10: logical channel number assignment by the card, b3 b2 b1 = yzt: max. number of logical channels: y, z, t not all set to 1 means 4y+2z+t+1, i.e. from one to seven; y = z = t = 1 means eight or more
Tag/Length	'81' or '82' or '83'	Compact header of Status Indicator - absent if Category Indicator Byte = '00' - present if Category Indicator Byte = '80'
LCS	'00' or '05' or '07'	Life Cycle Status - 1. Status Indicator byte if CI = '00' or - 1. Status Indicator byte if CI = '80' and Status Indicator header = '81' or '83' or - absent if CI = '80' and Status Indicator header = '82' - LCS = '05' recommended
SW1	'6x' or '9x'	1. Status Word - 2. Status Indicator byte if CI = '00' or - 1. Status Indicator byte if CI = '80' and Status Indicator header = '82' or - 2. Status Indicator byte if CI = '80' and Status Indicator header = '83' or - absent if CI = '80' and Status Indicator header = '81' - SW1 = '90' recommended
SW2	'xx'	2. Status Word - 3. Status Indicator byte if CI = '00' or - 2. Status Indicator byte if CI = '80' and Status Indicator header = '82' or - 3. Status Indicator byte if CI = '80' and Status Indicator header = '83' or - absent if CI = '80' and Status Indicator header = '81' - SW2 = '00' recommended
TCK	'xx'	Check Character: 'xx' = exclusive OR of all ATR bytes; Note: according to ISO/IEC 7816-3, TS is not part of the ATR and thus not taken as input of the XOR sum

Table 1 (N2001.00) shows the coding of the ATR for HPCs (T = 1 cards). Since EF.DIR is a structured EF and EF.ATR is a transparent EF and this diversity can not be properly coded in a "Card Service Data Byte" in accordance with ISO/IEC 7816-4, this byte must be absent in the Historical Bytes.

4.2 General Structure

The HPC contains

- the Root Application (MF) with 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 Cryptographic Information Application (CIA) related to the QES application as required by [EN14890-1]
- the ESIGN application for client/server authentication, document decipherment and transcipherment
- the Cryptographic Information Application (CIA) related to the ESIGN application as required by [EN14890-1]

The HPC may further contain the organization-specific authentication application (AUTO) for authentication mechanisms which cannot make use of the ESIGN application.

The general structure is shown in Figure 1 (N2002.00).

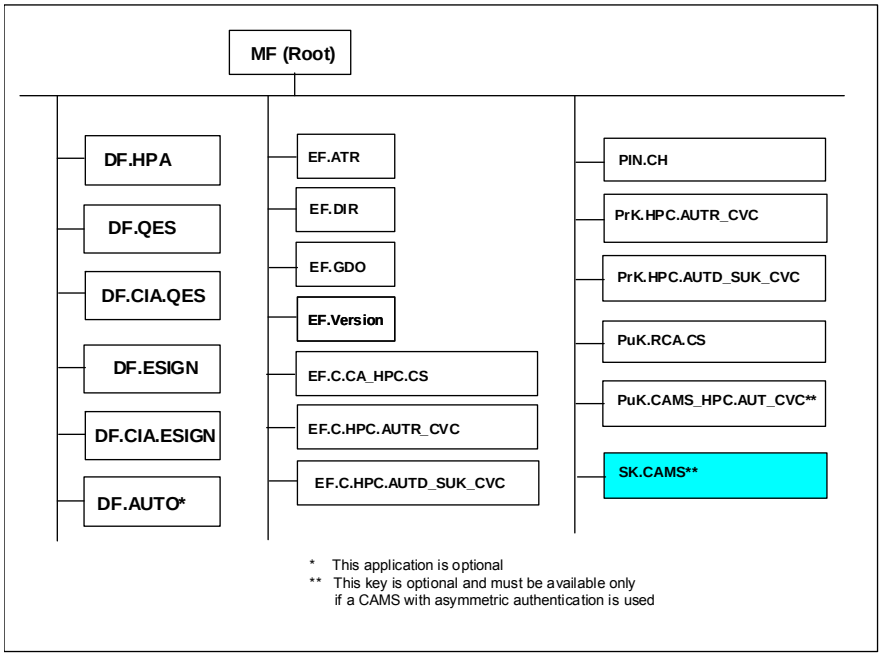

Figure 1 – (N2002.00) General file structure of an HPC

4.3 Root Application and elementary Files at MF-Level

4.3.1 MF

The MF is an Application Dedicated File (see Clause 8.3.1.3 of [HPC-P1]) and has got the characteristics denoted in Table 2 (N2003.00).

Table 2 – (N2003.00) Characteristics of MF

Attribute	Value	Note
Object type	Application Dedicated File	
Application Identifier	‘D276 00014601’	
File Identifier	‘3F00’	Optional
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION (after HPC issuing)	AUT(‘D27600014600' || '01’) AND SmMac AND SmCmdEnc	Only executable if a CAMS is used; see Chapter 13. If a CAMS with symmetric authentication is used, then the security condition must contain the key reference of the corresponding symmetric key, i.e. AUT('13') instead of AUT(‘D27600014600' || '01’).
ACTIVATE, DEACTIVATE, DELETE	NEVER	

4.3.2 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 characteristics of EF.ATR are shown in the following table.

Table 3 – (N2004.00) Characteristics of MF / EF.ATR

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	‘2F01’	Acc. to [ISO7816-4]
Short File Identifier	‘1D’ = 29	
Number of Bytes	COS specific	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 4 (N2005.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

The content of EF.ATR is specified in the following Table 4 (N2005.00), Table 5 (N2006.00), Table 6 (N2007.00), and Table 7 (N2008.00).

Table 4 – (N2005.00) Content of EF.ATR

Tag	Length	Value	Meaning
’E0’	‘xx’	’02 xx xxxx 02 xx xxxx 02 xx xxxx 02 xx xxxx’	I/O buffer sizes; see Table 5 (N2006.00)
‘66’	‘xx’	’46 xx …’ see Table 6 (N2007.00) || '47 03 x6 21 Dx' see Table 7 (N2008.00)	DO Card Data: DO Pre-issuing data || DO Card capabilities

The data object I/O buffer sizes shown in Table 5 (N2006.00) has 4 embedded DOs (Tag '02’ = Integer value, length field 1 Byte with value '02' or '03', value field = max. number of bytes of the respective APDU).

Table 5 – (N2006.00) Data object input/output buffer size

Tag	Length	Value
'E0'	'xx'	'02' -L-'xxxx' || '02' -L-'xxxx' || '02'-L-'xxxx' || '02'-L-'xxxx' = - DO max. length of command APDU without SM || - DO max. length of response APDU without SM || - DO max. length of command APDU with SM || - DO max. length of response APDU with SM

NOTE – In contrast to the Note 1 in Clause 11.5.5 of [HPC-P1] and Note 1 in Clause 11.5.6 of [HPC-P1] it is not possible to specify maximum length values, which depend on specific combinations of CLA, INS, P1 and P2. This may be defined in later versions of this document when appropriate data structures are admitted in [ISO7816-4].

Table 6 – (N2007.00) Value of DO Pre-issuing Data (Tag ‘46’)

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

Table 7 – (N2008.00) Value of DO Card Capabilities (Tag ‘47’)

b8	b7	b6	b5	b4	b3	b2	b1	**Meaning of 1st byte ('x6')**
1	-	-	-	-	-	-	-	DF selection by full DF name
-	x	-	-	-	-	-	-	DF selection by partial DF name (not determined)
-	-	x	-	-	-	-	-	DF selection by path (not determined)
-	-	-	1	-	-	-	-	DF selection by file identifier
-	-	-	-	0	-	-	-	Implicit DF selection (not supported)
-	-	-	-	-	1	-	-	Short EF identifier supported
-	-	-	-	-	-	1	-	Record number supported
-	-	-	-	-	-	-	0	Record identifier (not supported)
b8	b7	b6	b5	b4	b3	b2	b1	**Meaning of 2nd byte ('21')**
0	-	-	-	-	-	-	-	EFs of TLV structure (not supported)
-	0	1	-	-	-	-	-	Behavior of write functions (proprietary)
-	-	-	0	-	-	-	-	Value 'FF' for the first byte of BER-TLV tag fields invalid
-	-	-	-	0	0	0	1	Data unit size in quartets (power of 2, i.e. ’01’ = 2 quartets = one byte)
b8	b7	b6	b5	b4	b3	b2	b1	**Meaning of 3rd byte ('Dx')**
1	-	-	-	-	-	-	-	Command chaining supported; see Note 1
-	1	-	-	-	-	-	-	Extended Lc and Le fields
-	-	0	-	-	-	-	-	b6 is RFU (b6 = 0 recommended)
-	-	-	1	0	-	-	-	Logical channel number assignment by the card
					y	z	t	
-	-	-	-	-	x	x	x	Maximum number of logical channels; see Note 2

NOTE 1 – Command Chaining may be required for the LOAD APPLICATION command; see Clause 13.3.

NOTE 2 – The card capability bytes are set according to Clause 8.1.1.2.7 of [ISO7816-4]. In the 3rd capabilities byte the bits b3-b1 encode the maximum number of logical channels supported by the card: y, z, t not all set to 1 means 4y+2z+t+1, i.e. from one to seven, y = z = t = 1 means eight or more. The maximum number of supported logical channels shall be set to one of the following values: b3b2b1 = 011 (4 channels), 100 (5 channels), 101 (6 channels), 110 (7 channels) or 111 (≥ 8 channels).

4.3.3 EF.DIR

EF.DIR contains the application templates for MF, DF.HPA, DF.QES, DF.CIA.QES, DF.ESIGN, DF.CIA.ESIGN, and DF.AUTO according to ISO/IEC 7816-4. EF.DIR allows the addition of AIDs of further (downloaded) applications; see the following table for characteristics of EF.DIR.

Table 8 – (N2009.00) Characteristics of MF / EF.DIR

Attribute	**Value**	**Note**
Object type	Linear Variable Record Elementary File	
File Identifier	‘2F00’	
Short File Identifier	‘1E’ = 30	
Number of Bytes	190	10 * max. record length
Maximum Number of Records	10 (3 for future use)	
Maximum Record Length	19 bytes	
Flag Record LCS	False	
Flag Transaction Mode	True	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 9 (N2010.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ RECORD, SEARCH RECORD	ALWAYS	
APPEND RECORD, UPDATE RECORD	AUT((‘D27600014600' || '01’) AND SmMac	Only executable if a CAMS is used; see Chapter 13. If a CAMS with symmetric authentication is used, then the security condition must contain the key reference of the corresponding symmetric key, i.e. AUT('13') instead of AUT(‘D27600014600' || '01’).
ACTIVATE, ACTIVATE RECORD, DEACTIVATE, DEACTIVATE RECORD, DELETE, ERASE RECORD	NEVER	

The application templates contained in EF.DIR are shown in Table 9 (N2010.00).

Table 9 – (N2010.00) Application Templates in EF.DIR

Tag	**Length**	**Application Template**	**Meaning**
’61’	‘08’	‘4F 06 D27600014601’	Application Template with AID.MF
’61’	‘08’	‘4F 06 D27600014602’	Application Template with AID.HPA
‘61’	‘08’	‘4F 06 D27600006601’	Application Template with AID.QES (DINSIG)
‘61’	‘0D’	‘4F 0B E828BD080F D27600006601’	Application Template with AID.CIA.QES
‘61’	’0C’	‘4F 0A A000000167 455349474E’	Application Template with AID.ESIGN
‘61’	‘11’	‘4F 0F E828BD080F A000000167 455349474E’	Application Template with AID.CIA.ESIGN
‘61’	‘08’	‘4F 06 D27600014603’	Application Template with AID.AUTO (optional)

4.3.4 EF.GDO

Table 10 (N2011.00) shows the attributes of EF.GDO.

Table 10 – (N2011.00) Characteristics of MF / EF.GDO

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'2F02'	
Short File Identifier	'02' = 2	
Number of Bytes	12	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 11 (N2012.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

EF.GDO contains in compliance with [Resolution190] the DO ICC Serial Number; see Table 11 (N2012.00).

Table 11 – (N2012.00) DO ICCSN in EF.GDO

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

The structure of the DO ICCSN (Tag '5A') for health cards is shown in Figure 2 (N2013.00).

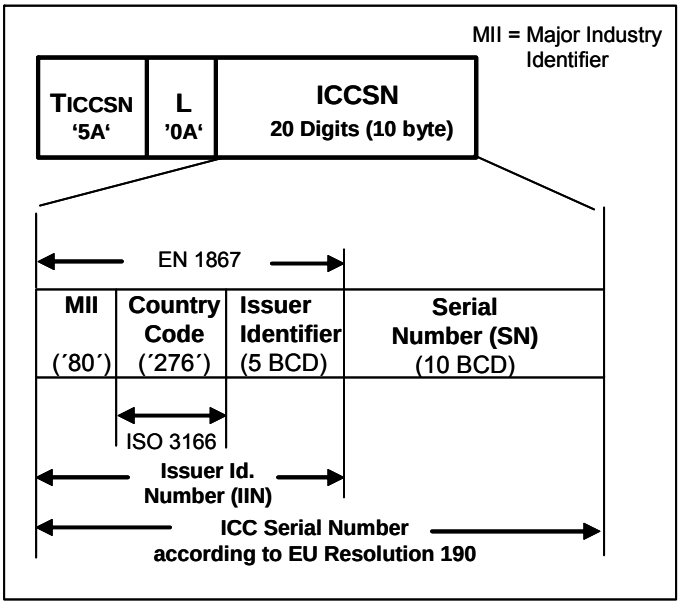

Figure 2 - (N2013.00) ICC Serial No. for health cards

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

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). The two leading BCDs of the serial number (following the issuer identifier) indicate the engaged CA; see Annex A.

4.3.5 EF.Version

The EF.Version with linear fixed record structure contains the version numbers of the specification parts and an SRQ number – indicating the upper limit of relevant SRQs – which the card is compliant to. The characteristics of the file EF.Version are shown in the subsequent table.

Table 12 – (N2014.00) Characteristics of MF / EF.Version

Attribute	**Value**	**Note**
Object type	Linear Fix Record Elementary File	
File Identifier	'2F10'	
Short File Identifier	'10' = 16	
Number of Bytes	20	
Maximum Number of Records	4	
Maximum Record Length	5 bytes	
Flag Record LCS	False	
Flag Transaction Mode	True	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 13 (N2015.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ RECORD, SEARCH RECORD	ALWAYS	
UPDATE RECORD	AUT(('D27600014600' || '01') AND SmMac	Only executable if a CAMS is used; see Chapter 13. If a CAMS with symmetric authentication is used, then the security condition contain the key reference of the corresponding symmetric key, i.e. AUT('13') instead of AUT('D27600014600' || '01').
ACTIVATE, ACTIVATE RECORD, APPEND RECORD, DEACTIVATE, DEACTIVATE RECORD, DELETE, ERASE RECORD	NEVER	

The file EF.Version contains 4 records of fixed length; see Table 13 (N2015.00). The first 3 records indicate the 3 parts of the HPC specification and an SRQ number related to the card. The triplet version number XX.YY.ZZ of the respective specification part together with the four-digit SRQ number SSSS is encoded as XXYYZZSSSS in BCDs. The last record is reserved for future use (RFU).

Table 13 – (N2015.00) Content of EF.Version

Rec No.	Value (5 Bytes)	Meaning
1	'0203020000'	Version of supported HPC Specification Part 1 followed by the upper SRQ number limit, which the card is compliant to. In this case version 2.3.2, no additional SRQ
2	'0203020000'	Version of supported HPC Specification Part 2 followed by the upper SRQ number limit, which the card is compliant to. In this case version 2.3.2, no additional SRQ
3	'0203020000'	Version of supported HPC Specification Part 3 followed by the upper SRQ number limit, which the card is compliant to. In this case version 2.3.2, no additional SRQ
4	'0000000000'	RFU

4.3.6 EF.C.CA_HPC.CS

EF.C.CA_HPC.CS contains the card verifiable certificate of the Certificate Service Provider, issued by the Root CA for Health Care for a CA_HPC. The file characteristics are outlined in Table 14 (N2016.00).

Table 14 – (N2016.00) Characteristics of MF / EF.C.CA_HPC.CS

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'2F04'	
Short File Identifier	'04' = 4	
Number of Bytes	331	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 17 (N2019.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

Table 15 **(N2017.00)** shows the coding of the CV certificates for a CA and for a card (eGK, HPC or SMC). The contained OIDs indicate which signature or authentication scheme is used by the certified public key pair; see Table 16 **(N2018.00)**.

Table 15 – (N2017.00) CPI values and CV field values

CPI (1 B)	**PK** (250 B)	**OID** (7 or 6 B)	**CHA** (0 or 7B)	**CHR** (8 or 12 B)	**CAR** (8 B)	Remark
'21'	Modulus (256 bytes) || Exponent (4 bytes)	'2B240304020204' (7 bytes)	-	CA Name (5 bytes) || Extension (3 bytes)	CA Name (5 bytes) || Extension (3 bytes)	CVC for a CA, issued by the RCA (signed key is used for verifying of further keys in a CVC chain; signed is the hash value of the CVC content), e.g. C.CA_HPC.CS
'22'	Modulus (256 bytes) || Exponent (4 bytes)	'2B2403050204' (6 bytes)	Prefix (6 bytes) || Role identifier (1 bytes)	'00' (1 bytes) || KeyRef (1 bytes) || ICCSN (10 bytes)	CA Name (5 bytes) || Extension (3 bytes)	CVC for an eGK, HPC or SMC, issued by a CA (signed key is used for Card-to-card authentication procedures), e.g. C.HPC.AUTR_CVC, C.HPC.AUTD_SUK_CVC

Table 16 – (N2018.00) Object Identifier

OID Name	**OID Number**	**OID Coding**	**Registration Authority**
sigS_ISO9796-2Withrsa_sha256 Signature scheme with RSA signature and DSI according to ISO/IEC 9796-2 and SHA-256	{1 3 36 3 4 2 2 4}	'2B240304020204'	TeleTrusT
authS_ISO9796-2Withrsa_sha256_mutual Authentication scheme with RSA signature and DSI according to ISO/IEC 9796-2 and SHA-256 for mutual authentication with or without establishment of a Trusted Channel	{1 3 36 3 5 2 4}	'2B2403050204'	TeleTrusT

NOTE – In [ISO9796-2] Annex A.4 is relevant.

The structure and content of the CVC in EF.C.CA_HPC.CS with CPI = '21' are defined in Clause 7.1 of [HPC-P1] and outlined in Table 17 (N2019.00). The CV certificate file contains a constructed certificate data object with tag '7F21' (RSA certificate with message recovery). The total length of the file content is 331 bytes.

Table 17 – **(N2019.00)** Structure and content of an EF containing a CV certificate with CPI = '21'

Tag	**Length**	**Value**		
'7F21'	'820146'	CV certificate (326 bytes)		
		Tag	**Length**	**Value**
		'5F37'	'820100'	SIG.CA (256 bytes)
				Digital Signature Input for SIG.CA ('6A' …'BC'):
				'6A' = Padding according to [ISO9796-2] (1 byte)
				'21' = CPI (1 byte)
				PK first part of modulus (221 bytes)
				SHA-256 Hash value (32 bytes) Input: CPI || PK || OID || CHR || CAR; see Table 15 (N2017.00)
				'BC' = Trailer according to [ISO9796-2] (1 byte)
		'5F38'	'3E'	Non-recoverable part of the signature SIG.CA (62 bytes):
				PK rest of modulus followed by exponent (39 bytes)
				OID (7 bytes)
				CHR (8 bytes)
				CAR (8 bytes)

4.3.7 EF.C.HPC.AUTR_CVC

EF.C.HPC.AUTR_CVC contains the card verifiable certificate of the HPC for card-to-card role authentication between HPC and eGK and for SMC-A, SMC-B authorization. The characteristics of the file are shown in the following table.

Table 18 – (N2020.00) Characteristics of MF / EF.C.HPC.AUTR_CVC

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'2F03'	
Short File Identifier	'03' = 3	
Number of Bytes	341	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 19 (N2021.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

The structure and content of the CVC in EF.C.HPC.AUTR_CVC with CPI = '22' are defined in Clause 7.1 of [HPC-P1] and outlined in Table 19 (N2021.00). The CV certificate file contains a constructed certificate data object with tag '7F21' (RSA certificate with message recovery). The total length of the file content is 341 bytes. The Certificate Holder Authorizations relevant for a C.HPC.AUTR_CVC are shown in Table 228 (N2623.00).

Table 19 – (N2021.00) Structure and content of an EF containing a CV certificate with CPI = '22'

Tag	Length	Value		
'7F21'	'820150'	CV certificate (336 bytes)		
		Tag	**Length**	**Value**
		'5F37'	'820100'	SIG.CA (256 bytes)
				Digital Signature Input for SIG.CA ('6A' …'BC'):
				'6A' = Padding according to [ISO9796-2] (1 byte)
				'22' = CPI (1 byte)
				PK first part of modulus (221 bytes)
				SHA-256 Hash value (32 bytes) Input: CPI || PK || OID || CHA || CHR || CAR; see Table 15 (N2017.00)
				'BC' = Trailer according to [ISO9796-2] (1 byte)
		'5F38'	'48'	Non-recoverable part of the signature SIG.CA (72 bytes):
				PK rest of modulus followed by exponent (39 bytes)
				OID (6 bytes)
				CHA (7 bytes)
				CHR (12 bytes)
				CAR (8 bytes)

4.3.8 EF.C.HPC.AUTD_SUK_CVC

EF.C.HPC.AUTD_SUK_CVC contains the card verifiable certificate of the HPC for card-to-card device authentication between HPC/SMC-A, HPC/SMC-B and HPC/SMC-K with HPC as signature card capable of stack and comfort signatures ("Stapel- und Komfortsignatur" SUK) to receive PIN data and data to be signed (DTBS).

Table 20 – (N2022.00) Characteristics of MF / EF.C.HPC.AUTD_SUK_CVC

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'2F05'	
Short File Identifier	'05' = 5	
Number of Bytes	341	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	see Table 19 (N2021.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

The structure and content of the CVC in EF.C.HPC.AUTD_SUK_CVC with CPI = '22' are defined in Clause 7.1 of [HPC-P1] and outlined in Table 19 (N2021.00). The Certificate Holder Authorization relevant for a C.HPC.AUTD_SUK_CVC is shown in Table 229 (N2624.00).

