German Office Identity Card

(Elektronischer Dienstausweis)

Version 1.0

06.07.2000

TeleTrusT Deutschland e.V.
Eichendorffstr. 16
D-99096 Erfurt

This specification was elaborated by the TeleTrusT Working Group 2 "Security Architecture and Smart Cards", headed by Dr. Gisela Meister, Giesecke & Devrient.

Contact Addresses for this Specification:

Chair:

Dr. Gisela Meister
Prinzregentenstr. 159
81607 München
e-mail: Gisela.Meister@gdm.de

Editor:

Bruno Struif
Rheinstr. 75
64295 Darmstadt
e-mail: Bruno.Struif@darmstadt.gmd.de

Content

1 The OIC and its File Structure

1.1 Scope

This specification defines the interface between a terminal (interface device) and an office identity card (OIC), which is in compliance with the German digital signature law and the EU directive for electronic signatures.
The OIC shall provide the following basic security services:

- digital signature service
- authentication service for logon, client/server authentication and ICC/IFD authentication
- encryption service for document encryption.

In addition the OIC contains identification data.

All defined codings for commands and data are based on ISO/IEC-Standard 7816, part 4, 5, 6, 8 and 9. Applications with other codings shall comply with the functional requirements.

1.2 References

EU Directive 1999/93/EC of the European Parliament and the council of 13 December 1999 on a Community framework for electronic signatures

DIN 66291-1: 1999
Chipkarten mit digitaler Signatur-Anwendung/-Funktion nach SigG/SigV
Teil 1: Anwendungsschnittstelle

ISO/IEC 7810: 1995
Identification cards - Physical characteristics

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

ISO/IEC 7816-3: 1997 (2nd edition)
Information technology - Identification cards - Integrated circuit(s) cards with contacts -
Part 3: Electronic signals and transmission protocols

ISO/IEC 7816-4: 1995
Information technology - Identification cards - Integrated circuit(s) cards with contacts -
Part 4: Interindustry commands for interchange
AM1: Impact of secure messaging on the structures of APDU messages

ISO/IEC 7816-5: 1995
Information technology - Identification cards - Integrated circuit(s) cards with contacts -
Part 5: Numbering system and registration procedure for application identifiers

ISO/IEC 7816-6: 1995
Information technology - Identification cards – Integrated circuit(s) cards with contacts -
Part 6: Interindustry data elements
AM1: IC manufacturer register (FDIS)

ISO/IEC 7816-8: 1999
Information technology - Identification cards - Integrated circuit(s) cards with contacts -
Part 8: Security related interindustry commands

ISO/IEC 7816-9: FDIS 2000
Information technology - Identification cards - Integrated circuit(s) cards with contacts -
Part 9: Additional interindustry commands and security attributes

ISO 9564:
Personal identification number management and security
Part 1: PIN protection principles and techniques
Part 2: Approved algorithms for PIN encipherment

ISO/IEC 9796-2: 1997
Information technology - Security techniques – Digital signature schemes giving message recovery
Part 2: Mechanisms using a hash-function

ISO/IEC 10118: 1997
Information technology - Security techniques - Hash functions
Part 3: Dedicated hash functions

CEN ENV 1375-1: 1994
Identification card systems - Intersector integrated circuit(s) card additional formats –
Part 1: ID-000 card size and physical characteristics

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

PKCS #1: RSA Encryption Standard
Version 1.5, Nov. 1993
Version 2.0, July 1998

PKCS #15: Cryptographic Token Information Format Standard

3rd Draft Version 1.1, May 4, 2000

German digital signature law - Gesetz zur digitalen Signatur (Signaturgesetz - SigG) vom 22.07.1997 (BGBl. I S. 1870, 1872), verkündet als Artikel 3 des "Gesetzes zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste (Informations- und Kommunikationsdienste-Gesetz - IuKDG)

German signature regulations -Verordnung zur digitalen Signatur (Signaturverordnung - SigV), 22.10.1997 (BGBl. I S. 2498)
Schnittstellenspezifikation zur Entwicklung interoperabler Verfahren u. Komponenten nach SigG/SigV (SigI)
Teil Zertifikate, Version 2.0, 31.07.1998

NIST: FIPS Publication 186-2:
Digital Signature Standard (DSS), 2000

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

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

IEEE P1363, Standard Specifications for Public Key Cryptography, 2000

ANSI X 9.42 Draft, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys using Diffie-Hellman and MQV Algorithms, 1999

ANSI X 9.62 Draft, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm, 1999

ANSI X 9.63, Draft, Version 2.0,
Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Key Transport Schemes, 1999

ISO/IEC 8824, Information technology - Abstract Syntax Notation One - ASN.1, 1998

ISO/IEC 8825, Information technology - Abstract Syntax Notation One - ASN.1-Encoding rules, 1998

ISO/IEC 9797-1: 1998
Information technology – Security techniques – Message authentication codes, Part 1: Mechanisms using a block cipher, 27. Mai 1998

ISO/IEC 9798-3:1993
Information technology – Security techniques – Entity authentication mechanisms, Part 3: Entity authentication using a public key algorithm

ISO/IEC 11770-3: 1998
Information technology – Security techniques – Key management, Part 3: Mechanisms using asymmetric techniques

Internet-Draft : draft-ietf-cat-kerberos-pk-init-11.txt, March 2000

RFC 1510: The Keberos Network Authentication Service (V5), September 1993

RFC 2246: The TLS Protocol, version 1.0, January 1999

RFC 2630: Cryptographic Message Syntax (CMS), June 1999

RFC 2631: Diffie-Hellman Key Agreement Method, June 1999

RFC 2785: Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME, March 2000

1.3 Abbreviations and Notations

1.3.1 Abbreviations

AID = Application Identifier
ASS = Additional Security Services
AKI = Authority Key Identifier
AT = CRT for Authentication
ATR = Answer-to-Reset
AUT = Authentication
B = Byte
BCD = Binary Coded Digit
BD = Basic Identification Data
BER = Basic Encoding Rules
BSS = Basic Security Services
C = Certificate
CA = Certification Authority
CAR = CA Reference
CC = Cryptographic Checksum
CG = Cryptogram
CH = Cardholder
CHA = Certificate Holder Authorisation
CHN = Cardholder Name
CHR = Certificate Holder Reference
CRT = Control Reference Template
CPI = Certificate Profile Identifier
CS = CertSign
CT = CRT for confidentiality
CTI = Cryptographic Token Information
CV = Card Verifiable
C/S = Client/Server
D() = Decipherment of

DE = Data Element
DF = Dedicated File
DH = Diffie-Hellman
DO = Data Object
DS = Digital Signature
DSA = Digital Signature Algorithm
DSI = Digital Signature Input
DST = CRT for Digital Signature
E() = Encipherment of
ED = Enhanced Identification Data
EF = Elementary File
ELC = Elliptic Curve
FCI = File Control Information
FID = File Identifier
GDO = Global Data Objects
h() = hash value of
HB = Historical Bytes
HT = CRT for hash code
ICC = Integrated Circuit(s) Card
ICCSN = ICC Serial Number
IDD = Identification Data
IFD = Interface Device
IFSC = Information Field Size Card
IFSD = Information Field Size Device
IVS = Identity Verification System
KE = Key Encipherment
KID = Key Identifier
MF = Master File
MSE = MANAGE SEC. ENVIRONMENT
NAD = Node Address
OIC = Office Identity Card
OID = Object Identifier
PCA = Policy Certification Authority
PK = Public Key
PKCS = Public Key Cryptography Standards
PI = Padding Indicator
PIN = Personal Identification Number
PRND = Padding Random Number
PROV = Provider
PPS = Protocol Parameter Selection
PSO = PERFORM SEC. OPERATION
PV = Plain Value
PW = Password
RC = Retry Counter
RCA = RootCA
RD = Reference Data
RegTP = Regulation Authority Telecom.& Post
RFU = Reserved for Future Use
RSA = Sig-Alg. of Rivest, Shamir, Adleman
SE = Security Environment
SIG() = Signature of
SigG = Signature law
SK = Secret Key
SN = Serial Number
SM = Secure Messaging
SP = Specific Privileges
SSC = Send Sequence Counter
SSD = Security Service Descriptor
TLS = Transport Layer Security
TLV = Tag, Length, Value
VD = Verification Data
VR = Verification Requirement
WTLS = Wireless TLS
ZZ = Diffie-Hellman shared value

1.3.2 Notations

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

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

<key descriptor>::=
<key>[<index1>].<keyholder>
[<index2>].<usage list>

<key>::=<asymmetric key> |
<symmetric key>
<asymmetric key>::= <secret key> |
<public key>
<secret key>::= SK
<public key>::= PK
<index1>::= 1 | 2 | ...

<keyholder>::= <cardholder> |
<device> | <organisation>
<device> ::= <integrated circuit(s) card> | <interface device> |
<personalisation system> | <server> |
<identity verification system>
<organisation> ::= <certification authority> | <card manufacturer>
<cardholder>::= CH
<certifcation authority>::= CA | RCA
<card manufacturer>::= CM
<integrated circuit(s) card>::= ICC
<interface device>::= IFD
<personalisation system>::= PS
<server> ::= S
<identity verification system> ::= IVS
<index2> used for indication of qualification or role

<usage list>::= <usage>[<index3>] |
<usage list> & <usage>[<index3>]
<usage>::= <digital signature> |
<authentication> | <key encipherment> |<CertSign>
<digital signature>::= DS
<authentication>::= AUT
<key encipherment>::= KE
<CertSign> ::= CS | CS-DS | CS-AUT | CS-KE
<index3> used for indication of algorithm or components (e.g. ICC/IFD or C/S)

<certificate descriptor>::=
<certificate>[<index1>].
<certholder>[<index2>].
<usage list>

<certificate>::= C | <CV certificate> | <X.509 certificate> | <X.509 attribute certificate>
<CV certificate> ::= C_CV
<X.509 certificate> ::= C_X509
<X.509 attribute certificate> ::= C_X509A

<certholder>::= <cardholder> | <device> | <organisation>

NOTE - Indexes shall only be applied for avoiding ambiguouity.

For the representation of data sequences the following notation is used:

|| = Concatenation of data

Examples:

1. SK.RCA.CS = Secret key of root CA for signing certificates, whereby no binding of the certified key pair to a special usage is specified
2. SK.RCA.CS-DS = Secret key of root CA for signing certificates, whereby the certified key pair shall be used for signing DS certificates only
3. SK_1.CH.DS_{RSA} = Secret key no. 1 of the cardholder for digital signature with RSA
4. SK.CH.$AUT_{C/S}$ = Secret key of the cardholder for client/server authentication
5. PK.CA.CS-$AUT_{C/S}$ = Public key of CA for verification of certificates related to client/server authentication
6. PK.CA.CS-$AUT_{ICC/IFD}$ = Public key of CA for verification of certificates related to ICC/IFD authentication
7. SK.CA.CS-AUT&AUT = Secret key of CA for signing of authentication certificates and for use in authentication procedures

1.4 OIC File Structure

The OIC has a general file structure according figure 1 or figure 2.

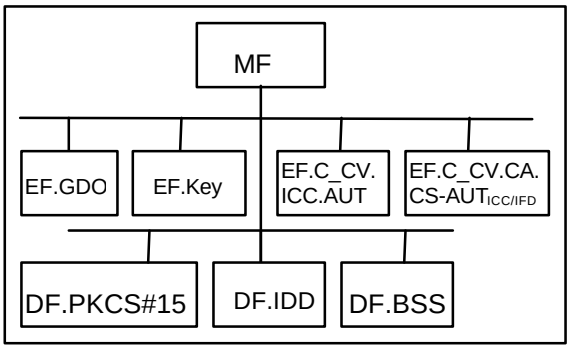

Fig. 1: OIC file structure (IDD+BSS type)

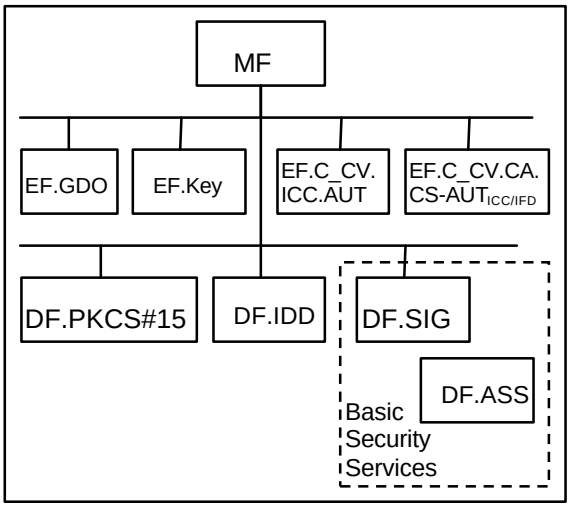

Fig. 2: OIC file structure (IDD+SIG/ASS type)

The logical place of DF.ASS may be under DF.SIG or under MF. The functionality of DF.SIG together with DF.ASS is that one of DF.BSS.

1.5 Application Identifier

Application Identifier relevant for the OIC are shown in Tab. 1.

Name	AID
DF.PKCS#15	'A000000063504B43532D3135'
DF.IDD	'D27600006603'
DF.BSS	'D27600006601'
DF.SIG	'D27600006601'
DF.ASS	'D27600006602'

Tab. 1: AIDs

DFs shall have a FID, so that e.g. an absolute path to an EF can be described in the crypto-graphic token information.

2 Technical Characteristics and Transmission Protocols

2.1 Scope

This clause specifies technical characteristics and transmission protocol related issues.

2.2 Technical Characteristics

Office identity cards are smartcards capable to process public key algorithms.
The location and dimensions of the contacts shall comply with ISO/IEC 7816-2. The format shall be ID-1 (normal size, see ISO 7810).
For bit transmission the 'direct convention' shall be applied. Programming power Vpp shall not be requested and will not be supported by the IFDs.

2.3 Answer-to-Reset

2.3.1 Global Interface Characters

The content and structure of the ATR is outlined in annex F.

2.3.2 Historical Bytes

The Historical Bytes (HB) shall comply with ISO/IEC 7816-4 and contain data objects like

- Category Indicator
- Pre-issuing Data Object
- Card Profile Data Object
- Card Life Cycle
- Status Bytes.

The content and structure of the HB is outlined in annex F.

2.4 Protocol Parameter Selection

The support of Protocol Parameter Selection (PPS) according to ISO/IEC 7816-3 is mandatory. It is used for selecting T=1, if T=0 is additionaly present in the card, and to negotiate the Fi/Di parameters for achieving higher transmission rates.

2.5 Transmission Protocols

2.5.1 General Aspects

As transmission protocol the asynchronous half-duplex Block Transmission Protocol T=1 shall be supported. Optionally the asynchronous half-duplex Character Transmission Protocol T=0 may be present in the card.

2.5.2 Transmission Protocol T = 0

If present, the implementation of T = 0 shall comply with ISO/IEC 7816-3.

2.5.3 Transmission Protocol T = 1

The implementation of T = 1 shall comply with ISO/IEC 7816-3. The support of chaining is mandatory.

The following simplifications are allowed:

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

The Information Field Size Card (IFSC) shall be indicated in the ATR (Character TA3, recommended value: min '80' = 128 Bytes).
The Information Field Size Device (IFSD) shall be indicated from the IFD immediately after ATR, i.e. the IFD shall transmit the S-Block IFS Request after ATR and the card shall send back S-Block IFS. The recommended value for IFSD is 254 bytes.

3 Global Keys, Global Data and Card Verifiable Certificates

3.1 Scope

This clause specifies the global keys, the global data objects and CV certificates.

3.2 Files on MF-Level

Fig. 3 shows the elementary files on MF level.

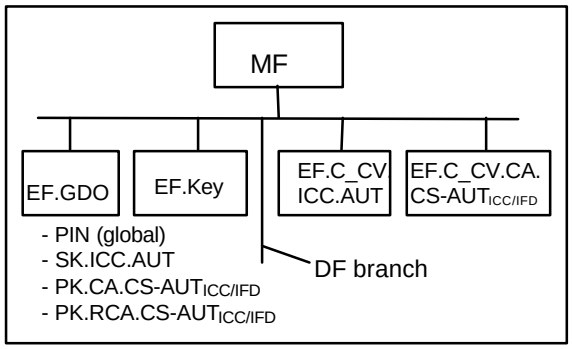

Fig. 3: EFs on MF-Level

3.2.1 EF.GDO

The EF.GDO contains

- DO 'ICC Serial Number (ICCSN)' as shown in fig. 4 or 5 (tag '5A', also referred as primary account number) and
- DO 'Cardholder Name (CHN)' as shown on the card cover (tag '5F20'). The DO CHN is optional.

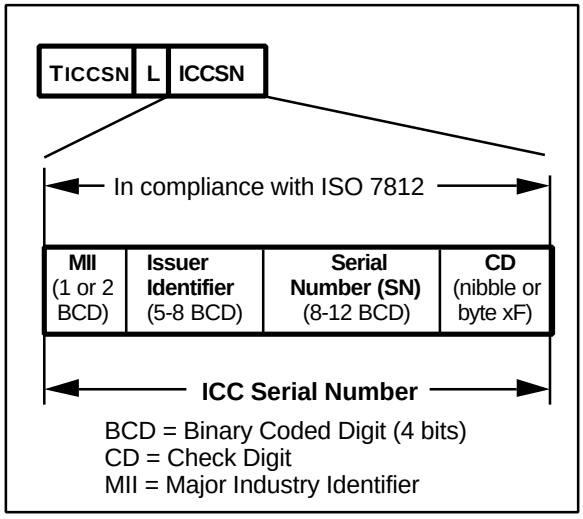

Fig. 4: ICCSN (in compliance with ISO/IEC 7812)

An alternative encoding for the ICCSN is shown in fig. 5.

Issuer Identification No.	Serial No.
'D2 76 nn nn nn' (5 Bytes, n= BCD)	'xx .. xx' (8 Bytes, e.g. chip serial no.)

Fig. 5: ICCSN (ISO/IEC 7816-5/6-oriented with Category D and Country Code)

NOTE - An IIN according to fig. 5 is assigned by CCG in Cologne.

3.2.2 EF.Key

There may be one or more key files with private keys. The following private keys are present:

Key name	KID
PIN (global, authentication and decryption PIN)	'01'
SK.ICC.AUT$_{ICC/IFD}$ (secret key of ICC for ICC/IFD authentication)	'01'

Tab. 2: Keys and KIDs

NOTE - If other KIDs are used, then the KID has to be indicated in the SSD file or in the cryptographic token information.

In addition to the private keys, the public key of the root CA and the CA which issued the CV certificate, are present in a key file at MF level:

- PK.RCA.CS-AUT$_{ICC/IFD}$
- PK.CA.CS-AUT$_{ICC/IFD}$

3.2.3 EF.C_CV.ICC.AUT

The EF.C_CV.ICC.AUT contains the CV certificate of the card for authentication.

