



Specification
of chipcard interface
with
digital signature application/function
acc. to SigG and SigV

DIN NI-17.4

Version 1.0

15.12.1998

Contents

1	Scope	4
2	References	4
3	Definitions, Abbreviations and Notations	5
3.1	Definitions	5
3.2	Abbreviations	5
3.3	Notations	6
4	Technical characteristics	6
5	Answer-to-Reset	6
5.1	Global Interface Characters	6
5.2	Historical Bytes	6
6	Protocol Parameter Selection	7
7	Transmission protocols	7
7.1	General aspects	7
7.2	Transmission protocol T = 0	7
7.3	Transmission protocol T = 1	7
8	Flow diagram and embedding of the SigG function	7
9	Certification authorities and certificates	9
10	File structure and data objects	10
10.1	File structure	10
10.2	EFGDO	11
10.3	EFKEY	12
10.4	EFSSD	12
10.5	EFDM	12
10.6	EFC.CH.DS	12
10.7	EFC.CA.DS	12
10.8	EFC.ICC.AUT	13
10.9	EFC.CA.AUT	13
10.10	EFPK.CA.DS	13
10.11	EFPROT	13
11	Chipcard opening	13
11.1	Command sequences	13
11.2	Reading the global data objects	13
12	Application opening phase	14
12.1	Command sequences	14
12.2	Application selection	14
12.3	Reading of SSD Templates	14
13	Authentication of the Card Holder	14
13.1	Command-overview	14
13.2	User authentication	15
13.3	Changing the reference data	15
13.3.1	Knowledge based user authentication	15
13.3.2	Biometrical user-authentication	16
13.4	Resetting the retry counter (RC) and setting new reference data	16
14	Computing a digital signature	16
14.1	Signature-Process and Command-Overview	16
14.2	Chipcards without Hash-Function	17
14.2.1	Chipcards with Hash-Function	17
14.3	Selection of a Security Environment	18
15	Reading of Certificates and Certificate Verification Keys (PK.CAs)	18
15.1	Command-Sequences	18
15.2	Selecting and Reading of Data Files	19
16	Verifying a Digital Signature	19

16.1	Command Overview	19
16.2	Delivering a Hash-Value	19
16.3	Selecting the PK to Verify a Signature	20
16.4	Signatur-Verification	20
17	Signature-Protocol	20
17.1	Command-Sequences	20
17.2	Selecting the Protocol-File.....	21
17.3	Reading and Writing of a protocol record	21
18	Interaction with Customer Terminals	21
18.1	Security Functions and Command Overview.....	21
18.2	Read Chipcard Certificate.....	23
18.3	Authentication of the Customer Terminal	23
18.3.1	One-step Process	23
18.3.2	Two step Process	24
18.4	Selecting the Keys.....	24
18.5	Chipcard-Authentication	25
18.6	Customer Terminal-Authentication.....	25
18.7	Display a Message	25
18.8	Changing the Display Message	26
19	MF-Selection	26
20	Use of the signature function simultaneously with other applications.....	27
21	GET RESPONSE	27
22	CLA-Byte and Status Bytes.....	27
22.1	CLA-Byte.....	27
22.2	Status Bytes.....	27
Annex A:	DS-Formats	28
Annex B:	Authentication Certifikats	33
Annex C:	SigG Files	42
Annex D:	Device Authentication, Session Key Agreement and Secure Messaging	45
Annex E:	Object Identifier	57
Annex F:	Security Service Descriptor Templates	60
Annex G:	Use of the Security Service Descriptors	63
Annex H:	ATR-Coding	64

1 Scope

This specification defines the interface between a terminal (interface device) and a digital signature card, which is in compliance with the German digital signature law. The specification takes into account

- the German legal regulations
- relevant standards for smartcards (especially ISO/IEC 7816).

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

2 References

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 und Komponenten nach SigG/SigV (SigI)

Teil Zertifikate, Version 2.0, 31.07.1998

ISO/IEC 11770: 1996

Information technology - Security techniques - Key management

Part 3: Mechanisms using asymmetric techniques

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

Part 4 / 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 registration

ISO/IEC 7816-8: Final DIS 1998

Information technology - Identification cards - Integrated circuit(s) cards with contacts -

Part 8: Security related interindustry commands

ISO/IEC 7816-9: CD2 1998

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

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

3 Definitions, Abbreviations and Notations

3.1 Definitions

1. Digital Signature-Application (SigG Application): Application with SigG-AID and Digital Signature Function and possibly additional functions.

2. Digitale Signature Function (SigG-Function): Denotation for all functions, which are required according to SigG, SigV and the relevant 'Manual for Digital Signatures', like entity-authentication, user authentication, production of digital signatures and provision of objects (certificates, PK of the CA, ...)