4.3.9 PIN.CH

PIN.CH is the global PIN of the cardholder. The PIN consists of 5 to 8 digits and is changeable. The retry counter shall have the initial value 3.

The usage of the 8 digit resetting code (personal unblocking key, PUK) is limited by a usage counter with an initial value of 10. The usage counter is decremented irrespective of whether the resetting code was correct or not. The successful presentation resets the retry counter of the related PIN.CH. The security status of PIN.CH can be used an unlimited number of times, i.e. the default number of SSEC is set to infinite. The PIN characteristics are shown in the following table.

Table 21 – (N2023.00) Characteristics of MF / PIN.CH

Attribute	Value	Note
Object type	Password	
Password Identifier	'01'	
Password Reference	'01'	
Secret	....	To be personalized
Minimum Length	5	
Start Retry Counter	3	
Retry Counter	3	
Transport Status	Transport PIN Random or Transport PIN Derived or Transport PIN Fixed Value or Regular Password	
Flag Enabled	True	
Start Security Status Evaluation Counter	Infinite	
PUK	...	To be personalized
PUK Usage	10	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
CHANGE RD (Option '00')	ALWAYS	
GET PIN STATUS	ALWAYS	
RESET RC (Option '00' and '01')	ALWAYS	
VERIFY	ALWAYS	
Other	NEVER	

According to Clause 14.6.1.4 and Clause 14.6.5.6 of [HPC-P1] the COS checks only the minimum length (5 digits) of PIN.CH, i.e. the COS does not control whether the maximum length of 8 digits is exceeded.

The transport PIN method Regular_Password may especially be relevant for follow-up cards. With regard to the other methods the command CHANGE REFERENCE DATA (see Clause 4.6.2) is used to establish the regular PIN.

4.3.10 PrK.HPC.AUTR_CVC

PrK.HPC.AUTR_CVC is the global private key for C2C-authentications between HPC/eGK and HPC/CAMS, and for authorization of SMC-A and SMC-B.

The further key characteristics are shown in the subsequent table.

Table 22 – (N2024.00) Characteristics of MF / PrK.HPC.AUTR_CVC

Attribute	**Value**	**Note**
Object type	Private RSA Object	
Key Identifier	'10'	
Key Reference	'10'	
Private Key	.... (2048 bit)	To be personalized
Algorithm Identifier	rsaRoleAuthentication rsaSessionkey4SM	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
INTERNAL AUTHENTICATE	PWD(PIN.CH)	
EXTERNAL AUTHENTICATE	ALWAYS	
Other	NEVER	

The public key associated with PrK.HPC.AUTR_CVC (with CVC holder profile 2,3,...) is contained in C.HPC.AUTR_CVC.

4.3.11 PrK.HPC.AUTD_SUK_CVC

PrK.HPC.AUTD_SUK_CVC is the global private key for C2C-authentications between HPC/SMC-A and HPC/SMC-B for PIN transfer and HPC/SMC-K for DTBS transfer to the HPC. The further key characteristics are shown in the subsequent table.

Table 23 – (N2025.00) Characteristics of MF / PrK.HPC.AUTD_SUK_CVC

Attribute	**Value**	**Note**
Object type	Private RSA Object	
Key Identifier	'11'	
Key Reference	'11'	
Private Key	.... (2048 bit)	To be personalized
Algorithm Identifier	rsaRoleAuthentication rsaSessionkey4SM rsaSessionkey4Intro	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
INTERNAL AUTHENTICATE	ALWAYS	
EXTERNAL AUTHENTICATE	ALWAYS	
Other	NEVER	

The public key associated with PrK.HPC.AUTD_SUK_CVC (with CVC holder profile 53) is contained in C.HPC.AUTD_ SUK_CVC.

4.3.12 PuK.RCA.CS

PuK.RCA.CS is the public key of the Root CA for Health Care for verification of CVCs issued by this RCA. The subsequent table shows the characteristics of the public key.

Table 24 – (N2026.00) Characteristics of MF / PuK.RCA.CS

Attribute	Value	Note
Object type	Public RSA Signature Verification Object	
Key Identifier	CAR of C.CA_HPC.CS: CA name (5 bytes) || extension (3 bytes)	To be personalized
Key Reference	-	
Public Key	.... (2048 bit)	To be personalized
OID	´2B240304020204´ = {1 3 36 3 4 2 2 4}	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
PSO: VERIFY CERTIFICATE	ALWAYS	
Other	NEVER	

The key reference of PuK.RCA.CS can be retrieved from the HPC as described in Figure 3 (N2049.00).

4.3.13 PuK.CAMS_HPC.AUT_CVC

PuK.CAMS_HPC.AUT_CVC (optional) is the public key for performing an HPC/CAMS authentication procedure with TC establishment. The subsequent table shows the characteristics of the public key.

Table 25 – (N2027.00) Characteristics of MF / PuK.CAMS_HPC.AUT_CVC

Attribute	Value	Note
Object type	Public RSA Authentication Object	
Key Identifier	'000000000000000000000013' (12 bytes)	To be personalized
CHA	'D2760001460001'	
Public Key	.... (2048 bit)	To be personalized
OID	'2B2403050204' = {1 3 36 3 5 2 4}	
Algorithm Identifier	rsaSessionkey4SM	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
INTERNAL AUTHENTICATE	ALWAYS	
EXTERNAL AUTHENTICATE	ALWAYS	
Other	NEVER	

PuK.CAMS_HPC.AUT_CVC must be available in the card, if and only if a CAMS with asymmetric authentication is used. PuK.CAMS_HPC.AUT_CVC is a global key with a uniform key identifier and a CAMS-specific key value, that may additionally depend, e.g. on the year of HPC issuance. The related CAMS is probably bound to identify each individual card, in order to use the proper private key. The key identifier is used as key reference in the authentication procedure between HPC and CAMS (see Clause 13.2.1), whereas the CHA is used in access rules, see Table 2 (N2003.00), Table 8 (N2009.00), Table 12 (N2014.00) and Table 101 (N2105.00).

4.3.14 SK.CAMS

SK.CAMS (optional) is the secret key for performing an HPC / CAMS authentication procedure with TC establishment. The subsequent table shows the characteristics of the key.

Table 26 – (N2028.00) Characteristics of MF / SK.CAMS

Attribute	**Value**	**Note**
Object type	3TDES Authentication Object	
Key Identifier	'13' = 19	
encKey	...	To be personalized
macKey	...	To be personalized
Algorithm Identifier	desSessionkey4SM	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
MUTUAL AUTHENTICATE	PWD(PIN.CH)	
Other	NEVER	

4.4 Security Environments at MF Level

At MF level only SE # 1 (default SE) is used. It is possible in SE # 1 to establish a trusted channel, e.g. for remote PIN entry.

4.5 HPC Opening

4.5.1 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, EF.GDO, and EF.Version and possibly also the Cryptographic Information Objects contained in the EFs of DF.CIA.QES and DF.CIA.ESIGN. The card type can be identified by the set of available applications indicated in EF.DIR. A more efficient way to identify a presented card is the MF selection with the SELECT command with empty data field and FCP in the response data field disclosing the AID of the Root application; see Clause 4.5.2.

Which generation a presented card belongs to, can be recognized by the specification version indicated in EF.Version.

Subsequently, the selection of root application and the reading of EF.ATR, EF.GDO, EF.DIR and EF.Version are described.

4.5.2 Selecting the Root Application

After reset the root application is selected by default. Afterwards, the root application can be selected, e.g. by using the SELECT command with application identifier as shown in Table 27 (N2029.00).

Table 27 - (N2029.00) SELECT command for MF

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 AID in the data field
Data field	‘D27600014601’ = AID of Root Application (MF)
Le	Absent

Table 28 - (N2030.00) SELECT response

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

NOTE 1 – The optional FID ‘3F00’ is not used for MF selection since only the current directory is searched for the specified file identifier; see Clause 14.2.6.10 in [HPC-P1].

NOTE 2 – The AID of the Root application can be disclosed via SELECT command with P2 = ‘04’ and empty data field. The response will include FCP with the corresponding AID; see Clause 8.3.3 of [HPC-P1]. This opens up an efficient way to identify the card type.

4.5.3 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 29 - (N2031.00) 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 30 - (N2032.00) READ BINARY response

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

4.5.4 Reading EF.DIR and EF.Version

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

Table 31 - (N2033.00) 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 ‘84’ = b8-b4: 10000 SFID of EF.Version: 16 b3-b1: 100 Read record P1
Lc	Absent
Data field	Absent
Le	‘00’ = Complete record

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

Table 32 – (N2034.00) READ RECORD Response

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

4.5.5 Reading HPC related CV Certificates

For reading the certificates C.CA_HPC.CS, C.HPC.AUTR_CVC and C.HPC.AUTD_SUK_CVC 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. Which roles/profiles are supported by a newly-presented card, the external software can find out without recovering CVC signature inputs as the Certificate Holder Authorization (CHA) is part of the non-recoverable plaintext; see Table 19 (N2021.00).

Table 33 - (N2035.00) READ BINARY command for reading a CV certificate

CLA	As defined in ISO/IEC 7816-4
INS	‘B0’ = READ BINARY
P1, P2	- P1 = b8-b6:100 b5-b1: 00100 SFID of EF.C.CA_HPC.CS: 4 b5-b1: 00011 SFID of EF.C.HPC.AUTR_CVC: 3 b5-b1: 00101 SFID of EF.C.HPC.AUTD_SUK_CVC: 5 P2 = Offset - ‘xxxx’ = Offset (bit b8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	‘000000’ = Read until end-of-file

Table 34 - (N2036.00) READ BINARY response

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

If the supported extended length is not sufficient to read the whole certificate with a single command, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

4.6 PIN Management

4.6.1 PIN Verification

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

Table 35 - (N2037.00) 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 Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 36 - (N2038.00) 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 bytes as specified in Clause 14.6.6.3 of [HPC-P1].

4.6.2 PIN Change

For PIN change the ISO/IEC 7816-4 command CHANGE REFERENCE DATA may be used at any time convenient for the HP. If PIN.CH is referenced, the command can only be used with P1 = ‘00’, i.e. with both old and new PIN in the data field. The successful verification of the old PIN does not set the security status of the referenced PIN; see Clause 14.6.1.4 of [HPC-P1].

The command is especially used to lift a PIN transport status and establish a regular PIN; see Clause 8.2.5 and Clause 14.6.1 of [HPC-P1].

Table 37 - (N2039.00) CHANGE REFERENCE DATA 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 Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 38 - (N2040.00) CHANGE REFERENCE DATA response

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

4.6.3 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 successful verification of the resetting code does not set the security status of the referenced PIN object; see Clause 14.6.5.6 of [HPC-P1].

Table 39 – (N2041.00) RESET RETRY COUNTER 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 bytes) followed by a new PIN (8 bytes) - If P1 = '01': Resetting code (8 bytes) (RC and PIN format: see Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 40 - (N2042.00) RESET RETRY COUNTER response

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

4.6.4 Query of the PIN.CH status

The software can query the security status as well as the transport status related to a PIN by using a GET PIN STATUS command as shown in the following table.

Table 41 – (N2043.00) GET PIN STATUS command

CLA	As defined in ISO/IEC 7816-4 for proprietary class
INS	'20' = GET PIN STATUS
P1	'00'
P2	'01' = PIN.CH reference
Lc	Absent
Data field	Absent
Le	Absent

Table 42 – (N2044.00) GET PIN STATUS response

Data field	Absent
SW1-SW2	'9000' Verification not required or specific status bytes; see [HPC-P1]

5 Channel Management

5.1 General Aspects

The support of four logical channels by an HPC is mandatory; see Clause 11.4 of [HPC-P1]. The maximum number of logical channels is indicated in EF.ATR; see Table 4 (N2005.00). 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.

5.2 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 43 – (N2045.00) 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 44 - (N2046.00) 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]

5.3 Closing a Logical Channel

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

Table 45 – (N2047.00) 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 46 – (N2048.00) MANAGE CHANNEL response

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

6 Interactions between HPC and eGK

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 (see Annex A) has been successfully presented”.
In the authentication procedure, no SM keys are established, since in the subsequent communication with the eGK (reading/writing data) an HPC is not involved.

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

6.2 CVC Key Management Authorities

The general model for CV Certificates and CVC Key Management Authorities is shown in Figure 3 (N2049.00). The figure denotes especially, where the key reference are found to be set for CVC verification and the subsequent authentication procedure.

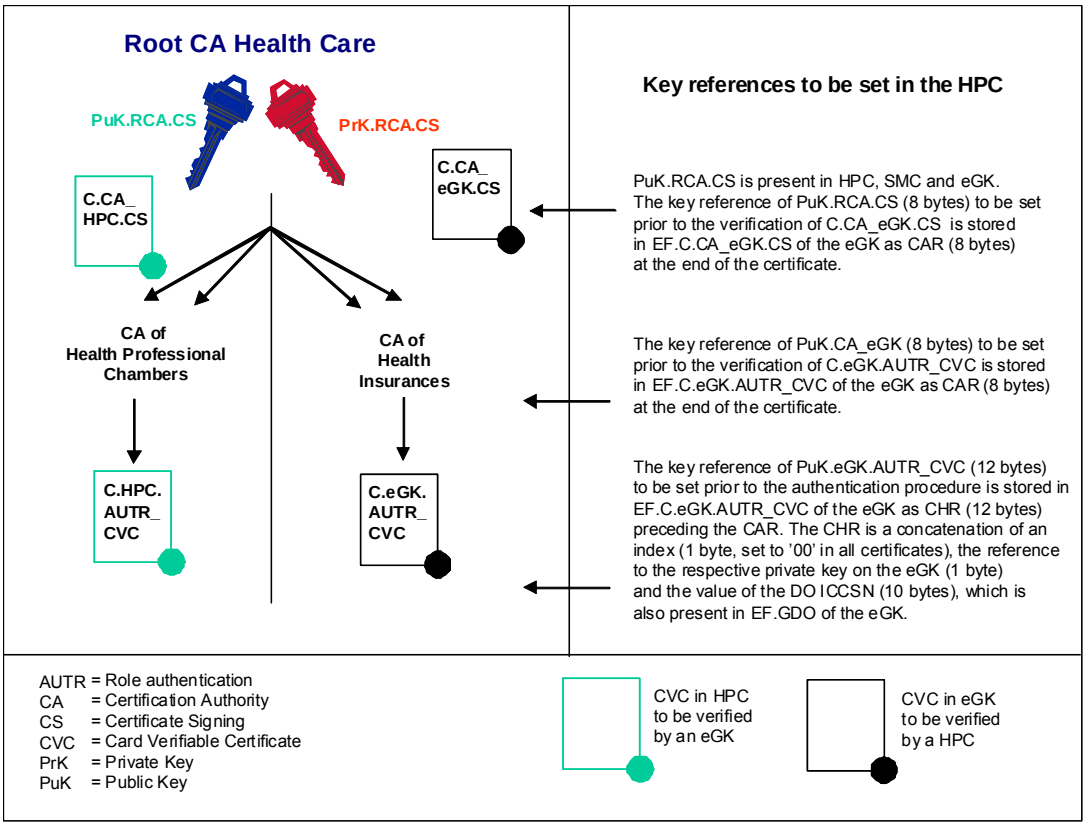

Figure 3 – (N2049.00) CVC Key Management Authorities, CV Certificates and Key Selection in an HPC

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 e.g. stored in and retrievable from a connector.

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).

6.3 Verification of eGK related CV Certificates

The verification may follow a bottom-up approach since the public key required for C2C authentication may already be available in the HPC; see Clause 9.1 of [HPC-P1] for persistent storage of public keys. For this purpose, the external software will try to set the public key of the eGK for starting directly the C2C authentication procedure; see Clause 6.4. If this public key is not available, then the public key of CA_eGK will be set, in order to verify the eGK certificate and import the public key of eGK. If this again is not possible, the public key of the Root CA will be set in order to verify the CA certificate and import the CA public key. Alternatively, the bottom-up procedure may start with the verification of the eGK certificate since there is a better chance of a stored CA public key than a stored eGK public key.

In the the following the conventional top-down approach is described starting with the verification of the CA certificate. For this purpose, the public key of the common Root CA is selected with the MSE command. The key reference can be found by the software as shown in Figure 3 (N2049.00).

Table 47 - (N2050.00) 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’ = Digital Signature Template
Lc	‘0A’= Length of subsequent data field
Data field	‘83 08’ || ‘xx..xx’ = DO KeyRef PuK.RCA.CS (for retrieval of the Key Reference see Figure 3 (N2049.00))
Le	Absent

Table 48 - (N2051.00) MSE response

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

After setting the Root PuK for verification of C.CA_eGK.CS, 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 non-recoverable part of the CVC (containing the final part of PK modulus, OID, CHR, CAR); see Clause 7.1.2 of [HPC-P1] and Table 17 (N2019.00).

Table 49 - (N2052.00) 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	‘000146’ = Length of subsequent data field = 326
Data field	‘5F37’ || ‘820100’ || SIG.RCA (256 bytes) || ‘5F38’ || ‘3E’ || Non-recoverable part of the certificate signature (62 bytes)
Le	Absent

Table 50 - (N2053.00) 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 C.eGK.AUT_CVC is now available and has to be selected.

Table 51 - (N2054.00) MSE command for selecting the PuK of CA_eGK

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for verification
P2	‘B6’ = Digital Signature Template
Lc	‘0A’= Length of subsequent data field
Data field	‘83 08’ || ‘xx..xx’ = DO KeyRef of PuK.CA_eGK.CS (for retrieval of the Key Ref see Figure 3 (N2049.00))
Le	Absent

Table 52 - (N2055.00) MSE response

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

Now the C.eGK.AUT_CVC can be verified.

Table 53 - (N2056.00) PSO: VERIFY CERTIFICATE command for verifying the C.eGK.AUT_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	‘000150’ = Length of subsequent data field = 336
Data field	‘5F37’ || ‘820100’ || SIG.RCA (256 bytes) || ‘5F38’ || ‘48’ || Non-recoverable part of the certificate signature (72 bytes)
Le	Absent

Table 54 - (N2057.00) PSO: VERIFY CERTIFICATE response

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

6.4 Asymmetric HPC / eGK Authentication without TC Establishment

In the command sequence described below, the eGK proves its authenticity to the HPC first. Then the HPC proves its access rights to the eGK. The procedure may also be performed in reversed order, since the COS does not control whether the internal or external authentication comes first. The external software controls the command sequence.

Before the authentication procedure is performed, PIN.CH must have been successfully presented. In a first step, the key references for PuK.eGK.AUT_CVC and PrK.HPC.AUTR_CVC and the corresponding algorithm identifiers are set.

Table 55 - (N2058.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for external authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘11’ = Length of subsequent data field
Data field	‘83 0C xx ... xx’ || ‘80 01 00’ = DO KeyRef of PuK.eGK.AUT_CVC (for retrieval of the Key Ref, see Figure 3 (N2049.00) and Note) || DO AlgID rsaRoleCheck
Le	Absent

NOTE – The CHR of the CV certificate of the eGK is used as key reference. It has a length of 12 bytes: index set to '00' (1 byte) || KeyRef of PrK.eGK.AUT_CVC (1 byte) || ICCSN.eGK (10 bytes); see Clause 8.5.2 of [HPC-P1] for CHR structure.

Table 56 - (N2059.00) MSE response

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

Table 57 - (N2060.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for internal authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘0A’ = Length of subsequent data field
Data field	’84 01 10’ || ‘80 01 00' = DO KeyRef of PrK.HPC.AUTR_CVC || DO AlgID rsaRoleAuthentication
Le	Absent

Table 58 - (N2061.00) MSE response

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

NOTE – Both MSE SET commands for internal and external authentication are performed successively, so that the sequence of authentication commands will not be interrupted by a MSE SET command; see Chapter 15 of [HPC-P1].

After that a challenge is retrieved from the HPC.

Table 59 - (N2062.00) 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' = Length of expected random number = 8

Table 60 - (N2063.00) GET CHALLENGE response

Data field	RND.HPC (8 bytes)
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 Clause 14.7.4 of [HPC-P1]) has to be sent to the eGK. The HPC has to verify the eGK signature.

Table 61 – (N2064.00) EXTERNAL AUTHENTICATE command for eGK authentication

CLA	As defined in ISO/IEC 7816-4
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'000100' = Length of subsequent data field = 256
Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
Le	Absent

Table 62 – (N2065.00) EXTERNAL 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 command.

Table 63 – (N2066.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	‘88’ = INTERNAL AUTHENTICATE
P1	‘00’ = Algorithm already known to the card
P2	‘00’ = Key reference already known to the card
Lc	‘000010’ = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	‘0100’ = Length of expected signature

Table 64 – (N2067.00) INTERNAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
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 ID 'xx' successfully presented”.

7 Interactions between HPC and SMC

7.1 General

For HPC/SMC interactions each card has to prove its authenticity to the other card. Therefore, a CV based authentication procedure has to be performed, so that in both cards the related security status can be set, i.e. “certificate holder authorization y (see Annex A) has been successfully presented”. In general, the HPC is able to perform the following C2C authentication procedures:

- Asymmetric authentication without agreement on session keys
- Asymmetric authentication with establishment of a trusted channel
- Asymmetric authentication with persistent storage of introduction keys
- Symmetric authentication using introduction keys with establishment of a trusted channel.

However, the asymmetric authentication keys each support only a subset of the three above-mentioned asymmetric algorithms; see Table 22 (N2024.00) and Table 23 (N2025.00). Which of the procedures is appropriate, depends on the intended HPC/SMC interaction (e.g. remote data transfer) and the affected access rules within the card (e.g. command data to be protected by secure messaging); see Table 65 (N2068.00). The appropriate procedure is set with its corresponding algorithm identifier in MSE SET commands.