3.2.4 EF.C_CV.CA.CS-AUT$_{ICC/IFD}$

The EF.C_CV.CA.CS-AUT$_{ICC/IFD}$ contains the CV certificate of the CA or card manufacturer, which issued the card certificate C_CV.ICC.AUT.

3.3 CV Certificates

Beside of the X.509v3 certificates, a CV certificate is needed in an OIC to enable especially

- advanced personalisation concepts
- access to the protected employee data and
- digital signatures on public customer service terminals, where a higher security level is required (see DIN V66291-1).

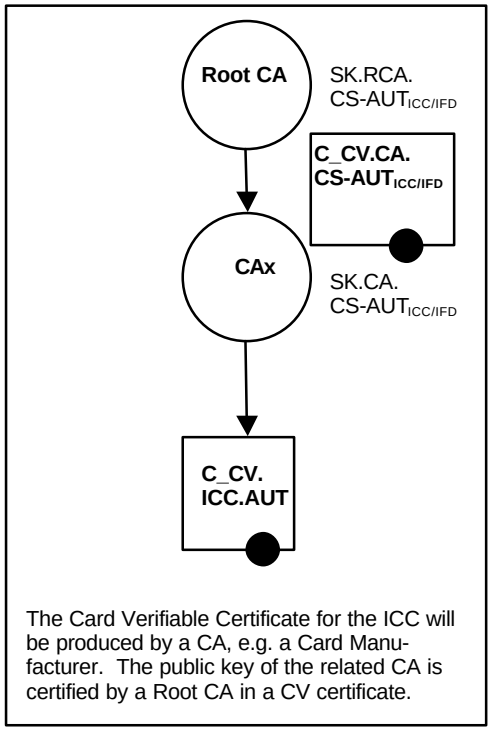

Fig. 6: CAs and CV certificates

3.4 Reading the Global Data Objects and CV Certificates

For reading the global data objects and CV certificates the ISO/IEC 7816-4 commands SELECT FILE and READ BINARY are used.

CLA	'00'
INS	'A4' = SELECT FILE
P1	'02' = EF file selection
P2	'0C' = No FCI to return
Lc	'02' = Length of subsequent data field
Data field	FID (see annex C) of one of the files: - EF.GDO - EF.C.ICC.AUT - EF.C.CA.CS-AUT$_{ICC/IFD}$
Le	Empty

Tab. 3: SELECT FILE command for EF selection

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

Tab. 4: SELECT FILE response

CLA	'00'
INS	'B0' = READ BINARY
P1,P2	'0000'
Lc	Empty
Data field	Empty
Le	'00' = Read until end-of-file

Tab. 5: READ BINARY command for reading the global data objects or CV certificates

Data field	- If EF.GDO: Global data objects - If EF.C_CV.ICC.AUT: CV certif. of ICC - If EF.C_CV.CA.CS-AUT$_{ICC/IFD}$: CV certificate of CA
SW1-SW2	'9000' or specific status bytes

Tab. 6: READ BINARY response

4 Identification Data

4.1 Scope

In this clause, the file structure and file content for identification data is outlined.

4.2 File Structure

The following figure shows the file structure of DF.IDD.

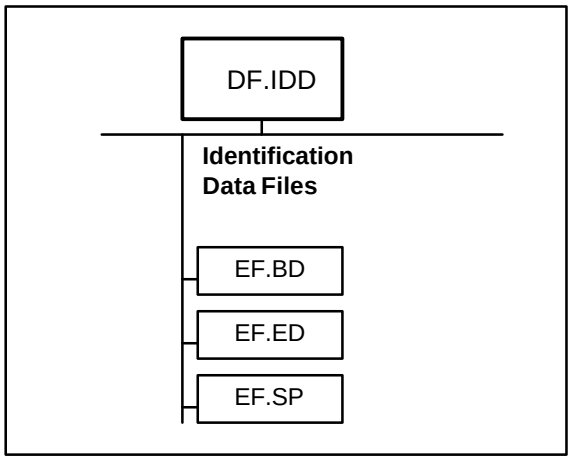

Fig. 7: DF.IDD file structure

4.2.1 EF.BD

The EF.BD contains the basic identification data of the cardholder:

- office name (max. 70 bytes)
- surname (max. 52 bytes)
- given names (max. 26 bytes)
- date of birth (10 bytes: DD.MM.YYYY)
- identity registration no. (8 bytes)
- validity (10 bytes: until DD.MM.YYYY)
- freetext 1 (max. 140 bytes)
- freetext 2 (max. 140 bytes)

Further data elements may be present.

The information is signed. The coding scheme of the file content (basic data with signature) is outside the scope of this document.

4.2.2 EF.ED

The EF.ED contains the enhanced identification data of the cardholder:

- foto
- signature image
- identity registration no.

Further or other data elements may be present.

The information is signed. The coding scheme of the file content is outside the scope of this document.

4.2.3 EF.SP

The EF.SP contains the specific privileges of the cardholder. The coding scheme of the file content is outside the scope of this document.

4.3 Application Selection

The ISO/IEC 7816-4 command for 'Direct Application Selection' is shown subsequently.

CLA	'00'
INS	'A4' = SELECT FILE
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'06' = Length of subsequent data field
Data field	'D27600006603' = AID of DF.IDD, see tab. 1
Le	Empty

Tab. 7: SELECT FILE command for application selection with AID

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

Tab. 8: SELECT FILE response

4.4 Access Control for the Identification Data

For getting read or write access to the identification data, an authentication procedure as described in clause 5.8 has to be performed.

4.5 Reading / Updating the Identification Data

For reading the identification data, the ISO/IEC 7816-4 commands SELECT FILE and READ BINARY are used.

CLA	'0C' = Command with SM
INS	'A4' = SELECT FILE
P1	'02' = EF file selection
P2	'0C' = No FCI to return
Lc	'xx' = Length of subsequent data field
Data field	'8102xxxx' || '8E'-L-'xx ... xx' = PV-DO with FID || CC-DO FID (see annex C) of one of the files: - EF.BD - EF.ED - EF.SP
Le	Empty

Tab. 9: SELECT FILE command for EF selection

Data field	'99029000' || '8E'-L-'xx ... xx' = Status-DO || CC-DO
SW1-SW2	'9000' or specific status bytes

Tab. 10: SELECT FILE response

CLA	'0C' = Command with SM
INS	'B0' = READ BINARY
P1,P2	'0000' or offset
Lc	'xx' = Length of subsequent data field
Data field	'97 01 08' || '8E'-L-'xx ... xx' = Le-DO || CC-DO
Le	'00'

Tab. 11: READ BINARY command for reading the identification data

Data field	'87'-L-'xx .. xx' || '8E'-L-'xx .. xx' = CG-DO || CC-DO The cryptogram contains the following data: - If EF.BD: Basic identific. data - If EF.ED: Enhanced id. data - If EF.SP: Specific privileges
SW1-SW2	'9000' or specific status bytes

Tab. 12: READ BINARY response

CLA	'0C' = Command with SM
INS	'D6' =UPDATE BINARY
P1,P2	'0000' or offset
Lc	'xx'= Length of subsequent data field
Data field	'87'-L-'xx ... xx' || '8E'-L-'xx ... xx' = CG-DO || CC-DO
Le	'00'

Tab. 13: UPDATE BINARY command for updating the identification data

Data field	'99029000' || '8E'-L-'xx ... xx' = Status-DO || CC-DO
SW1-SW2	'9000' or specific status bytes

Tab. 14: UPDATE BINARY response

5 Basic Security Services

5.1 Scope

The basic security services provided by DF.BSS covers the

- digital signature service for legally relevant signatures of the cardholder
- authentication service as required for specific authentication procedure
- Key decipherment service

The digital signature service is in full compliance to DIN V66291-1.

5.2 Certification Authorities and Certificates

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

- DS certificate(s) (X.509v3 digital signature certificate(s), i.e. public key certificate and attribute certificate(s)); key usage flag is set to "non-repudiation"
- $AUT_{C/S}$ certificate (X.509v3 authentication certificate) for proving access rights to servers; key usage flag is set to "digtal signature"
- KE certificate (X.509v3 key encipherment certificate); key usage flag is set to "key encipherment or key agreement".

NOTE - It is in principle also possible to set the flags for "digital signature" and "key encipherment/-agreement" in the same certificate, if appropriate.

The X.509v3 certificates have to be produced by certification authorities (CA) as shown in fig. 8.

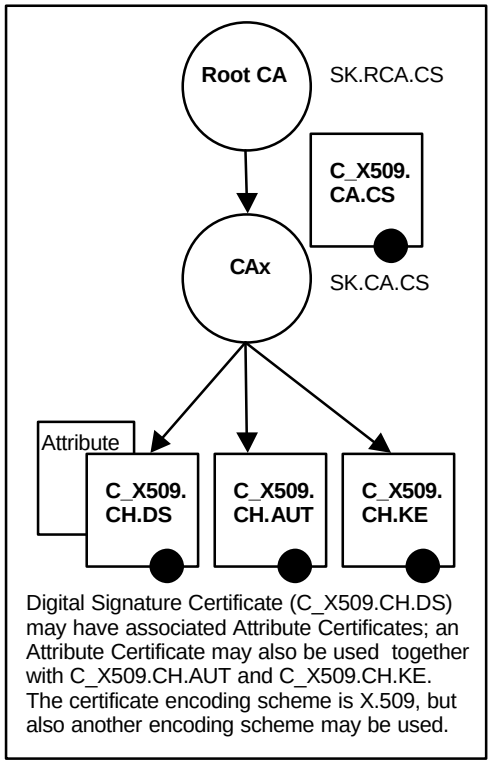

Fig. 8: CAs and X.509v3 certificates (example)

NOTE - The root CA for the digital signature sevice and for the other security services may be not the same. In case that there is a common root, it may be that different keys are used for signing DS-related certificates and certificates related to the other security services.

5.3 File Structure

5.3.1 General File Structure

Fig. 9 shows the BSS file structure.

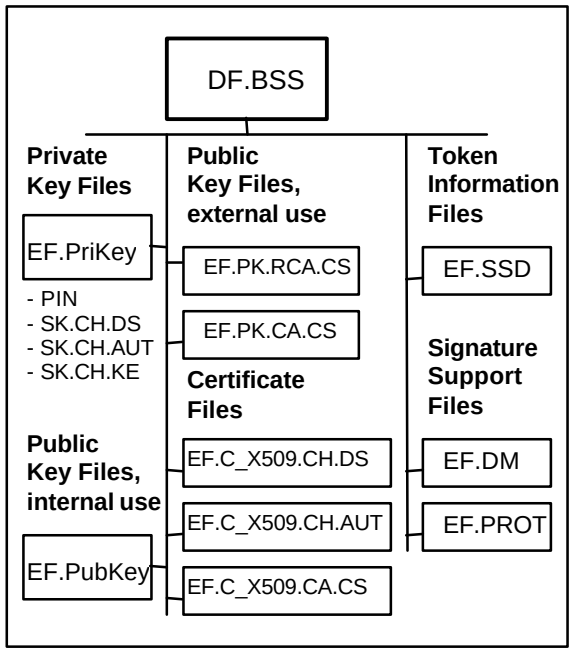

Fig. 9: BSS file structure (example)

NOTE - The certificate for key enciphering is normally stored in Directory Systems and not in the card. However, depending on the client/server authentication mechanism, also encipherment may play a role.

5.3.2 EF.PriKey

There may be one or more key files with private keys. The following private keys are present:

Key name	KID
PIN (specific, digital signature PIN)	'81'
SK.CH.DS (digital signature key)	implicit or '84'
SK.CH.AUT (authentication key)	'82'
SK.CH.KE (key encipherment key)	'83'

Tab. 15: Keys and KIDs

NOTE - If other KIDs are used, then the KID has to be indicated in the SSD file or in the cryptographic token information.

5.3.3 EF.PubKey

In the optional public key file(s) for internal use, PKs are stored to be applied by the card for signature verification (e.g. the PKs of special communication partners may be installed in the personalisation phase). For the signature verification service the command VERIFY SIGNATURE has to be applied as specified in DIN V66291-1.

5.3.4 EF.PK.RCA.CS

In public key file for external use, PKs are stored to be applied by the outside world:

- PK.RCA.CS (verification key for X.509v3 certificates of the root CA)

5.3.5 EF.PK.CA.CS

In public key file for external use, PKs are stored to be applied by the outside world:

- PK.CA.CS (verification key for X.509v3 certificates of the CA issuing the certificates C_X509.CH.x)

5.3.6 EF.C_X509.CH.DS

The EF.C_X509.CH.DS contains the DS-certificate of the cardholder.

5.3.7 EF.C_X509.CA.CS

The EF.C_X509.CA.CS contains the X.509v3 certificate of the CA, which issued the cardholder certificates.

5.3.8 EF.C_X509.CH.AUT

The EF.C_X509.CH.AUT contains the AUT-X.509v3 certificate of the cardholder.

5.3.9 EF.SSD

The EF.SSD contains the security service descriptor templates as defined in annex E.

5.3.10 EF.DM

The EF.DM contains the display message. It should consist of 8 characters in ASCII Code and be pronounceable (have a meaning for the cardholder) so that the card holder can recognize it easily when it is displayed.

5.3.11 EF.PROT

The optional file EF.PROT with cyclic file structure serves as a logfile for signatures. It is to be seen as an aid for the user, but cannot be used as legal evidence. A record contains:

- Date (12 Bytes, JJJJMMTTHHMM),
- Terminal ID (18 Bytes, e.g. ICCSN.IFD in printable form),
- Document-ID (20 Bytes) and
- Signature counter (3 Bytes, digits).

ASCII shall be used as the coding scheme.

5.4 Application Selection

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

CLA	'00'
INS	'A4' = SELECT FILE
P1	'04' = DF selection by AID
P2	'0C' = No FCI to return
Lc	'06' = Length of subsequent data field
Data field	'D27600006601' = AID of DF.BSS, see tab. 1
Le	Empty

Tab. 16: SELECT FILE command for application selection with AID

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

Tab. 17: SELECT FILE response

5.5 Reading of EF.SSD

For reading the security service descriptors needed by the outside world the following commands are needed:

CLA	'00'
INS	'A4' = SELECT FILE
P1	'02' = EF file selection
P2	'0C' = No FCI to return
Lc	'02' = Length of subsequent data field
Data field	FID of EF.SSD, see annex C
Le	Empty

Tab. 18: SELECT FILE command for selecting EF.SSD

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

Tab. 19: SELECT FILE response

CLA	'00'
INS	'B0' = READ BINARY
P1,P2	'0000'
Lc	Empty
Data field	Empty
Le	'00' = Read until end-of-file

Tab. 20: READ BINARY command

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

Tab. 21: READ BINARY response

5.6 Authentication of the Cardholder

The DF.BSS contains a specific local PIN dedicated to protect the COMPUTE DIGITAL SIGNATURE operation.
For protection of the client/server authentication function and the key decipherment function a global PIN is used.
In a future version of this specification, biometrical user authentication may be added.

CLA	'00'
INS	'20' = VERIFY
P1	'00'
P2	'81' = Specific PIN/PW reference for DS, see tab. 15 '01' = Global PIN/PW reference for AUT+KE, see tab. 2
Lc	'0x' = Length of subsequent data field
Data field	PIN or PW (min 6, max 8 ASCII digits or characters) or PIN in Format 2 PIN Block (see ISO 9564-1), if this format is indicated in the SSD file.
Le	Empty

Tab. 22: VERIFY command for cardholder authentication

NOTE - The VERIFY command referencing the global PIN may be sent at any time after answer-to-reset.

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

Tab. 23: VERIFY response

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

The following Status Bytes are of special importance:

- '6300': Warning - verification failed (no further information
- '63Cx': Warning - verification failed (x indicates the number of further allowed retries)
- '6983': Checking error: authentication method blocked (these status bytes shall be delivered, if the VERIFY command is sent and the RC is zero).

5.7 Digital Signature Service

5.7.1 General Aspects

The digital signature function is in full compliance with DIN V66291-1. A digital signature process consists of several steps as shown in fig. 10.

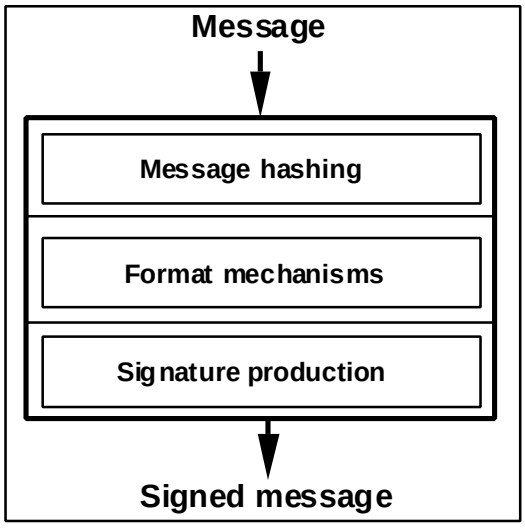

Fig. 10: Signature process

Step 1: Message hashing

The text to be signed has to be hashed by applying a hash function which is on the list of

the applicable hash functions (see annex A). The hash result can be considered as a compressed version of the text. The hashing will be done either

- outside the card or
- inside the card or
- partly outside and partly inside (last round of hashing) the card

depending on the card functionality and the software organization.

Step 2: Format mechanisms

Since the hash result may be shorter then the Digital Signature Input (DSI) for signature production, the hash value has to be padded, i.e. to be formatted. The padding, however, is not a simple filling up with a special padding character, since the construction of the DSI is highly security relevant. Several format mechanisms have been defined, some of them are evaluated as „weak“ and should not be used.

Step 3: Signature production

The final DSI is the string to be transformed by the signature algorithm. The result is the Digital Signature DS.

The command sequence for signature computation depends on the functionality of the respective card (cards without and with a hash algorithm as shown in fig. 11 and 12).