3. Verification data: knowledge based or biometrical user authentication data (PIN/ password or fingerprint, ...)

3.2 Abbreviations

AID = Application Identifier
AKI = Authority Key Identifier
AT = CRT for Authentication
ATR = Answer-to-Reset
B = Byte
BCD = Binary Coded Digit
BER = Basic Encoding Rules
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
CV = Card Verifiable
D() = Decipherment of
DE = Data Element
DF = Dedicated File
DIR = Directory

DO = Data Object
DS = Digital Signature
DSI = Digital Signature Input
DST = CRT for Digital Signature
E() = Encipherment of
EF = Elementary File
FCI = File Control Information
FID = File Identifier
h() = hash value of
HB = Historical Bytes
ICC = Integrated Circuit(s) Card
ICCSN = ICC Serial Number
IFD = Interface Device
IFSC = Information Field Size Card
IFSD = Information Field Size Device
IIN = Issuer Identification Number
KID = Key Identifier
MF = Master File
MII = Major Industry Identifier
MSE = MANAGE SECURITY ENVIRONMENT
NAD = Node Address
OID = Object Identifier
PCA = Policy Certification Authority
PK = Public Key
PKCS = Public Key Cryptography Standards
PI = Padding Indicator
PIN = Personal Identification Number
PPS = Protocol Parameter Selection
PRND = Padding Random Number
PROV = Provider
PSO = PERFORM SECURITY OPERATION
PV = Plain Value
PW = Password
RC = Retry Counter
RCA = RootCA
RD = Reference Data
RegTP = Regulierungsbehörde Telekom. u. Post
RFU = Reserved for Future Use
SE = Security Environment
SIG() = Signature of
SigG = Signature law
SM = Secure Messaging
SSD = Security Service Descriptor
SK = Secret Key
SM = Secure Messaging
SN = Serial Number
SSC = Send Sequence Counter
TLV = Tag, Length, Value
VD = Verification Data
VR = Verification Requirement

3.3 Notations

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

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

<key descriptor> ::=

<key>.<keyholder>.<usage>

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

<secret key> ::= SK

<public key> ::= PK

<keyholder> ::= <cardholder> |
<certification authority> | <integrated
circuit(s) card> | <interface device>

<cardholder> ::= CH

<certification authority> ::= CA | RCA

<integrated circuit(s) card> ::= ICC

<interface device> ::= IFD

<usage> ::= <digital signature> |

<authentication> | <CertSign>

<digital signature> ::= DS

<authentication> ::= AUT

<CertSign> ::= CS

<certificate descriptor> ::=

<certificate>.<certholder>.<usage>

<certificate> ::= C

<certholder> ::= <cardholder> |

<certification authority> | <integrated
circuit(s) card> | <interface device>

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

|| = Concatenation of data

4 Technical characteristics

Digital signature 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) or ID-000 (Plug-in-Card, see EN 1375-1) as shown in fig. 1.

For bit transmission the 'direct convention' shall be applied. Programming power V_{pp} shall not

be requested and will not be supported by the IFDs.

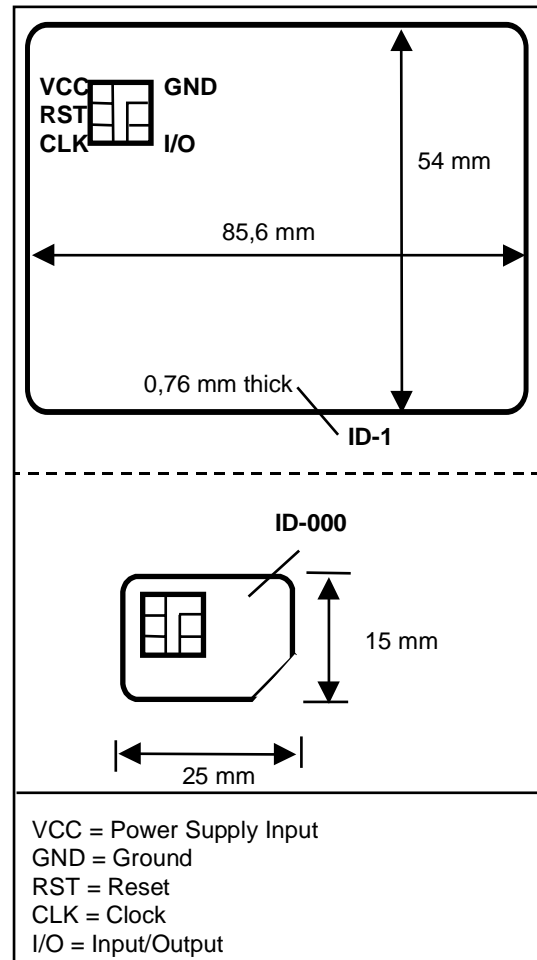


Fig. 1: Card formats

5 Answer-to-Reset

5.1 Global Interface Characters

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

5.2 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 H.

6 Protocol Parameter Selection

The support of Protocol Parameter Selection (PPS) according to ISO/IEC 7816-3 is strongly recommended. It may be used for selecting T=1 and to negotiate the Fi/Di parameter for achieving higher transmission rates.

7 Transmission protocols

7.1 General aspects

As transmission protocol the asynchronous half-duplex Character Transmission Protocol T=0 and/or the asynchronous half-duplex Block Transmission Protocol T=1 shall be supported.

7.2 Transmission protocol T = 0

The implementation shall comply with ISO/IEC 7816-3.

7.3 Transmission protocol T = 1

The implementation 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.

8 Flow diagram and embedding of the SigG function

Figure 2 depicts the logical flow chart for a chipcard with SigG application. The diagram demonstrates that after application selection there are two parallel branches, which show the use of the chipcard at different types of terminals:

- at a private or company user terminal
- at a customer terminal.

The chipcard cannot recognize at once the environment, in which it is used. At a customer terminal (i.e. a terminal, which, in the way of business, is offered to third party users), however, the use of the authentication branch is mandatory. The security module in the customer terminal, which may, for example, contain a plug-in-card, shall only allow the authentication procedure, if the security status of the customer remains unchanged.