Table 65 – (N2068.00) Possible authentication procedures involving HPC and/or SMC (informative)

Source card	Access condition of authentication key on source card	Target card	Access condition of authentication key on target card	Note
SMC-A (profile 54: PIN sender)	Authentication of target card	HPC (profile 53: SSCD)	Always	Use Case of PIN transfer
		SMC-B (profile 55: PIN receiver)	Always	Use Case of PIN transfer
		RFID Token (profile 55: PIN receiver)	Always	Use Case of PIN transfer
SMC-B (profile 54: PIN sender)	Authentication of target card	HPC (profile 53: SSCD)	Always	No use case
		SMC-B (profile 55: PIN receiver)	Always	No use case
		RFID Token (profile 55: PIN receiver)	Always	Use Case of PIN transfer
SMC-K (profile 51: SAK)	Authentication of target card	HPC (profile 53: SSCD)	Always	Use Case of DTBS transfer
RFID Token (profile 52: comfort feature)	Always	SMC-K (profile 51: SAK)	Authentication of source card	Use Case of comfort signature trigger

For HPC/SMC authentication procedures each side uses a private key for device authentication. On part of the HPC the certificate C.HPC.AUTD_SUK_CVC (profile 53 of stack and comfort signature card) and the corresponding private key PrK.HPC.AUTD_SUK_CVC are used towards an SMC-A, SMC-B or SMC-K; see Table 229 (N2624.00) for definition of profiles.

The authentication procedures are performed as described in Clause 15.4 of [HPC-P1]. Which side has to authenticate first, depends on the access rules of the involved authentication keys. In all

authentication procedures denoted in Table 65 (N2068.00) the HPC has to authenticate first, e.g. the HPC has to authenticate to the SMC-A first, in order that a trusted channel for remote PIN entry can be established.

The asymmetric authentication procedure with storage of introduction keys is supported by HPC, SMC-A, SMC-B and SMC-K. It provides the keys for symmetric authentication procedures that usually are more efficient than the asymmetric algorithms.

The introduction keys assume the attributes of the public and private key used in the asymmetric authentication procedure, i.e. the verified role ID, the key reference of the public key (with adapted index), the applied access rule and algorithm identifier; see Clause 8.5.2 of [HPC-P1]. Thus, an asymmetric authentication procedure with TC establishment may be replaced by the symmetric procedure using stored introduction keys.

7.2 Preparatory steps

In Clause 4.5.5 is specified, how the HPC CV certificates – located at MF level – can be retrieved. It is assumed, that reading is performed only once by the software environment, which stores the CVCs e.g. associated with the respective ICCSN.

The CV certificates related to the involved SMC are to be verified, so that the corresponding public keys are available in the HPC, before establishing the trusted channel. The following CVCs have to be verified:

- C.CA_SMC.CS
- C.SMC.AUTD_RPS_CVC (SMC-A, SMC-B) or C.SAK.AUTD_CVC (SMC-K)

The procedure is equivalent to the procedure described in Clause 6.3.

If the external software has to manage the security statuses of many authentication keys without having implemented a finite state machine (that could map the internal statuses of different cards), the software may query from the card the security status related to an external presentation of a CHA (with role identifier not equal ‘00’) using a GET SECURITY STATUS KEY command as shown in the following table.

Table 66 – (N2069.00) GET SECURITY STATUS KEY command for querying a certain security status

CLA	As defined in ISO/IEC 7816-4 for proprietary class
INS	‘82’ = GET SECURITY STATUS KEY
P1	‘80’ = Query of the security status
P2	‘00’
Lc	‘0A’
Data field	- ‘5F4C 07’ || ‘D27600004000 36’ = DO CHA of profile 54 (SMC-A, SMC-B) or - ‘5F4C 07’ || ‘D27600004000 33’ = DO CHA of profile 51 (SMC-K)
Le	Absent

Table 67 - (N2070.00) GET SECURITY STATUS KEY response

Data field	Absent
SW1-SW2	- ‘9000’ Security status set; see [HPC-P1] - ‘63CF’ Security status not set; see [HPC-P1]

7.3 Asymmetric Authentication with TC Establishment

In the first part of the procedure the HPC proves its authenticity to the SMC. The software system will require a challenge from the SMC prior to sending the subsequent commands. Before the authentication commands are sent to the HPC, the key reference for PrK.HPC.AUTD_SUK_CVC and the algorithm identifier are set.

Table 68 - (N2071.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for internal authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘06’ = Length of subsequent data field
Data field	’84 01 11’ || ‘80 01 54' = DO KeyRef of PrK.HPC.AUTD_SUK_CVC of HPC || DO AlgID rsaSessionkey4SM
Le	Absent

Table 69 - (N2072.00) MSE response

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

Furthermore, the key used for external authentication and the appropriate algorithm identifier has to be set in the HPC. Both MSE SET commands for internal and external authentication are performed successively, so that the sequence of authentication commands will not be interrupted by a MSE SET command; see Chapter 15 of [HPC-P1].

Table 70 - (N2073.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for external authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘11’ = Length of subsequent data field
Data field	‘83 0C xx ... xx’ || ‘80 01 54’ = DO KeyRef of PuK.SMC.AUTD_RPS_CVC (SMC-A, SMC-B) or PuK.SAK.AUTD_CVC (SMC-K); see Note || DO AlgID rsaSessionkey4SM
Le	Absent

Table 71 - (N2074.00) MSE response

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

NOTE – The CHR of the CV certificate of the counterpart is taken as key reference. It has a length of 12 bytes: index set to '00' (1 byte) || KeyRef of PrK.SMC.AUTD_RPS_CVC or PrK.SAK.AUTD_CVC (1 byte) || ICCSN.SMC (10 byte); see Clause 8.5.2 of [HPC-P1] for CHR structure.

Table 72 - (N2075.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	‘88’ = INTERNAL AUTHENTICATE
P1	‘00’ = Algorithm already known to the card
P2	‘00’ = Key reference already known to the card
Lc	‘000010’ = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	‘0100’ = Length of expected signature

Table 73 - (N2076.00) INTERNAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
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 SMC and sets in the SMC the security status “CHA with role ID 'xx' successfully presented”.

The second part of the procedure comprises the external authentication of the SMC. First, a challenge is retrieved from the HPC.

Table 74 - (N2077.00) 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’ = Length of expected random number = 8

Table 75 - (N2078.00) GET CHALLENGE response

Data field	RND.HPC (8 bytes)
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 resulting digital signature of the INTERNAL AUTHENTICATE performed by the SMC is presented to the HPC in the EXTERNAL AUTHENTICATE command. The HPC has to verify the SMC signature.

Table 76 - (N2079.00) EXTERNAL AUTHENTICATE command for SMC authentication

CLA	As defined in ISO/IEC 7816-4
INS	‘82’ = EXTERNAL AUTHENTICATE
P1	‘00’ = Algorithm already known to the card
P2	‘00’ = Key reference already known to the card
Lc	‘000100’ = Length of subsequent data field = 256
Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
Le	Absent

Table 77 - (N2080.00) EXTERNAL AUTHENTICATE response

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

The authentication related data contain data elements for key computation. The secure messaging keys are computed as described in Clause 13.1 of [HPC-P1].

7.4 Asymmetric Authentication with Storage of Introduction Keys

In the authentication sequence of agreement on introduction keys, the HPC receives the first INTERNAL AUTHENTICATE and the SMC receives the first EXTERNAL AUTHENTICATE, in order to set in the SMC the security status that is required for the use of the private authentication key of the SMC. Before the authentication commands are sent to the HPC, the key reference for PrK.HPC.AUTD _SUK_CVC and the appropriate algorithm identifier are set.

Table 78 – (N2081.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for internal authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘06’ = Length of subsequent data field = 6
Data field	’84 01 11’ || ‘80 01 94' = DO KeyRef of PrK.HPC.AUTD_SUK_CVC of HPC || DO AlgID rsaSessionkey4Intro
Le	Absent

Table 79 – (N2082.00) MSE response

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

Furthermore, the key used for external authentication and the appropriate algorithm identifier have to be set in the HPC. Both MSE SET commands for internal and external authentication are performed successively, so that the sequence of authentication commands will not be interrupted by a MSE SET command; see Chapter 15 of [HPC-P1].

Table 80 – (N2083.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for external authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘11’ = Length of subsequent data field = 17
Data field	‘83 0C xx ... xx’ || ‘80 01 94’ = DO KeyRef of PuK.SMC.AUTD_RPS_CVC (SMC-A, SMC-B) or PuK.SAK.AUTD_CVC (SMC-K); see Note || DO AlgID rsaSessionkey4Intro
Le	Absent

Table 81 – (N2084.00) MSE response

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

NOTE – The CHR of the CV certificate of the counterpart is taken as key reference. It has a length of 12 bytes: index set to '00' (1 byte) || KeyRef of PrK.SMC.AUTD_RPS_CVC or PrK.SAK.AUTD_CVC (1 byte) || ICCSN.SMC (10 byte); see Clause 8.5.2 of [HPC-P1] for CHR structure.

The first part of the procedure comprises the internal authentication of the HPC. First, a challenge is retrieved from the SMC. Then, an INTERNAL AUTHENTICATE command with RND.SMC || ICCSN8.SMC in the data field has to be sent to the HPC.

Table 82 – (N2085.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'000010' = Length of subsequent data field = 16
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	'0100' = Length of expected signature = 256

Table 83 – (N2086.00) INTERNAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
SW1-SW2	'9000' or specific status bytes; see [HPC-P1]

The authentication token (content of the response data field) is verified in the related SMC that temporarily sets a security status. The second part of the procedure comprises the external authentication of the SMC. Therefore, the software system requires a challenge from the HPC.

Table 84 – (N2087.00) 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' = Length of expected random number = 8

Table 85 – (N2088.00) GET CHALLENGE response

Data field	RND.HPC (8 bytes)
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 resulting digital signature of the INTERNAL AUTHENTICATE performed by the SMC is presented to the HPC in the following EXTERNAL AUTHENTICATE command that verifies the SMC signature.

Table 86 – (N2089.00) EXTERNAL AUTHENTICATE command for SMC authentication

CLA	As defined in ISO/IEC 7816-4
INS	‘82’ = EXTERNAL AUTHENTICATE
P1	‘00’ = Algorithm already known to the card
P2	‘00’ = Key reference already known to the card
Lc	‘000100’ = Length of subsequent data field = 256
Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
Le	Absent

Table 87 – (N2090.00) EXTERNAL AUTHENTICATE response

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

The HPC temporarily sets a security status. The authentication related data contain data elements for key computation. The introduction keys are derived and their attributes are set during the first command following the authentication procedure as described in Clause 13.1 of [HPC-P1]. The CHR of the involved SMC CVC certificate is stored as reference of the introduction keys after adjusting the index (first byte of CHR) to the computed key material, i.e. ‘02’ for 3TDES keys; see Clause 8.5.2 of [HPC-P1]. During the derivation of introduction keys the security status is deleted and secure messaging is not enabled. The introduction keys are used in a symmetric authentication, in order to establish secure messaging keys, see next clause.

7.5 Symmetric authentication with TC Establishment

If a certain HPC and a certain SMC have been introduced to each other before, i.e. had performed an asymmetric authentication including the persistent storage of introduction keys, then both cards can perform a symmetric authentication by using the shared introduction keys.

During a successful symmetric authentication the security status “Successful verification of the SMC role identifier” is set, since the verified role identifier, the used key identifier and the access rule of the private key have been assigned to the introduction keys during the successful asymmetric authentication; see [HPC-P1], Clause 8.5.2, Clause 13.1.1 and Clause 15.6.

The command sequence at the HPC side starts with setting the introduction keys and the appropriate algorithm identifier for internal authentication.

Table 88 - (N2091.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for internal authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘11’ = Length of subsequent data field
Data field	’83 0C 02 xx...xx’ || ‘80 01 54' = DO KeyRef of introduction keys of HPC; see Note || DO AlgID desSessionkey4SM
Le	Absent

NOTE – The key reference has a length of 12 bytes: index set to '02' indicating 3TDES (1 byte) || KeyRef of PrK.SMC.AUTD_RPS_CVC or PrK.SAK.AUTD_CVC (1 byte) || ICCSN.SMC (10 bytes); see Clause 8.5.2 of [HPC-P1] for structure of the introduction key reference.

Table 89 - (N2092.00) MSE response

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

Furthermore, the key used for external authentication and the appropriate algorithm identifier has to be set in the HPC. Both MSE SET commands for internal and external authentication are performed successively, so that the sequence of authentication commands will not be interrupted by a MSE SET command; see Chapter 15 of [HPC-P1].

Table 90 - (N2093.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for external authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘11’ = Length of subsequent data field
Data field	‘83 0C 02 xx ... xx’ || ‘80 01 54’ = DO KeyRef of introduction keys of HPC; see Note || DO AlgID desSessionkey4SM
Le	Absent

NOTE – The key reference has a length of 12 bytes: index set to '02' indicating 3TDES introduction keys (1 byte) || KeyRef of PrK.SMC.AUTD_RPS_CVC or PrK.SAK.AUTD_CVC (1 byte) || ICCSN.SMC (10 bytes); see Clause 8.5.2 of [HPC-P1] for structure of the introduction key reference.

Table 91 - (N2094.00) MSE response

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

After that a challenge is retrieved from the SMC. Then, an INTERNAL AUTHENTICATE command with RND.SMC || ICCSN8.SMC in the data field has to be sent to the HPC.

Table 92 - (N2095.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	‘88’ = INTERNAL AUTHENTICATE
P1	‘00’ = Algorithm already known to the card
P2	‘00’ = Key reference already known to the card
Lc	‘10’ = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	‘68’ = Length of expected authentication related data = 104

Table 93 - (N2096.00) INTERNAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
SW1-SW2	‘9000’ or specific status bytes; see [HPC-P1]

Then, a MUTUAL AUTHENTICATE command follows on the side of the SMC. This command delivers the authentication related data of the HPC to the SMC. The SMC has to verify the HPC data and

compute the SMC authentication related data. The SMC authentication related data (content of the response data field) will be verified in the HPC during an EXTERNAL AUTHENTICATE command.

Table 94 – (N2097.00) EXTERNAL AUTHENTICATE command for SMC authentication

CLA	As defined in ISO/IEC 7816-4
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'68' = Length of subsequent data field = 104
Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
Le	Absent

Table 95 – (N2098.00) EXTERNAL AUTHENTICATE response

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

A successful verification sets in the HPC the security status "CHA with role ID 'xx' successfully presented". A trusted channel has been established, i.e. data can be transferred to the HPC in secure messaging mode.

7.6 Authorizing procedures

According to [GMG], a health professional can authorize other dedicated persons to access an eGK. This is achieved by using an authorized SMC-A or SMC-B. The authorization of the SMC is linked to the use of PrK.SMC.AUTR_CVC in card-to-card authentication procedures. The usage of the key PrK.SMC.AUTR_CVC for a SMC/eGK authentication requires according to the its access rule an HPC authentication with the appropriate role ID present in the CHA of the C.HPC.AUTR_CVC; see Clause 5.3.9 and Clause 5.7 of [HPC-P3] for more details.

Alternatively for the SMC-B, the HP authentication can be achieved by a successful presentation of PIN.SMC; see Clause 6.3.10, Clause 6.3.11 and Clause 6.7 of [HPC-P3] for further details.

The following table summarizes the possible authorization procedures (towards SMCs) denoting the involved cards and the access conditions of the authentication keys.

Table 96 – (N2099.00) Possible authorization procedures (informative)

Possible Use	Authenti-cating Card	Security Condition of Authentication Key of Authenticating card	Target card	Security Condition of Authentication Key of Target card
Authorization of SMC for access to eGK data	HPC of profile 2 (or other role profile)	PWD(PIN.CH)	SMC-A of related profile	AUT(CHA of related profile)
			SMC-B of related profile	PWD(PIN.SMC) OR AUT(CHA of related profile)
	SMC-B of profile 2 (or other role profile)	PWD(PIN.SMC) OR AUT(CHA of profile 2)	SMC-A of related profile	AUT(CHA of related profile)
			SMC-B of related profile	PWD(PIN.SMC) OR AUT(CHA of related profile)
	SMC-A of profile 2 (or other role profile)	AUT(CHA of profile 2)	SMC-A of related profile	AUT(CHA of related profile)
			SMC-B of related profile	PWD(PIN.SMC) OR AUT(CHA of related profile)

Authorization of SMC for PIN transfer to authenticating card	HPC of profile 53	ALWAYS	SMC-A with profile 54	AUT(CHA of profile 55)
			SMC-B with profile 54	AUT(CHA of profile 55)
	SMC-B of profile 55	ALWAYS	SMC-A with profile 54	AUT(CHA of profile 55)
			SMC-B with profile 54	AUT(CHA of profile 55)
	RFID Token profile 55	PWD(PIN.Token) OR ALWAYS	SMC-A with profile 54	AUT(CHA of profile 55)
			SMC-B with profile 54	AUT(CHA of profile 55)
Authorization of SMC for DTBS transfer to HPC	HPC profile 53	ALWAYS	SMC-K with profile 51	AUT(CHA of profile 53)

NOTE – This table denotes authorization procedures which are technically feasible on basis of the access rules of involved authentication keys. It is not defined here, whether they represent use cases to be permitted.

The authorization of the SMC-A or SMC-B for remote PIN transfer to the authenticating card (HPC, SMC-B or RFID Token) or authorization of the SMC-K for transfer of DTBS to the HPC is part of the C2C authentication procedure, i.e. the target card authenticates first to enable the use of the SMC's authentication key for sending the data; see Clause 7.1.

Before an HPC will authorize an SMC, PIN.CH must be successfully presented. The PIN.CH authentication and the SMC authorization can be performed in any channel of the HPC. In a first step of the HPC authentication, the private key for role authentication or the corresponding introduction keys (if previously stored during a role authentication with the private key) together with the appropriate algorithm have to be set.

Table 97 - (N2100.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for internal authentication
P2	'A4' = Authentication template
Lc	'06' or '11' = Length of subsequent data field
Data field	'84 01 10 || 80 01 00' = DO KeyRef of PrK.HPC.AUTR_CVC || DO AlgID rsaRoleAuthentication
Le	Absent

Table 98 - (N2101.00) MSE response

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

After that a challenge is retrieved from the SMC. Then, an INTERNAL AUTHENTICATE command with RND.SMC || ICCSN8.SMC in the data field has to be sent to the HPC.

Table 99 – (N2102.00) INTERNAL AUTHENTICATE command for proving the authenticity of an HPC for authorization of an SMC

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'000010' = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	'0100' = Length of expected authentication related data = 256

Table 100 – (N2103.00) INTERNAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
SW1-SW2	'9000' or specific status bytes; see [HPC-P1]

The authentication related data produced by the HPC will be verified by the SMC, which sets – if authentication successful – the security status required for the usage of the PrK.SMC.AUTR_CVC.

8 The Health Professional Application HPA

8.1 File Structure and File Content

The file structure of DF.HPA is shown in Figure 4 (N2104.00).

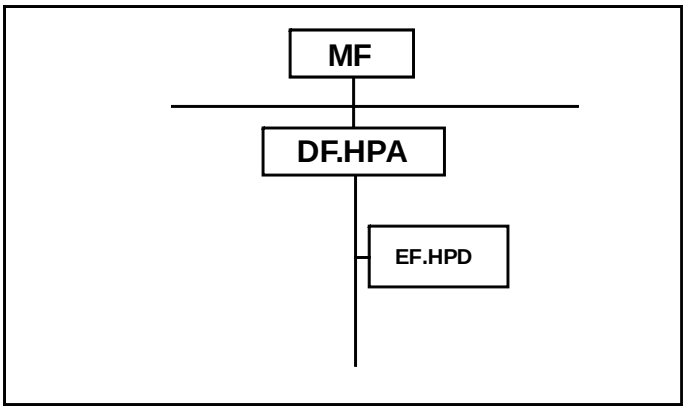

Figure 4 – (N2104.00) File structure of DF.HPA

8.1.1 DF.HPA (Health Professional Application)

DF.HPA is an application according to Clause 8.3.1.1 of [HPC-P1], i.e. selectable by using the application identifier. Table 101 (N2105.00) shows the characteristics of the application directory.

Table 101 – (N2105.00) Characteristics of MF / DF.HPA

Attribute	Value	Note
Object type	Application Directory	
Application Identifier	'D276 00014602'	
File Identifier	-	Manufacturer-specific; if supported, then out of interval ['1000', 'FEFF']; see Clause 8.1.1 of [HPC-P1]
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION (after HPC issuing)	AUT('D27600014600' || '01') AND SmMac AND SmCmdEnc	Only executable if a CAMS is used; see Chapter 13. If a CAMS with symmetric authentication is used, then the security condition must contain the key reference of the corresponding symmetric key, i.e. AUT('13') instead of AUT('D27600014600' || '01').
ACTIVATE, DEACTIVATE, DELETE	NEVER	

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 Chapter 13.

8.1.2 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; see Table 102 (N2106.00) for further characteristics of the file.

Table 102 – (N2106.00) Characteristics of MF / DF.HPA / EF.HPD

Attribute	**Value**	**Note**
Object type	Transparent Elementary File	
File Identifier	'D001'	
Short File Identifier	'01' = 1	
Number of Bytes	2048	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized.
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
UPDATE BINARY	PWD(PIN.CH)	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

8.2 Security Environments

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

8.3 Application Selection

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

Table 103 - (N2107.00) 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	'D27600014602' = AID of DF.HPA
Le	Absent

Table 104 - (N2108.00) SELECT response

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

8.4 Reading and Updating of EF.HPD

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

Table 105 - (N2109.00) READ BINARY command for reading EF.HPD

CLA	As defined in ISO/IEC 7816-4
INS	‘B0’ = READ BINARY
P1, P2	- P1 = b8-b6: 100 b5-b1: 00001 SFID of EF.HPD: 1 P2 = Offset - ‘xxxx’ = Offset (bit 8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	'00' or ‘000000’

Table 106- (N2110.00) READ BINARY response

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

If the supported extended length is not sufficient to read the data with a single command, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

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

Table 107 - (N2111.00) UPDATE BINARY command for updating EF.HPD

CLA	As defined in ISO/IEC 7816-4
INS	‘D6’ = UPDATE BINARY
P1, P2	- P1 = b8-b6: 100 b5-b1: 00001 SFID of EF.HPD: 1 P2 = Offset - ‘xxxx’ = Offset (bit 8 of P1 = 0)
Lc	‘xx’ or ‘00xxxx’ = Length of subsequent data field
Data field	Data
Le	Absent

Table 108 - (N2112.00) UPDATE BINARY response

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

If the supported extended length is not sufficient to write the data with a single command, the UPDATE BINARY command shall be repeated specifying the respective offset in P1-P2.

9 The Qualified Electronic Signature Application

9.1 File Structure and File Content

The general structure of the QES application, which is in accordance with DIN66291, is shown in Fig. 5 (N2113.00).