ICC Commands	Meaning
PSO: COMPUTE DS <hash value or digestinfo> → Digital Signature ←	Computation of a digital signature (command can be repeated)

Fig. 11: Digital signature phase – card without hash algorithm

ICC Commands	Meaning
PSO: HASH <Intermediate hash result || textblock> → OK ←	Hashing the last round in the card
PSO: COMPUTE DS → Digital Signature ←	Computation of a digital signature (command can be repeated)

Fig. 12: Digital signature phase – card with hash algorithm

5.7.2 Signature Computation without Hashing

The ISO/IEC 7816-8 command for digital signature computation is shown in tab. 24. If there is only one signature key, then this key may be implicitly selected (if not, the KID has to be set with the MSE command).
Signature algorithm and allowed Digital-Signature-Input formats (DSI formats) are outlined in annex A.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	‘2A’ = PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE
P1	‘9E’ = Return digital signature
P2	‘9A’ = Data field contains data to be integrated in the DSI
Lc	‘xx’ = Length of subsequent data field
Data field	Data to be integrated in the DSI: hash value or digestinfo, see annex A (the length of the data field shall not exceed 40% of the length of the modulus in case of RSA)
Le	‘00’ or ‘xx’ = Length of expected digital signature

Tab. 24: PSO: COMPUTE DS command

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

Tab. 25: PSO: COMPUTE DS response

5.7.3 Signature Computation with Hashing

Cards with a hash function (see annex A) can compute the final hash value by processing the last round(s) in the card. The ISO/IEC 7816-8 command for hashing is PSO: HASH.
The command can also be used for delivering the complete hash value to the card.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'2A' = PERFORM SECURITY OPERATION: HASH
P1	'90' = Compute hash value
P2	'A0' = Data field contains DOs relevant for hashing
Lc	'xx' = Length of subsequent data field
Data field	- If data shall be hashed: '90' - L - Intermediate hash result (see annex x) || '80' - L - Data to be hashed - If the final hash value shall be transmitted: '90' - L - Hash value
Le	Empty

Tab. 26: PSO: HASH command

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

Tab. 27: PSO: HASH response

After hashing the PSO: COMPUTE DIGITAL SIGNATURE command has to be sent.

CLA	As defined in ISO/IEC 7816-4 and -8
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
Lc	Empty
Data field	Empty (i.e. DSI already present in the card)
Le	'00' or 'xx' = Length of expected digital signature

Tab. 28: PSO: COMPUTE DS command

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

Tab. 29: PSO: COMPUTE DS response

5.7.4 Selection of different Algorithms and Input Formats

If the card supports several digital signature algorithms or different Digital Signature Input formats (e.g. according to PKCS #1 (default) and ISO/IEC 9796-2rnd (see annex A)), then the algorithm respectively the DSI format has to be selectable with the MSE command according to DIN V66291-1 (there are 2 ways described: selecting an SE or setting the AlgID; the first one is outlined in DIN V66291-1, the latter one is outlined subsequently).

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for computation
P2	'B6' = DST (CRT for digital signature)
Lc	'03' = Length of subsequent data field
Data field	'8001xx' = DO for AlgID, see tab. A.1
Le	Empty

Tab. 30: MANAGE SECURITY ENVIRONMENT command for selection of signature algorithm or DSI

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

Tab. 31: MANAGE SECURITY ENVIRONMENT response

In a similar way a hash algorithm may be selected with the MSE command, if the card supports more than one type of hash function.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	‘22’ = MANAGE SECURITY ENVIRON-MENT
P1	‘41’ = SET for computation
P2	‘AA’ = HT (CRT for hash code)
Lc	‘03’ = Length of subsequent data field
Data field	‘8001xx’ = DO for AlgID, see tab. A.1
Le	Empty

Tab. 32: MANAGE SECURITY ENVIRON-MENT command for selection of hash algorithm

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 33: MANAGE SECURITY ENVIRON-MENT response

5.7.5 Reading and Writing Protocol Records

First the file EF.PROT must be selected.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'A4' = SELECT FILE
P1	'02' = EF file selection
P2	'0C' = No FCI to be returned
Lc	'02' = Length of data field
Data field	FID of EF.PROT, see annex C
Le	Empty

Tab. 34: SELECT FILE command to select the protocol file

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

Tab. 35: SELECT FILE response

To read the last protocol record the following command has to be used:

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'B2' = READ RECORD
P1	'01' = Record no. 1 (last written record)
P2	'04' = Read record P1
Lc	Empty
Data field	Empty
Le	'xx' = Length of record

Tab. 36: READ RECORD command to read a protocol record

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

Tab. 37: READ RECORD response

Further protocol records can be read by setting the record number in P1 to a higher value.

To write a protocol record the following command has to be used:

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'E2' = APPEND RECORD
P1,P2	'0000'
Lc	'xx' = Length of record
Data field	Protocol data
Le	Empty

Tab. 38: APPEND RECORD command to write a protocol record

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

Tab. 39: APPEND RECORD response

5.7.6 Reading the Display Message

Signatures at a public customer service terminal require a mutual authentication with session key agreement for secure messaging (see annex D).
After the mutual authentication has been processed successfully according to clause 5.10, the commands SELECT FILE and READ BINARY must be used to read the display message (display text 8 bytes, ASCII coded), individually for each chipcard, which is shown on the display, so that the user also recognizes that the mutual authentication was successful.

Access to the 'Display Message' is only allowed after successful mutual authentication.

NOTE - Confirmation of the message by the user is considered sensitive, although no interaction with the card follows.

The commands SELECT FILE and READ BINARY in the SM-Mode are structured as follows:

CLA	'0C' = Command with SM
INS	'A4' = SELECT FILE
P1	'02' = EF file selection
P2	'0C' = No FCI to be returned
Lc	'xx' = Length of subsequent data field
Data field	'81 02 D0 00' || '8E'- L-'xx ... xx' = PV-DO with FID of EF.DM || CC-DO
Le	'00'

Tab. 40: SELECT FILE command to select the EF.DM

Data field	'99 02 90 00'|| '8E'-L-'xx ... xx' = Status-DO || CC-DO
SW1-SW2	'9000' or specific status bytes

Tab. 41: SELECT FILE response

CLA	'0C' = Command with SM
INS	'B0' = READ BINARY
P1,P2	'0000'
Lc	'xx'= Length of subsequent data field
Data Field	'97 01 08' || '8E'-L-'xx ... xx' = Le-DO || CC-DO
Le	'00'

Tab. 42: READ BINARY command to read the Display Message

Data field	'87'-L-'xx .. xx' || '8E'-L-'xx .. xx' = CG-DO || CC-DO
SW1-SW2	'9000' or specific status bytes

Tab. 43: READ BINARY response

Note: The command READ BINARY is also allowed after user authentication on private or company terminals. In this case SM is not applied.

5.7.7 Updating the Display Message

Changing the Display Message is only allowed after user authentication. It is done via the command UPDATE BINARY.

CLA	'0C' = Command with SM
INS	'D6' = UPDATE BINARY
P1,P2	'0000'
Lc	'xx' = Length of subsequent data field
Data field	'87'-L-'xx ... xx' || '8E'-L-'xx ... xx' = CG-DO || CC-DO
Le	'00'

Tab. 44: UPDATE BINARY command to change the Display Message

Data field	'99 02 90 00' || '8E'-L-'xx ... xx' = Status-DO || CC-DO
SW1-SW2	'9000' or specific status bytes

Tab. 45: UPDATE BINARY response

NOTE - The command UPDATE BINARY is also allowed on private or company terminals. In this case SM is not applied.

5.8 Authentication with CV Certificates

5.8.1 CV Certificate Verification

Before performing the authentication procedure, the certificate C_CV.ICC.AUT and possibly the certificate C_CV.CA.CS-AUT$_{ICC/IFD}$ read from the card before selecting the DF.BSS has to be verified in the IFD. It is assumed that the

- PK.RCA.CS-AUT$_{ICC/IFD}$ and possibly the
- PK.CA.CS-AUT$_{ICC/IFD}$ is present in the IFD.

The real procedure from the viewpoint of the card starts with verifying the CV-certificate of the IFD. The public key of the CA for verification, i.e. the PK.CA.CS-AUT$_{ICC/IFD}$ is expected to be in the card (if not then this key has to be delivered in a CV certificate which can be verified with the PK.RCA.CS-AUT$_{ICC/IFD}$ present in the card). For CV-certifcate verification the following commands have to be performed:

- MANAGE SECURITY ENVIRONMENT for setting the public key of the CA (i.e. PK.CA.CS-AUT$_{ICC/IFD}$)
- VERIFY CERTIFICATE for checking the CV-certificate presented by the IFD.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	‘22’ = MANAGE SECURITY ENVIRONMENT
P1	‘81’ = SET for verification
P2	‘B6’ = DST (CRT for digital signature)
Lc	‘0A’= Length of subsequent data field
Data field	‘8308 ...’ = DO for KeyRef of $PK.CA.CS\text{-}AUT_{ICC/IFD}$
Le	Empty

Tab. 46: MSE command

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 47: MSE response

After the $PK.CA.CS\text{-}AUT_{ICC/IFD}$ is set, then the VERIFY CERTIFICATE command is sent whereby command chaining is used (for command chaining, see ISO/IEC 7816-8).

a) Certificate with signature giving message recovery

CLA	'1x'
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	Length of subsequent data field
Data field	'5F37'-L-SIG.CA (see annex B)
Le	Empty

Tab. 48: PSO: VERIFY CERTIFICATE Command

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

Tab. 49: PSO: VERIFY CERTIFICATE Response

CLA	'0x'
INS	'2A' = PERFORM SECURITY OPERATION: VERIFY CERTIFICATE
P1	'00'
P2	'AE' (= Certificate in the data field, signed signature input consists of non-BER-TLV-coded data, i.e. the certificate content is a concatenation of DEs)
Lc	'xx'= Length of subsequent data field
Data field	'5F38'-L-PK-remainder (see annex B)
Le	Empty

Tab. 50: PSO:VERIFY CERTIFICATE Command

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

Tab. 51: PSO: VERIFY CERTIFICATE Response

b) Certificate with signature not giving message recovery

CLA	‘1x’
INS	‘2A’ = PERFORM SECURITY OPERATION: VERIFY CERTIFICATE
P1	‘00’
P2	‘AE’ = Certificate in the data field, signed signature input consists of non-BER-TLV-coded data, i.e. the certificate content is a concatenation of DEs
Lc	‘xx’= Length of subsequent data field
Data field	‘5F4E’-L-Certificate content (see annex B)
Le	Empty

Tab. 52: PSO: VERIFY CERTIFICATE command

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 53: PSO: VERIFY CERTIFICATE response

CLA	'0x'
INS	'2A' = PERFORM SECURITY OPERATION: VERIFY CERTIFICATE
P1	'00'
P2	'AE' = Certificate in the data field, signed signature input consists of non-BER-TLV-coded data, i.e. the certificate content is a concatenation of DEs
Lc	'xx' = Length of subsequent data field
Data field	'5F37'-L-CA signature (see annex B)
Le	Empty

Tab. 54: PSO: VERIFY CERTIFICATE command

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

Tab. 55: PSO: VERIFY CERTIFICATE response

5.8.2 Setting the Keys for Authentication

Before the INTERNAL/EXTERNAL AUTHENTICATE commands are performed the keys to be applied by the card for these commands have to be set.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'C1' = SET for int./ext. authentication
P2	'A4' = AT (CRT for authentication)
Lc	'xx'= Length of subsequent data field
Data field	'830Cxx ...xx' || '840101' = DO KeyRef of PK.IFD.AUT, i.e. ICCSN.IFD || DO KeyRef of SK.ICC.AUT, see tab. 2
Le	Empty

Tab. 56: MSE command

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

Tab. 57: MSE response

5.8.3 ICC/IFD Authentication

For ICC/IFD authentication (see annex D) the card has to perform the INTERNAL AUTHENTICATE command in a first step.

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

Tab. 58: INT. AUTHENTICATE command

Data field	- for RSA: Digital signature (DSI format ISO/IEC 9796-2, see DIN V66291-1, tab. D.1: '6A' || PRND1 || K1 || h(PRND1 || K1 || RND.IFD || ICCSN.IFD, least significant 8 bytes) || 'BC' with h = SHA-1) - for other algorithms: see annex D
SW1-SW2	'9000' or specific status bytes

Tab. 59: INT. AUTHENTICATE response

After the card has been successfully authenticated, the commands

- GET CHALLENGE and
- EXTERNAL AUTHENTICATE

are send to the card.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'84' = GET CHALLENGE
P1, P2	'0000'
Lc	Empty
Data field	Empty
Le	'08' = Length of expected RND

Tab. 60: GET CHALLENGE command

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

Tab. 61: GET CHALLENGE response

After GET CHALLENGE the command EXTERNAL AUTHENTICATE follows, which delivers the digital signature of the IFD to the card.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'82' = EXTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'xx' = Length of subsequent data field
Data field	Authentication related data: - For RSA: digital signature of IFD (DSI format ISO/IEC 9796-2, see DIN V66291-1, tab. D.1: '6A' || PRND2 || K2 || h(PRND2 || K2 || RND.ICC || ICCSN.ICC, least significant 8 bytes) || 'BC' with h = SHA-1) - For other algorithms: see annex D
Le	Empty

Tab. 62: EXT. AUTHENTICATE command

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

Tab. 63: EXT. AUTHENTICATE response

After successfull authentication of the external entity the respective security status is set in the card.

5.9 Client/Server Authentication

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

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

This specification covers only the case, where the card performs a digital signature computation applying the private key for authentication in an INTERNAL AUTHENTICATE command to the authentication input contained in the data field of the command after formatting the input. In case of RSA, the authentication input T is formatted according to PKCS #1, Version 2.0:

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

where PS is an octet string consisting of octets with value 'FF'. The length of PS must be at least 8 octets. The formatted octet string must consist of k-1 octets where k is the length in octets of the modulus of the private key for authentication.

The other parts on the local side have to be performed by the software in the cardholders system.

Before the INTERNAL AUTHENTICATE command can be executed, the related SK has to be selected.
Furthermore, the related algorithm reference denoting the type of authentication protocol may be set additionally.

CLA	'00'
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for internal authentication
P2	'A4' = AT (CRT for authentication)
Lc	'xx'= Length of subsequent data field
Data field	'84 01 82' [|| '80 01 xx'] = DO KeyRef denoting SK.CH.AUT, see tab. 15 [|| DO AlgID, value see table A.2]
Le	Empty

Tab. 64: MANAGE SECURITY ENVIRONMENT command

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

Tab. 65: MSE response

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'88' = INTERNAL AUTHENTICATE
P1	'00'
P2	'00'
Lc	'xx' = Length of subsequent data field
Data field	For RSA: - DigestInfo, used for KERBEROS - H_MD5 || H_SHA1, used for SSL/TLS - H_SHA1, used for WTLS and NETSCAPE The formatting of the authentication input is according to PKCS#1 (the length of the data field shall not exceed 40% of the length of the modulus) For other algorithms: Hash value
Le	'00' or 'xx' = length of expected digital signature

Tab. 66: INT. AUTHENTICATE command

NOTE - The use of the key SK.CH.AUT requires, that the global PIN has been successfully presented.

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

Tab. 67: INT. AUTHENTICATE response

5.10 Encryption Key Decipherment

The key decipherment service is used for the following security protocols:

? Application level encryption key decipherment
? Application level key agreement
? Key agreement or key exchange in PK Kerberos' preauthentication phase

In this specification it is decribed how this key decipherment service can be used for confidential document exchange.

5.10.1 Key Management with RSA

For confidential document exchange, the following scheme is applied:

- key transport is organized by enciphering the content encryption key with the receiver's PK
- document enciphering with a symmetrical algorithm (e.g. DES-3).

If an enciphered document is sent, the card is not involved: the software computes the content encryption key, enciphers the document and finally enciphers the content encryption key by applying the receiver's public key taken from the receiver's KE certificate.

This specification covers only the case, where the card performs a key decryption applying the private key for key decryption in a DECIPHER command to the cryptogram contained in the data field of the command and retrieving the key from the formatted clear text after decryption. The encryption key K may be formatted according to PKCS #1, Version 2.0 or on the basis of ISO/IEC 9796-2.
Formatting according to PKCS #1:

'02' || PS || '00' || K

where PS is an octet string consisting of pseudorandomly generated nonzero octets. The length of PS must be at least 8 octets. The formatted octet string must consist of k-1 octets where k is the length in octets of the modulus of the private key for decryption.
Formatting based on ISO/IEC 9796-2:

'60 00 … 00 01' || RND (8 bytes) || K || 'BC'

The formatted octet string must consist of k octets where k is the length in octets of the modulus of the private key for decryption.

If an enciphered document is received the card has to be applied for deciphering the content encryption key.

Before key decipherment can be performed, the secret key has to be selected with the MSE command.

CLA	'00'
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for decipherment
P2	'B8' = CT (confidentiality template)
Lc	'xx' = Length of subsequent data field
Data field	'840183' [|| '8001xx'] = DO key reference of SK.CH.KE, see tab. 15 [|| DO AlgId, see tab. A.2]
Le	Empty

Tab. 68: MANAGE SECURITY ENVIRONMENT command with SET option

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

Tab. 69: MSE response

After the key is set, the decipher operation can be executed.

CLA	'00'
INS	'2A' = PERFORM SECURITY OPERATION: DECIPHER
P1	'80' = Return plain value
P2	'86' = Enciphered data present in the data field
Lc	'xx' = Length of subsequent data field
Data field	Padding indicator byte followed by cryptogram (PI coding and cryptogram format see tab. A.4)
Le	'00' or 'xx' = Length of content encryption key

Tab. 70: PERFORM SECURITY OPERATION command with DECIPHER operation

NOTE - The use of the key SK.CH.KE requires, that the global PIN has been successfully presented.

Data field	Content encryption key
SW1-SW2	'9000'or specific status bytes

Tab. 71: DECIPHER response

5.10.2 Diffie-Hellman

The requirements for the card are based on the ephemeral mode of the Diffie-Hellman key agreement method, see RFC 2631, as referenced in CMS (Cryptographic Message Syntax). It can be summarized as follows:

- The document is encrypted by a content-encryption key, which is generated at random.
- The recipient's public key and the sender's private key are used to generate a symmetric key, then the content-encryption key is encrypted in this symmetric key.
- The recipient can use his/her private key together with the sender's public key to generate the same symmetric key, which was used for encryption of the content-encryption key

It is assumed that the recipient has a certificate for his/her public key, but the sender generates an ephemeral key pair each time he/she wants to communicate to the recipient. If a document is encrypted by the sender, the card is not involved.
The card supports this scheme by deriving the shared symmetric key from the recipient's secret key and from the sender's public key.
This shared symmetric key is stored in the card. Subsequently, the card computes hash values of this shared symmetric key concatenated with additional data. These hash values are used for deriving the key-encryption key. This derivation of the key-encryption key and the decryption of the content-encryption key are done in software.

The notations of the RFC 2631 are used: The sender's public key ya is sent to the card. The card derives the shared value ZZ = ya^{xb} mod p. The parameter p is the recipient's public modulus. ZZ is stored in the card. The card computes subsequently the keying material, which is the hash value H(ZZ || OtherInfo). The additional data to be hashed (OtherInfo) have to be delivered to the card. This command is repeated with different additional data to be hashed (OtherInfo) until enough bytes are produced for deriving the key-encryption key.