During the authentication session keys are exchanged to protect the further communication via secure messaging (see Annex D). To ensure the user's confidence in this procedure, a card specific display info is read from the card and then displayed at the terminal. This display info can only be read from the card, after mutual authentication has successfully taken place.

Then the cardholder must prove either by a knowledge-based identification (input of PIN or password) and/or a biometric identification, that he is the legitimate user of the relevant chipcard. Knowledge-based and biometric authentication procedures can co-exist in the chipcard.

Depending on the configuration during the personalisation and/or the capabilities of the respective card operating systems chipcards can work in two different ways:

1. The card requires a user authentication every time before a digital signature is produced.
2. The card requires only a single user authentication, i.e from the point of view of the card any number of signatures may be produced after this one authentication.

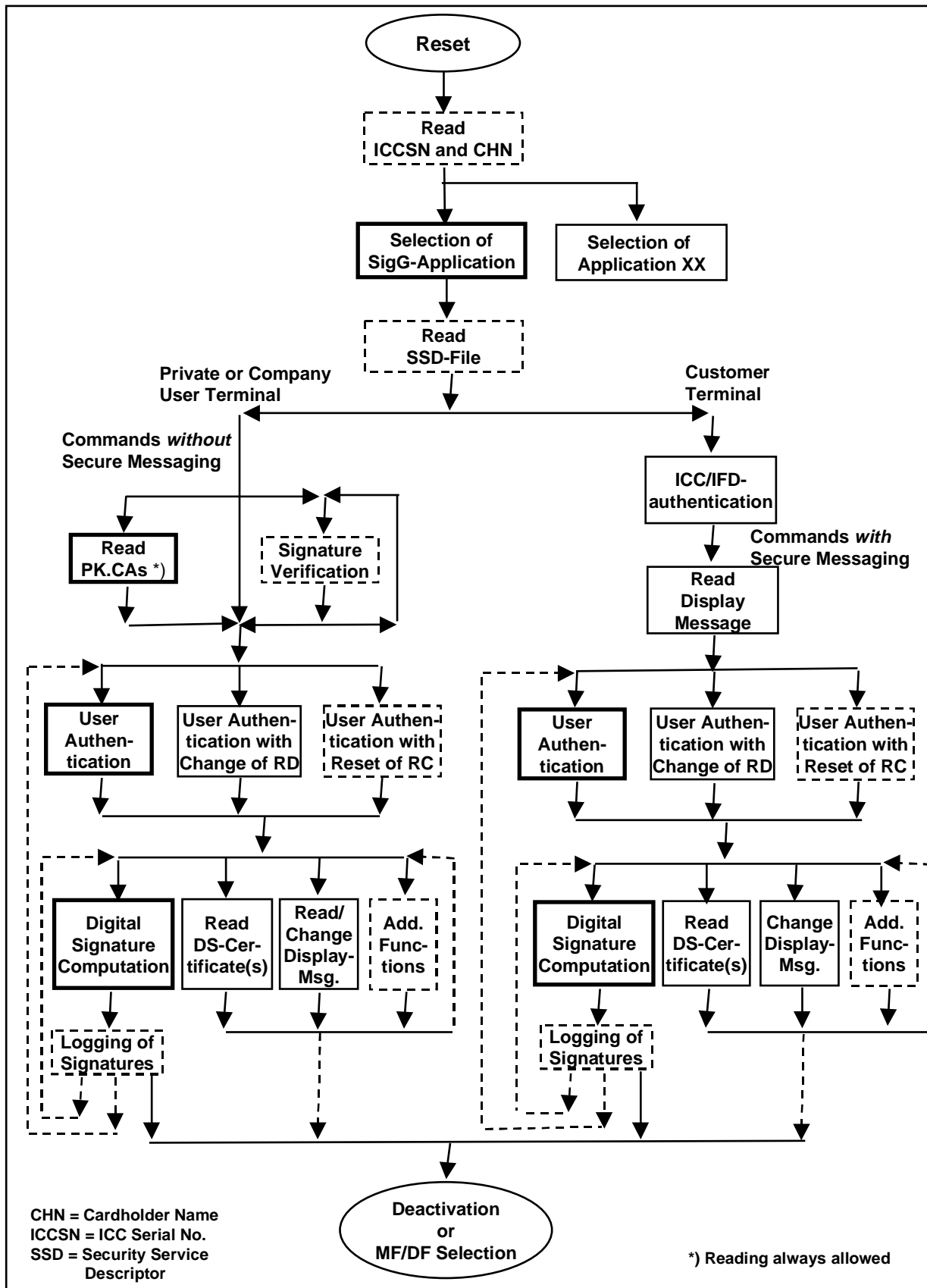


Fig. 1: Flowchart for Chipcards with SigG-Application

It would be helpful to have support of a counter which cuts off after a configurable number of signatures and thus reinstates the security status. Before a further signature can be used a successful user authentication is then required.

The PIN usage policy can be displayed in the SSD file (DO '5F2F').

With the second version the card holder can, however, install the required variants by means of the appropriate user operation or configuration of the application system in application environments where

- before every digital signature or
- before a digital signature after n digital signatures or
- before a digital signature after expiry of a definable time x after the last digital signature

the user authentication procedure must be applied again (protection against unauthorised change is required). System-wise this means that after the specified occurrence happens (signature created, n signatures created or time expired) the application system either causes

- a reset to the chipcard or
- a jump back to the superordinate DF (i.e. usually to the MF) by selection of the superordinate DF

so that when the user wishes to sign again a new application selection occurs and thereafter the user authentication is again requested.