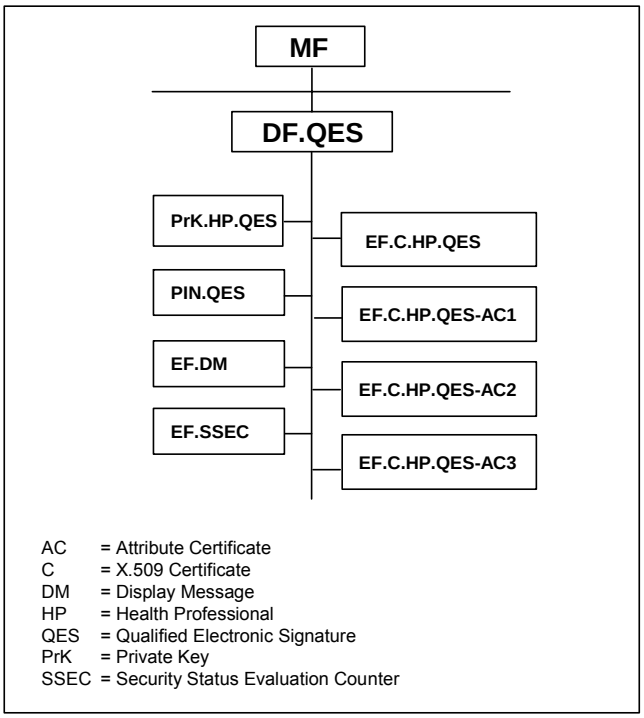

Figure 5 – (N2113.00) General structure of the QES application

The QES application contains EFs for the X.509-QES certificate and up to 3 attribute certificates. Furthermore, an EF for a display message that may be relevant for remote usage of an HPC and an EF for indication of the supported maximum SSEC are present.

9.1.1 DF.QES (Qualified Electronic Signature Application)

DF.QES is an application according to Clause 8.3.1.1 of [HPC-P1], i.e. selectable by using the application identifier; see Clause 9.4. The following table shows the attributes and access rules that apply for the QES application directory.

Table 109 – (N2114.00) Characteristics of MF / DF.QES

Attribute	**Value**	**Note**
Object type	Application Directory	
Application Identifier	'D27600006601'	
File Identifier	-	Manufacturer-specific; if supported, then out of interval ['1000', 'FEFF']; see Clause 8.1.1 of [HPC-P1]
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION, ACTIVATE, DEACTIVATE, DELETE	NEVER	

9.1.2 PrK.HP.QES

PrK.HP.QES is the private key for the computation of qualified electronic signatures. The key reference and other characteristics of the key are shown in the subsequent table. The PIN characteristics are outlined in Clause 9.1.3.

Table 110 – (N2115.00) Characteristics of MF / DF.QES / PrK.HP.QES

Attribute	**Value**	**Note**
Object type	Private RSA Object	
Key Identifier	'04' = 4	To be personalized
Key Reference	'84'	
Private Key	.... (2048 bit)	To be personalized
Key Available	True	
Algorithm Identifier	signPKCS1_V1_5 signPSS sign9796_2_DS2	
Access Rule in SE # 1		
Access Mode	**Security Condition**	**Note**
COMPUTE DIGITAL SIGNATURE (P2 = '9E' or 'AC')	PWD(PIN.QES)	Single signature mode
Other	NEVER	
Access Rule in SE # 2		
Access Mode	**Security Condition**	**Note**
COMPUTE DIGITAL SIGNATURE (P2 = '9E' or 'AC')	PWD(PIN.QES) AND AUT('D27600004000' || '33') AND SmMac AND SmCmdEnc AND SmRespEnc See Note	Stack and comfort signature mode; see [TR-03114] and [TR-03115] Device authentication of SMC-K with profile 51 (SAK)
Other	NEVER	

NOTE – In spite of the AND relation of the security conditions, the HPC by itself cannot rule out the possibility that the authentication with profile 51 had been performed without agreement on SM keys and the SM keys had been agreed with a counterpart's profile other than profile 51 (authentication of profile 51 may remain set). This is ruled out by two restrictions: Firstly, the private key of SMC-K with profile 51 (SAK) can only be used with either agreement on SM keys or storage of introduction keys. Secondly, the access rule of the SMC-K's authentication key related to profile 51 demands that the HPC with profile 53 has to authenticate first. Since the storage of introduction keys does not set a security status, the possibly available SM keys in the HPC after authentication of the SMC-K must result from the very authentication.

9.1.3 PIN.QES

PIN.QES is a DF-specific PIN, which is used only for protection of the SigG/SigV-related private electronic signature key of the health professional (PrK.HP.QES). The PIN consists of 6 to 8 digits.
The usage of the 8 to 12 digit resetting code (personal unblocking key, PUK) is limited by a usage counter with an initial value of 10. The security status of PIN.QES can be used only a limited number of times, i.e. the maximum value of SSEC is limited.

The PIN reference as used in the VERIFY, CHANGE REFERENCE DATA and RESET RETRY COUNTER command and the other PIN attributes are shown in the following table.

Table 111 – (N2116.00) Characteristics of MF / DF.QES / PIN.QES

Attribute	Value	Note
Object type	Password	
Password Identifier	'01'	
Password Reference	'81'	
Secret	....	To be personalized
Minimum Length	6	
Start Retry Counter	3	
Retry Counter	3	
Transport Status	Transport PIN Random or Transport PIN Derived or Transport PIN Fixed Value or Regular Password	
Flag Enabled	True	
Start Security Status Evaluation Counter	SE # 1: SSEC = 1 SE # 2: 1 ≤ SSEC ≤ 250	Maximum value in SE # 2 as indicated in EF.SSEC
PUK	...	To be personalized
PUK Usage	10	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
CHANGE RD (Option '00')	ALWAYS	
GET PIN STATUS	ALWAYS	
RESET RC (Option '01')	ALWAYS	
VERIFY	ALWAYS	
Other	NEVER	

According to Clause 14.6.1.4 and Clause 14.6.5.6 of [HPC-P1] the COS checks only the minimum length (6 digits) of PIN.QES, i.e. the COS does not control whether the maximum length of 8 digits is exceeded.

The initialization of PIN.QES e.g. by using a transport PIN has to comply with BNA regulations. The command CHANGE REFERENCE DATA (see 4.6.2) is used to establish a regular PIN.

9.1.4 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. Further characteristics of the EF.DM are shown in the subsequent table.

Table 112 – (N2117.00) Characteristics of MF / DF.QES / EF.DM

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'D004'	
Short File Identifier	'04' = 4	
Number of Bytes	8 bytes	
Flag Transaction Mode	True	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
READ BINARY	{AUT('D27600004000' || '33') OR AUT('D27600004000' || '36') } AND SmMac AND	Role authentication of SMC-K with profile 54 (SAK) or SMC-A or SMC-B with profile 51

	SmRspEnc	(PIN sender)
UPDATE BINARY	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

9.1.5 EF.SSEC

The transparent file EF.SSEC indicates the maximum values of SSEC that are defined for an individual application context of the HPC in compliance with the evaluation and approvement of the HPC as secure signature creation device. The File characteristics are shown in Table 113 (N2118.00).

Table 113 – (N2118.00) Characteristics of MF / DF.QES / EF.SSEC

Attribute	**Value**	**Note**
Object type	Transparent Elementary File	
File Identifier	'D005'	
Short File Identifier	'05' = 5	
Number of Bytes	46 bytes	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized; see Table 114 (N2119.00)
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

The content of EF.SSEC is stored during personalization; see Table 114 (N2119.00). The external signature application component may read the content of EF.SSEC in order to optimize the stack sizes. It is not possible to retrieve the current values of the SSECs, whose maxima are implemented during card production and current states managed by the COS in RAM. The implemented SSEC maximum values must be consistent with the values indicated in EF.SSEC.

Table 114 – (N2119.00) Content of EF.SSEC

Tag	**Length**	**Meaning**					
'7B'	'2C'	Set of security environment data objects					
		Tag	**Length**	**Value**	**Meaning**		
		'80'	'01'	'01'	Security Environment ID = 1		
		'A4'	'11'	Authentication Template			
				Tag	**Length**	**Value**	**Meaning**
				'82'	'06'	'D27600006601'	DF name: DF.QES
				'83'	'01'	'81'	Key reference: PIN.QES
				'95'	'01'	'08'	Usage Qualifier: User authentication
				'C0'	'01'	'01'	Maximum value of SSEC = 1
		Tag	**Length**	**Value**	**Meaning**		
		'80'	'01'	'02'	Security Environment ID = 2		
		'A4'	'11'	Authentication Template			
				Tag	**Length**	**Value**	**Meaning**
				'82'	'06'	'D27600006601'	DF name: DF.QES
				'83'	'01'	'81'	Key reference: PIN.QES
				'95'	'01'	'08'	Usage Qualifier: User authentication
				'C0'	'01'	'xx'	Maximum value of SSEC, e.g. 250

NOTE 1 – The set of data objects under tag '7B' is used according to [ISO7816-4] except for the SSEC object. The maximum value of SSEC could alternatively have been expressed in the CIA.QES file EF.PrKD as common

object attribute userConsent. However, a value of e.g. 250 would be beyond the upper bound value cia-ub-user-Consent = 15 defined in [ISO7816-15]. Furthermore, the attribute userConsent can hardly be linked to an individual security environment.

NOTE 2 – Maximum SSEC values in the range 251 to 254 ('FB'-FE') should not be set, since these values may be reserved by the COS for other purposes. If a default SSEC of infinite is required, this must be indicated in EF.SSEC by coding 'FF' in the SSEC field.

9.1.6 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. The file characteristics are shown in Table 115 (N2120.00).

Table 115 – (N2120.00) Characteristics of MF / DF.QES / EF.C.HP.QES

Attribute	**Value**	**Note**
Object type	Transparent Elementary File	
File Identifier	'C000'	
Short File Identifier	'10' = 16	
Number of Bytes	1536 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized.
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

9.1.7 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). The characteristic file attributes and access rules of the files are shown in the subsequent tables.

Table 116 – (N2121.00) Characteristics of MF / DF.QES / EF.C.HP.QES-AC1

Attribute	**Value**	**Note**
Object type	Transparent Elementary File	
File Identifier	'C001'	
Short File Identifier	'01' = 1	
Number of Bytes	1024 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	... If empty, first byte set to '00'	To be personalized
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
UPDATE BINARY	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

Table 117 – (N2122.00) Characteristics of MF / DF.QES / EF.C.HP.QES-AC2

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'C002'	
Short File Identifier	'02' = 2	
Number of Bytes	1024 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	... If empty, first byte set to '00'	To be personalized
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
UPDATE BINARY	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

Table 118 – (N2123.00) Characteristics of MF / DF.QES / EF.C.HP.QES-AC3

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'C003'	
Short File Identifier	'03' = 3	
Number of Bytes	1024 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	... If empty, first byte set to '00'	To be personalized
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
UPDATE BINARY	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

All TLV coded X.509 certificates start with tag '30' (of a sequence object) in the first byte. If a file does not contain a certificate, this shall be indicated: the first byte shall be set to '00'.

9.2 Creation of a Qualified Electronic Signature

An electronic signature process consists of several steps as shown in Figure 6 (N2124.00).

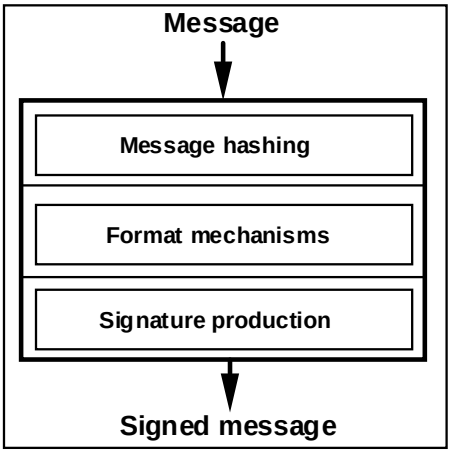

Figure 6 - (N2124.00) Signature process according to ISO/IEC 9796-2

Regarding the signature process defined for the HPC the complete hashing is performed outside 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 is set with the MSE command.

9.3 Security Environments

In DF.QES, two security environments are used:

- SE # 1: Default SE for use of the signature function in single signature mode. A use of a trusted channel is not required. It is possible to establish a trusted channel though.

- SE # 2: SE for use of the signature function in stack and comfort signature mode. A trusted channel is used between HPC/SMC-K for transmission of data to be signed in a health professional environment (verified by the card).

Note – The PIN for stack and comfort signature is transferred to the HPC in SE # 2 with or without trusted channel. If the PIN is to be entered remotely, the external software will control that the PIN is transferred in a trusted channel.

9.4 Selecting the QES Application

Prior to application selection it may be neccessary to open a further logical channel; see Clause 5.2. The QES application has to be selected with the SELECT command.

Table 119 - (N2125.00) 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 120 - (N2126.00) SELECT Response

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

After DF selection, SE # 1 is implicitly selected.

9.5 Selecting the Security Environment

If the QES application is to be used in the stack and comfort signature mode, prior to any command processing after application selection, the SE # 2 shall be selected with the following MSE command using the RESTORE function.

Table 121 - (N2127.00) MSE command for selecting the SE relevant for stack and comfort signature

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 122 - (N2128.00) MSE response

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

9.6 Reading and Updating the Display Message

When the authentication procedure has established a trusted channel, prior to the QES computation the display message may be read and shown to the health professional as feedback that subsequent commands can be sent in secure messaging mode. The reading of the display message, however, is not controlled by the HPC. For reading the display message, the READ BINARY command is used.

Table 123 - (N2129.00) 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' = Length of expected display message = 8

NOTE – Each TC relevant application (DF.QES and DF.ESIGN) has its own EF.DM with SFID 4.

Table 124 - (N2130.00) 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 125 - (N2131.00) 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 bytes)
Le	Absent

Table 126 - (N2132.00) UPDATE BINARY response

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

9.7 PIN Management

9.7.1 PIN Verification

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

Table 127 - (N2133.00) 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 Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 128 - (N2134.00) 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 bytes as specified in Clause 14.6.6.3 of [HPC-P1].

9.7.2 PIN Change

For PIN change the ISO/IEC 7816-4 command CHANGE REFERENCE DATA may be used at any time convenient for the HP. If PIN.QES is referenced, the command can only be used with P1 = '00', i.e. with both old and new PIN in the data field. The successful verification of the old PIN does not set the security status of the referenced PIN; see Clause 14.6.1 of [HPC-P1].

The command is especially used to lift a PIN transport status and establish a regular PIN; see Clause 8.2.5 and Clause 14.6.1 of [HPC-P1].

Table 129 - (N2135.00) CHANGE REFERENCE DATA 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 Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 130 - (N2136.00) CHANGE REFERENCE DATA response

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

9.7.3 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 successful verification of the resetting code does not set the security status of the referenced PIN object; see Clause 14.6.5 of [HPC-P1]. The successful verification of PIN.QES is still required before using PrK.HP.QES. The use of the resetting code must comply with the operative regulations of Bundesnetzagentur. According to the current regulations a new PIN.QES must not be set in a RESET RETRY COUNTER command.

Table 131 – (N2137.00) RESET RETRY COUNTER 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 Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 132 - (N2138.00) RESET RETRY COUNTER response

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

9.7.4 Query of the PIN.QES status

The software can query the status related to a PIN by using a GET PIN STATUS command as shown in the following table.

Table 133 – (N2139.00) GET PIN STATUS command

CLA	As defined in ISO/IEC 7816-4 for proprietary class
INS	'20' = GET PIN STATUS
P1	'00'
P2	'81' = PIN.QES reference
Lc	Absent
Data field	Absent
Le	Absent

Table 134 – (N2140.00) GET PIN STATUS response

Data field	Absent
SW1-SW2	'9000' Verification not required or specific status bytes; see [HPC-P1]

9.8 Computation of a QES

The signing is performed after complete hashing outside the HPC. Before the PSO: COMPUTE DIGITAL SIGNATURE command is sent to the HPC, the private signature key and the algorithm have to be selected.

Table 135 - (N2141.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for computation
P2	'B6' = Digital Signature Template
Lc	'06' = Length of subsequent data field
Data field	'84 01 84' || '80 01 yy' = DO KeyRef of PrK.HP.QES || DO AlgID 'yy' = '02': signPKCS1_V1_5 'yy' = '05': signPSS 'yy' = '07': sign9796_2_DS2
Le	Absent

Table 136 - (N2142.00) MSE response

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

Table 137 - (N2143.00) PSO: COMPUTE DIGITAL SIGNATURE 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 contains data to be signed or integrated in the DSI (for PKCS#1 SSA-V1.5 and PKCS#1 PSS) - 'AC' = Data field contains data objects with data elements to be signed or integrated in the DSI (for ISO9796-2 DS2)
Lc	'00xxxx' = Length of subsequent data field
Data field	- DSI according to ISO9796-2 DS2 or - DSI according to PKCS#1 SSA-V1.5 or - DSI according to PKCS#1 PSS
Le	'0100' = length of expected digital signature = 256

Table 138 - (N2144.00) PSO: COMPUTE DIGITAL SIGNATURE response

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

9.9 Reading of QES related Certificates

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

Table 139 - (N2145.00) 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 00001 SFID of EF.C.HP.QES-AC1: 1 00010 SFID of EF.C.HP.QES-AC2: 2 00011 SFID of EF.C.HP.QES-AC3: 3 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	'000000'

Table 140 - (N2146.00) READ BINARY response

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

If the supported extended length is not sufficient to read the whole certificate with a single command, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

9.10 Writing of Attribute Certificates

For updating attribute certificates (see access conditions in Table 116 (N2121.00) , Table 117 (N2122.00) and Table 118 (N2123.00)), the ISO/IEC 7816-4 command UPDATE BINARY shall be used.

Table 141 - (N2147.00) 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 00010 SFID of EF.C.HP.QES-AC2: 2 00011 SFID of EF.C.HP.QES-AC3: 3 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	'00xxxx' = Length of subsequent data field
Data field	Data
Le	Absent

Table 142 - (N2148.00) UPDATE BINARY response

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

If the supported extended length is not sufficient to write the whole certificate with a single command, then the UPDATE BINARY command shall be repeated specifying the respective offset in P1-P2.

10 The ESIGN Application

10.1 File Structure and File Content

The general structure of the ESIGN application, which is in accordance with [EN14890-1], is shown in Figure 7 (N2149.00).

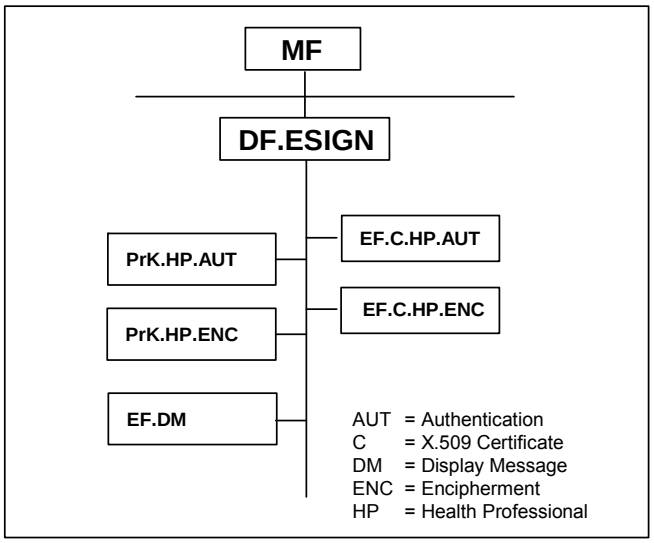

Figure 7 – (N2149.00) General structure of DF.ESIGN

10.1.1 DF.ESIGN

DF.ESIGN is an application according to Clause 8.3.1.1 of [HPC-P1], i.e. selectable by using the application identifier; see Clause 10.3. The characteristics of the application directory are shown in Table 143 (N2150.00).

Table 143 – (N2150.00) Characteristics of MF / DF.ESIGN

Attribute	Value	Note
Object type	Application Directory	
Application Identifier	'A000000167 455349474E'	
File Identifier	-	Manufacturer-specific; if supported, then out of interval [‘1000’, ‘FEFF’]; see Clause 8.1.1 of [HPC-P1]
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION, ACTIVATE, DEACTIVATE, DELETE	NEVER	

10.1.2 PrK.HP.AUT

PrK.HP.AUT is the private key for client/server authentication. The key reference and other characteristics of the private key are shown in the following table.

Table 144 – (N2151.00) Characteristics of MF / DF.ESIGN / PrK.HP.AUT

Attribute	Value	Note
Object type	Private RSA Object	
Key Identifier	'02' = 2	To be personalized
Key Reference	'82'	
Private Key	.... (2048 bit)	To be personalized
Key Available	True	
Algorithm Identifier	INTERNAL AUTHENTICATE: rsaClientAuthentication PSO: COMPUTE DIGITAL SIGNATURE: signPKCS1_V1_5 signPSS sign9796_2_DS2	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
INTERNAL AUTHENTICATE, PSO: COMPUTE DIGITAL SIGNATURE (P2 = '9E' or 'AC')	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
Other	NEVER	

With respect to key length the same conventions are used as for legally binding electronic signature keys; see [ALGCAT] and [TR-03116] .

10.1.3 PrK.HP.ENC

PrK.HP.ENC is the private key for document cipher key decipherment. The key reference and other characteristics of the private key are shown in the following table.

Table 145 – (N2152.00) Characteristics of MF / DF.ESIGN / PrK.HP.ENC

Attribute	Value	Note
Object type	Private RSA Object	
Key Identifier	'03' = 3	To be personalized
Key Reference	'83'	
Private Key	.... (2048 bit)	To be personalized
Key Available	True	
Algorithm Identifier	rsaDecipherOaep rsaDecipherPKCS1_V1_5	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
PSO: DECIPHER, PSO: TRANSCIPHER	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
Other	NEVER	

With respect to key length the same conventions are used as for legally binding electronic signature keys; see [ALGCAT] and [TR-03116].

10.1.4 EF.DM

The transparent file EF.DM contains the same display message as described in 9.1.4. The file characteristics are shown in the subsequent table.