NOTES
- Care has to be taken in order to protect the recipient's secret key against the so called "small-subgroup" attacks, see RFC 2785.
- If the sender wants to use the same key pair more than once, it is highly recommended that the card should also take part in the encryption on the sender's side.

First, the secret key SK.CH.KE has to be selected with the MSE command:

CLA	'00'
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'41' = SET for decipherment
P2	'B8' = CT (confidentiality template)
Lc	'xx' = Length of subsequent data field
Data field	'840183' [|| '80010B'] = DO Key reference of SK.CH.KE, see tab. 15 [|| DO Algorithm reference, see tab. A.3]
Le	Empty

Tab. 72: MANAGE SECURITY ENVIRONMENT command with SET option

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

Tab. 73: MSE response

After the key is set, the derivation of the shared secret can be done. The computed value ZZ is kept in the card and is used by the subsequent HASH command.

CLA	'00'
INS	'2A' = PERFORM SECURITY OPERATION: DECIPHER
P1	'80' = Return plain value
P2	'A6' = Decipher input template
Lc	'xx' = Length of subsequent data field
Data field	'9C'-L -xxxx = DO PK of sender
Le	Empty

Tab. 74: PERFORM SECURITY OPERATION command with DECIPHER operation

NOTE - The use of the key SK.CH.KE requires, that the global PIN has been successfully presented.

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

Tab. 75: DECIPHER response

NOTE - The tag 'A6' will be proposed to ISO by DIN.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'2A' = PERFORM SECURITY OPERATION: HASH
P1	'90' = Compute hash value
P2	'A0' = Data field contains DOs relevant for hashing
Lc	'xx' = Length of subsequent data field
Data field	'8000' || '80'-L-xxx = DO PV with length 0, i.e. data already in the card || DO PV contaning additional information to be hashed, i.e. hash input = ZZ as internal data || OtherInfo as value of DO PV
Le	'xx' = Length of hash value

Tab. 76: PSO: HASH command

NOTE - The HASH command shall not be executed, if no data referenced with '8000' are available, i.e. if no DECIPHER command was executed immediately before.

Data field	Hash value representing the key material
SW1-SW2	'9000' or specific status bytes

Tab. 77: PSO: HASH response

5.11 Reading X.509 Certificates and the Public Key of CAs

The subsequent tables show the commands for reading such objects.

CLA	'00'
INS	'A4' = SELECT FILE
P1	'02' = EF file selection
P2	'0C' = No FCI to return
Lc	'02' = Length of subsequent data field
Data field	FID (see annex C) of one of the files: - EF.C.CH.DS - EF.C.CA.CS - EF.C.CH.AUT - EF.PK.RCA.CS - EF.PK.CA.CS
Le	Empty

Tab. 78: SELECT FILE command

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 79: SELECT FILE response

CLA	‘00’
INS	‘B0’ = READ BINARY
P1,P2	‘0000’ or offset
Lc	Empty
Data field	Empty
Le	‘xx’ = Length of data to be read or '00' = read until end-of-file

Tab. 80: READ BINARY command for reading certificates or public keys

Data field	Certificate or PublicKeyInfo
SW1-SW2	‘9000’ or specific status bytes

Tab. 81: READ BINARY response

5.12 Change of Reference Data

The CHANGE RD command for changing the global PIN may be used at any time convenient for the cardholder. The change of the specific PIN can only be performed after selection of DF.BSS.

CLA	'00'
INS	‘24’ = CHANGE REFERENCE DATA
P1	‘00’ = Exchange reference data
P2	‘81’ = Specific PIN/PW reference for DS, see tab. 15 '01' = Global PIN/PW reference for AUT+KE, see tab. 2
Lc	‘xx’ = Length of subsequent data field
Data field	- Existing reference data (6 to 8 bytes) followed by new reference data (6 to 8 bytes, ASCII coding) or - coding of the existing and new reference data according to the Format 2 PIN Block, if indicated in the SSD file or in the CTI
Le	Empty

Tab. 82: CHANGE RD command

The length of the existing Reference Data is known in the card, so that neither a delimiter nor padding for filling up fixed formats is necessary.

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 83: CHANGE RD response

5.13 Reset of RC and Setting a new PIN or Password

After three subsequent false presentations of the PIN or password, the RC has the value 0 and does not allow further usage of the protected functions (see fig. 2).

With the ISO/IEC 7816-8 command RESET RETRY COUNTER, the cardholder can initiate the reset of the RC to its initial value 3. The Resetting Code shall have a minimum length of 6 bytes (digits delivered as ASCII characters).

It is also possible to define a new PIN or password with the RESET RC command.

The support of this command is mandatory.

CLA	‘00’
INS	‘2C’ = RESET RETRY COUNTER
P1	‘00’ = Reset retry counter and set new reference data ‘01’ = Reset retry counter
P2	‘81’ = Specific PIN/PW reference for DS, see tab. 15 ‘01’ = Global PIN/PW reference for AUT+KE, see tab. 2
Lc	‘xx’ = Length of subsequent data field
Data field	- if P1 = ‘01’: Resetting code (min 6 bytes) - if P1 = ‘00’: Resetting code (min 6 bytes) followed by new reference data in ASCII coding (6 to 8 bytes) or in format 2 PIN Block (see ISO 9564-1), if this format is indicated in the SSD file or in the CTI.
Le	Empty

Tab. 84: RESET RETRY COUNTER command

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 85: RESET RC response

After successfull presentation of the Resetting Code, the retry counter is automatically reset to its initial value (3).

6 Cryptographic Token Information

6.1 Scope

This clause specifies the file structure and file content for DF.PKCS#15 on the basis of PKCS#15: Cryptographic Token Information Format Standard, 3rd Draft v.1.1.

6.2 File Structure

The following figure shows the file structure of DF.PKCS#15.

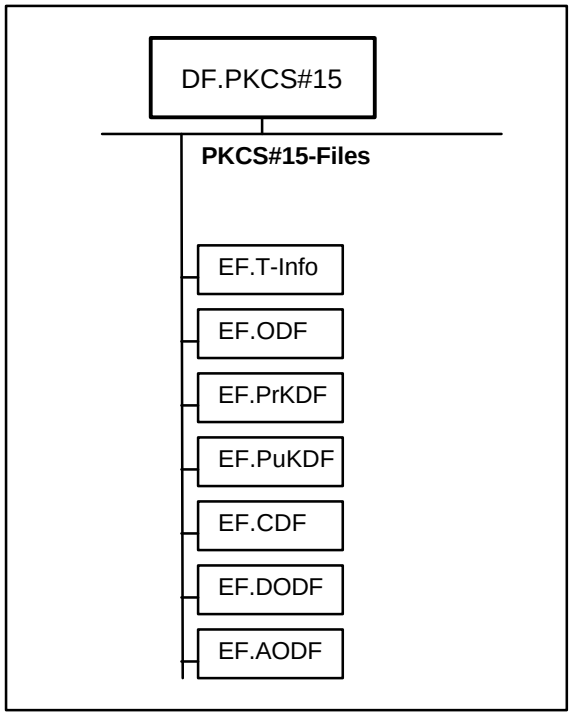

Fig. 13: DF.PKCS file structure

6.2.1 EF.T-Info

The EF.T-Info contains the token information.

6.2.2 EF.ODF

The EF.ODF contains the object directory information.

6.2.3 EF.PrKDF

The EF.PrKDF contains the private key directory information.

6.2.4 EF.PuKDF

The EF.PuKDF contains the public key directory information.

6.2.5 EF.CDF

The EF.CDF contains the certificate directory information.

6.2.6 EF.DODF

The EF.DODF contains the data object directory information.

6.2.7 EF.AODF

The EF.AODF contains the authentication object directory information.

6.3 Application Selection

The ISO/IEC 7816-4 command for ‘Direct Application Selection’ is shown subsequently.

CLA	‘00’
INS	‘A4’ = SELECT FILE
P1	‘04’ = DF selection by AID
P2	‘0C’ = No FCI to return
Lc	‘0C’ = Length of subsequent data field
Data field	'A000000063504B43532D3135' = AID of DF.PKCS#15, see tab. 1
Le	Empty

Tab. 86: SELECT FILE command for application selection with AID

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 87: SELECT FILE response

6.4 Reading Cryptographic Token Information

For reading the cryptrographic token information the ISO/IEC 7816-4 commands SELECT FILE and READ BINARY are used.

CLA	‘00’
INS	‘A4’ = SELECT FILE
P1	‘02’ = EF file selection
P2	‘0C’ = No FCI to return
Lc	‘02’ = Length of subsequent data field
Data field	FID (see annex C) of one of the files: - EF.T-Info - EF.ODF - EF.PrKDF - EF.PuKDF - EF.CDF - EF.DODF - EF.AODF
Le	Empty

Tab. 88: SELECT FILE command for EF selection

Data field	Empty
SW1-SW2	‘9000’ or specific status bytes

Tab. 89: SELECT FILE response

CLA	‘00’
INS	‘B0’ = READ BINARY
P1,P2	‘0000’
Lc	Empty
Data field	Empty
Le	‘00’ = Read until end-of-file

Tab. 90: READ BINARY command for reading cryptographic token information

Data field	Cryptographic token information
SW1-SW2	‘9000’ or specific status bytes

Tab. 91: READ BINARY response

Annex A
(normative)

Digital Signature Formats, AlgIDs and OIDs

1 Hash-Input-Formats

When hashing is performed partly inside and partly outside the card, the data field of the command PSO: HASH for the hash algorithms SHA-1 and RIPEMD-160 should be structured as follows:

a) PSO:HASH without command chaining

- DO '90' with 20 Bytes intermediate hash value followed by a counter of 8 bytes (number of bits already hashed)
- DO '80' contains the text still to be hashed without padding (length of text up to 64 bytes)

b) PSO:HASH with command Chaining

- first command: DO '90' (28 bytes, structure as in a)
- second to $(n-1)^{th}$ command: DO '80' with 64 bytes text each
- n^{th} command: DO '80' with the last textblock without padding (length of text up to 64 bytes)

INS-P1-P2 stay unchanged for all chained commands. The coding of the CLA byte is described in chapter 22.1.

2 Formats of the Digital Signature

In the following text SigG conform DSI formats of the digital signature are described for different signature algorithms. It is mandatory to support at least one of these formats, i.e. if only the RSA algorithm is available in a card, then the card must be able to compute a signature on the basis of the SigG formats for RSA. Optionally the card may be able to support other signature formats.

The given DSI formats describe the string, that is used directly by the signature algorithm to compute the digital signature.

2.1 RSA

2.1.1 DSI according to ISO/IEC 9796-2 with Random Number

The DSI format based on ISO/IEC 9796-2 and integrating a random number has the following structure:

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

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

2.1.2 DSI according to PKCS #1

The DSI format according to PKCS #1 has the following structure:

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

The DigestInfo to be delivered to the card when using this signature format has the following coding:

a) SHA-1 with OID: ? 1 3 14 3 2 26 ?

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

b) RIPEMD-160 with OID: ? 1 3 36 3 2 1 ?

DigestInfo: 3021 3009 06052B24030201 0500 0414 || hash value (20 bytes)

2.1.3 Further RSA-DSI-Formats

Further RSA DSI formats may be added when necessary.

2.2 DSA

The DSI consists of the hash value calculated using SHA-1 or RIPEMD-160.

2.3 Elliptic Curves

The DSI consists of the hash value which was calculated using SHA-1 or RIPEMD-160. If the DSI is longer than the hash value (e.g. if q is longer than 160 bits), then the DSI should be filled with leading zero bits.

3 Signature Certificates

Signature certificates are provided in transparent files (see Annex C). The verification of a DS certificate is carried out by the recipient of a signed document. Security software is usually used for this. The structure of a DS certificate (see interoperability specification for signature certificates) is not known to the card.

4 Structure of the PK-Information in EF.PK.RCA.DS

Public keys of root certification authorities for different security services (signature function according to SigG, ...) are stored in the file EF.PK.RCA.DS. Public keys for the verification of CV certificates are not stored in this file.
For the signature function according to SigG the file shall contain at least one public key of the root certification authority (RegTP) as a 'security anchor' (e.g PK from 1999 and maybe PK from 2000).
The coding of the respective data fields complies with those in a certificate according to ISO/IEC 9594-8 (ITU X.509). In DIN V66291-1 a coding example is presented.

5 Security Environments and Headerlists

Cards supporting only one security environment, one signature algorithm and one DSI format, use SE #0 implicitly. For cards which support more than one signature algorithm or several RSA.DSI formats, it can be described in the signature template of the SSD file which algorithm complies with which security environment. SE #1 is default. The use of headerlists for a card internal description of different DSI formats is outlined in DIN V66291-1.

6 Algorithm Identifier

6.1 Algorithm Identifier for COMPUTE DS

If the first nibble is set to 0, it means that no PS: HASH command is sent to the card prior to the command PSO: COMPUTE DS.
If the first nibble is not 0, it means that a PS: HASH command shall be sent to the card prior to the command PSO: COMPUTE DS and that the card is managing the DSI including setting of OIDs or associated value, if needed.

AlgID	Meaning
'0x'	No hash-function
'1x'	SHA-1, OID = ? 1 3 14 3 2 26 ?
'2x'	RIPEMD-160, OID = ? 1 3 36 3 2 1 ?
'x1'	RSA with DSI acc. to ISO/IEC 9796-2 with RND (format see Annex A 2.1.1, parameter hash value, if x = 0)
'x2'	RSA with DSI acc. to PKCS #1 (format see Annex A 2.1.2, parameter DigestInfo, if x = 0)
'x3'	DSA (parameter hash value, if x = 0)
'x4'	ELC (parameter hash value, if x = 0)

Table A.1: AlgIDs for hash-functions and signature algorithms for COMPUT DS

6.2 Algorithm Identifier for INTERNAL AUTHENTICATE

Tab. A.2 shows the AlgIds relevant for INTERNAL AUTHENTICATION.

AlgID	Meaning
'0x'	No hash-function
'1x'	SHA-1, OID = ? 1 3 14 3 2 26 ?
'2x'	RIPEMD-160, OID = ? 1 3 36 3 2 1 ?
'x1'	RSA with DSI acc. to ISO/IEC 9796-2 with RND (format see Annex A 2.1.1, parameter hash value, if x = 0)
'x2'	RSA with DSI acc. to PKCS #1 (format see Annex A 2.1.2, parameter DigestInfo, if x = 0)
'x3'	DSA (parameter hash value, if x = 0)
'x4'	ELC (parameter hash value, if x = 0)
'05'	RSA framing according to PKCS#1 without OID
'17'	RSA with SHA-1and session key exchange
'18'	DSA with SHA-1 and DH session key agreement
'19'	ELC with SHA-1 and DH session key agreement

Table A.2: AlgIDs for hash-functions and signature algorithms for INTERNAL AUTHENTICATE used for device authentication and client/server authentication

6.3 Algorithm Identifier for DECIPHER

Tab. A.3 and A.4 show the AlgIds relevant for decipherment.

AlgID	Meaning
'0x'	x?A, Padding method not indicated
'0A'	RSA , PI = '00'
'1A'	RSA , PI ? '00', see tab. A.4
'0B'	DH Key Agreement

Table A.3: AlgIDs for decipherment

PI	KEI	Specification
'00'	No further information	ISO/IEC 7816-4
'81'	'02' || RND (all bytes ? 0, number key length dependend) || '00' || document cipher key	PKCS #1, V2.0
'82'	'60 00 .. 01' || RND (8 bytes) || document cipher key || 'BC')	ISO/IEC 9796-2

Table A.4: Key Encipherment Input Formats

7 Object Identifiers

In the following table only OIDs are listed, which are relevant for the card.

OID ASN.1-Syntax	OID Autho-rity	Name Objects	Relevant for		
			COMP. DS	VERIFY CERT	INT/EXT AUTH
?1 2 840 113549 1 1 5?	rsadsi	sha1WithRSAENC – SHA1 with RSA and PKCS#1 padding	x	(x)	
?1 3 36 3 3 1 2?	TTT	rsasignatureWithripemd160 – RSA with RIPEMD160 and PKCS#1 pad-ding	x	(x)	
?1 3 36 3 4 2 2 1?	TTT	sigS_ISO9796-2Withsha1 – SHA1 with RSA and DSI according to ISO/IEC 9796-2 and trailer 'BC'		x	
?1 3 36 3 4 2 2 2?	TTT	sigS_ISO9796-2Withripemd160 – RIPEMD160 with RSA and DSI according to ISO/IEC 9796-2 and trailer 'BC'		(x)	
?1 3 36 3 4 3 2 1?	TTT	sigS_ISO9796-2rndWithsha1 - SHA1 with RSA and DSI according to ISO/IEC 9796-2 with random number and trailer 'BC'	x		
?1 3 36 3 4 3 2 2?	TTT	sigS_ISO9796-2rndWithripemd160 - RIPEMD160 with RSA and DSI according to ISO/IEC 9796-2 with random number and trailer 'BC'	x		
?1 3 14 3 2 27?	OIW	DSAWithSHA1	x	x	
?1 2 840 10045 1?	ANSI	ecdsa-with-SHA1 – Elliptic curves digital signature algorithm with SHA1	x	(x)	
?1 3 36 3 3 2 1?**	TTT	ecSignWithsha1 – Elliptic curve with SHA1	x	x	
?1 3 36 3 3 2 2?**	TTT	ecSignWithripemd160 – Elliptic curve with RIPEMD160	x	(x)	
?1 3 36 7 2 1 1?*	TTT	ktEncsigS_ISO9796-2WithrsaWith-sha1 – key transport with RSA encryption and RSA signature with DSI according to ISO9796-2 and SHA1			x
?1 3 36 7 1 2 1 1?*	TTT	kaDHWithdsaWithsha1 – Diffie-Hell-man key agreement with DSA and SHA1			x
?1 3 36 7 1 2 2 1?**	TTT	kaDHWithecWithsha1 – Diffie-Hell-man key agreement with elliptic curve and SHA1			x

Table A.5: OIDs relevant for signatures acc. to SigG/SigV, signatures for CV certificates and PK-based device authentication with key transport or key agreement

NOTE - The OIDs in brackets will not be used at the moment (see Table B.10).

*) = To be finally assigned by TTT
**) = Meaning and coding to be precised and OID to be finally assigned by TTT

Annex B
(normative)

Authentication Certificates

1 Structure of Authentication Certificates

1.1 Certificate-Data Elements

The following table shows data elements relevant to CV certificates (Card Verifiable Certificates) and thus for terminal and card authentication certificates.