The digital signature function (SigG function, see definition in 3.1) can also occur within the framework of another application, as shown in Fig. 3.

Within the signature application additional functions such as document encryption, may occur.

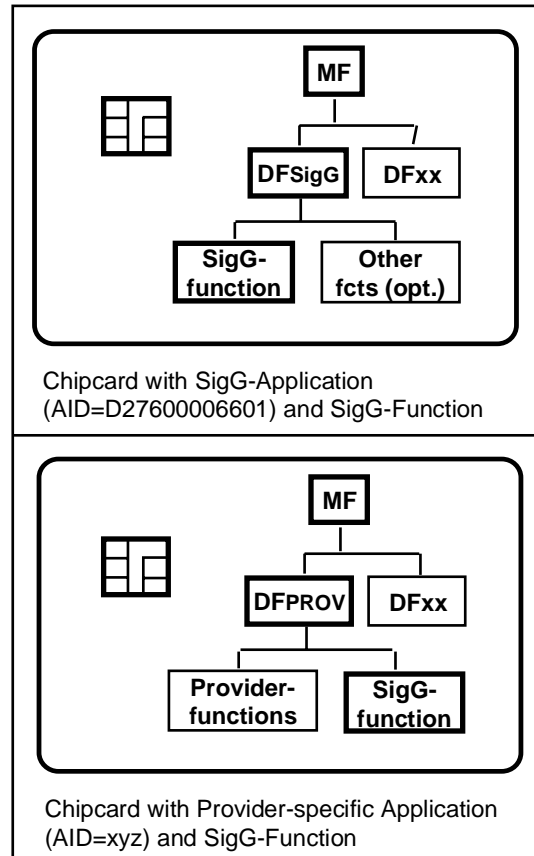


Fig. 2: Applications with SigG function

9 Certification authorities and certificates

The Regulatory Authority RegTP acts as root CA (RootCA) and provides certificates for the subordinate CAs. These then issue certificates for users with SigG chipcards (see Fig. 4).

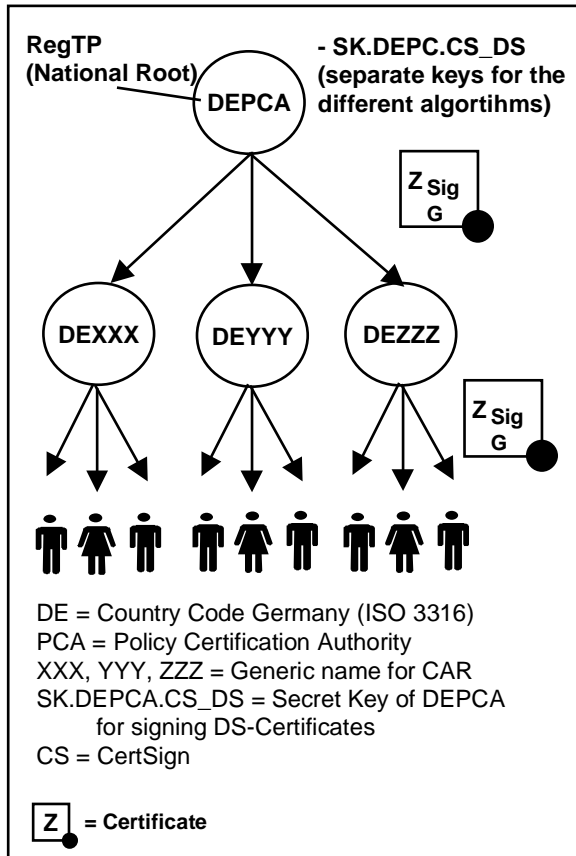


Fig. 3: Certificate authorities and certificates for SigG certificates

To use SigG chipcards with customer terminals a mutual authentication between chipcard and customer terminal is required. As again PK algorithms are to be employed ICC and IFD authentication certificates are required. Separate key pairs are used for signature and authentication. The chipcard is the carrier of the ICC authentication certificate; carrier of the IFD authentication certificate is a plug-in card as an example for the realisation of a terminal security module (see Fig. 5).

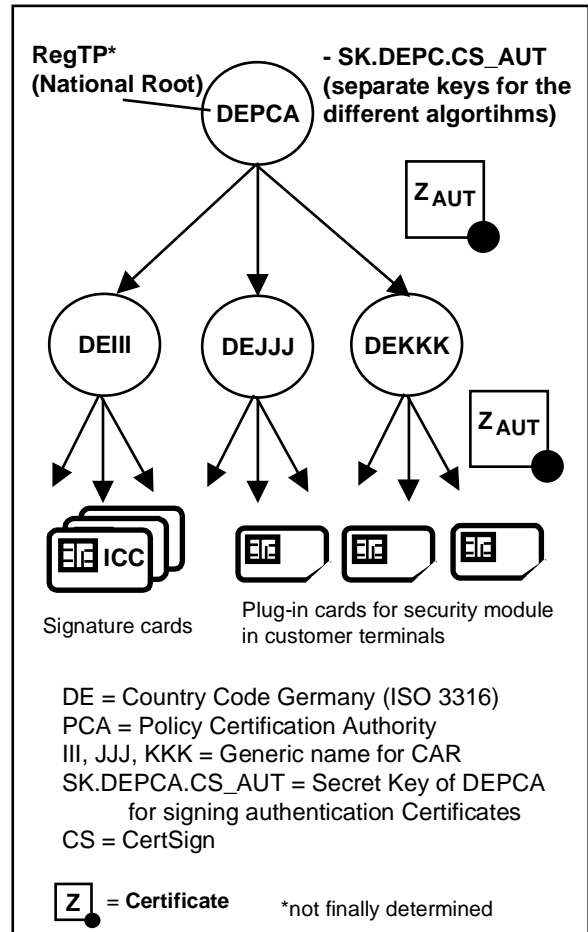


Fig. 4: Certificate authorities and certificates for ICC and IFD authentication certificates

10 File structure and data objects

10.1 File structure

The file organisation in the chipcard must be compatible with ISO/IEC 7816-4. The file structure of the chipcard is shown in Fig. 6. The characteristics of the individual files are described in Annex C.