Table 146 – (N2153.00) Characteristics of MF / DF.ESIGN / EF.DM

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	‘D004’	
Short File Identifier	‘04’ = 4	
Number of Bytes	8 bytes	
Flag Transaction Mode	True	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
READ BINARY	{AUT(‘D27600004000’ || '33') OR AUT(‘D27600004000’ || '36') } AND SmMac AND SmRspEnc	Role authentication of SMC-K with profile 54 (SAK) or SMC-A or SMC-B with profile 51 (PIN sender)
UPDATE BINARY	PWD(PIN.CH)	The access rule for PIN.CH is specified at MF level.
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

10.1.5 EF.C.HP.AUT

EF.C.HP.AUT contains the X.509 AUT certificate of the health professional. The characteristics of the certificate file are shown in the subsequent table.

Table 147 – (N2154.00) Characteristics of MF / DF.ESIGN / EF.C.HP.AUT

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	‘C500’	
Short File Identifier	‘01’ = 1	
Number of Bytes	1536 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized; see Annex C
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

10.1.6 EF.C.HP.ENC

EF.C.HP.ENC contains the X.509 ENC certificate of the health professional. The characteristics of the certificate file are shown in the subsequent table.

Table 148 – (N2155.00) Characteristics of MF / DF.ESIGN / EF.C.HP.ENC

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	‘C200’	
Short File Identifier	‘02’ = 2	

Number of Bytes	1024 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized; see Annex C
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

10.2 Security Environments

In DF.ESIGN only SE # 1 (default SE) is used. It is possible in SE # 1 to establish a trusted channel, e.g. for remote HPC configurations.

10.3 ESIGN Application Selection

The ESIGN application selection is performed with the SELECT command.

Table 149 - (N2156.00) 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 150 - (N2157.00) SELECT Response

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

10.4 Reading an X.509 Certificate

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

Table 151 - (N2158.00) 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 00010 SFID of EF.C.HP.ENC: 2 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	'000000'

Table 152 - (N2159.00) READ BINARY response

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

If the supported extended length is not sufficient to read the whole certificate with a single command, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

10.5 PIN Management

Prior to the usage of PrK.HP.AUT or PrK.HP.ENC, PIN.CH has to be successfully presented; see Clause 4.6.1.

10.6 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.

The card computes the authentication related data 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 private key and the algorithm identifier denoting the type of authentication protocol have to be set.

Table 153 – (N2160.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for internal authentication
P2	‘A4’ = Authentication Template
Lc	‘06’ = Length of subsequent data field
Data field	‘84 01 82’ || ’80 01 05’ = DO KeyRef of PrK.HP.AUT || DO AlgID rsaClientAuthentication
Le	Absent

Table 154 - (N2161.00) MSE response

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

Table 155 – (N2162.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	‘88’ = INTERNAL AUTHENTICATE
P1	‘00’
P2	‘00’
Lc	‘00xxxx’ = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	‘0100’ = length of expected digital signature = 256

Table 156 – (N2163.00) INTERNAL AUTHENTICATE response

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

PrK.HP.AUT can also be used with the PSO: COMPUTE DIGITAL SIGNATURE command. The private authentication key and the algorithm identifier denoting the type of authentication protocol have to be set before.

Table 157 (N2164.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for computation
P2	'B6' = Digital Signature Template
Lc	'06' = Length of subsequent data field
Data field	'84 01 82' || '80 01 yy' = DO KeyRef of PrK.HP.AUT || DO AlgID 'yy' = '02': signPKCS1_V1_5 'yy' = '05': signPSS 'yy' = '07': sign9796_2_DS2
Le	Absent

Table 158 - (N2165.00) MSE response

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

Table 159 - (N2166.00) PSO: COMPUTE DIGITAL SIGNATURE 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 contains data to be signed or integrated in the DSI for PKCS#1 SSA-V1.5 or PKCS#1 PSS - 'AC' = Data field contains data objects with data elements to be signed or integrated in the DSI for ISO9796-2 DS2
Lc	'00xxxx' = Length of subsequent data field
Data field	- DSI according to ISO9796-2 DS2 or - DSI according to PKCS#1 SSA-V1.5 or - DSI according to PKCS#1 PSS
Le	'0100' = length of expected digital signature = 256

Table 160 - (N2167.00) PSO: COMPUTE DIGITAL SIGNATURE response

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

10.7 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 and the algorithm have to be selected with the ISO/IEC 7816-4 command MSE.

Table 161 - (N2168.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for decipherment
P2	'B8' = Confidentiality Template
Lc	'06' = Length of subsequent data field
Data field	'84 01 83' || '80 01 yy' = DO KeyRef of PrK.HP.ENC || DO AlgID 'yy' = 85': rsaDecipherOaep 'yy' = 81': rsaDecipherPKCS1_V1_5
Le	Absent

Table 162 - (N2169.00) MSE response

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

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

Table 163 – (N2170.00) 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	'000101' = Length of subsequent data field = 257
Data field	'00' (Padding indicator) || cryptogram (256 bytes)
Le	'0000' or '00xx' = Length of document cipher key

Table 164 – (N2171.00) PSO: DECIPHER response

Data field	Document cipher key
SW1-SW2	'9000'or specific status bytes; see [HPC-P1]

10.8 Document Cipher Key Transcipherment

For exchange of already enciphered documents with a third party, the following scheme applies:

- cipher key decipherment with PrK.HP.ENC
- cipher key encipherment with the third party's public key.

Both steps are performed in a single TRANSCIPHER command, so that the plaintext data do not need to appear at the card interface.

Before the transcipherment of the cipher key can be performed, the PIN.CH has to be presented successfully. In addition, the private key PrK.HP.ENC and the algorithm have to be selected with the ISO/IEC 7816-4 command MSE.

Table 165 - (N2172.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for decipherment
P2	'B8' = Confidentiality Template
Lc	'06' = Length of subsequent data field
Data field	'84 01 83' || '80 01 yy' = DO KeyRef of PrK.HP.ENC || DO AlgID 'yy' = '85': rsaDecipherOaep 'yy' = '81': rsaDecipherPKCS1_V1_5
Le	Absent

Table 166 - (N2173.00) MSE response

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

After the key and the algorithm have been set, the transcipher operation can be executed with the proprietary command PSO: TRANSCIPHER; see Clause 14.8.7 of [HPC-P1]. The command data field contains a constructed cryptogram DO (tag 'A6') with the cipher key to be transciphered as well as a constructed plaintext DO (tag 'A0') providing the algorithm identifier (tag '80') and the public key of the third party (tag '7F49') for the encipherment part of the operation; see Clause 14.8.7.4 of [HPC-P1].

Table 167 – (N2174.00) PSO: TRANSCIPHER command

CLA	As defined in ISO/IEC 7816-4 for proprietary class
INS	'2A' = PERFORM SECURITY OPERATION: TRANSCIPHER
P1	'86' = Return cryptogram
P2	'B8' = Confidentiality Template
Lc	'00021F' = Length of subsequent data field = 543
Data field	'A6' || lengthA6 (3 B) || '86' || length86 (3 B) || Padding indicator (1 B) || cryptogram (256 B) || 'A0' || lengthA0 (3 B) || '80' || length80 (1 B) || AlgID (1 B) || '7F49' || length7F49 (3 B) || '81' || length81 (3 B) || Modulus (256 B) || '82' || length82 (1 B) || Exponent (4 B)
Le	'0000' or '0101' = Length of the enciphered document cipher key with preceding PI = 257

Table 168 – (N2175.00) PSO: TRANSCIPHER response

Data field	Padding Indicator || Enciphered document cipher key
SW1-SW2	'9000'or specific status bytes; see [HPC-P1]

11 The Cryptographic Information Applications

11.1 General Structure

The general structures of the Cryptographic Information Applications (CIAs) associated to the QES application and the ESIGN application, are shown in Figure 8 (N2176.00).

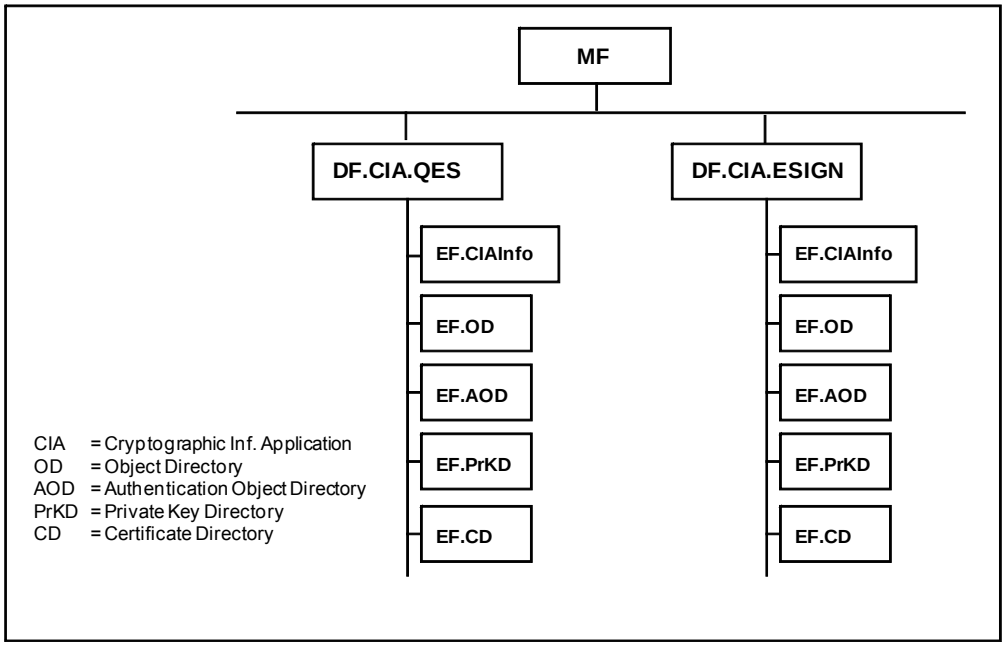

Figure 8 – (N2176.00) DF.CIA applications and their substructures

11.2 DF.CIA.QES and DF.CIA.ESIGN (Cryptographic Information Applications)

DF.CIA.QES and DF.CIA.ESIGN are applications according to Clause 8.3.1.1 of [HPC-P1], i.e. selectable by using the respective application identifier; see Clause 11.4. The following tables show the attributes and access rules that apply for the application directories.

Table 169 – (N2177.00) Characteristics of MF / DF.CIA.QES

Attribute	**Value**	**Note**
Object type	Application Directory	
Application Identifier	'E828BD080F D27600006601'	
File Identifier	-	Manufacturer-specific; if supported, then out of interval [‘1000’, ‘FEFF’]; see Clause 8.1.1 of [HPC-P1]
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION, ACTIVATE, DEACTIVATE, DELETE	NEVER	

Table 170 – (N2178.00) Characteristics of MF / DF.CIA.ESIGN

Attribute	Value	Note
Object type	Application Directory	
Application Identifier	'E828BD080F A000000167 455349474E'	
File Identifier	-	Manufacturer-specific; if supported, then out of interval ['1000', 'FEFF']; see Clause 8.1.1 of [HPC-P1]
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION, ACTIVATE, DEACTIVATE, DELETE	NEVER	

11.3 Files with Cryptographic Information Objects (CIOs)

The logical file names, the FIDs, the SFIDs and the content of the files (see Annex D and Annex E) are in compliance with ISO/IEC 7816-15. The corresponding CIO files in both MF / DF.CIA.QES and MF / DF.CIA.ESIGN have the same characteristic attributes and access rules as shown in the subsequent tables.

Table 171 – (N2179.00) Characteristics of EF.CIAInfo (Cryptographic Information Application Info)

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'5032'	
Short File Identifier	'12' = 18	
Number of Bytes	256 bytes or limited to length of CIOs	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Annex D and Annex E
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

Table 172 – (N2180.00) Characteristics of EF.OD (Object Directory)

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'5031'	
Short File Identifier	'11' = 17	
Number of Bytes	64 bytes or limited to length of CIOs	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Annex D and Annex E
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

Table 173 – (N2181.00) Characteristics of EF.AOD (Authentication Object Directory)

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'5034'	
Short File Identifier	'14' = 20	
Number of Bytes	128 bytes or limited to length of CIOs	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Annex D and Annex E
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

Table 174 – (N2182.00) Characteristics of EF.PrKD (Private Key Directory)

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'5035'	
Short File Identifier	'15' = 21	
Number of Bytes	128 bytes or limited to length of CIOs	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Annex D and Annex E
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

Table 175 – (N2183.00) Characteristics of EF.CD (Certificate Directory)

Attribute	Value	Note
Object type	Transparent Elementary File	
File Identifier	'5038'	
Short File Identifier	'16' = 22	
Number of Bytes	128 bytes or limited to length of CIOs	
Flag Transaction Mode	False	
Flag Checksum	True	
Life Cycle Status	Operational state (activated)	
Content	...	see Annex D and Annex E
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY, UPDATE BINARY	NEVER	

11.4 Application Selection

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

Table 176 – (N2184.00) SELECT command for DF.CIA.QES and DF.CIA.ESIGN

CLA	'00'
INS	'A4' = SELECT
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'0B' or '0F' = Length of subsequent data field
Data field	'E828BD080F D27600006601' = AID of DF.CIA.QES or '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.QES or AID.ESIGN.

Table 177 – (N2185.00) SELECT response

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

11.5 Reading the CIA Files

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

Table 178 – (N2186.00) 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 b5-b1: 10001 SFID of EF. OD: 17 b5-b1: 10100 SFID of EF. AOD: 20 b5-b1: 10101 SFID of EF. PrKD: 21 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 bytes)

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

Table 179 - (N2187.00) SELECT response

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

12 The Organization-specific Authentication Application

The Organization-specific Authentication Application is an optional application. It is left to the discretion of the organization issuing the HPC (professional chamber), whether the application is available to be harnessed after card issuance. The actual use of the application is left to the discretion of the card holder. If the organization-specific authentication application is used, then the content of this chapter is normative.

12.1 File Structure and File Content

DF.AUTO is used for

- organization-specific authentication procedures (e.g. Windows Logon with smart card) which can not make use of the ESIGN application due to technical differences (e.g. proprietary certificate extensions) or an incompatible policy (e.g. mandatory PIN caching).

The general structure of the AUTO application is shown in Figure 9 (N2188.00).

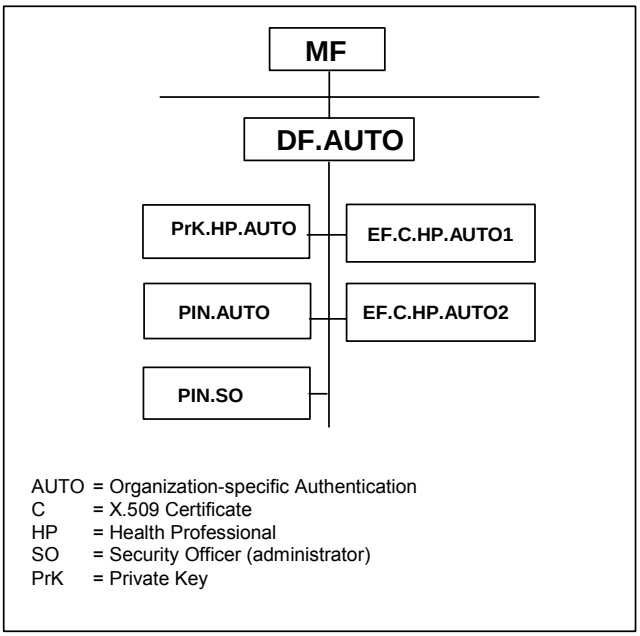

Figure 9 – (N2188.00) General structure of DF.AUTO

12.1.1 DF.AUTO (Organization-specific Authentication Application)

DF.AUTO is an application according to Clause 8.3.1.1 of [HPC-P1], i.e. selectable by using the application identifier. The following table shows the attributes and access rules that apply for the application directory.

Table 180 – (N2189.00) Characteristics of MF / DF.AUTO

Attribute	Value	Note
Object type	Application Directory	
Application Identifier	'D27600014603'	
File Identifier	-	Manufacturer-specific; if supported, then out of interval ['1000', 'FEFF']; see Clause 8.1.1 of [HPC-P1]
Life Cycle Status	Operational state (activated)	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT	ALWAYS	
LOAD APPLICATION, ACTIVATE, DEACTIVATE, DELETE	NEVER	

12.1.2 PrK.HP.AUTO

PrK.HP.AUTO is the private key for client/server authentication. The attributes and access rules of the private key are shown in the following table.

Table 181 – (N2190.00) Characteristics of MF / DF.AUTO / PrK.HP.AUTO

Attribute	Value	Note
Object type	Private RSA Object	
Key Identifier	'02' = 2	To be personalized
Key Reference	'82'	
Private Key	.... (2048 bit)	To be personalized
Key Available	True	
Algorithm Identifier	INTERNAL AUTHENTICATE: rsaClientAuthentication PSO: COMPUTE DIGITAL SIGNATURE: signPKCS1_V1_5 signPSS sign9796_2_DS2	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
GENERATE ASYMMETRIC KEY PAIR	PWD(PIN.SO)	
INTERNAL AUTHENTICATE, PSO: COMPUTE DIGITAL SIGNATURE (P2 = '9E' or 'AC')	PWD(PIN.AUTO)	
Other	NEVER	

Note – PrK.HP.AUTO is a private RSA object, which supports the GENERATE ASYMMETRIC KEY PAIR command (see Clause 8.5.3 of [HPC-P1]). Since the organization-specific certificate information is probably not known to the personalizer, it may be necessary to use this command when the card is already in use in order to enable the generation of certificates.

With respect to key length the same conventions apply as for legally binding electronic signature keys; see [ALGCAT] and [TR-03116].

12.1.3 PIN.AUTO

PIN.AUTO is a DF-specific PIN, which is used only for protection of the private electronic authentication key for organization-specific authentication mechanisms of the health professional (PrK.HP.AUTO). PIN.AUTO shall consist of exactly 5 digits.

The usage of the 8 digit resetting code (personal unblocking key, PUK) is limited by a usage counter with an initial value of 10. The security status of PIN.AUTO can be used an unlimited number of times, i.e. the maximum number of SSEC is set to infinite. The PIN reference as used in the VERIFY,

CHANGE REFERENCE DATA and RESET RETRY COUNTER command and the other PIN attributes are shown in the subsequent table.

Table 182 – (N2191.00) Characteristics of MF / DF.AUTO / PIN.AUTO

Attribute	Value	Note
Object type	Password	
Password Identifier	'01'	
Password Reference	'81'	
Secret	....	To be personalized
Minimum Length	5	
Start Retry Counter	3	
Retry Counter	3	
Transport Status	One of the methods given in Clause 8.2.5 of [HPC-P1]	
Flag Enabled	True	
Start Security Status Evaluation Counter	Infinite	
PUK	...	To be personalized
PUK Usage	10	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
CHANGE RD (Option '00')	ALWAYS	
GET PIN STATUS	ALWAYS	
RESET RC (Option '01')	ALWAYS	
VERIFY	ALWAYS	
Other	NEVER	

The initialization of PIN.AUTO e.g. by using a transport PIN depends on the policy of the responsible organization. If a transport PIN method is used, one of the methods given in Clause 8.2.5 of [HPC-P1] must apply.

12.1.4 PIN.SO

PIN.SO is a DF-specific PIN, which is used for administration purposes of DF.AUTO, i.e. for generating the asymmetric key pair and updating the organization-specific authentication certificates. PIN.SO consists of 6 to 8 digits.

The usage of the 8 digit resetting code (personal unblocking key, PUK) is limited by a usage counter with an initial value of 10. The security status of PIN.SO can be used an unlimited number of times, i.e. the maximum number of SSEC is set to infinite. The PIN reference as used in the VERIFY, CHANGE REFERENCE DATA and RESET RETRY COUNTER command and the other PIN attributes are shown in the subsequent table.

Table 183 – (N2192.00) Characteristics of MF / DF.AUTO / PIN.SO

Attribute	Value	Note
Object type	Password	
Password Identifier	'03'	
Password Reference	'83'	
Secret	....	To be personalized
Minimum Length	6	
Start Retry Counter	3	
Retry Counter	3	
Transport Status	One of the methods given in Clause 8.2.5 of [HPC-P1]	
Flag Enabled	True	
Start Security Status Evaluation Counter	Infinite	
PUK	...	To be personalized

PUK Usage	10	
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
CHANGE RD (Option '00')	ALWAYS	
GET PIN STATUS	ALWAYS	
RESET RC (Option '01')	ALWAYS	
VERIFY	ALWAYS	
Other	NEVER	

According to Clause 14.6.1.4 and Clause 14.6.5.6 of [HPC-P1] the COS checks only the minimum length of PIN.SO, i.e. the COS does not control whether the maximum length of 8 digits is exceeded. The initialization of PIN.SO e.g. by using a transport PIN depends on the policy of the responsible organization. If a transport PIN method is used, one of the methods given in Clause 8.2.5 of [HPC-P1] must apply.

12.1.5 EF.C.HP.AUTO1 and EF.C.HP.AUTO2

EF.C.HP.AUTO1 and EF.C.HP.AUTO2 contain the organization-specific X.509 AUT certificates of the health professional. Therewith, two different identities related to the health professional and associated to the same private key PrK.HP.AUTO are possible. The certificates can be updated after successful authentication of PIN.SO; see Table 184 (N2193.00) and Table 185 (N2194.00).

Table 184 – (N2193.00) Characteristics of MF / DF.AUTO / EF.C.HP.AUTO1

Attribute	**Value**	**Note**
Object type	Transparent Elementary File	
File Identifier	'E001'	
Short File Identifier	'01' = 1	
Number of Bytes	1536 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized.
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
UPDATE BINARY	PWD(PIN.SO)	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

Table 185 – (N2194.00) Characteristics of MF / DF.AUTO / EF.C.HP.AUTO2

Attribute	**Value**	**Note**
Object type	Transparent Elementary File	
File Identifier	'E002'	
Short File Identifier	'02' = 2	
Number of Bytes	1536 or limited to length of certificate	
Flag Transaction Mode	False	
Flag Checksum	False	
Life Cycle Status	Operational state (activated)	
Content	...	To be personalized.
Access Rule in all SEs		
Access Mode	**Security Condition**	**Note**
SELECT, READ BINARY	ALWAYS	
UPDATE BINARY	PWD(PIN.SO)	
ACTIVATE, DEACTIVATE, DELETE, ERASE BINARY	NEVER	

12.2 Security Environments

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

12.3 AUTO Application Selection

The AUTO application selection is performed with the SELECT command.