Tag	Data element	Defined in
'7F21'	CV certificate, constructed, e.g. cardholder certificate	ISO/IEC 7816-6/8
'5F4E'	Certificate content	ISO/IEC 7816-8
'5F29'	Interchange profile descriptor, e.g. Certificate Profile Identifier (CPI)	ISO/IEC 7816-6
'42'	Certification authority reference (CAR), e.g. name or ID of issuer authority	ISO/IEC 7816-6/8
'5F20'	Certificate holder reference (CHR), e.g. cardholder name or ICCSN	ISO/IEC 7816-6/8
'5F49'	Certificate holder public key, e.g. PK.IFD (primitive DO)	ISO/IEC 7816-6/8
'7F49'	Certificate holder public key, e.g. PK.IFD (constructed DO)	ISO/IEC 7816-8 (proposed)
'06'	Object Identifier (OID) for signature algorithm of issuer and certificate holder	ISO/IEC 7816-6
'5F4A'	Public Key of CA or its reference (Authority Key Identifier AKI)	ISO/IEC 7816-6
'5F4B'	Certificate holder authorisation (CHA)	ISO/IEC 7816-9
'5F37'	Signature of a certificate, produced by the related CA	ISO/IEC 7816-6/8
'5F38'	PK-Remainder	ISO/IEC 7816-6

Table B.1: Interindustry DOs for CV Certificates

In application specific contexts further data elements may be present, such as

- Expiry Date
- Validity Label

The expiry date delimits the period in which the public key is bound to the key holder (cannot be checked in a card at the moment).
The validity label denotes the actual status of a valid certificate by means of a number. The use of these data elements will be described in a later version of this specification.

1.2 Certificate Profile Identifier

The certificate profile identifier (CPI) delineates the exact structure of an authentication certificate. It can be used as a card internal identifier of the relevant headerlist. Thus several headerlists can be supported and the matching headerlist to an incoming certificate can be found dynamically. A certificate headerlist describes the concatenation of DEs within one certificate. The coding scheme for CPI is shown in tab. B.2.

CPI Coding	Meaning
'00'	RFU
'01' - '7E'	Standardized CPIs
'80' - 'FE'	Application specific CPIs
'FF'	Escape, subsequent bytes contain the CPI

Tab. B.2: CPI Coding

1.3 Certification Authority Reference (Authority Key Identifier)

The „Certification Authority Reference“ (CAR) has the purpose of identifying the certificate issuing CA in such a way that the DE can be used at the same time as an authority key identifier. The CAR consists of

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

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

Table B.3: Structure of the Certification Authority Reference (Authority Key Identifier)

The extension has the following structure:

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

Table B.4: Structure of the extension for key referencing

The Service Indicator (Key usage) has the value 1 = entity authentication (see also Annex C, Table C.1).
The Discretionary Data may have a value at the discretion of the related CA.
The algorithm Reference/Id can be individually assigned by a CS for distinguishing different PK algorithms.
The Date consists of the last two digits of the year, in which the key pair for certificate signing was produced. If in one year more than one keypair has been generated for the same algorithm, the keypairs may be distinguished by using the discretionary data field.

1.4 Certificate Holder Reference (Subject Key Identifier)

The „Certificate Holder Reference“ (CHR) is used to denote the certificate holder uniquely in such a way that the DE can be used as an subject key identifier to reference the PK of the certificate holder. The field has a fixed length of 12 bytes. The value depends on the entity for which the certificate is issued. The CHR consists of

- CV-certificate for CS: CA Reference CAR (5 Bytes) || extension for key referencing (3 Bytes)
- CV-certificate for ICC and IFD: Filler (0-4 Bytes) || ICCSN (8-12 Bytes)

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

Table B.5: Structure of the Certificate Holder Reference, if Certificate Holder is a CA

A filler-byte is coded '00'.

The "Extension for key referencing“ has the same structure as shown in table B.3. The field "date“ contains the last two digits of that year, in which the PK certified in the CV-certificate (PK.CA.CS-$AUT_{ICC/IFD}$) was issued.

Filler (0-4 B)	ICCSN.ICC or ICCSN.IFD (8-12 B)

Table B.6: Structure of Certificate Holder Reference, if certificate holder is a card

During verification of the CV certificate the PK is stored temporarily, whereby the unique certificate holder reference is used as key reference (for the certified PK) in the command MSE.

1.5 Certificate Holder Authorisation

The „Certificate Holder Authorisation“ (CHA) is a general DE in a CV certificate, which can be used to identify (i.e. access-) rights of the certificate holder. The meaning of CHA can be compared with a role based key identifier when applying symmetrical authentication algorithms.

The CHA-DE consists of

- a prefix, which denotes the entity assigning the authorisation (role id) (to the card holder) or the application reference/ID for which the authorisation is valid and
- the role identifier.

Prefix (AID of related DF, see tab.1) (6 B)	Role ID (1B)

Table B.7: Structure of Certificate Holder Authorisation

NOTE - The coding of the Role ID should be organized as flag list, so that more than one role or authorisation can be encoded.

The table below shows the CHA values relevant to the applications DF.IDD, DF.BSS respectivly DF.SIG and DF.ASS.

CHA (AID || Role ID)	Meaning	Used in			
		C_CV.CA. AUT	C_CV.ICC. AUT	C_CV.IFD. AUT	C_CV.IVS. AUT
'D2760000660100'	CHA with no access right to data	X	X		
'D2760000660101'	CHA for proving the right for read access to EF.DM (CHA used by IFD, i.e. a customer service terminal)			X	
'D2760000660102'	CHA for proving the right for read/write access to certificate files and EF.DM, if present (CHA used by IFD of a CA)			X	
'D2760000660301'	CHA for proving the right for read/write access to EF.BD/SP				X
'D2760000660303'	CHA for proving the right for read/write access to EF.BD/ED/SP				X

Table B.8: CHA coding for the applications DF.IDD and DF.BSS (respectivly DF.SIG and DF.ASS)

1.6 Object Identifier for Signature Algorithm of the Certificate Holder

The Object Identifiers are described in tab. A.5.

1.7 PK of Certificate Holder

1.7.1 General Construction

The Public Key PK.IFD.AUT (or PK.ICC.AUT) in a CV certificate consists of a concatenation of parameters. Which PK parameters at what length are present in the CV certificate can be described in the DO PK (Tag '7F49', constructed) in a certificate specific headerlist. The tags of the parameters are always context sensitive (starting with '81'). Their interpretation depends on the algorithm identified by the OID. The values of the parameters have to be coded as an octet string.

1.7.2 Public Key RSA

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

1.7.3 Public Key DSA

- Tag '81': p prime (length L.p)
- Tag '82': q prime that divides p-1 (length 20 bytes)
- Tag '83': g element of order q (length L.p)
- Tag '84': y public key, y = g exp(x) mod p (length L.p)

1.7.4 Public Key ELC

1. Public Key ELC-p:

An elliptic curve E over a field F with p elements, p prime (p > 3), is given by the equation

$Y^2 = X^3 + aX + b$ in F with a,b ? F.

Let k be the smallest number with p ? 2^{8k}. L.F is a number ? k.
The number p and the elements in F are represented as byte sequence of length L.F, whereby the leading (L.F-k) bytes are zero bytes. A point Q on E (except the point at infinity) is represented as the concatenation of the sequence of bytes which represents the x-component of Q, followed by the sequence of bytes which represents the y-component of Q. The length of this concatenated sequence is equal to 2*L.F.

- Tag '81': p prime (length L.F)
- Tag '82': a coefficient of curve (length L.F)
- Tag '83': b coefficient of curve (length L.F)
- Tag '84': generator-point PB, PB is point of curve (length 2*L.F)
- Tag '85': q prime, order of generator-point PB (length L.F)
- Tag '86': public key point PP, PP= x*PB, PP is point of curve (length 2*L.F)

2. Public Key ELC-2:

An elliptic curve E over a field F with 2^m elements is given by the equation

$Y^2 + XY = X^3 + aX^2 + b$ with a,b ? F.

Let k be the smallest number with m ? 8k. L.F is a number ? k.
An element in Z is represented as the concatenation of a sequence of zero bytes, length (L.F-k), followed by z, represented as bytes sequence of length k with respect to a basis as described in ANSI X.9.62. A point Q on E (except the point at infinity) is represented as the concatenation of the sequence of bytes which represents the x-component of Q, followed by the sequence of bytes which represents the y-component of Q. The length of this concatenated sequence is equal to 2*L.F.
The number m is represented as byte sequence of length L.m.

case a, optimal normal basis:

- Tag '87': m (length L.m)
- Tag '82' - '86': as above

case b, polynominal basis with respect to a trinomial polynome:

- Tag '82' - '87': as in case a
- Tag '88': value k of trinomial polynome X^m+X^k+1 (length L.m)

case c, polynomial basis with respect to a pentanomial polynome:

- Tag '82' - '87': as in case a
- Tag '89': value k1 of pentanomial polynome $X^m+X^{k3}+X^{k2}+X^{k1}+1$ (length L.m)
- Tag '8A': value k2 of pentanomial polynome $X^m+X^{k3}+X^{k2}+X^{k1}+1$ (length L.m)
- Tag '8B': value k3 of pentanomial polynome $X^m+X^{k3}+X^{k2}+X^{k1}+1$ (length L.m)

1.8 PK Remainder

In reversible algorithms a DSI format with message recovery can be used. That means, the recovered message is then the complete certificate with the exception of that part of the PK, which had to be exported for memory reasons. This part is called PK Remainder (PK-Rest, Tag '5F38').

1.9 Signature-Formats for the CA

The following signature formats are relevant for the command VERIFY CERTIFICATE.

1.9.1 RSA

a) **DSI according to ISO/IEC 9796-2 for RSA**

In the DSI for CV certificates based on RSA no random padding is needed; as they are not created dynamically, but are only used for verification. Therefore the original DSI format according to ISO 9796-2 can be used:

- Header: 2 bits (= 01)
- More-data bit (= 0, if Public Remainder field empty; = 1, if Public Remainder field non empty)
- Padding field according to ISO/IEC 9796-2
- Hash field: hash-code (for SHA-1 and RIPEMD-160: 160 bits)
- Trailer: 1 byte: 'BC'

b) **DSI according to PKCS #1**

The same conventions apply as for the DS signature format (see Annex A, chapter 2.1.2).

1.9.2 DSA

The same conventions apply as for the DS signature format (see Annex A, chapter 2.2).

1.9.3 ELC

The same conventions apply as for the DS signature format (see Annex A, chapter 2.3).

1.10 Object Identifier of the Signature-Algorithms of the CA

The Object Identifier of the signature algorithm of the certificate issuing CA is not given in the CV certificate. The signature algorithm is known to the root CA, as the algorithm reference is stored in the chipcard together with the PK.RCA. The OID of the signature algorithm of the subordinate authorities is specified in the certificate of the root authority.

2 Coding of Authentication Certificates

2.1 Certificate-Headerlist

CV certificates may be constructed as a concatenation of DOs or DEs. Only the DE construction is used here, because the formatting scheme is uniquely specified by means of the CPI. The headerlist scheme allows the description of the CV certificate within the chipcard, so that the DEs to be used can be found within the certificate contents by the card itself. Table B.8 shows the contents of the headerlist and the corresponding certificate structure.

Certificate Content	CPI (1 B)	CAR (8 B)	CHR (12 Bytes)	CHA (7 Byte)	OID (x B)	PK (n Bits) (x B)
Headerlist Content	‘5F29 01‘	’42 08‘	‘5F20 0C‘	'5F4B 07'	’06 0x‘	‘7F49 xx‘ || PK-Parameter Tag-Lengths, see 1.8

Table B.8: Headerlist for Certificate Contents

NOTE - The DO public key (Tag '7F49') is a constructed data object. The length is the number of bytes of the parameter description (Tag-Length-pairs).
The certificate contents is the "Sequence to be signed".

If a card supports certificates with different structures for the signature application, then the CPI value for the relevant headerlist can be used as the selector.

2.2 Structure of the Signature with Message Recovery

In CV certificates with signatures, which allow message recovery, the DSI is structured as follows:

'6A' || Mr || hash value || 'BC'

with Mr = CPI || CAR || CHR || CHA || OID || PK (first part)
and Mx = PK-Remainder and hash value = h(Mr || Mx).

Since the PK cannot be completely integrated, the rest can be found in the PK remainder.

The DSI structure is described by the following headerlist:

DSI	Auxiliary Data (1 B)	Hash input Template	Plain value (x B)	PK-Remainder (0 B)	Hash value (20 B)	Auxiliary Data (1 B)
Headerlist Content	'8A 01'	'A0 00'	'80 xx'	'5F38 00' (value to be taken from PK-Remainder DO)	'90 14'	'8A 01'

Table B.9: Headerlist for DSI

The contents of the certificate is identical to the one described in table B.8.

2.3 Coding of Certificates

The following table shows the defined CPIs with the corresponding certificate coding. It should be noted, that the OID in the certificate of the CA identifies the signature algorithm, whereas the OID in the certificates C_CV.ICC or C_CV.IFD identifies the authentication procedure including key transport or key agreement.

CPI (1 B)	**CAR** (8 B)	**CHR** (12 B)	**CHA** (7 B)	**OID***	**PK**	**Remark**
'01'	...	...	AID || '00'	'2B240304020201'	Modulus (96 B) || Exponent (4 B)	C.CA: RSA, 768 bit, SHA-1, 9796-2
'02'			AID || '00' AID || '01'	'2B2407020101' '2B2407020101'	as immediately above	C.ICC: RSA, 768 bit, SHA-1, 9796-2 C.IFD: RSA, 768 bit, SHA-1, 9796-2
'03'	...	...	AID || '00'	'2B240304020201'	Modulus (128 B) || Exponent (4 B)	C.CA: RSA, 1024 bit, SHA-1, 9796-2
'04'			AID || '00' AID || '01'	'2B2407020101' '2B2407020101'	as immediately above	C.ICC: RSA, 1024 bit, SHA-1,9796-2 C.IFD: RSA, 1024 bit, SHA-1,9796-2
'05'	...	...	AID || '00'	'2B0E03021B'	p (128 B) || q (20 B) || g (128 B) || y (128 B)	C.CA: DSA, L.p=128, SHA-1
'06'			AID || '00' AID || '01'	'2B240701020101' '2B240701020101'	as immediately above	C.ICC: DSA, L.p=128, SHA-1 C.IFD: DSA, L.p=128, SHA-1
'07'	...	...	AID || '00'	'2B2403030201'	p (32 B) || a (32 B) || b (32 B) || PB (64 B) || q (32 B) || PP (64 B)	C.CA: ELC-p, L.F=32, SHA-1
'08'			AID || '00' AID || '01'	'2B240701020201' '2B240701020201'	as immediately above	C.ICC: ELC-p, L.F=32, SHA-1 C.IFD: ELC-p, L.F=32, SHA-1
'09'	...	...	AID || '00'	'2B2403030201'	a (32 B) || b (32 B) || PB (64 B) || q (32 B) || PP (64 B) || m (1 B) || k (1B)	C.CA: ELC-2b, L.F=32,L.m=1,SHA-1
'0A'			AID || '00' AID || '01'	'2B240701020201' '2B240701020201'	as immediately above	C.ICC: ELC-2b,L.F=32,L.m=1,SHA-1 C.IFD: ELC-2b,L.F=32,L.m=1,SHA-1
'0B'	...	...	AID || '00'	'2B2403030201'	PP (2*L.F B)	C.CA: ELC (fixed params.**), SHA-1
'0C'			AID || '00' AID || '01'	'2B240701020201' '2B240701020201'	as immediately above	C.ICC: ELC (fixed params.**), SHA-1 C.IFD: ELC (fixed params.**), SHA-1

Table B.10: CPI Values and CV field Values

*) = The same CPI value may be used, if another OID of the same length is applied.
**) = Parameters to be defined by the root.

NOTE - The Role ID '02' for CA-Certificates (see Table B.7) is only used in connection with an OID, which denotes an authentication procedure using key transport or key agreement. The use of such a certificate, however, is not within the scope of this specification.

3 Contents of File(s) with Authentication Certificate

3.1 Certificate with Message Recovery

If message recovery is applied, the file contains the following data objects:

CV-Certificate	Signature with Certcontent	PK-Remainder	CAR
'7F21'-L-	'5F37'-L-'xx ... xx'	'5F38'-L-'xx ... xx'	'42' -L- 'xx ... xx'

Table B.11: Certificate with Message Recovery

Note: CAR must be listed seperately, because CAR is hidden by the signature. PK.CA must, however, be named, in order to be selected uniquely for the verification of the certificate.

3.2 Certificate without Message Recovery

Files with a certificate without message recovery have the following contents:

CV-Certificate	Certcontent	Signature
‘7F21'-L-	‘5F4E'-L-'xx .. xx’	‘5F37'-L-'xx xx ... xx’

Table B.12: Certificate without Message Recovery

4 Certificates and Key Management

In DIN V66291-1 two figures are presented, which show the processing of one step and two step certification verification applies.

Annex C
(normative)

File Parameter

1 FIDs for Certificate Files

A certificate file consists of exactly one certificate. The FID contains:

- ID for Certificate Files
- Service ID
- SE #
- Certificate Type

Figure C.1 shows the general structure of a certificate FID.

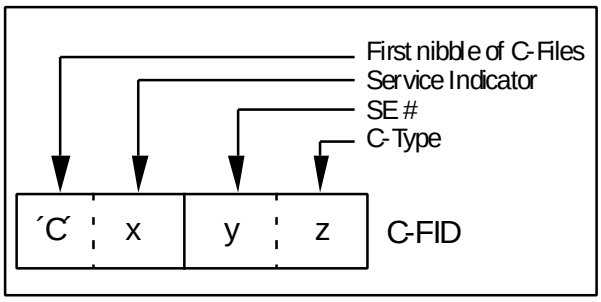

Fig. C.1: Generic structure of a Certificate FID

The coding of the service indicator is shown in Table C.1.

Value	Service
'0'	Digital Signature according to SigG
'1'	Entity Authentication (IFD/ICC)
'2'	Key Encipherment
'3'	Data Encipherment
'4'	Key Agreement
'5'	Entity Authentication (C/S)
'6'	CertSign (no service indicated)
'7'	CertSign for Authentication and Key Encipherment
'8'	CertSign for Authentication and Key Agreement
	Other values RFU

Table C.1: Values for Service Indicator

The SE# (value = 0 in the FID means, that the card does not support or use SE numbers) identifies the SE with the corresponding signature algorithm.
The certificate type denotes the kind of certificate (see Table C.2)

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

Table C.2: Values of certificate type

NOTE - If the value in C.1 is set to '0'-'5' and the value in C.2 is set to '8' - 'D', then it denotes a specific single service for CertSign.

2 OIC Files with File Characteristics and Access Rights

The following table shows the OIC files and their characteristics. If different FIDs are used in the scope of a signature application with additional functions (see Figure 2), then these FIDs must be indicated in the SSD file.