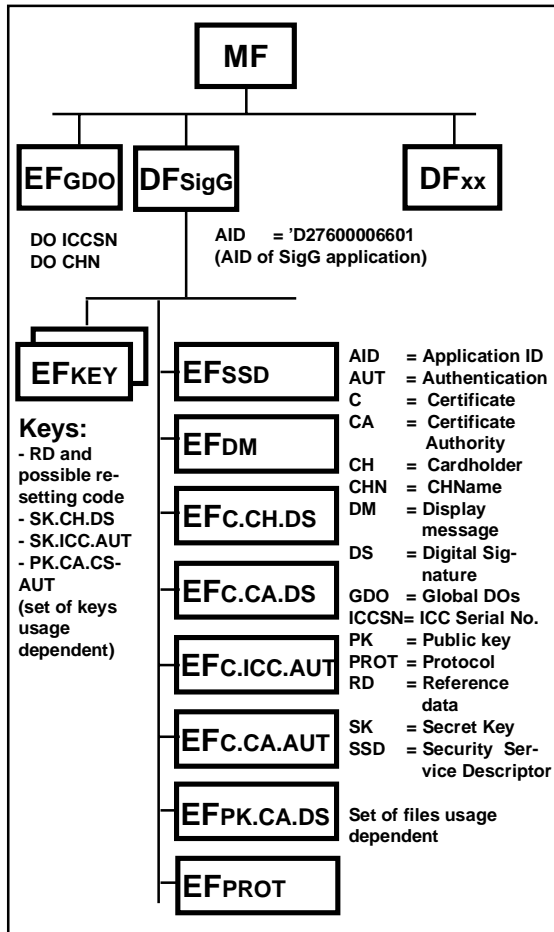


Fig. 5: File structure of a chipcard with SigG application

If a chipcard supports more than one signature algorithm (e.g. RSA and DSA or more signature formats <e.g. RSA acc.to ISO 9796-2 and PKCS #1>), the various user certificates are stored in separate files (see Annex C).

The SigG application (DFSigG) has an ISO/IEC 7816-5 conform application identifier (AID), whose structure and coding is shown in Fig. 7.

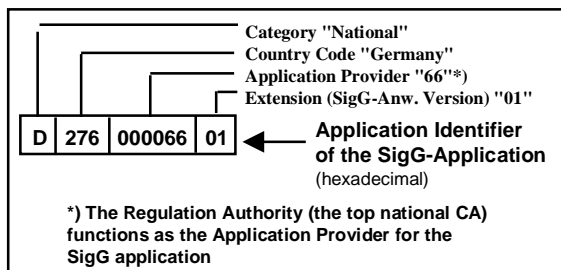


Fig. 6: AID of the SigG application

10.2 EFGDO

The DO 'ICC Serial Number (ICCSN)' (tag '5A', see Fig. 8) and the DO 'Cardholder Name' (tag '5F20', contents as on the card cover) are to be stored in the EFGDO.

The ICCSN consists of the card manufacturer identifier followed by a serial number and a check byte acc. to the Luhn formula (see ISO 7812). The ICCSN is a unique chipcard identifier and can, for example, be used for chipcard revocation lists. The exact structure of the ICCSN can vary according to the application. The figs. 8 and 9 show possible versions. Use of the version shown in fig. 9, however, is recommended.

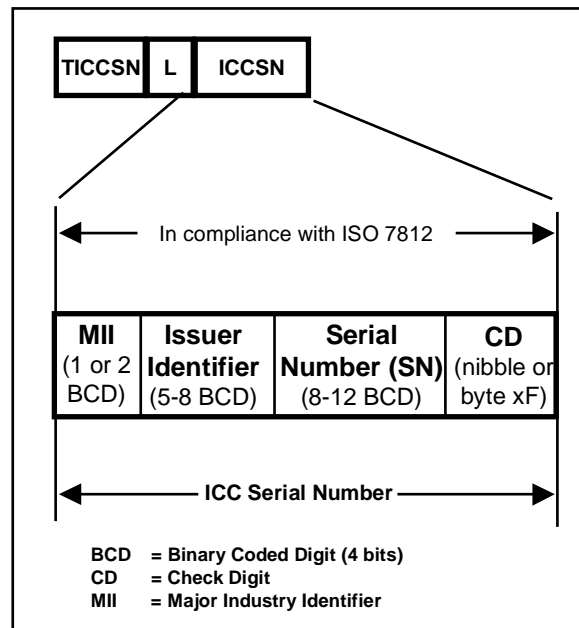


Fig. 7: ICCSN (ISO 7812 orientated)

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

Fig. 8: ICCSN (ISO/IEC 7816-5/6 orientated with Category D and Country Code)

Note: CCG in Cologne, as the accredited authority with DIN, is responsible for the issue of an IIN as per fig. 9.