Table 186 - (N2195.00) 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	‘06’ = Length of subsequent data field
Data field	'D27600014603’ = AID of DF.AUTO
Le	Absent

Table 187 - (N2196.00) SELECT Response

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

After DF selection, SE # 1 is implicitly selected.

12.4 PIN Management

12.4.1 PIN Verification

Prior to the usage of PrK.HP.AUTO, PIN.AUTO has to be successfully presented. For updating the certificate files PIN.SO has to be successfully presented. For PIN verification the ISO/IEC 7816-4 command VERIFY is used.

Table 188 - (N2197.00) VERIFY command for PIN verification

CLA	As defined in ISO/IEC 7816-4
INS	‘20’ = VERIFY
P1	‘00’
P2	‘81’ = PIN.AUTO reference or ‘83’ = PIN.SO reference
Lc	‘08’ = Length of subsequent data field
Data field	PIN (Format: see Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 189 - (N2198.00) 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 bytes as specified in Clause 14.6.6.3 of [HPC-P1].

12.4.2 PIN Change

For PIN.AUTO or PIN.SO change the ISO/IEC 7816-4 command CHANGE REFERENCE DATA may be used at any time convenient for the HP. The command can only be used with P1 = '00', i.e. with both old and new PIN in the data field. The successful verification of the old PIN does not set the security status of the referenced PIN object; see Clause 14.6.1.4 of [HPC-P1].

Table 190 - (N2199.00) CHANGE REFERENCE DATA command

CLA	As defined in ISO/IEC 7816-4
INS	'24' = CHANGE REFERENCE DATA
P1	'00' = Exchange reference data
P2	'81' = PIN.AUTO reference or '83' = PIN.SO reference
Lc	'10' = Length of subsequent data field
Data field	PIN_old || PIN_new (PIN format: see Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 191 - (N2200.00) CHANGE REFERENCE DATA response

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

12.4.3 Reset of Retry Counter

For resetting the retry counter to its initial value and – if used in this way – setting a new PIN, the ISO/IEC 7816-4 RESET RETRY COUNTER command is used. The successful verification of the re-setting code does not set the security status of the referenced PIN object; see Clause 14.6.5.6 of [HPC-P1].

Table 192 – (N2201.00) RESET RETRY COUNTER command for RC reset

CLA	As defined in ISO/IEC 7816-4
INS	'2C' = RESET RETRY COUNTER
P1	'00' or '01'
P2	- '81' = Reference of PIN.AUTO, to which the resetting code ("PUK") according to ISO/IEC 7816-4 belongs or - '83' = Reference of PIN.SO, to which the resetting code ("PUK") according to ISO/IEC 7816-4 belongs
Lc	'10' or '08' = Length of subsequent data field
Data field	- If P1 = '00': Resetting code (8 bytes) followed by a new PIN (8 bytes) - If P1 = '01': Resetting code ("PUK", Format: see Clause 8.1.7 of [HPC-P1])
Le	Absent

Table 193 - (N2202.00) RESET RETRY COUNTER response

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

NOTE – The command may also be used to lift the PIN transport status; see Clause 14.6.5.1 and Clause 14.6.5.3 of [HPC-P1].

12.5 Reading the Organization-specific Certificates

For reading the certificates EF.C.HP.AUTO1 and EF.C.HP.AUTO2, the READ BINARY command is used.

Table 194 - (N2203.00) 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.AUTO1: 1 00010 SFID of EF.C.HP.AUTO2: 2 P2 = Offset - ‘xxxx’ = Offset (bit b8 of P1 = 0)
Lc	Absent
Data field	Absent
Le	'000000'

Table 195 - (N2204.00) READ BINARY response

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

If the supported extended length is not sufficient to read the whole certificate with a single command, the READ BINARY command shall be repeated specifying the respective offset in P1-P2.

12.6 Writing of Organization-specific Certificates

For updating organization-specific certificates (see access conditions in Table 184 (N2193.00) and Table 185 (N2194.00)), the ISO/IEC 7816-4 command UPDATE BINARY shall be used.

Table 196 - (N2205.00) UPDATE BINARY command for writing a new certificate

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.AUTO1: 1 00010 SFID of EF.C.HP.AUTO2: 2 P2 = Offset - 'xxxx' = Offset (bit b8 of P1 = 0)
Lc	'00xxxx' = Length of subsequent data field
Data field	Data
Le	Absent

Table 197 - (N2206.00) UPDATE BINARY response

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

If the supported extended length is not sufficient to write the whole certificate with a single command, then the UPDATE BINARY command shall be repeated specifying the respective offset in P1-P2.

12.7 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.AUTO, PuK.HP.AUTO) 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.

The card computes the authentication related data 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 and the algorithm identifier denoting the type of authentication protocol have to be set.

Table 198 – (N2207.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for internal authentication
P2	'A4' = Authentication Template
Lc	'06' = Length of subsequent data field
Data field	'84 01 82' || '80 01 05' = DO KeyRef of PrK.HP.AUTO || DO AlgID rsaClientAuthentication
Le	Absent

Table 199 - (N2208.00) MSE response

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

Table 200 – (N2209.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	‘88’ = INTERNAL AUTHENTICATE
P1	‘00’
P2	‘00’
Lc	‘00xxxx’ = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	‘0100’ = length of expected digital signature = 256

Table 201 – (N2210.00) INTERNAL AUTHENTICATE response

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

PrK.HP.AUTO may also be used with the COMPUTE DIGITAL SIGNATURE command. The private authentication key and the algorithm identifier denoting the type of authentication protocol have to be set before.

Table 202 - (N2211.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for computation
P2	‘B6’ = Digital Signature Template
Lc	‘06’ = Length of subsequent data field
Data field	‘84 01 82’ || ’80 01 yy’ = DO KeyRef of PrK.HP.AUTO || DO AlgID ‘yy’ = ‘02’: signPKCS1_V1_5 ‘yy’ = ‘05’: signPSS ‘yy’ = ‘07’: sign9796_2_DS2
Le	Absent

Table 203 - (N2212.00) MSE response

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

Table 204 - (N2213.00) PSO: COMPUTE DIGITAL SIGNATURE 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 contains data to be signed or integrated in the DSI for PKCS#1 SSA-V1.5 or PKCS#1 PSS - 'AC' = Data field contains data objects with data elements to be signed or integrated in the DSI for ISO9796-2 DS2
Lc	'00xxxx' = Length of subsequent data field
Data field	- DSI according to ISO9796-2 DS2 or - DSI according to PKCS#1 SSA-V1.5 or - DSI according to PKCS#1 PSS
Le	'0100' = length of expected digital signature = 256

Table 205 - (N2214.00) PSO: COMPUTE DIGITAL SIGNATURE response

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

13 Loading a new Application or Creation of an EF after Issuing of the HPC

13.1 Identification of the HPC COS Platform

It is assumed that the loading of a new application or the creation of a new EF on MF level or DF.HPA after issuing of the HPC is performed by a Card Application Management System (CAMS). This is an optional procedure. Similarly a CAMS is optional. The contents of this chapter are however normative, if loading of a new application or creation of a new EF after issuing are to be performed.

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, EF.DIR and EF.Version.

13.2 Establishing a Trusted Channel between CAMS and HPC

A trusted channel may be achieved with a symmetric or an asymmetric authentication procedure depending on the involved CAMS. Both CAMS authentication procedures are allowed and considered equivalent. 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 an HPC, an asymmetric authentication procedure with SM key computation has to be performed; see the following section. Since the PuK.CAMS_ HPC.AUT_CVC 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 10 (N2600.00).

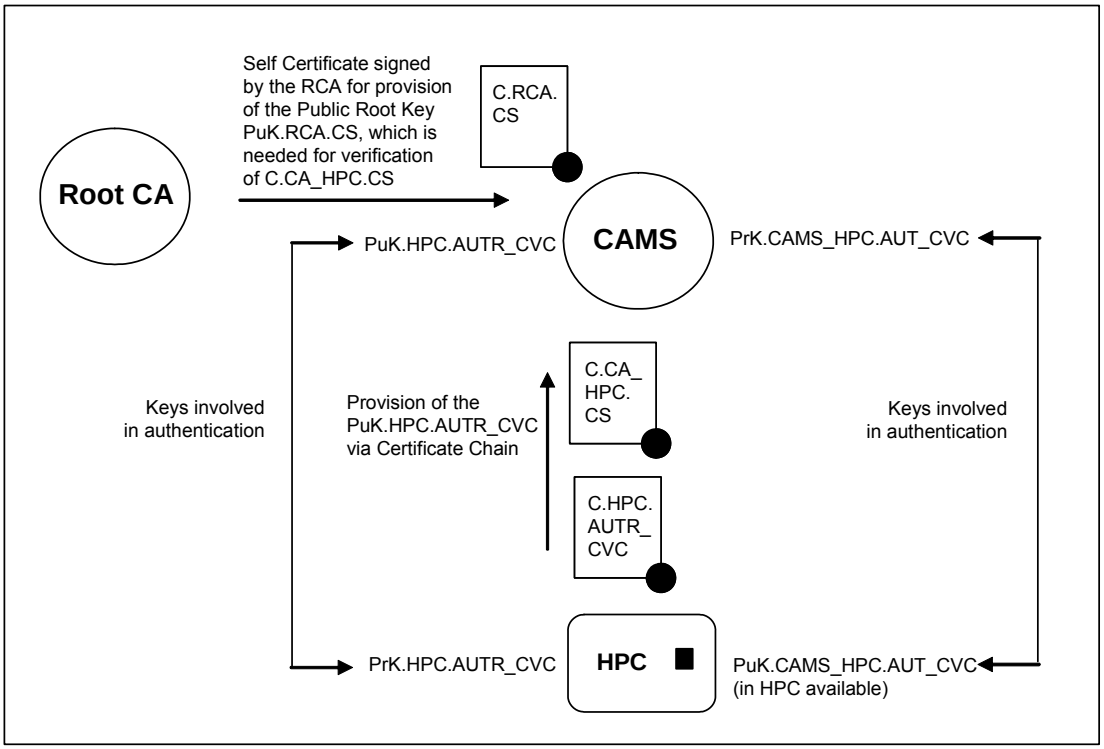

Figure 10 – (N2600.00) Keys and key import for HPC/CAMS authentication with TC establishment

13.2.1 Performing the asymmetric authentication

In the first part of the procedure the HPC proves its authenticity to the CAMS. The external software requests a challenge from the CAMS prior to sending the command for internal authentication to the HPC. In the HPC the keys and algorithms of both authentication commands are selected first.

NOTE – Both MSE SET commands for internal and external authentication are performed successively, so that the sequence of authentication commands will not be interrupted by a MSE SET command; see Chapter 15 of [HPC-P1].

Table 206 - (N2601.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘41’ = SET for internal authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘06’ = Length of subsequent data field
Data field	’84 01 10’ || ‘80 01 54' = DO KeyRef of PrK.HPC.AUTR_CVC || DO AlgID rsaSessionkey4SM
Le	Absent

Table 207 - (N2602.00) MSE response

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

Table 208 - (N2603.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for external authentication
P2	‘A4’ = Authentication Template in data field
Lc	‘11’ = Length of subsequent data field
Data field	‘83 0C 00000000 00000000 00000013’ || ‘80 01 54’ = DO KeyRef of PuK.CAMS_HPC.AUT_CVC || DO AlgID rsaSessionkey4SM
Le	Absent

Table 209 - (N2604.00) MSE response

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

Table 210 – (N2605.00) INTERNAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	'88' = INTERNAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'000010' = Length of subsequent data field
Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
Le	'0100' = Length of expected signature

Table 211 – (N2606.00) INTERNAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.4 of [HPC-P1]
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 CAMS. The second part of the procedure comprises the external authentication of the CAMS. First a challenge is retrieved from the HPC.

Table 212 - (N2607.00) 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' = Length of expected random number = 8

Table 213 - (N2608.00) GET CHALLENGE response

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

The challenge is presented to the CAMS together with the 8 LSB of HPC's ICC Serial number. The resulting digital signature of the internal authentication performed by the CAMS is presented to the HPC in the EXTERNAL AUTHENTICATE command. The HPC has to verify the CAMS signature.

Table 214 – (N2609.00) EXTERNAL AUTHENTICATE command for CAMS authentication

CLA	As defined in ISO/IEC 7816-4
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'000100' = Length of subsequent data field = 256
Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
Le	Absent

Table 215 – (N2610.00) EXTERNAL AUTHENTICATE response

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

The authentication related data contain data elements for key computation that is performed as described in Clause 13.1 and 14.7.1 of [HPC-P1]. A successful verification sets in the HPC the security status "Private key of CAMS successfully presented". Then, a trusted channel has been established, i.e. new application data can be transferred to the HPC in secure messaging mode.

13.2.2 Performing the symmetric authentication

If the symmetric key SK.CAMS is available in the card, the symmetric authentication procedure between HPC and CAMS must be performed as follows. The command sequence starts with setting the symmetric key and the appropriate algorithm for mutual authentication.

Table 216 - (N2611.00) MSE command for key and algorithm selection

CLA	As defined in ISO/IEC 7816-4
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'81' = SET for mutual authentication
P2	'A4' = Authentication Template in data field
Lc	'06' = Length of subsequent data field = 6
Data field	'83 01 13' || '80 01 54' = DO KeyRef of SK.CAMS || DO AlgID desSessionkey4SM
Le	Absent

Table 217 - (N2612.00) 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 218 - (N2613.00) 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' = Length of expected random number = 8

Table 219 - (N2614.00) GET CHALLENGE response

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

Then, an INTERNAL AUTHENTICATE command with RND.HPC || ICCSN8.HPC in the data field is sent to the CAMS. A MUTUAL AUTHENTICATE command follows on the side of the HPC. This command delivers the authentication related data of the CAMS to the HPC. The HPC verifies the data of the CAMS and computes the HPC authentication related data.

Table 220 – (N2615.00) MUTUAL AUTHENTICATE command

CLA	As defined in ISO/IEC 7816-4
INS	'82' = MUTUAL AUTHENTICATE
P1	'00' = Algorithm already known to the card
P2	'00' = Key reference already known to the card
Lc	'68' = Length of subsequent data field = 104
Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
Le	'68' = Length of expected authentication related data = 104

Table 221 - (N2616.00) MUTUAL AUTHENTICATE response

Data field	Authentication related data; see Clause 14.7.1 of [HPC-P1]
SW1-SW2	'9000' or specific status bytes; see [HPC-P1]

A successful verification sets in the HPC the required security status. The HPC authentication related data (content of the response data field) is verified in the related CAMS during an EXTERNAL AUTHENTICATE command. Finally, a trusted channel is established that enables a data transmission with secure messaging to the HPC.

13.3 Loading a new Application or a new File

Relevant for the loading process is the LOAD APPLICATION command specified in Clause 14.2.5 of [HPC-P1]. If a single command is not sufficient to transfer the whole application or file data, command chaining may be used.

Table 222 – (N2617.00) LOAD APPLICATION command for loading a new application or file

CLA	As defined in ISO/IEC 7816-4
INS	'EA' = LOAD APPLICATION
P1	'00'
P2	'00'
Lc	'xx'or 00xxxx' = Length of subsequent data field
Data field	New application related data
Le	Absent

Table 223 – (N2618.00) LOAD APPLICATION response

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

The LOAD APPLICATION command can also be used to create a new EF; see Clause 14.2.2 and Clause 14.2.5 of [HPC-P1]. If a new access rule for a created EF applies, then this access rule has to be integrated into the COS.

13.4 Registering the new Application in EF.DIR

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

Table 224 - (N2619.00) 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 9 (N2010.00)
Le	Absent

Table 225 – (N2620.00) APPEND RECORD response

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

Annex A (normative)

Issuer Identifier, Number spaces of Certification Authorities, and Certificate Holder Authorizations

A.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 [ISO3166]. The issuer identifier (“Kartenausgeberschlüssel”) is assigned in Germany on behalf of DIN by GS1 Germany GmbH (www.gs1-germany.de). 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 backside 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 [EN1867] . 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 value ‘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 some issuer identifiers used in German health care for Health Professional Cards (HPC) and Security Module Cards (SMC). The complete list of registered issuers is not publicly available but may be requested from GS1 Germany.

Table 226 - (N2621.00) 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’ Bundeszahnärztekammer
		’00279’ Kassenzahnärztliche Bundesvereinigung
		'00xxx' Landeskrankenhausgesellschaften (DKTIG)

A.2 Number spaces of Certification Authorities

Each certification authority is assigned an individual serial number space, in order to ensure that the ICCSN as well as the attribute serialNumber of the SubjectDN (part of the X.509 certificate Subject field) are unique accross all CAs. For both purposes, the leading two digits of the serial number are

used as a prefix to indicate the certification authority that ensures the uniqueness of the remaining digits. The CA prefixes are shown in Table 227 (N2622.00).

Table 227 – (N2622.00) CA prefixes for serial number spaces

Prefix	Certification Authority
10	D-TRUST
11	Signtrust
12	T-Systems Telesec
13	S-Trust
14	TC TrustCenter
15	DGN
16	medisign

The following examples are ICCSNs of cards issued by Bundesärtzekammer (issuer identifier '00108') with serial numbers assigned by D-TRUST and Signtrust:

'80' '276' '00108' '10 12345678' ICCSN with serial number of D-TRUST
'80' '276' '00108' '11 12345678' ICCSN with serial number of Signtrust

A.3 Certificate Holder Authorizations and Profiles

The Certificate Holder Authorization (CHA) is present in a card verifiable certificate of each card, i.e. HPC, SMC-A, SMC-B and other cards of the German health care. It is used especially in access rules related to cryprographic keys and data files so that for a given access mode (e.g. compute digital signature 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. an HPC of a specific health professional group) 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 in the HPC, so that the authentication procedure is legitimated.

The CHA consists of 7 bytes including an application identifier (AID, first 6 bytes) and a role identifier (Role ID, last 1 byte). The AID consists of the Registered Application Provider Identifier (RID, first 5 bytes) and a Proprietary Application Provider Extension (PIX). For CHAs of the German health care, the application identifier 'D27600004000' is used, which is worldwide unique. The RID 'D276000040' has been assigned by the RID German National Registration Authority (now Fraunhofer Institute SIT on behalf of DIN) to ZI (Zentralinstitut für die kassenärztliche Versorgung in der Bundesrepublik Deutschland) for usage in German health care. The PIX of the AID is set to '00'.
The Role ID encodes the actual authorization of the certificate holder, i.e. a profile that implies specific authorizations to cryptographic keys or access rights to data files of other cards. The profile definition as well as the mapping of profiles to specific professional groups is out of scope of this specification. Two groups of profiles are defined for the German health care (see [PKI-Reg] for further details):

- Profiles for the role authentication of professional groups (e.g. physicians and pharmacists), personnel of professional groups (e.g. personnel of physicians and pharmacists) and the institutions (e.g. practices, pharmacies and hospitals). If several professional groups have the same access rights to specific card data, then they may be assigned the same profile. This leads to smaller sized access rules in the cards. Currently, ten profiles (0 - 9) for role authentication are defined for the german health care; see Table 228 (N2623.00)

- Profiles for the device authentication of specific cards and functional units (e.g. PIN receiving units and signature cards). Currently, five profiles (51 - 55) for device authentication are defined for the german health care; see Table 229 (N2624.00).

Table 228 – (N2623.00) CHAs for Role Authentication of Persons / Organizations (informative)

CHA Prefix	CHA Role ID	Profile of CV Certificate holder	Certification authority	Access Condition of Authentication key (e.g. Authentication with CHA of related card)
'D27600004000'	'00'	Profile 0 for role authentication of the insurant without access right to protected eGK data (eGK)	CA_eGK	eGK: ALWAYS
'D27600004000'	'01'	Profile 1 for role authentication of the eKiosk as environment of exercising the insurant's rights (SMC-B)	CA_eKiosk	SMC-B: tbd
'D27600004000'	'02'	Profile 2 for role authentication of - the physician / dentist in an institution, e.g. own practice, group practice or hospital (HPC), - the personnel of a medical institution (e.g. medical practice or hospital) (HPC or Professional Card), - the personnel of a medical institution (e.g. medical practice or hospital) with authorization and logging according to § 291a Abs. 5 Satz 4 SGB V (SMC-A, SMC-B). The personnel of medical institution represent the institution of the physician/ dentist towards the telematics infrastructure.	CA_HPC CA_SMC	HPC, Professional Card: PWD(PIN.CH) SMC-A, SMC-B in practice: AUT('D27600004000' || '02') SMC-A, SMC-B in hospital: AUT('D27600004000' || '02') OR AUT('D27600004000' || '03') OR AUT('D27600004000' || '04') OR AUT('D27600004000' || '05') See Note 2
'D27600004000'	'03'	Profile 3 for role authentication of - the pharmacist in a public pharmacy or hospital pharmacy, each based in Germany (HPC), - the personnel of a pharmacy as professional assistant or person on vocational preparation according to § 291a Abs. 5 Satz 4 SGB V (HPC or Professional Card) - the personnel of a pharmacy with authorization and logging according to § 291a Abs. 5 Satz 4 SGB V (SMC-A, SMC-B). The personnel of pharmacy represent the institution of the pharmacist towards the telematics infrastructure.	CA_HPC CA_SMC	HPC, Professional Card: PWD(PIN.CH) SMC-A and SMC-B: AUT('D27600004000' || '03')
'D27600004000'	'04'	Profile 4 for role authentication of - the psychological psychotherapist, child psychotherapist (the medical psychotherapist uses profile 2) (HPC), - the personnel of a psychological practice tbd (SMC-A, SMC-B)	CA_HPC CA_SMC	HPC: PWD(PIN.CH) SMC-A, SMC-B: AUT('D27600004000' || '04')
'D27600004000'	'05'	Profile 5 for role authentication of - the professional for therapeutic measures (HPC or Professional Card) - the professional for treatment aids (Professional Card)	tbd	HPC, Professional Card: PWD(PIN.CH)
'D27600004000'	'06'	Profile 6 for role authentication of the entity tbd (SMC-A or SMC-B)	tbd	SMC-A or SMC-B: tbd
'D27600004000'	'07'	Profile 7 for role authentication of the medical personnel working in an ambulance service (Emergency Medical Technician, parame-	tbd	HPC, Professional Card: PWD(PIN.CH)

		dic) (HPC or Professional Card). These are members of a further health profession, which have graduated in a registered professional training for practising the profession or holding the occupational title (§ 291a Abs. 4 Satz 1 Nr. 2 e).		
‘D27600004000’	‘08’	Profile 8 for role authentication of - the personnel of health institutions without an own HPC or Professional Card (SMC-A, SMC-B) - the personnel of health insurances (SMC-A, SMC-B)	tbd	SMC-A, SMC-B: tbd
‘D27600004000’	‘09’	Profile 9 for role authentication of - the personnel of health institutions without an own HPC or Professional Card (SMC-A, SMC-B) - the secure operational environment for the insurant (SMC-A, SMC-B)	tbd	SMC-A, SMC-B: tbd
‘D27600004000’	‘0A’ – ‘32’	Profiles 10 – 50: Reserved for future use	tbd	tbd

NOTE 1 – Many or all CAs in Table 228 (N2623.00) may be identical. All CV certificates of an HPC or SMC are derived from the same CV Root Certification Authority.