2.1 Files on MF Level

File	**FID**	**Struc-ture**	**Size** (length of data)	**Access Condition**	**Presence**
EF.GDO (Global Data Objects)	'2F02'	trans-parent	64 Bytes	Read: always Update: never	mandatory
EF.C_CV.ICC. AUT (AUT Certificate of ICC)	'2F03'	trans-parent	256 bytes or length of C_CV.ICC.AUT	Read: always Update: never (or IFD authentication (CHA with RoleID ='02') and SM)	mandatory
EF.C_CV.CA.CS-$AUT_{ICC/IFD}$ (CV Certificate of CA)	'2F04'	trans-parent	256 bytes or length of C_CV.CA.CS-$AUT_{ICC/IFD}$	Read: always Update: never (or IFD authentication (RoleID ='02') and SM)	mandatory

Table C.3: Files with their characteristics and access rights

NOTE - The certificates C_CV.ICC.AUT and C_CV.CA.CS-$AUT_{ICC/IFD}$ can also be used for personalisation.

2.2 EFs in DF.IDD

File	**FID**	**Struc-ture**	**Size** (length of data)	**Access Condition**	**Presence**
EF.BD (Basic Id Data)	'D001'	trans-parent	Size dependent on content	Read or Update: IVS Authentication (CHA with RoleID= '01' or '03') with SM	mandatory
EF.ED (Enhanced Id Data)	'D002'	trans-parent	Size dependent on content	Read or Update: IVS Authentication (CHA with RoleID = '03') with SM	mandatory
EF.SP (Special Privileges)	'D003'	trans-parent	Size dependent on content	Read or Update: IVS Authentication (CHA with RoleID= '01' or '03') with SM	mandatory

Table C.4: Files with their characteristics and access rights

NOTE - The content of the identification files is specified in a separate document.

2.3 EFs in DF.BSS

File	**FID**	**Struc-ture**	**Size** (length of data)	**Access Condition**	**Presence**
EF.SSD (Security Service Descriptors)	'1F00'	trans-parent	256 bytes or card specific length	Read: always Update: never	optional
EF.DM (Display message)	'D000'	trans-parent	8 Bytes	Read: IFD authentication (CHA with RoleID ='01' or '02') and SM or user auth. Update: user authentic.	mandatory
EF.C_X509.CH.DS (DS Certificate of Cardholder issued by CA of CH)	'C00x' (x = 0 - 7, see tab.C.2)	trans-parent	2k bytes or length of C.CH.DS	Read / Update: - IFD authentication (CHA with RoleID ='02') and SM) or - user authentication	mandatory
EF.C_X509.CH.AUT (AUT Certificate of Cardholder issued by CA of CH)	'C50x' (x = 0 - 7, see tab.C.2)	trans-parent	1k bytes or length of C.CH.AUT	Read / Update: - IFD authentication (CHA with RoleID ='02') and SM) or - user authentication	mandatory
EF.C_X509.CA.CS (Certificate of CA issued by RCA)	'Cy0x' (x = 8 – E, see tab.C.2, y see tab. C.1)	trans-parent	1k bytes or length of C.CA.CS	Read: always Update: - IFD authentication (CHA with RoleID ='02') and SM) or - user authentication	optional
EF.PK.RCA.CS (Public Key(s) of RCA)	'B000'	trans-parent	512 bytes or length of stored PKs	Read: always Update: - IFD authentication (CHA with RoleID ='02') and SM)	mandatory (if no RCA self-certif. present)
EF.PK.CA.CS (Public Key(s) of CAs)	'B001'	trans-parent	512 bytes or length of stored PKs	Read: always Update: - IFD authentication (CHA with RoleID ='02') and SM)	optional
EF.PROT (Signature log)	'A000'	cyclic	20 Records, each 53 bytes	Read: user authentic. Update: user authentic.	optional

Table C.5: Files with their characteristics and access rights

NOTES -
1. When, e.g. for memory reasons, the certificates are not present in the card, then the SSD file contains the reference to the certificate (the structure of this reference is not within the scope of this specification).
2. Read access to EF.C_X509.CH.DS/AUT may be granted always due to the security policy of the application provider.
3. There may be a difference between update access and write access due to the security policy of the application provider.

2.5 EFs in DF.PKCS#15

If DF.PKC#15 is present in the card, then tab. C.6 is relevant.

File	**FID**	**Struc-ture**	**Size** (length of data)	**Access Condition**	**Presence**
EF.T-Info (TokenInfo)	'5032'	trans-parent	card specific length	Read: always Update: user authentication	mandatory
EF.ODF (ObjectInfo)	'5031'	trans-parent	card specific length	Read: always Update: never	mandatory
EF.PrKDF (PrivateKeyInfo)	'5034'	trans-parent	card specific length	Read: always Update: never	mandatory
EF.PuKDF (Public KeyInfo)	'5035'	trans-parent	card specific length	Read: always Update: user authentication	mandatory
EF.CDF (CertificateInfo)	'5036'	trans-parent	card specific length	Read: always Update: user authentication	mandatory
EF.DODF (DatafileInfo)	'5037'	trans-parent	card specific length	Read: always Update: user authentication	mandatory
EF.AODF (AuthenticationObjectInfo)	'5038'	trans-parent	card specific length	Read: always Update: user authentication	mandatory

Table C.6: Files with their characteristics and access rights

Annex D
(normative)

Device Authentication, Session Key Agreement and Secure Messaging

1 Device Authentication with RSA

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

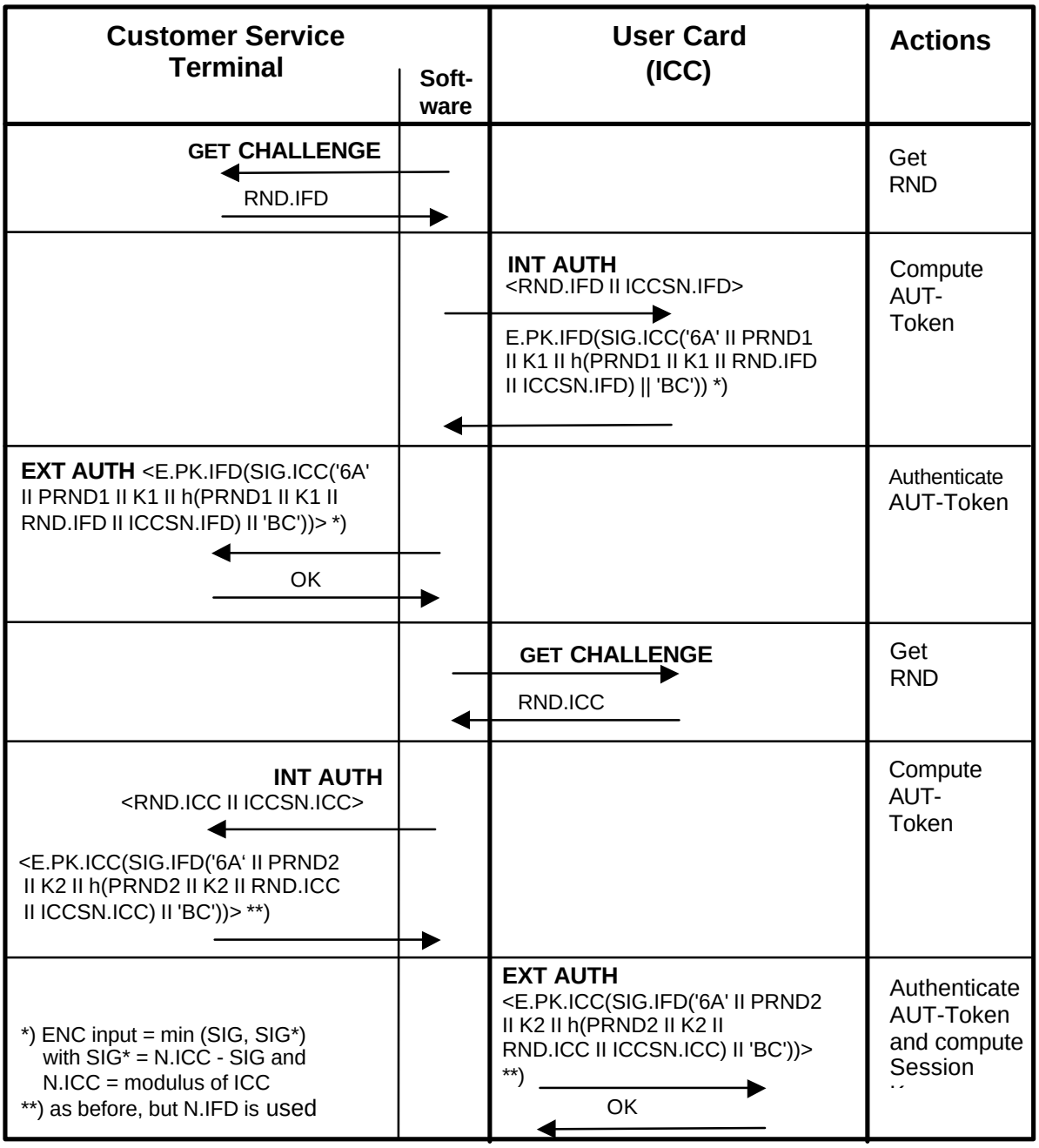

Fig. D.1: Diagram of the authentication procedure with RSA according to ISO/IEC 9796-2 (description of exchange / verification of the certificates and the use of the MSE commands was omitted)

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

Table D.1 shows the data fields in the INTERNAL AUTHENTICATE command/response. Before the command INTERNAL AUTHENTICATE is sent to the card, the keys SK.ICC.AUT and PK.IFD.AUT must be set via the command MSE.

NOTE - If SK.ICC.AUT was not selected, the command INTERNAL AUTHENTICATE is not successful. If PK.ICC.AUT instead of PK.IFD.AUT was selected, the command EXTERNAL AUTHENTICATE is not successful.

Command data field	RND.IFD (8 B) || ICCSN.IFD (8 LSB)
Response data field	E.PK.IFD (SIGMIN) with SIGMIN = min (SIG, SIG*) SIG = SIG.SK.ICC ('6A' || PRND1 (x B) || K1 (32 B) || h(PRND1 (x B) || K1 (32 B) || RND.IFD (8 B) || ICCSN.IFD (8 LSB)) || 'BC') SIG* = N.ICC – SIG with N.ICC = modulus of ICC

Table D.1: Data fields of the command INTERNAL AUTHENTICATE with RSA

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

Preconditions:

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

Command procedure:

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

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

Command data field	E.PK.ICC (SIGMIN) with SIGMIN = min (SIG, SIG*) SIG = SIG.SK.IFD ('6A' || PRND2 (x B) || K2 (32 B) || h (PRND2 (x B) || K2 (32 B) || RND.ICC (8 B) || ICCSN. ICC (8 LSB)) || 'BC') SIG* = N.IFD – SIG with N.IFD = modulus of IFD
Response data field	Empty

Table D.2: Data fields of the command EXTERNAL AUTHENTICATE with RSA

Command procedure:

- Decipher the cryptogram

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

3 Device Authentication with DSA

Cards which use DSA signatures for authentication with customer terminals, must support the authentication procedure acc. to ISO/IEC 9798-3 as depicted in figure D.2. Integrated into this procedure is a key agreement acc. to ISO/IEC DIS 11770-3 (1997) Annex B.5. The names were taken from this standard. The necessary system parameters (the global parameters p and g, depiction of a number mod p, derivation of the SM keys, especially the used hash function h.SM) must be known to the card as well as to the customer terminal. These system parameters are specified in the card authentication certificates in the object identifier for the authentication algorithm. The default values for p, g (incl. L.p) and the hash function h.SM are the respective parameters 'from' the DSA signature of the card.

Preconditions:

- Length of p in bytes is L.p ? 128 bytes
- SIG.ICC is a DSA-signature with length = 40 bytes
- h is the Hash-function of the DSA signature of the chipcard
- SIG.IFD is described in the certificate of the IFD (length L.SIG.IFD), e.g. a DSA signature as well with a length of 40 bytes
- KT.A1 = g^{ra} mod p, KT.A1 has the length L.p
- KT.B1 = g^{rb} mod p, KT.B1 has the length L.p

Before the command INTERNAL AUTHENTICATE can be sent, the keys SK.ICC.AUT and PK.IFD.AUT must be set.

NOTE - If SK.ICC.AUT was not selected, the command INTERNAL AUTHENTICATE is not successful. If PK.ICC.AUT instead of PK.IFD.AUT was selected, the command EXTERNAL AUTHENTICATE is not successful.

Command data field	RND.IFD (8 B) || ICCSN.IFD (8 LSB)
Response data field	KT.A1 (L.p B) || SIG (40 B) with SIG = SIG.ICC(h(RND.IFD (8 B) || ICCSN. IFD (8 LSB) || KT.A1 (L.p B)))

Table D.3: Data fields of the command INTERNAL AUTHENTICATE with DSA

Command procedure:

- Verify, that the Key Reference of PK.IFD is identical to ICCSN.IFD
- Generate KT.A1, store it temporarily for session key calculation
- Store temporarily RND.IFD
- Build hash input and calculate the hash value
- Calculate digital signature
- Build the response data field

Command data field	KT.B1 (L.p B) ?? SIG (L.SIG.IFD B) with SIG = SIG.IFD(h(RND.ICC (8 B) ?? ICCSN.ICC (8 LSB) ??KT.B1 (L.p B)))
Response data field	Empty

Table D.4: Data fields of the command EXTERNAL AUTHENTICATE with DSA

Command procedure:

- Build hash input and calculate the hash value
- Verify the digital signature
- Build session keys

4 Device Authentication with ELC

Figure D.2 shows the general procedure of mutual authentication using ELC including agreement on session keys.

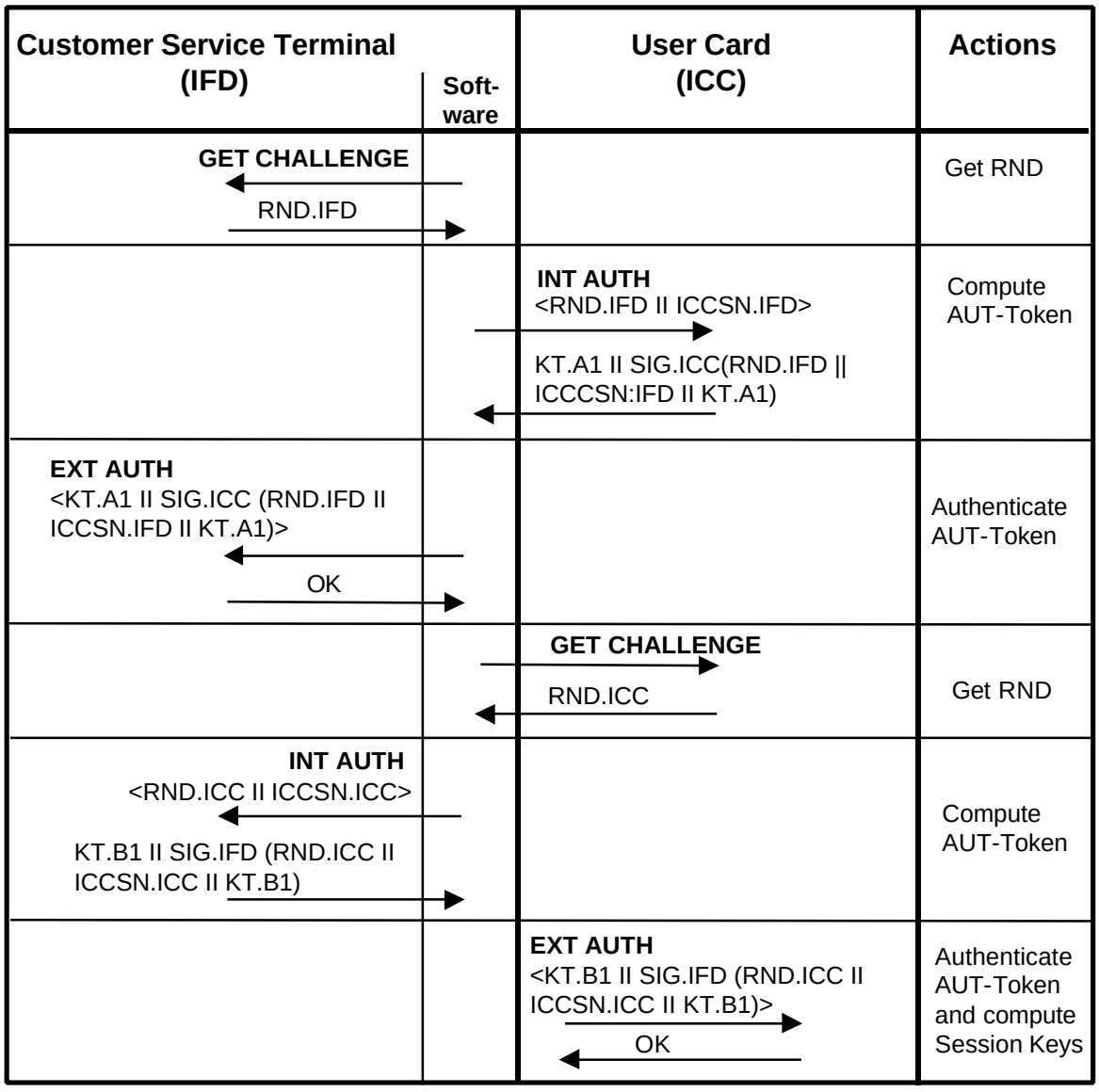

Fig. D.2: Diagram of the authentication procedure with ELC and DSA according to ISO/IEC 11770-3 (without description of exchange and verification of certificates and the use of MSE commands)

Cards, which use ELC signatures for authentication with customer terminals, must support the authentication procedure acc. to ISO/IEC 9798-3 as depicted in figure D.2. Integrated into this procedure is a key agreement acc. to ISO/IEC DIS 11770-3 (1997) Annex C.4. The names were taken from this standard. The necessary system parameters (the elliptic curve E over a field F, the base point PB, representation of a point on E, derivation of the SM keys, especially the used hash function h.SM) must be known to the chipcard as well as to the customer terminal. These system parameters are specified in the card authentication certificate in the object identifier for the authentication algorithm. The default settings are described below (the names are acc. to Annex B 1.8.4).

Default settings:

- E, F, L.F, L.m and PB are the corresonding parameters from the ELC signature of the card.
- Representation of an element in F and a point on E as in Annex B 1.8.4.
- Hash function h.SM is the hash function h from the ELC signature of the chipcard.