The DO CHN contains the cardholder name in the same form as on the cover of an ID 1 chipcard. ASCII shall be used as the coding scheme for CHN.

It should be possible to read the global data objects with SELECT FILE and READ BINARY, whereby the presence of DO CHN in chipcards with the format ID-1 is optional, but for chipcards with the format ID-000 mandatory.

10.3 EFKEY

The key/reference data required for the SigG application are organised in 3 groups:

1. reference data for user authentication

- Reference data for the knowledge based user authentication (PIN or password, resetting code)
- As required reference data for biometric user authentication (biometric data, resetting code)

2. Secret keys

- SK.CH.DS for the digital signature function (when required several secret keys)
- SK.ICC.AUT of the chipcard for authentication

3. Public keys

- PK.CA.AUT for terminal authentication (when required several public keys)
- When required one or several PK.IFD.AUT

The structuring of key files and the way of using them is specific to the operating system.

The reading of secret keys and reference data must be prevented, the tampering with of installed secret keys also. This also applies equally to the resetting code. Moreover it must be ensured that each key can only be used for the appropriate service (e.g. a signature key may only be used to create the digital signature).

Public keys are usually stored in a separate PK key file. PKs are entered either during personalisation of the card or by the operating system as a sub-operation of VERIFY CERTIFICATE (e.g. entering a PK.IFD.AUT).

A KID, usually 1 byte long, is used as a key reference for reference data and secret keys. With PK.CA.AUT keys the identifier of the certification authority is used as the key identifier. With PK.IFD.AUT keys the IFD identifier is used. PKs are write-protected objects.

10.4 EFSSD

The EFSSD provides for the terminal information to address and control the SigG function. This information delineates the permissible processes including:

- indication of the algorithms that can be used for an external authentication (terminal authentication)
- indication, how the chipcard creates the digital signature (e.g. hash function available, various DSI formats)
- indication whether signatures can also be verified
- indication whether for the user authentication only the knowledge based identification procedure (PIN/password procedure) or whether also a biometric identification procedure is available. When using the knowledge based identification procedure it can be indicated whether the chipcard uses the PIN format "Format 2 PIN block" additionally or alternatively.

The available security functions are described by means of security service descriptors. For this reason SSD templates or a security service profile identifier can be found.

Structure and contents of the EFSSD are shown in Annex F.

10.5 EFDM

In the file EFDM the Display Message is kept. It should consist of 8 letters 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.

10.6 EFC.CH.DS

The file EFC.CH.DS contains the 'signature'-certificate of the cardholder. The structure of the DS certificate is described in SigI (Signature-Interoperability Specification). If the cardholder has more than one signature certificate, they are stored in separate files.

10.7 EFC.CA.DS

The file EFC.CA.DS contains the 'signature'-certificate of the certification authority, which has produced the signature certificate C.CH.DS of the cardholder. The signature certificate

C.CA.DS is produced by the highest certification authority (RegTP).

10.8 EFC.ICC.AUT

The file EFC.ICC.AUT contains the 'authentication' certificate C.ICC.AUT (structure see Annex B).

10.9 EFC.CA.AUT

The file EFC.CA.AUT contains the 'authentication'-certificate C.CA.AUT of the certification authority (structure see Annex B).

10.10 EFPK.CA.DS

The file EFPK.CA.DS contains those public keys of certification authorities, which serve as 'security anchors' for the software, when verifying certificates. The contents of EFPK.CA.DS is described in Annex A.

10.11 EFPROT

The optional file EFPROT 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.

11 Chipcard opening

11.1 Command sequences

After the chipcard has been inserted a reset must follow. Conditionally (see Chapter 10.2) after reset the global data objects ICCSN and CHN can be read.

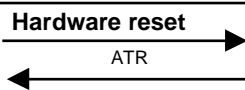
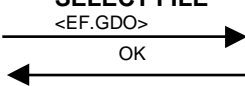
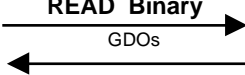
ICC Commands	Meaning
Hardware reset 	Cold reset and Answer-to-Reset with indication of the selection method
SELECT FILE 	Selection of EF.GDO
READ Binary 	Reading DO ICC Serial No. and DO CardholderName

Fig. 9: Chipcard opening

11.2 Reading the global data objects

To read the global data objects the ISO/IEC 7816-4 commands SELECT FILE and READ BINARY are used.

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 subsequent data field
Data field	FID of GDO file, see annex C
Le	Empty

Tab. 1: SELECT FILE command to select the GDO File

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

Tab. 2: SELECT FILE response

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'B0' = READ BINARY
P1	'00'
P2	'00'
Lc	Empty
Data field	Empty
Le	'00' = Read until end-of-file

Tab. 3: READ BINARY command to read the global data objects

Data field	DO ICCSN DO CHN
SW1-SW2	'9000' or specific status bytes

Tab. 4: READ BINARY response

12 Application opening phase

12.1 Command sequences

The application opening phase consists of the application selection and optionally of the reading of the SSD file.

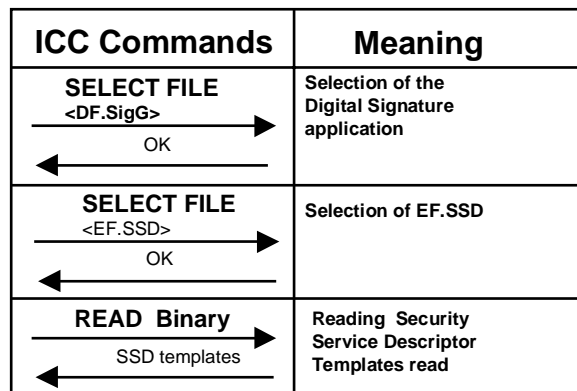


Fig. 10: Application opening phase

12.2 Application selection

For application selection the ISO/IEC 7816-4 command SELECT FILE (selection by AID) is used (direct application selection).