NOTE 2 – An SMC in a hospital has the same access rights to an eGK as an SMC in a physician or dentist practice, but the SMC in a practice can only be authorized (irrespective of PIN.SMC entry) by a physician’s or dentist’s HPC whereas several health professional profiles may have the right to authorize an SMC in a hospital. The denoted profiles are the maximum set of permitted profiles. The respective institution may reduce this set to a subset that is allowed to authorize a specific SMC.

Table 229 – (N2624.00) CHAs for Device Authentication of Functional Units (informative)

CHA Prefix	CHA Role ID	Profile of CV Certificate holder	Certification authority	Access Condition of Authentication key (e.g. Authentication with CHA of related card)
‘D27600004000’	‘33’	Profile 51 for device authentication of secure signature environment of SAK (SMC-K)	CA_SAK	SMC-K: AUT(‘D27600004000’ || ‘35’)
‘D27600004000’	‘34’	Profile 52 for device authentication of comfort signature trigger (RFID Token or Biometric Device)	CA_KM	RFID-Token: PWD(PIN.CH) or ALWAYS Biometric device: PWD(Biometric feature)
‘D27600004000’	‘35’	Profile 53 for device authentication of SSCD capable of stack and comfort signature (HPC)	CA_HPC	HPC: ALWAYS
‘D27600004000’	‘36’	Profile 54 for device authentication of remote PIN sender (SMC-A, SMC-B)	CA_SMC	SMC-A and SMC-B: AUT(‘D27600004000’ || ‘35’) OR AUT(‘D27600004000’ || ‘37’)
‘D27600004000’	‘37’	Profile 55 for device authentication of remote PIN receiver (SMC-B, RFID Token)	CA_SMC CA_KM	SMC-A, SMC-B: ALWAYS

NOTE – Many or all CAs in Table 229 (N2624.00) may be identical. All CV certificates of an HPC or SMC are derived from the same CV Root Certification Authority.

The HPC owns two CV certificates:

- C.HPC.AUTR_CVC for the role authentication (see Clause 4.3.7) with a profile of the card holder as associated member of a professional group, and

- C.HPC.AUTD_SUK_CVC for device authentication (see Clause 4.3.8) with profile 53 for SSCD capable of stack and comfort signature. This profile is used for receiving the data to be signed as well as the PIN data.

The SMC-A owns two CV certificates:

- C.SMC.AUTR_CVC for role authentication (see Clause 5.3.7 in [HPC-P3]) with a profile of the personnel or institution and

- C.SMC.AUTD_RPS_CVC for device authentication (see Clause 5.3.8 in [HPC-P3]) with profile 54 for sending PIN data.

The SMC-B owns three CV certificates:

- C.SMC.AUTR_CVC for role authentication (see Clause 6.3.7 in [HPC-P3]) with a profile of the personnel or institution,

- C.SMC. AUTD_RPS_CVC for device authentication (see Clause 6.3.8 in [HPC-P3]) with profile 54 for sending PIN data, and

- C.SMC.AUTD_RPE_CVC for device authentication (see Clause 6.3.9 in [HPC-P3]) with profile 55 for receiving PIN data.

Annex B (normative)

Structure and Content of QES Certificates

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

B.1.1 General Aspects

The coding scheme of HP QES-certificates shall comply with

- standardized coding schemes for certificates (X.509v3),
- legal requirements (the German digital signature act SigG and the associated signature ordinance SigV), and
- the Common PKI specification for interoperable PKI applications, especially Part 1 "Certificate and CRL Profiles" [COM-PKI-1], and its optional profile "SigG-Profile" [COM-PKI-9].

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 Common PKI specification for certificates. The attribute certificates are described in Clause B.7.

B.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 }
```

B.2 ToBeSignedCertificate

B.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.

B.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.

B.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 [COM-PKI-1] 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 [COM-PKI-1] and have to be taken into account for HP QES-certificates.

B.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 B.3).

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

```
Algorithmidentifier         ::=   SEQUENCE {
```

```
    algorithm                       OBJECT IDENTIFIER,
    parameters                      ANY DEFINED BY algorithm OPTIONAL }
```

B.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 [RFC3629], 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. [COM-PKI-1] 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 Table 7 of [COM-PKI-1].

B.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 [ISO3166] „DE“.

B.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))
```

B.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.

B.2.7 Subject

The *subject* field identifies the certificate holder in the sense of technical identification (the juridical identification can be provided according to Common PKI specification in the *subjectDirectoryAttributes* extension; see B.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 Common PKI specification or relevant RFCs is optional.

The coding scheme for these *subject* attribute types in HP QES-certificates is defined in Table 7 of [COM-PKI-1].

B.2.7.1 Country name

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

B.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 B.2.5).

B.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 B.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 B.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.

B.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))
```

B.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 in the header of 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 algorithms of the SHA-2 family (especially SHA-256)
- the padding formats, i.e. the DSI formats PKCS#1 SSA-V1.5, PKCS#1 PSS and ISO/IEC 9796-2 DS2.

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 can be compared with the hash value as part of the DSI. The hash algorithm indication is supported by PKCS#1 SSA-V1.5 (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 9796-2 DS2, if option 2 is used (with trailer of 2 bytes comprising a hash-function identifier and 'CC'). It is neither provided in the DSI-format ISO/IEC 9796-2 DS2 using option 1 (with trailer 'BC') nor in the DSI-format of PKCS#1 PSS. Since the HPC supports all three signature algorithms, the OID has to denote RSA and the hash algorithm SHA-256.

B.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. Empty extensions shall not be set.

```
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 [COM-PKI-9]
- 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 branch

```
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 branch

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

Common PKI SigG Private X.509 Extensions:

- admission
- restriction
- additional information

Common PKI SigG private X.509 certificate extensions are identified by the initial object identifier branch

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

BNA Private X.509 Extensions:

- validity model.

B.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 }
```

B.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.

B.2.9.3 Certificate Policies

This field specifies the certificate policies that were applied when issuing the certificate. The extension MUST be present, and SHOULD NOT be marked critical. The values have to be set complying with

[COM-PKI-1]. 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
 -- commonpki branch for certificate policies
id-commonpki-cp OBJECT IDENTIFIER                    ::= { id-commonpki 1  }
id-commonpki-cp-accredited OBJECT IDENTIFIER   ::= { id-commonpki-cp 1 }
```

B.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"

B.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 branch

```
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 [COM-PKI-1], that contains the value of the card holder's name at his/her birth.

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

B.2.9.6 Authority Key Identifier

This field is mandatory according to [COM-PKI-1], and denotes the reference to the PK of the CA to be applied for verification of the CA signature. This extension MUST NOT be marked critical.

ASN.1-Definition of Type *AuthorityKeyIdentifier*:

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

Notes:
Both identification methods, described in [RFC2459] MAY be used in the same certificate. [COM-PKI-1] requires that the *keyIdentifier* field MUST contain exactly the same ID as the *subjectKeyIdentifier* of the CA certificate (see B.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.

B.2.9.7 Subject Key Identifier

This field denotes a reference to the PK of the certificate holder and is optional in the Common PKI specification, 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
```

B.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 in the *accessMethod* component to indicate the OCSP-service. The OCSP-URL is specified in the *accessLocation* field.

B.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 }
```

B.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 B.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.

B.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 [COM-PKI-1], if indirect CRLs are issued, and it should be present if direct CRLs are issued.

B.2.9.12 Professional Admission

Professional admission information can be expressed either in the base QES certificate by means of the [COM-PKI-9] 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 [COM-PKI-9]. 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-commonpki-at-admission OBJECT IDENTIFIER              ::= { commonpki-at 3 }
id-commonpki-at-namingAuthoritiesOBJECT IDENTIFIER       ::= { commonpki-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* support 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 B.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.

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, the professionOID and the registrationNumber must always be present, whereas the other indications, i.e. namingAuthority and addProfessionInfo, are optional.In encryption and authentication certificates the field *registrationNumber* contains a unique number („Telematik-ID“) used to identify the card holder and correlate his certificates over card generations. It is recommended (subject to the decision of the cardholder), that this number remains stable. A registration number must not be assigned to more than one person. For this reason a prefix must be used to ensure, that the issuing organisations are using distinct number ranges; see Table 230 (N2625.00). The prefix is separated from the actual number with a minus („-“) sign (ASCII 45). For example the Telematik-ID of a physician could be 1-12100123456789012; see [TID]. The component *addProfessionInfo* may contain additional application-specific information in DER-encoded form.

Table 230 – (N2625.00) Prefixes of registration numbers

Prefix	Meaning
1	Registered physicians (Ärzteschaft)
2	Registered dentists (Zahnärzteschaft)
3	Registered pharmacists (Apothekerschaft)
4	Registered psychotherapists (Psychotherapeutenschaft)
5	Hospital (Krankenhaus)
6	Other health care sectors or other health professionals and related institutions

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. The field *professionOIDs* 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 be 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, if this information is not provided in an attribute certificate; see Note. Then, 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 alternatively (i.e. instead of being provided in the base QES certificate) be provided in an attribute certificate for qualifications (see B.7.1), and in an attribute certificate for authorizations (see B.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 B.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" } } } } } }
```

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

The encoding of the DirectoryString shall be UTF-8.

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

The encoding of the DirectoryString shall be UTF-8.

B.3 Signature algorithm

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

B.4 Signature

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

B.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.

Table 231 – (N2626.00) Content of the QES-certificate of the health professional

Certificate Field	Content	Common PKI, reference	This Spec., Clause	This Spec., Usage
version	X.509v3	T2 #2 #12	B.2.2	mandatory
serialNumber	Serial number of certificate	T2 #13	B.2.3	mandatory
signature	Algorithm identifier for CA signature	T2 #4, T4	B.2.4	mandatory
issuer	Certification authority	T2 #5, T5	B.2.5	mandatory
countryName	DE for Germany	T7 #13	B.2.5.1	mandatory
organizationName	e.g. DEZGW for German CA in health care	T7 #6	B.2.5.2	mandatory
validity	Validation period	T2 #6, T3	B.2.6	mandatory
notBefore	UTC Time	T3 #2	B.2.6	mandatory
notAfter	UTC Time	T3 #3	B.2.6	mandatory
subject	Certificate holder	T2 #7, T5	B.2.7	mandatory
countryName	DE for Germany	T7 #13	B.2.7.1	mandatory
stateOrProvince	Name of German state	T7 #12	B.2.7.2	optional
commonName (CN)	Common name in its full form	T7 #1	B.2.7.3	mandatory
serialNumber	Serial number for unique naming of subject	T7 #4	B.2.7.4	optional
subjectPublicKeyInfo	PK data	T2 #8 #14	B.2.8	mandatory
algorithm	OID of algorithm incl. parameters if any	T2 #15	B.2.8	mandatory
subjectPublicKey	Coding of PK with modulus and publicExponent	T2 #16	B.2.8	mandatory
extensions	Extensions	T2 #11, T9	B.2.9	mandatory
basicConstraints	Classification as end user certificate	T10, T18	B.2.9.1	optional
keyUsage	nonRepudiation, i.e. usage of certificate restricted to digital signatures according to SigG requirement	T10, T12	B.2.9.2	mandatory
certificatePolicies	Indication of SigG Conformance	T10, T14	B.2.9.3	optional
subjectAltName	alternative technical name with possibly e-Mail address of certificate holder	T10, T16 #1	B.2.9.4	optional
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	T10, T17	B.2.9.5	optional
authorityKeyIdentifier	Reference to PK of the CA for verification of the CA signature	T10, T11	B.2.9.6	mandatory
subjectKeyIdentifier	Reference to PK of the certificate holder	T10, T11	B.2.9.7	optional
authorityInfoAccess	Identification of how and where information about the status of the certificate can be obtained	T10, T13, T23	B.2.9.8	optional
validityModel	Information about the mechanisms that must be used in order to prove the validity of the certificate	-	B.2.9.9	optional
qcStatements	Indication that the certificate is a qualified certificate, and where applicable monetary limit of transactions	T10, T25	B.2.9.10	mandatory
cRLDistributionPoints	Identification of how CRL information can be obtained	T10, T22	B.2.9.11	optional
admission	Professional admission information	T29, T29b	B.2.9.12	mandatory
restriction	Other restrictions regarding the usage of the certificate	T29, T29e	B.2.9.13	optional
additionalInformation	Non-restrictive information regarding the usage of the certificate	T29, T29f	B.2.9.14	optional
signatureAlgorithm	Algorithm identifier for CA signature (Value identical to signature field in Certificate Content)	T1 #3, T4	B.3	mandatory
signature	Signature of CA	T1 #4	B.4	mandatory

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

```
Certificate DEFINITIONS IMPLICIT TAGS ::=
-- IMPLICIT TAGS: Since all objects are tagged, the tags of universal data types are omitted

BEGIN

IMPORTS Controls, BasicLatin, Latin-1Supplement FROM ASN.1-CHARACTER-MODULE
        {joint-iso-itu-t asn1(1) specification(0) modules(0) iso10646(0)}
    -- Further character sets of [ISO10646] may be imported in order to form an internationally
    -- usable character subset of the [ISO10646] Universal Multiple-Octet Coded Character Set (UCS).


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

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

```
        encipherOnly                    (7),
        decipherOnly                    (8) }

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

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

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

CertPolicyId                    ::=     OBJECT IDENTIFIER

-- subject alternative name extension
SubjectAltName                  ::=     GeneralNames

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

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

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

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

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

-- subject directory attributes extension
SubjectDirectoryAttributes      ::=     Attributes

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

DateOfBirth                     ::=     GeneralizedTime

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

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

```
CountryOfCitizenship            ::=   PrintableString (SIZE(2)) – ISO 3166

CountryOfResidence              ::=   PrintableString (SIZE(2)) – ISO 3166
NameAtBirth                     ::=   DirectoryString (SIZE(1..64))

-- authority key identifier extension
AuthorityKeyIdentifier          ::=   SEQUENCE {
    keyIdentifier                     [0] 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 }

-- Common PKI 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))

END
```

B.6 Object Identifiers Used in this Specification

The subsequent table shows the relevant object identifiers.

Table 232 – (N2627.00) Overview of OIDs

Object Identifier		**Meaning**
Name	Value	
id-at	{ 2 5 4 }	branch for X.500 attributes
id-at-title	{ 2 5 4 12 }	title
id-ce	{ 2 5 29 }	branch 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 }	branch for pkix
id-pe	{ id-pkix 1 }	branch 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 }	branch 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-commonpki	{ 1 3 36 8 }	branch for Common PKI
id-qcs	{ id-pkix 11 }	branch for QC statements
id-qcs-pkixQCSyntax-v1	{ id-qcs 1 }	RFC3039 QC statement
id-commonpki-at	{ id-commonpki 3 }	branch for Common PKI attributes and extensions
id-commonpki-at-admission	{ id-commonpki-at 3 }	Indicates professional admission information
id-commonpki-at-restriction	{ id-commonpki-at 8 }	Indicates other (non-monetary) restriction on certificate usage
id-commonpki-at-namingAuthorities	{ id-commonpki-at 11 }	branch for naming authorities for professions
id-commonpki-at-nameAtBirth	{ id-commonpki-at 14 }	name at birth
id-commonpki-at-additionalInformation	{ id-commonpki-at 15 }	Indicates non-restricitve information about certificate usage
id-commonpki-cp	{ id-commonpki 1 }	branch for Common PKI certificate policies
id-commonpki-cp-accredited	{ id-commonpki-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 }	branch 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 }	branch 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

B.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. Only v1 attribute certificates conforming to the former Common PKI v1.1 specification are currently used.

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 [COMPKI-9]. 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 B.2.9.12.
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 extensions 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.

B.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.

B.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" } } } } }
```

B. 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.

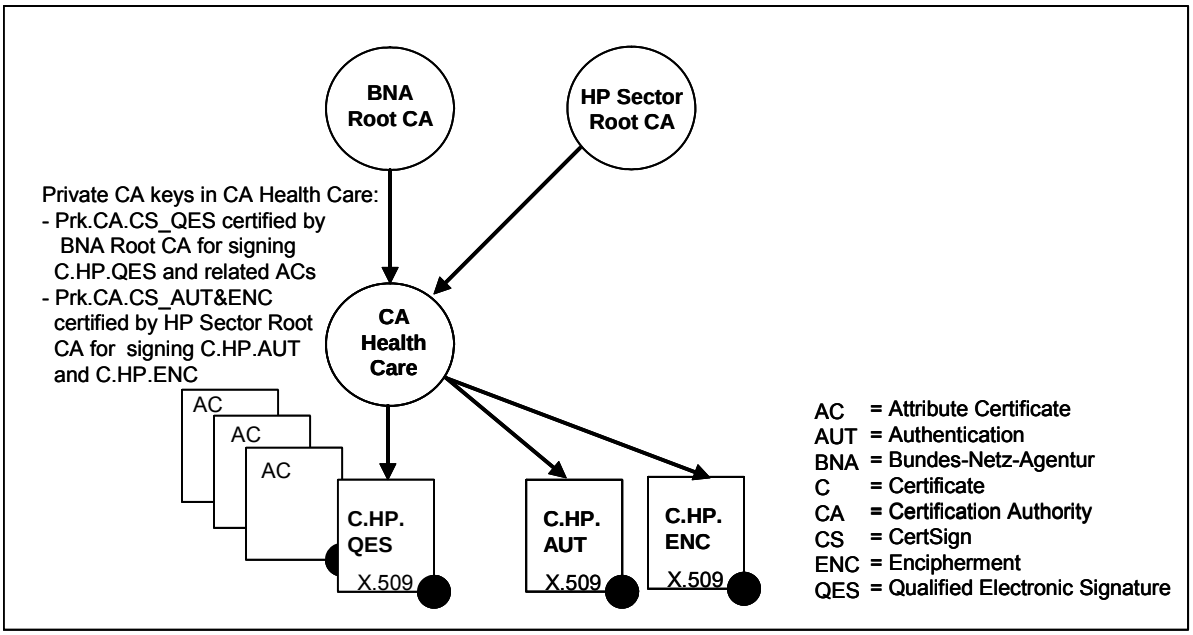

Figure 11 – (N2628.00) X.509 Certificates and Certification Authorities

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

Annex C (normative)

Certificates for Authentication

and Key Encipherment

C.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.

C.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 Common PKI 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 Common PKI 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 D (informative)

QES Cryptographic Information Objects

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

D.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).