Preconditions:

- E is an elliptic curve over a field F. The elements in F are represented as byte sequence of length L.F ? 20. A point Q on E is represented as the concatenation of the sequence of bytes which represents the x-component of Q, followed by the sequence of bytes which represents the y-component of Q. The length of this concatenated sequence is equal to 2?L.F ? 40.
- SIG.ICC is an ELC-signature of length L.SIG.ICC ? 40 bytes
- h is the hash-function of the ELC-signature of the chipcard
- SIG.IFD is described in the certificate of the IFD, e.g. also an ELC-signature of length L.SIG.IFD ? 40 bytes
- KT.A1 is point on E.
- KT.B1 is point on E.

Before the command INTERNAL AUTHENTICATE can be sent, the keys SK.ICC.AUT and PK.IFD.AUT must be set.

NOTE - If SK.ICC.AUT was not selected, the command INTERNAL AUTHENTICATE is not successful. If PK.ICC.AUT instead of PK.IFD.AUT was selected, the command EXTERNAL AUTHENTICATE is not successful.

Command data field	RND.IFD (8 B) || ICCSN.IFD (8 LSB)
Response data field	KT.A1 (2*L.F B) || SIG (L.SIG.ICC B) with SIG = SIG.ICC(h(RND.IFD (8 B) || ICCSN.IFD (8 LSB) || KT.A1 (2*L.F B)))

Table D.5: Data fields of the command INTERNAL AUTHENTICATE with ELC

Command procedure:

- Verify, that the Key Reference of PK.IFD is identical to ICCSN.IFD
- Generate KT.A1, store it temporarily for session key calculation
- Store temporarily RND.IFD
- Build hash input and calculate the hash value
- Calculate digital signature
- Build the response data field

Command data field	KT.B1 (2*L.F B) ?? SIG (L.SIG.IFD B) with SIG = SIG.IFD(h(RND.ICC (8 B) ?? ICCSN.ICC (8 LSB) ??KT.B1 (2*L.F B)))
Response data field	Empty

Table D.6: Data fields of the command EXTERNAL AUTHENTICATE with ELC

Command procedure:

- Build hash input and calculate the hash value
- Verify the digital signature
- Build session keys

4 Key Agreement

4.1 Key agreement using RSA

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

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

K = K1 ? K2

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

K = Ka(ENC) || Kb (ENC) || Ka (MAC) || Kb (MAC)

The use of the keys Ka and Kb is depicted in figure D.3.

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

4.2 Key Agreement using Diffie-Hellman

DSA:

As described in SO/IEC 11770-3 both the customer terminal and the chipcard calculate the common secret K.AB from KT.A1 and KT.B1.

$K.AB = g^{ra*rb} \bmod p$

Let K.AB(1) be the sequence of bytes, which represents K.AB. The length of K.AB(1) is L.p ? 128 bytes.

ELC:

As described in SO/IEC 11770-3 both the customer terminal and the chipcard calculate the common point K.AB on the curve.
Let K.AB(1) be the sequence of bytes, which represents K.AB. The length of K.AB(1) is 2*L.F ? 40 bytes.

Key derivation:

Key derivation from the common secret acc to ANSI X 9.63, Elliptic Curve Key Agreement and Transport Protocols. Let c be a 32 bit counter. Both customer terminal and the chipcard calculate

HASH1 = h.SM (K.AB(1) || c) with c=1 and
HASH2 = h.SM (K.AB(1) || c) with c=2.

with h.SM as the hash function mentioned above.

The bits b64 – b1 of HASH1 build the key Ka(ENC), the bits b128 – b65 build the key Kb(ENC).
The bits b64 – b1 of HASH2 build the key Ka(MAC), the bits b128 – b65 build the key Kb(MAC).

5 Secure Messaging

5.1 SM-DOs

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

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

Table D.7: SM Data objects

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

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

5.2 Commands and Responses with SM

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

Command:

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

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

Response with data:

TPV	LPV	PV	TCC	Lcc	CC	SW1-SW2

Response without data:

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

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

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

- READ BINARY (Display Message, Identification Data)
- UPDATE BINARY (Display Message, Identification Data)
- VERIFY

- CHANGE RD
- RESET RC

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

Command without cryptogram (READ BINARY):

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

Response with cryptogram:

TCG	LCG	PI, CG	TCC	Lcc	CC	SW1-SW2

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

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

Response:

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

5.3 Treatment of SM-Errors

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

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

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

When the chipcard returns status bytes without SM DOs or with an erroneous SM DO the session is aborted by the customer terminal.

5.4 Padding for checksum calculation

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

5.5 DES-Mode, Initial Value and Send Sequence Counter

5.5.1 Cryptograms

Cryptograms are build with DES-3 in CBC-Mode with the Null vector as Inital Check Block.

5.5.2 Cryptographic Checksums

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

- Initial stage: The initial check block y_0 is E(Ka, SSC).
- Sequential Stage: The check blocks y_1, .. , y_n are calculated using Ka.

- Final Stage: The cryptographic checksum is calculated from the last check block y_n as follows: E(Ka, D(Kb, y_n)).

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

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

The starting value for the SSC is

SSC = RND.ICC (4 least significant bytes) || RND.IFD (4 least significant bytes).

6 Use of DES

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

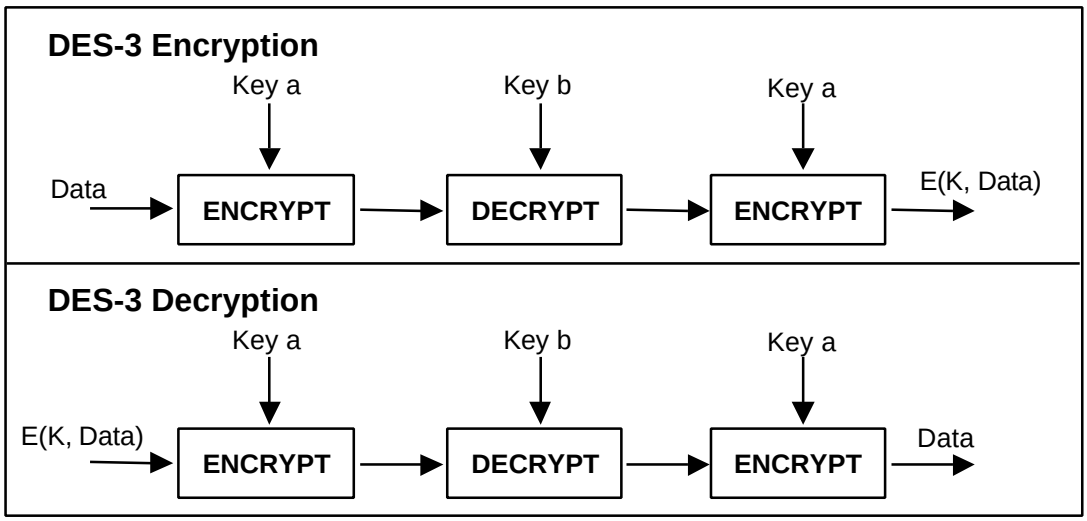

Fig. D.3: DES-3-Encryption/Decryption

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

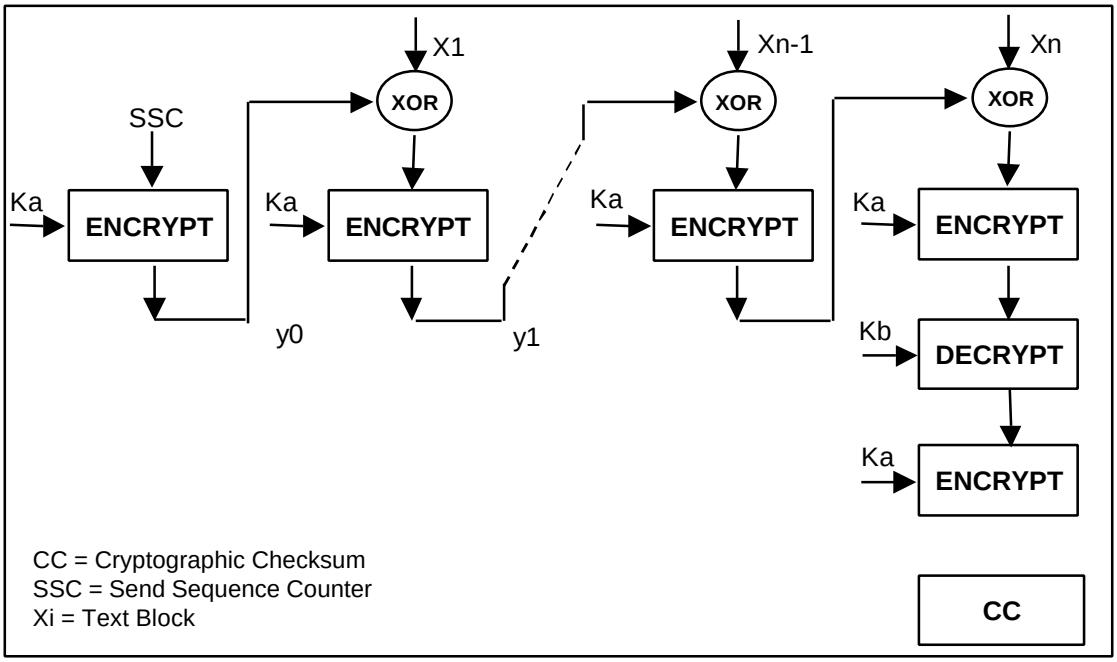

Fig. D.4: Calculation of the retail MAC

Annex E
(normative)

Security Service Descriptor Templates

1 Security Service Descriptor Concept

For supporting interoperability and coexistence of chipcards with differences, e.g. in command sequences, as well as to facilitate migration in an easier way, an SSD file should exist in the SigG application. The information about available security services are provided in SSD templates to be interpreted by the terminal. The SSD file contains either

- One or more SSD-Templates for each security service (see Table E.1) or
- a DO 'SSD Profile identifier' (see DO with tag '8D').

2 SSD Data Objects

The SSD templates have context-specific tags, whereby the context is given by the SSD file. The DO 'Instruction set mapping' is mandatory in the value part of each SSD template. All other DOs are optional.

Tag	**Meaning**	**Related ISO/IEC commands** (commands in ( ) optional)
'A0'	User authentication service	VERIFY or CHANGE RD or ENABLE VR or DISABLE VR or RESET RC
'A1'	Internal authentication service	(MANAGE SE) INT. AUTHENTICATE
'A2'	External authentication service (sym.)	(MANAGE SE) GET CHALLENGE EXT. AUTHENTICATE
'A3'	External / Mutual authentication service (asym.)	(MANAGE SE) PSO: VERIFY CERTIF. (INT. AUTHENTICATE) GET CHALLENGE EXT. AUTHENTICATE
'A4'	Digital signature computation service	(MANAGE SE) (PSO: HASH) PSO: COMPUTE DS
'A5'	Digital signature verification service	(MANAGE SE) (PSO: HASH) PSO: VERIFY DS
'A6'	Certificate verification service	(MANAGE SE) PSO: VERIFY CERTIFICATE
'A7'	Checksum computation service	(MANAGE SE) PSO: COMPUTE CC
'A8'	Checksum verification service	(MANAGE SE) PSO: VERIFY CC
'A9'	Encipherment service	(MANAGE SE) PSO: ENCIPHER
'AA'	Decipherment service	(MANAGE SE) PSO: DECIPHER
'AB'	File management service	SELECT FILE

Table E.1: SSD templates

- DO Instruction set mapping (ISM), tag '80'
  This DO contains the regular command to provide the respective security service. If the security service requires a sequence of commands, then this DO is repeated. The DO contains at least the CLA-byte and the INS-byte of the command as defined in main part of this specification. If appropriate, P1 and P2 or further parts of the command APDU may be present. It is assumed that the outside world knows the command in its complete form and which data - if any - have to be inserted in the body part of the command.

- DO Command to perform (CTP), tag '52' (see ISO/IEC 7816-6)
  This DO - if present - informs the outside world which command is supported by the card to provide the same security service as achievable with the command indicated in the DO ISM. If several commands are necessary then the DO CTP is repeated. It shall follow - if used - immediately behind the last DO ISM.

- DO Algorithm object identifier (OID), tag '06' (see ISO/IEC 7816-6)
  The value field of this DO contains the object identifier of the related crypto algorithm available in the card. The encoding rule of the OID is according to ISO 8825 as follows:

  - first subidentifier is multiplied by 40 and coded as binary number in one byte; the value of the second subidentifier is added to this byte
  - the following subidentifiers are each represented as a binary number in a sequence of bytes from which bit b8 is the chaining bit (1 = not last byte, 0 = last or only byte)

- DO Algorithm reference, tag '81'
  The value field of this DO contains the algorithm reference as used in the card for the algorithm denoted by the DO OID or specified in an application context.

- DO Key reference, tag '82'
  The value field contains the key reference of the key to be applied by the card when performing the related security operation. If this DO is present, then the value has to be used in the command related to this security service.

- DO FID key file, tag '83'
  The value field contains the file id of a file containing the key to be applied by the card when performing the related security operation. If this DO is present, a SELECT FILE command with the respective FID has to be sent as first command before using a security service.

- DO Key group, tag '84'
  This DO is only relevant for symmetric encipherment algorithms which work with individual keys and master keys. When this DO is used, then the value denotes the entity using the master key (e.g. '01' = physician, '02' = pharmacist, i.e. the values can be considered as group ids)

- DO FID base certificate file, tag '85'
  The value field of the DO contains the file id of a file containing the base certificate related to the respective security service.

- DO FID adjoint certificate file, tag '86'
  The value field contains the file id of a file containing an adjoint certificate related to the base certificate of the same security service template.

- DO Certificate reference, tag '87'
  If no FID for a certificate file is given, then this DO - if present - contains a reference to the related certificate which is not stored in the card. The reference may be a key identifier (tag '88') and/or the distinguished name (or the ID) of the CA issuing the certificate (tag '89') followed by the certificate serial number (tag '8A').

- DO Certificate qualifier, tag '88'
  The value field contains the information whether the certificate is a non-ICC certificate (e.g. a X.509 certificate) to be verified outside a card (value '00') or an ICC certificate, which may be interpreted by a card (value '01'). The default value is '00'.

- DO FID for file with public key of the certification authority PK(CA), tag '89'
  This DO - if present - contains the FID of the file in which the DO public key of the certification authority is stored (tag '5F4A').

- DO PIN usage policy, tag '5F2F'
  (see ISO/IEC 7816-6)
  This DO - if present - indicates the PIN usage policy. The value filed of this DO (2 bytes) is application specific.

For DF.SIG and DF.BSS the following values are defined:

First byte:

'80' i.e. PIN relevant for application, see clause 6.4 of ISO/IEC 7816-6

Second byte:

- msn: Coding scheme
- lsn: Limitation

'0x': ASCII coding (default for PIN reference ? '80')
'1x': Format 2 PIN block (default for PIN reference = '80')

'x0': after PIN presentation no limitation
'x1'-'xF': the number indicates the amount of signatures possible with one PIN presentation

Other values RFU

NOTE - If only one byte is present in the value field, then only the coding scheme and limitation indication are indicated.

- DO PIN reference, tag '8A'
  The value field of this DO contains one byte coding the qualifier of the reference for the cardholder verification data, if the specified value is not used (i.e. for the OIC the values '01' and '81').
  If bit b8 of the value field is set to 1, the DO contains the PIN reference for the specific PIN (i.e. for the OIC the DS-PIN). If bit b8 of the value field is set to 0, the DO contains the PIN reference for the global PIN (i.e. for the OIC the AUT+KE-PIN).

- DO Application identifier (AID), tag '4F' (see ISO/IEC 7816-6)
  This DO indicates, to which application the related template belongs and shall only be used, if an SSD file on the MF level is needed (e.g. if a PIN presentation before selection of the application with the specified AID is required).

- DO CLA coding, tag '8B'
  The value field of this DO contains the CLA coding which has to be used instead of the CLA coding given in the DO ISM.

- DO Status information (SW1-SW2), tag '42' (see ISO/IEC 7816-6)
  If relevant status bytes (e.g. from the VERIFY command) differ from those of the command described in the DO ISM, then the mapping shall be given, i.e. a SW1-SW2 of the ISM command is followed by SW1-SW2 coding send by the card.

- DO Discretionary data, tag '53' (see ISO/IEC 7816-6)
  The contents of this DO - if present - is defined by the application provider.

- DO SE number, tag '8C'
  The value field of this DO contains the security environment number.

- DO SSD profile identifier, tag '8D'
  The value of this DO identifies for the respective application an SSD set available in the IFD. The externally stored SSD set describes the security service supported by the card.

- DO FID mapping, tag '8E'
  The value of this DO contains one or more pairs of FIDs: the first FID gives the value assigned in this specification, the second FID indicates the FID to be used.

Annex F
(informative)

ATR-Coding

Figure F.1 shows the recommended coding for the ATR for cards of classes A, AB and B (5V, 5V/3V and 3V technology) with T=0 or T=1 transmission protocol.

T=0 ATR

TS	T0	TD1	TD2	TA3	T1-TK	TCK
´3B´	´8x´	´80´	´1F´	´xx´	HB	XOR

TS = Initial Character (direct convention)
T0 = Format Character (TD1 indic., x HB)
TD1 = Interface Character (T=0, TD2 indication)
TD2 = Interface Character (T=15,TA3 indic.)
TA3 = Interface Character (XI- and UI-indication for clock stop mode and voltage class)
T1-TK = Historical Characters (Hist. Bytes, max. 15)

T=1 ATR

TS	T0	TD1	TD2	TA3	TB3	TD3	TA4	T1-TK	TCK
´3B´	´8x´	´81´	´B1´	´20´ to ´FE´	´42´	´1F´	´xx´	HB	XOR

TS = Initial Character (direct convention)
T0 = Format Character (TD1 indic., x = no. of HB)
TD1 = Interface Character (T=1, TD2 indication)
TD2 = Interface Character (T=1,TA3,TB3 indic.)
TA3 = Interface Character (IFSC)
TB3 = Interface Character (BWI, CWI)
TD3 = Interface Character (T=15,TA4 indic.)
TA4 = Interface Character (XI- and UI-indication for clock stop mode and voltage class)
T1-TK = Historical Characters (Hist. Bytes, max. 15)
TCK = Check Character (exclusive OR)

NOTES
1. Other parameters may be present or optional parameters shown absent
2. TB1 may be set to ´00´ or absent. Default values for Vpp support are not valid.

Fig. F.1: ATR-Coding for Cards class A, AB and B (example)

Cards, which support higher transmission rates, shall indicate the respective values for FI and DI in TA1. The default operation mode should be the ‘negotiable mode’. Those cards shall support the protocol parameter selection acc. to ISO/IEC 7816-3.

The historical bytes contain information which are not interpreted by the terminal. Figure F.2 shows the recommended structure for the historical bytes.