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'A4' = SELECT FILE
P1	'04' = DF selection by AID
P2	'0C' = No FCI to be returned
Lc	'06' = Length of subsequent data field
Data field	'D27600006601' = AID of the SigG-application
Le	Empty

Tab. 5: SELECT FILE command for application selection with AID

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

Tab. 6: SELECT FILE response

12.3 Reading of SSD Templates

To read the Security Service Descriptors the following commands are needed:

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 subsequent data field
Data field	FID of SSD file, see Annex C
Le	Empty

Tab. 7: SELECT FILE command to select the SSD File

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

Tab. 8: SELECT FILE response

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'B0' = READ BINARY
P1	'00'
P2	'00'
Lc	Empty
Data field	Empty
Le	'00' = Read until end-of-file

Tab. 9: READ BINARY command to read the SSD Templates

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

Tab. 10: READ BINARY response

13 Authentication of the Card Holder

13.1 Command-overview

The commands for user authentication VERIFY, CHANGE REFERENCE DATA und RESET RETRY COUNTER (see Fig. 12) have different functionality and can be used independently of each other.

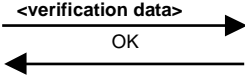
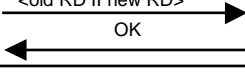
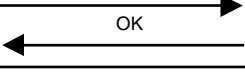
ICC Commands	Meaning
Verify <verification data> 	User authentication
CHANGE RD <old RD new RD> 	User authentication with Changing the reference data
RESET RC <resetting code new RD> 	User authentication with Resetting the retry counter and setting new Reference Data

Fig. 12: Commands for user-authentication

If the user authentication commands are sent to a customer terminal, then the data field of each command has to be transmitted as a cryptogram and the command has to be sent with a cryptographic checksum (See Annexe D).

13.2 User authentication

For user authentication the ISO/IEC 7816-4 the command VERIFY is used. The initial value of the retry counter is 3 for PIN/password presentation. For biometrical authentication the initial value must be defined for each individual procedure.

The way how the reference data are stored in the chipcard is not within the scope of this specification.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'20' = VERIFY
P1	'00'
P2	'81' = PIN/PW reference '91' = Biometrical data reference
Lc	'xx' = Length of subsequent data field
Data field	If P2 = '81': PIN or PW (min 6, max 8 ASCII characters) If P2 = '91': Biometrical verification data
Le	Empty

Tab. 11: VERIFY command

Further PIN formats may be used such as

P2 = '80': PIN in the format 2 PIN block (see ISO 9564-1).

Such additional formats can be indicated in the SSD-file.

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

Tab. 12: VERIFY response

The following status bytes are of special importance:

- '9000': user authentication OK
- '6300': Warning - verification failed (no further information)
- '6983': Checkingerror: authentication method blocked (these status bytes shall be delivered, if the VERIFY command is sent and the retry counter is at zero)
- '63CX': Warning - verification failed, 'X' indicates the number of retries still allowed

13.3 Changing the reference data

13.3.1 Knowledge based user authentication

To change the reference data (PIN or password) the ISO/IEC 7816-8-command CHANGE REFERENCE DATA is used. The CHANGE RD command may be used at any time after application selection.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'24' = CHANGE REFERENCE DATA
P1	'00' = Exchange reference data
P2	'81' = PIN/PW reference
Lc	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)
Le	Empty

Tab. 13: CHANGE REFERENCE DATA command

The length of the old reference data is known in the chipcard, so that neither a delimiter nor padding for filling up fixed formats is necessary. If the length of the old reference data is 6 and the length of the new reference data is 7, then the Lc field must be '0D' (= 13).

If the format 2 PIN block (see 13.2) is used in the VERIFY command, this format has to be used here too with the parameter P2 = '80'.

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

Tab. 14: CHANGE REFERENCE DATA response

13.3.2 Biometrical user-authentication

To change biometrical reference data, if applicable to and possible with the relevant procedure (e.g. with the fingerprint procedure), the ISO/IEC 7816-8 command CHANGE REFERENCE DATA is used. In this case the command is only allowed after successful user authentication, because here only the new reference data are transferred (to the command) ?.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'24' = CHANGE REFERENCE DATA
P1	'01' = Change reference data
P2	'91' = Biometrical data reference
Lc	Length of subsequent data field
Data field	New reference data
Le	Empty

Tab. 15: CHANGE RD command

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

Tab. 16: CHANGE RD response

13.4 Resetting the retry counter (RC) and setting new reference data

After three successive false presentations of the reference data the use of the SigG function is blocked.

In this case with the ISO/IEC 7816-8 command RESET RETRY COUNTER the retry counter can be reset to its initial value and thus the function SigG can be used again. The resetting code must be entered. (8 digits, initial value for the retry counter is 3.).

For knowledge based user authentication it is also possible to define a new PIN or password (P1 = '00').

For biometrical user authentication the support of P1 = '01' only is recommended.