D.1.1 ASN.1 Value notation

```
1   ciaInfo CIAInfo ::= {
2       version v2,
3       label  “Qualified Signature Application”,
4       cardflags { authRequired, prnGeneration }
5       seInfo {
6           SecurityEnvironmentInfo {
7               se 1,
8               aid 'D27600006601’ H
            }
9           SecurityEnvironmentInfo {
10              se 2,
11              aid 'D27600006601’ H
            }
        },
12      supportedAlgorithms {
            -- Hash algorithm: SHA-256
13          algorithmInfo {
14              reference 1, -- unique reference for cross-referencing
15              algorithm 592, -- PKCS#11 mechanism type CKM_SHA256 = 0x250
16              parameters NULL: NULL, -- type of parameters is NULL and value is NULL
17              supportedOperations {hash},
18              objId {2 16 840 1 101 3 4 2 1},-- SHA-256
                -- no algRef as no HASH command is used
            },
            -- Signature algorithm
            -- RSA with DSI according to PKCS#1 V.1.5  and SHA-256
19          algorithmInfo {
20              reference 2, -- unique reference for cross-referencing
21              algorithm 64, -- PKCS#11 mechanism type CKM_SHA256_RSA_PKCS = 0x40
22              parameters NULL: NULL, -- type of parameters is NULL and value is NULL
23              supportedOperations {compute-signature},
24              objId {1 2 840 113549 1 1 11},-- SHA256 and RSA with DSI according to PKCS#1
25              algRef 2 -- is equivalent to 0x02; see [HPC-P1], Table (N1218.00)
            },
            -- Signature algorithm
            -- RSA with DSI according to PKCS#1 PSS  and SHA-256
26          algorithmInfo {
27              reference 3, -- unique reference for cross-referencing
28              algorithm 67, -- PKCS#11 mechanism type CKM_SHA256_RSA_PKCS_PSS = 0x43
29              parameters NULL: NULL, -- type of parameters is NULL and value is NULL
30              supportedOperations {compute-signature},
31              objId {1 2 840 113549 1 1 11},-- SHA256 and RSA with DSI according to PKCS#1
```

```
                algRef 5 -- is equivalent to 0x05; see [HPC-P1], Table (N1218.00)
            },
            -- Signature algorithm
            -- RSA with DSI according to ISO/IEC 9796-2 DS2 (randomized) and SHA-256
            algorithmInfo {
                reference 4, -- unique reference for cross-referencing
                algorithm 2147483649, -- algorithm not defined in PKCS#11,
                -- vendor defined 0x80000001
                parameters NULL: NULL, -- type of parameters is NULL and value is NULL
                supportedOperations {compute-signature},
                objId {1 3 36 3 4 3 2 4},  -- 2B240304030204
                -- SHA-256 and RSA with DSI according to ISO/IEC 9796-2 with random number
                algRef 7 -- is equivalent to 0x07; see [HPC-P1], Table (N1218.00)
            },
            -- Device authentication algorithm
            algorithmInfo {
                reference 5, -- unique reference for cross-referencing
                algorithm 2147483650, -- algorithm not defined in PKCS#11,
                -- vendor defined 0x80000002
                parameters NULL: NULL, -- type of parameters is NULL and value is NULL
                supportedOperations {compute-signature, verify-signature},
                objId {1 3 36 3 5 2 4}, -- authentication scheme with RSA signature and DSI
                -- according to ISO/IEC 9796-2 and SHA-256 for mutual authentication with or
                -- without establishment of a Trusted Channel; see [HPC-P1], Table (N48.50) and
                Table (N1216.00)
            },
            -- Card Verifiable (CV) certificate signature verification
            algorithmInfo {
                reference 6, -- unique reference for cross-referencing
                algorithm 2147483651, -- algorithm not defined in PKCS#11,
                -- vendor defined 0x80000003
                parameters NULL: NULL, -- type of parameters is NULL and value is NULL
                supportedOperations {verify-signature},
                objId {1 3 36 3 4 3 2 4},  -- 2B240304030204
                -- SHA-256 and RSA with DSI according to ISO/IEC 9796-2 with random number
                -- algRef is not used (key and algorithm are selected by the Certification
                -- Authority Reference CAR provided in the Card Verifiable CV certificate)
            }
        }
    }
```

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

```
1  CIAInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 227
2    version INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
       1
3    label Label UTF8String: tag = [0] primitive; length = 31
       0x5175616C6966696564205369676E617475726520417070706C69636174696F6E
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 = 6
           0xD27600006601
9      SecurityEnvironmentInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 11
10       se INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
           0x02
11       aid OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 6
           0xD27600006601

12   supportedAlgorithms SEQUENCE OF: tag [UNIVERSAL 16] constructed; length = 156
```

```
13    algorithmInfo SEQUENCE: tag [UNIVERSAL 16] constructed; length = 24
14     reference Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x01
15     algorithm Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
         0x0250
16     parameter NULL: tag [UNIVERSAL 5] primitive; length = 0
17     supportedOperations Operations BIT STRING: tag [UNIVERSAL 3] primitive; length = 2
         0x0102
18     objId OBJECTIDENTIFIER: tag [UNIVERSAL 6] primitive; length = 9
         0x608648016503040201

19    algorithmInfo SEQUENCE: tag [UNIVERSAL 16] constructed; length = 23
20     reference Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x02
21     algorithm Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
         0x40
22     parameter NULL: tag [UNIVERSAL 5] primitive; length = 0
23     supportedOperations Operations BIT STRING: tag [UNIVERSAL 3] primitive; length = 2
         0x0604
24     objId OBJECTIDENTIFIER: tag [UNIVERSAL 6] primitive; length = 6
         0x60864883770D
25     algRef Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x02

26    algorithmInfo SEQUENCE: tag [UNIVERSAL 16] constructed; length = 23
27     reference Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x03
28     algorithm Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
         0x43
29     parameter NULL: tag [UNIVERSAL 5] primitive; length = 0
30     supportedOperations Operations BIT STRING: tag [UNIVERSAL 3] primitive; length = 2
         0x0604
31     objId OBJECTIDENTIFIER: tag [UNIVERSAL 6] primitive; length = 6
         0x60864883770D
32     algRef Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x05

33    algorithmInfo SEQUENCE: tag [UNIVERSAL 16] constructed; length = 27
34     reference Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x04
35     algorithm Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 4
         0x80000001
36     parameter NULL: tag [UNIVERSAL 5] primitive; length = 0
37     supportedOperations Operations BIT STRING: tag [UNIVERSAL 3] primitive; length = 2
         0x0604
38     objId OBJECTIDENTIFIER: tag [UNIVERSAL 6] primitive; length = 7
         0x2B240304030204
39     algRef Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x07

40    algorithmInfo SEQUENCE: tag [UNIVERSAL 16] constructed; length = 23
41     reference Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
         0x03
42     algorithm Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 4
         0x80000002
43     parameter NULL: tag [UNIVERSAL 5] primitive; length = 0
44     supportedOperations Operations BIT STRING: tag [UNIVERSAL 3] primitive; length = 2
         0x0405
45     objId OBJECTIDENTIFIER: tag [UNIVERSAL 6] primitive; length = 6
         0x2B2403050204
```

```
46      algorithmInfo SEQUENCE: tag [UNIVERSAL 16] constructed; length = 24
47        reference Reference INTEGER: tag [UNIVERSAL 2] primitive; length = 1
            0x04
48        algorithm Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 4
            0x80000003
49        parameter NULL: tag [UNIVERSAL 5] primitive; length = 0
50        supportedOperations Operations BIT STRING: tag [UNIVERSAL 3] primitive; length = 2
            0x0401
51        objId OBJECTIDENTIFIER: tag [UNIVERSAL 6] primitive; length = 7
            0x2B240304020203
```

D.1.3 Hexadecimal DER-encoding

```
1  30 81 E3
2     02 01
         01
3     80 1F
         51 75 61 6C 69 66 69 65 64 20 53 69 67 6E 61 74 75 72 65 20 41 70 70 6C 69 63 61 74 69 6F 6E
4     03 02
         05 60

5     30 1A
6        30 0B
7           02 01
               01
8           04 06
               D2 76 00 00 66 01
9        30 0B
10          02 01
               02
11          04 06
               D2 76 00 00 66 01

12    30 81 9C
13       30 18
14          02 01
               01
15          04 02
               02 50
16          05 00
17          03 02
               01 02
18          06 09
               60 86 48 01 65 03 04 02 01

19       30 17
20          02 01
               02
21          04 01
               40
22          05 00
23          03 02
               06 04
24          06 06
               60 86 48 83 77 0D
25          02 01
               02

26       30 17
27          02 01
               03
28          04 01
               43
29          05 00
30          03 02
               06 04
31          06 06
               60 86 48 83 77 0D
32          02 01
               05
```

```
33          30 1B
34              02 01
                    04
35              04 04
                    80 00 00 01
36              05 00
37              03 02
                    06 04
38              06 07
                    2B 24 03 04 03 02 04
39              02 01
                    07

40          30 17
41              02 01
                    05
42              04 04
                    80 00 00 02
43              05 00
44              03 02
                    04 05
45              06 06
                    2B 24 03 05 02 04

46          30 18
47              02 01
                    06
48              04 04
                    80 00 00 03
49              05 00
50              03 02
                    04 01
51              06 07
                    2B 24 03 04 02 02 03
```

D.2 EF.OD

The “Object Directory” EF contains information which CIO-EFs are present and how they are referenced by SFID.

D.2.1 ASN.1 Value notation

```
1       authObjects :
2               path : {
3                       efidOrPath '14'H -- SFID of EF.AOD
                        },
4       privateKeys :
5               path : {
6                       efidOrPath '15'H -- SFID of EF.PrKD
                        },
7       certificates :
8               path : {
9                       efidOrPath '16'H -- SFID of EF.CD
                        }
```

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

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

4   CIOChoice CHOICE
      privateKeys : tag = [0] constructed; length = 5
5       PrivateKeys CHOICE
          path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
```

```
6       efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
         0x15
7  CIOChoice CHOICE
    certificates : tag = [4] constructed; length = 5
8    Certificates CHOICE
       path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
9       efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
         0x16
```

D.2.3 Hexadecimal DER-encoding

```
1  A8 05
2       30 03
3            04 01
                14
4  A0 05
5       30 03
6            04 01
                15
7  A4 05
8       30 03
9            04 01
                16
```

D.3 EF.AOD

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

D.3.1 ASN.1 Value notation

```
1       pwd : {
2               commonObjectAttributes {
3                       label "PIN.QES",
4                       flags { private },
5                       authId '13'H,
6                       accessControlRules {
7                               accessControlRule {
8                                       accessMode { execute },
9                                       securityCondition authReference: {
10                                              authMethod {secureMessaging, extAuthentication},
11                                              seIdentifier 2
                                        }
                                }
                        }
                },
12              classAttributes {
13                      authId '03'H
                },
14              typeAttributes {
15                      passwordAttributes {
16                              pwdFlags { local, initialized, exchangeRefData },
17                              pwdType iso9564-1,
18                              minLength 6,
19                              storedLength 0,
20                              maxLength 8,
21                              pwdReference ‘81’H                      -- P2 value of the VERIFY command
22                              path {
23                                      efidOrPath ‘3F0004’H
                                        -- Qualified path: Last byte = P1 of SELECT with DF name
                                }
                        }
                }
        },
```

```
pwd : {
        commonObjectAttributes {
                label "PUK.QES",
                flags { private },
                accessControlRules {
                        accessControlRule {
                                accessMode { execute },
                                securityCondition authReference: {
                                        authMethod {secureMessaging, extAuthentication},
                                        seIdentifier 2
                                }
                        }
                }
        },
        classAttributes {
                authId '13'H
        },
        typeAttributes {
                passwordAttributes {
                        pwdFlags { local, change-disabled, unblock-disabled, initialized,
                            unblockingPassword },
                        pwdType iso9564-1,
                        minLength 8,
                        storedLength 8,
                        maxLength 8,
                        pwdReference ‘81’H          -- P2 value of the RESET RETRY COUNTER command
                        path {
                                efidOrPath ‘3F0004’H
                                -- Qualified path: Last byte = P1 of SELECT with DF name
                        }
                }
        }
}
```

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

```
1  AuthenticationObjectChoice CHOICE
     pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 47
2      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 16
3        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 7
           0x50494E2E514553
4        flags CnovellommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
           0x0780
5        authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
           0x13
6      accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 15
7        acessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 13
8          accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
             0x0520
9          securityCondition CHOICE
             authReference SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 7
10             authMethod BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                 0x06C0
11             seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
                 2
12 classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
     tag = [UNIVERSAL 16] constructed; length = 3
13     authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
       0x03
14     typeAttributes : tag = [1] constructed; length = 22
15       PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 20
16         pwdFlags PasswordFlags BIT STRING; length = 3
           0x044810
```

```
17        pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length = 1
          4
18        minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          6
19        storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          0
20        maxLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          8
21        pwdReference Reference INTEGER: [0] primitive; length = 1
          0x81
22        path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 5
23          efidOrPath OCTED STRING: tag = [UNIVERSAL 4]; length = 3
              0x3F0004
24  AuthenticationObjectChoice CHOICE
      pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 43
25      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
          constructed; length = 13
26        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 7
          0x50554B2E514553
27        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
          0x0780
28      accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 15
29        acessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 13
30          accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x0520
31          securityCondition CHOICE
            authReference SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 7
32            authMethod BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                0x06C0
33            seIdentifier INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
                2
34      classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
        tag = [UNIVERSAL 16] constructed; length = 3
35        authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x13
36      typeAttributes : tag = [1] constructed; length = 21
37        PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 19
38          pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
            0x017A
39          pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
            4
40          minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            8
41          storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            8
42          maxLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
            8
43          pwdReference Reference INTEGER: [0] primitive; length = 1
            0x81
44          path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 5
45            efidOrPath OCTED STRING: tag = [UNIVERSAL 4]; length = 3
                0x3F0004
```

D.3.3 Hexadecimal DER-encoding

```
1  30 2F
2     30 10
3        0C 07
               50 49 4E 2E 51 45 53
```

```
4           03 02
                07 80
5           04 01
                13
6       30 0F
7           30 0D
8               03 02
                    05 20
9               30 07
10                  03 02
                        06 C0
11                  02 01
                        02
12      30 03
13          04 01
                03
14      A1 1E
15          30 1B
16              03 03
                    04 48 10
17              0A 01
                    04
18              02 01
                    06
19              02 01
                    00
20              02 01
                    08
21              80 01
                    81
22              30 05
23                  04 03
                        3F 00 04
24  30 2B
25      30 0D
26          0C 07
                50 55 4B 2E 51 45 53
27          03 02
                07 80
28      30 0F
29          30 0D
30              03 02
                    05 20
31              30 07
32                  03 02
                        06 C0
33                  02 01
                        02
34      30 03
35          04 01
                12
36      A1 15
37          30 13
38              03 02
                    01 7A
39              0A 01
                    04
40              02 01
                    08
41              02 01
                    08
42              02 01
                    08
43              80 01
                    81
44              30 05
45                  04 03
                        3F 00 04
```

D.4 EF.PrKD

The “private key directory” EF provides necessary information about the private key PrK.HP.QES.

D.4.1 ASN.1 Value notation

```
privateRSAKey : {
    commonObjectAttributes {
        label "PrK.HP.QES",
        flags { private },
        authId '03'H            -- pointer to the AOD-entry of PIN.QES
        accessControlRules {
            accessControlRule {
                accessMode { execute },
                securityCondition or : {
                    securityCondition {
                        authId : '03'H,    -- pointer to the AOD-entry of PIN.QES
                        authReference : {
                            authMethod { userAuthentication }
                            seIdentifier 1
                        }
                    }
                    securityCondition {
                        authId : '03'H,    -- pointer to the AOD-entry of PIN.QES
                        authReference : {
                            authMethod { secureMessaging,
                             extAuthentication, userAuthentication }
                            seIdentifier 2 }
                        }
                    }
                }
            }
        }
    },
    classAttributes {
        iD '84'H,               -- cross-reference to the CD-entry of C.HP.QES
        usage { sign, signRecover, nonRepudiation },
        keyReference '84'H      -- used in MSE-SET-command
    },
    typeAttributes {
        privateRSAKeyAttributes {
            value { efidOrPath "H },
            modulusLength 2048
        }
    }
}
```

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

```
1  PrivateKeyChoice CHOICE
     privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 64
2      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 38
3        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
           0x50724B2E48502E514553
4        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
           0x0780
5        authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
           0x03
6        accessControlRules SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 32
7          accessControlRule SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 30
8            accessMode BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
               0x0520
9            securityCondition CHOICE
               or SEQUENCE OF: tag = [2] constructed; length = 24
               securityCondition SecurityCondition CHOICE
10               authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                 0x03
11               authReference SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 7
```

```
12              authMethod BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                  0x0520
13              seIdentifier INTEGER: tag = {UNIVERSAL 2] primitive; length = 1
                  0x01
            securityCondition SecurityCondition CHOICE
14            authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
              0x03
15            authReference SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 7
16              authMethod BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
                  0x0520
17              seIdentifier INTEGER: tag = {UNIVERSAL 2] primitive; length = 1
                  0x02
18      classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 10
19        iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x84
20        usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
          0x0520
21        keyReference KeyReference INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          0x84
22      typeAttributes : tag = [1] constructed; length = 10
23        PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 8
24          value Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 2
              efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 0
25          modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
              2048
```

D.4.3 Hexadecimal DER-encoding

```
1  30 40
2      30 26
3          0C 0A
               50 72 4B 2E 48 50 2E 51 45 53
4          03 02
               07 80
5          04 01
               03
6          30 20
7              30 1E
8                  03 02
                       05 20
9                  A2 18
10                     04 01
                           03
11                     30 07
12                         03 02
                               05 20
13                         02 01
                               01
14                     04 01
                           03
15                     30 07
16                         03 02
                               05 20
17                         02 01
                               02
18         30 0A
19             04 01
                   84
20             03 02
                   05 20
21             02 01
                   84
22         A1 0A
23             30 08
24                 30 02
                       04 00
25                 02 02
                       08 00
```

D.5 EF.CD

The “certificate directory” EF contains the file references to the X.509 certificates C.HP.QES, C.HP.QES-AC1, C.HP.QES-AC2, and C.HP.QES-AC3.

D.5.1 ASN.1 Value notation

```
x509Certificate : {
        commonObjectAttributes {
                label "C.HP.QES",
                flags { private }
        },
        classAttributes {
                iD '84'H                -- cross-reference to the PrKD-entry of PrK.HP.QES
        },
        typeAttributes {
                value indirect :
                        path : {
                                efidOrPath '10'H --SFID of C.HP.QES
                        }
        }
},
x509Certificate : {
        commonObjectAttributes {
                label "C.HP.QES-AC1",
                flags { private }
        },
        classAttributes {
                iD '84'H                -- cross-reference to the PrKD-entry of PrK.HP.QES
        },
        typeAttributes {
                value indirect :
                        path : {
                                efidOrPath '01'H --SFID of C.HP.QES-AC1
                        }
        }
},
x509Certificate : {
        commonObjectAttributes {
                label "C.HP.QES-AC2",
                flags { private }
        },
        classAttributes {
                iD '84'H                -- cross-reference to the PrKD-entry of PrK.HP.QES
        },
        typeAttributes {
                value indirect :
                        path : {
                                efidOrPath '02'H --SFID of C.HP. QES-AC2
                        }
        }
},
x509Certificate : {
        commonObjectAttributes {
                label "C.HP.QES-AC3",
                flags { private }
        },
        classAttributes {
                iD '84'H                -- cross-reference to the PrKD-entry of PrK.HP.QES
        },
        typeAttributes {
                value indirect :
                        path : {
                                efidOrPath '03'H --SFID of C.HP. QES-AC3
```

```
                    }
                }
            }
```

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

```
1  CertificateChoice CHOICE
     x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 30
2      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 14
3        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 8
           0x432E48502E514553
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
           0x84
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
                   0x10
10 CertificateChoice CHOICE
     x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 34
11     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 18
12       label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 12
           0x432E48502E5145532D414331
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
           0x84
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
                 0x01
19 CertificateChoice CHOICE
     x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 34
20     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 18
21       label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 12
           0x432E48502E5145532D414332
22       flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
           0x0780
23     classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 3
24       iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
           0x84
25     typeAttributes : tag = [1] constructed; length = 7
         X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
         length = 5
```

```
26        value CHOICE
          indirect ReferencedValue CHOICE
            path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
27              efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                0x02
28  CertificateChoice CHOICE
    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 34
29      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 18
30        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 12
          0x432E48502E5145532D414333
31        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
          0x0780
32      classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 3
33        iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x84
34      typeAttributes : tag = [1] constructed; length = 7
        X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 5
35        value CHOICE
          indirect ReferencedValue CHOICE
            path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 3
36              efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
                0x03
```

D.5.3 Hexadecimal DER-encoding

```
1  30 1E
2      30 0E
3          0C 08
               43 2E 48 50 2E 51 45 53
4          03 02
               07 80
5      30 03
6          04 01
               84
7      A1 07
           30 05
8              30 03
9                  04 01
                       10
10 30 22
11     30 12
12         0C 0C
               43 2E 48 50 2E 51 45 53 2D 41 43 31
13         03 02
               07 80
14     30 03
15         04 01
               84
16     A1 07
           30 05
17             30 03
18                 04 01
                       01
19 30 22
20     30 12
21         0C 0C
               43 2E 48 50 2E 51 45 53 2D 41 43 32
22         03 02
               07 80
23     30 03
24         04 01
               84
25     A1 07
           30 05
26             30 03
27                 04 01
```

```
                    02
28 30 22
29    30 12
30       0C 0C
               43 2E 48 50 2E 51 45 53 2D 41 43 33
31       03 02
               07 80
32    30 03
33       04 01
               84
34    A1 07
            30 05
35             30 03
36                04 01
                     03
```

Annex E (informative)

ESIGN Cryptographic Information Objects

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

E.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).

E.1.1 ASN.1 Value notation

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

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

```
1  CIAInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 45
2   version INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
     1
3   label Label UTF8String: tag = [0] primitive; length = 17
     0x455349474E204170706C69636174696F6E
4   cardflags CardFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
     0x0560
5   seInfo SEQUENCE OF: tag = [UNIVERSAL 16] constructed; length = 17
6    SecurityEnvironmentInfo SEQUENCE: tag = [UNIVERSAL 16] constructed;
    length = 15
7      se INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
        0x01
8      aid OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 10
        0xA000000167455349474E
9      se INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
        0x02
10     aid OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 10
        0xA000000167455349474E
```

E.1.3 Hexadecimal DER-encoding

```
1  30 3E
2     02 01
         01
3     80 11
         45 53 49 47 4E 20 41 70 70 6C 69 63 61 74 69 6F 6E
4     03 02
         05 60
5     30 22
6        30 0F
7           02 01
               01
8           04 0A
               A0 00 00 01 67 45 53 49 47 4E
         30 0F
9           02 01
               02
10          04 0A
               A0 00 00 01 67 45 53 49 47 4E
```

E.2 EF.OD

The “Object Directory” EF contains information which CIO-EFs are present and how they are referenced by SFID.

E.2.1 ASN.1 Value notation

```
1     authObjects :
2           path : {
3                 efidOrPath '14'H -- SFID of EF.AOD
                  },
4     privateKeys :
5           path : {
6                 efidOrPath '15'H -- SFID of EF.PrKD
                  },
7     certificates :
8           path : {
9                 efidOrPath '16'H -- SFID of EF.CD
                  }
```

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

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

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

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

E.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
```

E.3 EF.AOD

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

E.3.1 ASN.1 Value notation

```
pwd : {
    commonObjectAttributes {
        label "PIN.CH",
        flags { private },
        authId '12'H
        },
    classAttributes {
        authId '02'H
        },
    typeAttributes {
        pwdFlags { initialized, exchangeRefData },
        pwdType iso9564-1,
        minLength 5,
        storedLength 0,
        maxLength 8,
        pwdReference 1          -- P2 value of the VERIFY command
        }
    },
pwd : {
    commonObjectAttributes {
        label "PUK.CH",
        flags { private }
        },
    classAttributes {
        authId '12'H
        },
    typeAttributes {
        pwdFlags {  global, change-disabled, unblock-disabled, initialized, unblockingPassword },
        pwdType iso9564-1,
        minLength 8,
        storedLength 8,
        maxLength 8,
        pwdReference 1          -- P2 value of the RESET RETRY COUNTER command
        }
    }
```

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

```
AuthenticationObjectChoice CHOICE
  pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 46
    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
    constructed; length = 15
```

```
3       label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 6
          0x50494E2E4348
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 = 42
16      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 12
17        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 6
          0x50554B2E4348
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
```

E.3.3 Hexadecimal DER-encoding

```
1  30 2E
2       30 0F
3            0C 06
                  50 49 4E 2E 43 48
```

```
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 2A
16      30 0C
17          0C 06
                50 55 4B 2E 43 48
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
```

E.4 EF.PrKD

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

E.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                               accessControlRule {
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                      accessControlRule {
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
                  }
            }
```

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

```
1  PrivateKeyChoice CHOICE
     privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 64
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 = 63
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
```

E.4.3 Hexadecimal DER-encoding

```
1  30 40
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
                   06 C0

20 30 3F
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
                   06 C0
```

E.5 EF.CD

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

E.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
                              }
                  }
```

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

```
1  CertificateChoice CHOICE
     x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 30
2      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 14
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 = 30
11     commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
       constructed; length = 14
```

```
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
```

E.5.3 Hexadecimal DER-encoding

```
1  30 1E
2     30 0E
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 0E
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
```