Historical Bytes

CI	PIDO	CPDO	CLS	SW1-SW2

CI = Category indicator (´00´)
PIDO = Pre-issuing data object
CPDO = Card profile data object
CLS = Card life status (1 byte)
SW1-SW2 = Status bytes

Pre-issuing Data Object

TPI	ICM	ICT	OSV	DD

TPI = Tag pre-issuing DO (´6x´)
ICM = IC manufacturer Id (1 byte)
ICT = IC type (1 byte, if b8 = 0; 2 bytes, if b8 = 1 of first byte)
OSV = Operating system version (2 bytes)
DD = Discretionary data (x bytes)

Card Profile Data Object

TCP	CP

TCP = Tag card profile DO (´3x´)
CP = Card profile according to ISO/IEC 7816-4, 8.3.2 (1 byte)

Fig. F.2: Structure of the Historical Bytes (example)

For the coding of the DOs acc. to Figure E.2 the following conventions apply:

CI = '00' according to ISO/IEC 7816-4
TPI = '6x' according to ISO/IEC 7816-4 (x codes the length of the DO)
ICM = IC Manufacturer ID (see Table 1)
ICT = Coding manufacturer specific
OSV = Coding manufacturer specific
DD = Coding manufacturer specific (usually not used)
TCP = 3x’ according to ISO/IEC 7816-4 (x codes the length of the DO)
CP = Coding according to ISO/IEC 7816-4 (i.e. ‘80’ for ‘direct application selection’))
CLS = Card Life Cycle (Default-Wert '00')
SW1-SW2 = '9000'

Table F.1 shows the actual values for ICM.

ICM	IC Manufacturer according to ISO/IEC 7816-6/AM 1
'01'	Motorola
'02'	ST Microelectronics
'03'	Hitachi
'04'	Philips Semiconductors
'05'	Infineon Technologies
'06'	Cylinc
'07'	Texas Instruments
'08'	Fujitsu
'09'	Matsushita
'0A'	NEC
'0B'	Oki
'0C'	Toshiba
'0D'	Mitsubishi
'0E'	Samsung
'0F'	Hyundai
'10'	LG

Table F.1: ICM-Coding

Annex G
(informative)

Content of Cryptographic Token Information Files

The following notation is an example for an OIC card, where all PKCS#15 EFs are located under DF.PKCS#15 and where the card supports:

- SHA-1,
- RSA with DSI according to ISO/IEC 9796-2rnd with Random Number,
- RSA with DSI according to PKCS#1,
- Triple-DES,
- Retail-MAC,
- RSA encryption according to ISO/IEC 9796-2rnd

EF.DIR located under MF, optional

Value notation:

```
{
aid 'A000000063504B43532D3135'H,
label "PKCS#15 application",
path '3F005015'H –– absolute path to DF.PKCS#15
},
{
aid 'D27600006601'H, -- AID of DF.BSS
label "office identity card",
path '3F004016'H -- absolute path to DF.BSS, FID='4016' example
},
{
aid 'D27600006603'H, -- AID of DF.IDD
label "identification data",
path '3F004017'H -- absolute path to DF.IDD, FID='4017' example
}
```

DF.PKCS#15 has the following entries:

EF.T-Info

Value notation:

```
{
version v1,
serialNumber 'xx..xx'H,
manufacterID "XY, Inc.",
label "OIC",
tokenflags {prnGeneration},
supportedAlgorithms {
      {reference 1,
       algorithm 1,
       -- SHA-1
       parameters NULL : NULL,
       supportedOperations {hash},
       algId {13143226} -- OID assigned by OIW (Open Systems Implementors
                           Workshop)
      },
```

```
      {reference 2,
       algorithm 2,
       -- RSA, Hash: SHA-1, Padding: ISO/IEC 9796-2rnd
       parameters NULL : NULL,
       supportedOperations {compute-signature, verify-signature},
       algID {133634321} -- OID assigned by TeleTrust
      },
      {reference 3,
       algorithm 3,
       -- RSA, Hash: RIPEMD-160, Padding: PKCS#1
       parameters NULL : NULL,
       supportedOperations {compute-signature, verify-signature},
       algID {13363312} -- OID assigned by TeleTrust
      },
      {reference 4,
       algorithm 4,
       -- Triple-DES in CBC-Modus with IV='00'
       parameters '0000000000000000'H,
       supportedOperations {encipher,
                            decipher},
 algId {1284011354937} -- OID assigned by RSA Data Security Inc.
      },
      {reference 5,
       algorithm 5,
       -- Retail-MAC with DES
       parameters NULL : NULL,
       supportedOperations {compute-checksum,
                           verify-checksum},
                              algID {} -- to be inserted
      },
      {reference 6,
       algorithm 6,
       --  key transport with RSA encryption and RSA signature
       parameters NULL : NULL,
       supportedOperations {encipher,
                            decipher},
      algId {13367211} -- OID assigned by TeleTrust
      }
      },
issuerId "wxy", -- contains information about the token issuer
}
```

EF.ODF

Value notation:

```
{
privateKeys : path : {
      path '5034'H, -- relative path to EF.PrKDF
      },
trustedPublicKeys : path : {
      path '5035'H, -- relative path to EF.PuKDF
      },
trustedCertificates : path : {
      path '5036'H, -- relative path to EF.CDF
      },
dataObjects : path : {
      path '5037'H, -- relative path to EF.DODF
      },
authObjects : path : {
      path '5038'H, -- relative path to EF.AODF
      }
```

```
}
```

EF.PrKDF

Value notation:

```
{
privateRSAKey : {
      CommonObjectAttributes {
            label "SK.CH.DS", -- digital signature of the CH
            flags {private},
            authId '07'H -- presentation of the specific PIN required
            },
      CommonKeyAttributes {
            iD '01'H, -- PKCS#15 identifier for the related key
            usage {nonRepudiation},
            keyReference '84'H,
            startDate "20000101000000Z", -- validity from 1.1.2000, 00:00h
            endDate "20010101000000Z" -- to 1.1.2001, 00:00h
            },
      CommonPrivateKeyAttributes {
            subjectName "..." -- DN of PrK owner as specified in the
                              -- X.509 certificate containing the PuK
            },
      PrivateRSAKeyAttributes {
            value indirect : path {
                  path ''H –- no FID given
                  },
            modulusLength 1024
            }
      },
privateRSAKey : {
      CommonObjectAttributes {
            label "SK.ICC.AUT_ICC/IFD" -- PrK of ICC for authent.
            },
      CommonKeyAttributes {
            iD '02'H,
            usage {sign}, -- corresponds to cv certificate
            keyReference '01'H  -- Key Id in the card of PrK.ICC.AUT
            },
      PrivateRSAKeyAttributes {
            value indirect : path {
                  path ''H
                  },
            modulusLength 1024
            }
      },
privateRSAKey : {
      CommonObjectAttributes {
            label "SK.CH.KE", -- key encipherment key of the CH
            flags {private},
            authId '0A'H -- presentation of the global PIN required
            },
      CommonKeyAttributes {
            iD '0C'H,
            usage {unwrap},
            keyReference '83'H
},
      PrivateRSAKeyAttributes {
            value indirect : path : {
                  path ''H
```

```
                  },
            modulusLength 1024
            }
      },
privateRSAKey : {
      CommonObjectAttributes {
label "SK.CH.AUT", -- authentication key in combination with
                              -- x509 certificate of the CH
            flags {private},
            authId '0A'H
            },
      CommonKeyAttributes {
            iD '0D'H,
            usage {sign},
            keyReference '82'H
            },
      PrivateRSAKeyAttributes {
            value indirect : path : {
                  path ''H
                  },
            modulusLength 1024,
            }
      }
}
```

EF.PuKDF

Value notation:

```
{
publicRSAKey : {
      CommonObjectAttributes {
            label "PK.CA.CS-AUT_ICC/IFD" -- verification key for
                                         -- cv-certificates of the CA
            },
      CommonKeyAttributes {
            iD '03'H,
            usage {verify},
            keyReference 'xx..xx'H -- key Ref as used in the card, i.e.
                                   -- the certificate authority reference,
                                   -- which is taken as authority key id
            },
      CommonPublicKeyAttributes {
            subjectName "..." -- DN of the CA
            },
      PublicRSAKeyAttributes {
            value indirect : path : {
                  path ''H -- no FID given
                  },
modulusLength 1024
            }
      },
publicRSAKey : {
      CommonObjectAttributes {
            label "PK.RCA.CS-AUT_ICC/IFD" -- verification key for
                                          -- cv-certificates of root CA
            },
      CommonKeyAttributes {
            iD '04'H,
            usage {verify},
            keyReference 'xx..xx'H
```

```
            },
      CommonPublicKeyAttributes {
            subjectName "..." -- DN of the root CA
            },
      PublicRSAKeyAttributes {
            value indirect : path : {
                  path ''H
                  },
            modulusLength 1024
            }
      },
publicRSAKey : {
      CommonObjectAttributes {
            label "PK.CA.CS", -- verification key for x509 certificates
                              -- of the root CA
            flags {modifiable},
            authId '12'H
            },
      CommonKeyAttributes {
            iD '05'H,
            usage {verify},
            keyReference 'xx..xx'H
            },
      PublicRSAKeyAttributes {
            value indirect : path : {
                  path 'B001'H, -- relative path to EF.PK.RCA.CS
                  length xxx  -- length in bytes, optional
                  }
            }
      },
publicRSAKey : {
      CommonObjectAttributes {
label "PK.RCA.CS", -- verification key for x509 certificates
                               -- of the root CA
            flags {modifiable},
            authId '12'H
            },
      CommonKeyAttributes {
            iD '06'H,
            usage {verify},
            keyReference 'xx..xx'H
      },
      PublicRSAKeyAttributes {
            value indirect : path : {
                  path 'B000'H, -- relative path to EF.PK.RCA.CS
                  length xxx  -- length in bytes, optional
                  }
            }
      }
}
```

EF.CDF

Value notation:

```
{
x509Certificate : {
      CommonObjectAttributes {
            label "C_X509.CH.DS", -- DS-certificate of the CH
            accessControlRules {
                  {accessMode {read},
                   securityCondition or {authId '0A'H,
```

```
                                    authId '12'H}
                },
                {accessMode {update},
                 securityCondition or {authId '0A'H,
                                    authId '12'H}
      }}
            },
      CommonCertificateAttributes {
            iD '01'H, -- related to private RSA key with id '01'H
            ancestor '05'H  -- id of the previous (higher) CA-certificate
            },
      X509CertificateAttributes {
            value indirect : path : {
                  path 'C000'H, -- relative path to EF.C_X509.CH.DS
                  length xxx  -- length of certificate in bytes, optional
                  },
            subject "..", -- certificate subject name
            issuer ".." -- certificate issuer name
            }
      },
x509Certificate : {
      CommonObjectAttributes {
            label "C_X509.CA.CS", -- the x509 certificate of the CA, which
                                  -- issued cardholders certificate
            accessControlRules {
                  {accessMode {update},
                   securityCondition or {authId '0A'H,
                                    authId '12'H}
                  }}
       },
      CommonCertificateAttributes {
            iD '05'H,
            authority TRUE
            },
      X509CertificateAttributes {
            value indirect : path : {
                  path   'C008'H, -- relative path to EF.C_X509.CA.CS
                  length xxx  -- optional
                  }
            }
      },
x509Certificate : {
      CommonObjectAttributes {
            label "C_X509.CH.AUT", -- the AUT-x509 certificates of the CH
            accessControlRules {
                  {accessMode {read},
                   securityCondition or {authId '0A'H,
                                    authId '12'H}
                  },
                  {accessMode {update},
                   securityCondition or {authId '0A'H,
                                    authId '12'H}
                  }}
            },
      CommonCertificateAttributes {
            iD '0D'H,
            ancestor '05'H
            },
      X509CertificateAttributes {
            value indirect : path : {
                  path 'C500'H, -- relative path to EF.C_X509.CH.AUT
                  length xxx  -- optional
                  }
```

```
            }
      },
cvCertificate : {
      CommonObjectAttributes {
            label "C_CV.ICC.AUT",
            flags {modifiable},
            authId '12'H
            },
      CommonCertificateAttributes {
            iD '02'H, -- related to private RSA key with id '02'H
            ancestor '03'H  -- id of the previous (higher) CA-certificate
            },
      CVCertificateAttributes {
            value indirect : path : {
                  path '3F002F03'H -- absolute path to EF.C_CV.ICC.AUT
                  }
            }
      },
cvCertificate : {
      CommonObjectAttributes {
            label "C_CV.CA.CS-AUT_ICC/IFD",
            flags {modifiable},
            authId '12'H
            },
      CommonCertificateAttributes {
            iD '03'H,
            authority TRUE
            },
      CVCertificateAttributes {
            value indirect : path : {
                  path '3F002F04'H, -- absolute path to EF.C_CV.CA.CS-AUT
                  length xxx  -- optional
                  }
            }
      }
}
```

EF.DODF

Value notation:

```
{
opaqueDO : {
      CommonObjectAttributes {
            label "EF.GDO"
            },
      Opaque indirect : path : {
            path '3F002F02'H, -- absolute path to EF.GDO
            length xxx -- length of content in bytes, optional
            }
      },
opaqueDO : {
      CommonObjectAttributes {
            label "EF.PROT",
            flags {private, modifiable},
            authId '0A'
            },
      Opaque indirect : path : {
            path 'A000'H, -- relative path to EF.PROT
            index '01'H, -- index number
            length 53 -- length of a record in bytes, optional
            }
```

```
      },
opaqueDO : {
      CommonObjectAttributes {
            label "EF.SSD"
            },
      Opaque indirect : path : {
            path '1F00'H, -- relative path to EF.SSD
            length xxx -- length of content in bytes, optional
            }
      },
opaqueDO : {
      CommonObjectAttributes {
            label "display message",
            accessControlRules {
                  {accessMode {read}, -- readable after IFD auth.
                   securityCondition or {authId '11'H,
                                         authId '12'H,
                                         authId '0A'H}
                  },
                  {accessMode {update}, -- writable after user auth.
                   securityCondition '0A'H
                  }}
            },
      Opaque indirect : path : {
            path 'D000'H, -- relative path to EF.DM
            length xxx -- optional
            }
      },
opaqueDO : {
      CommonObjectAttributes {
            label  "basic data",
            accessControlRules {
                  {accessMode {read},
                   securityCondition or {authId '11'H,
                                         authId '13'H}
                  },
                  {accessMode {update},
                  securityCondition or {authId '11'H,
                                        authId '13'H}
}}
            },
      Opaque indirect : path : {
            path 'D001'H, -- relative path to EF.BD
            length xxx  -- optional
            }
      },
opaqueDO : {
      CommonObjectAttributes {
            label  "extended data",
            accessControlRules {
                  {accessMode {read},
                   securityCondition {authId '13'H}
                  },
                  {accessMode {update},
                  securityCondition {authId '13'H}
}}
            },
      Opaque indirect : path : {
            path 'D002'H, -- relative path to EF.ED
            length xxx  -- optional
            }
      },
opaqueDO : {
```

```
      CommonObjectAttributes {
            label  "special privileges",
                        accessControlRules {
                  {accessMode {read},
                   securityCondition or {authId '11'H,
                                         authId '13'H}
                  },
                  {accessMode {update},
                  securityCondition or {authId '11'H,
                                        authId '13'H}
                  }}
            },
      Opaque indirect : path : {
            path 'D003'H, -- relative path to EF.SP
            length xxx  -- optional
            }
      }
}
```

EF.AODF

Value notation:

```
{
pinAuthenticationObject : {
      CommonObjectAttributes {
            label "specific PIN for DS", -- PIN to protect the COMPUTE DS
                                          -- command
            flags {private, modifiable}
            },
      CommonAuthenticationObjectAttributes {
            authID '07'H
            },
      PinAttributes {
            pinFlags {case-sensitive, -- no conversion to uppercase
                      local, -- i.e. DF-specific
                  initialized},
            pinType {iso9564-1},  -- PIN in format 2 PIN Block
            minLength 6,
            storedLength 0, -- not information given
            maxLength 8,
            pinReference '81'H-- P2 of VERIFY/CHANGE RD command of specific
                              -- PIN
            }
      },
pinAuthenticationObject : {
      CommonObjectAttributes {
            label "resetting code for specific PIN",
            flags {private}
            },
      CommonAuthenticationObjectAttributes {
            authID '08'H
            },
      PinAttributes {
            pinFlags {local,
                      initialized,
                      unblockingPin},
            pinType {utf8},
            minLength 6,
            storedLength 0,
            pinReference '81'H
            }
```

```
      },
pinAuthenticationObject : {
      CommonObjectAttributes {
            label "global PIN for client/server authentication and
                   key decipherment",
            flags {private, modifiable}
            },
      CommonAuthenticationObjectAttributes {
            authID '0A'H
            },
      PinAttributes {
            pinFlags {case-sensitive, -- no conversion to uppercase
                initialized},
            pinType {iso9564-1},  -- PIN in format 2 PIN Block
            minLength 6,
            storedLength 0,
            maxLength 8,
            pinReference '01'H-- P2 of VERIFY/CHANGE RD command of global
                              -- PIN
            }
      },
pinAuthenticationObject : {
      CommonObjectAttributes {
            label "resetting code for global PIN",
            flags {private}
            },
      CommonAuthenticationObjectAttributes {
            authID '0B'H
            },
      PinAttributes {
            pinFlags {initialized,
                      unblockingPin},
            pinType {utf8},
            minLength 6,
            storedLength 0,
            PinReference '01'H-- P2 of VERIFY/CHANGE RD command
            }
      },
externalAuthenticationObject : {
      CommonObjectAttributes {
            label "certificateholder authorisation"
            },
      CommonAuthenticationObjectAttributes {
            authID '11'H
            },
      CertBasedAuthenticationAttributes {
                                        CHA 'D2760000660301'H  -- AID with
                              role Id 01 as defined for
                                                               -- the CV
                              certifcate to be presented as
                                                               -- part of
                              the authentication procedure
                                                               -- for
                              getting access to EFs
            }
      },
externalAuthenticationObject : {
      CommonObjectAttributes {
            label "certificateholder authorisation"
            },
      CommonAuthenticationObjectAttributes {
            authID '12'H
            },
```

```
      CertBasedAuthenticationAttributes {
            CHA 'D2760000660102'H  -- AID with role Id 02
            }
      },
externalAuthenticationObject : {
      CommonObjectAttributes {
            label "certificateholder authorisation"
            },
      CommonAuthenticationObjectAttributes {
            authID '13'H
            },
      CertBasedAuthenticationAttributes {
            CHA 'D2760000660303'H  -- AID with role Id 03
            }
      }
}
```