The support of this command is optional.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'2C' = RESET RETRY COUNTER
P1	'00' = Reset retry counter and set new reference data '01' = Reset retry counter

P2	'81' = PIN/PW reference '91' = Biometrical data reference
Lc	Length of subsequent data field
Data field	- if P1 = '00': Resetting code (8 bytes) followed by new reference data (6 to 8 bytes, ASCII coding) - if P1 = '01': Resetting code (8 bytes)
Le	Empty

Tab. 17: RESET RETRY COUNTER command

If the format 2 PIN block (see 13.2) is used in the VERIFY command, this format has to be used here too with the parameter P2 = '80'.

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

Tab. 18: RESET RETRY COUNTER response

14 Computing a digital signature

14.1 Signature-Process and Command-Overview

The following figure shows in general the different steps of a digital signature process. The function 'Format Mechanisms' is empty for algorithms, which use the 'plain' hash value for the digital signature production (e.g. DSA).

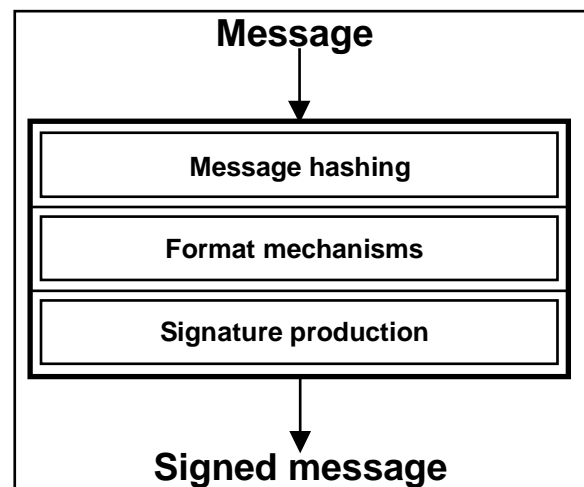


Fig. 13: Signature-process

There are various alternatives when producing the digital signature: The hashing is done

case 1: outside the chipcard.

- case 2: partly inside (last round of hashing) the chipcard.
- case 3: completely inside the chipcard.

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

Fig. 14: Sequence of signature production without hashing in the chipcard

ICC Commands	Meaning
PSO: HASH <Intermediate hash result II textblock> 	Hashing the last round in the chipcard
PSO:COMPUTE DS 	Computation of a digital signature (command can be repeated)

Fig. 15: Sequence of signature production with hashing in the chipcard (last round)

ICC Commands	Meaning
PSO: HASH <Hash blocks> 	Computation of the hash value in the card (several HASH-Commands with Command chaining)
PSO:COMPUTE DS 	Computation of a digital signature (command can be repeated)

Fig. 16: Sequence of signature production with hashing in the chipcard (complete hashing)

14.2 Chipcards without Hash-Function

The ISO/IEC 7816-8 command for digital signature computation is shown in the table below. The hash value (or if required the DigestInfo) is delivered in the data field of the command. Signature key as well as signature algorithm and the related Digital-Signature-Input formats (DSI formats, see Annex A) are either

- pre-set and thus implicitly selected or

- must be set explicitly with the command **MANAGE SECURITY ENVIRONMENT** (see 16.3).

Which algorithm and which DSI-format are pre-set, is defined in the SSD file.

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

Tab. 19: PSO: COMPUTE DS command

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

Tab. 20: PSO: COMPUTE DS-response

14.2.1 Chipcards with Hash-Function

a) Hashing of the last round(s) before signature computation

Chipcards with an integrated hash function 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 to deliver 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	Length of subsequent data field
Data field	<ul style="list-style-type: none"> - If data shall be hashed: '90' - L - Intermediate hash result (see annex A) '80' - L - Data to be hashed - If the final hash value shall be transmitted: '90' - L - Hash value

Le	Empty
----	-------

Tab. 21: PSO: HASH command

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

Tab. 22: PSO: HASH response

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

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'2A' = PERFORM SECURITY OPERATION: COMPUTE DIG. 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 length of expected digital signature

Tab. 23: PSO: COMPUTE DS command

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

Tab. 24: PSO: COMPUTE DS response

b) Complete Hashing in the Chipcard

Chipcards with an integrated hash function can compute the complete hash value in the card. The ISO/IEC 7816-8 command for hashing is PSO: HASH. In this case command-chaining (see ISO/IEC 7816-8) may be necessary.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'2A' = PERFORM SECURITY OPERATION: HASH
P1	'90' (=compute hash value)
P2	'80' (=data field contains data to be hashed)
Lc	Length of subsequent data field
Data field	Data to be hashed (one or more blocks)
Le	Empty

Tab. 25: PSO: HASH command

Note: The last transferred block may be smaller than the normal block length. The padding of the hash input is done by the card.

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

Tab. 26: PSO: HASH response

After hashing the command PSO: COMPUTE DS has to be sent (see Tab. 23 and 24).

14.3 Selection of a Security Environment

If the chipcard supports more than one signature algorithm (e.g. RSA or DAS) or several signature formats for one signature algorithm, the relevant security environment (SE) can be set using the MSE command. The default value is SE no 1. Which SEs are supported is defined in the SSD file (see Annexes F and G).

Alternatively an alg ID can be set using the SET option of the MSE command (see Annex F).

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'F3' (= RESTORE)
P2	'xx' (= SE number)
Lc	Empty
Data Field	Empty
Le	Empty

Tab. 27: MANAGE SECURITY ENVIRONMENT command

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

Tab. 28: MANAGE SECURITY ENVIRONMENT response

15 Reading of Certificates and Certificate Verification Keys (PK.CAs)

15.1 Command-Sequences

To read a certificate or a certificate key the following commands are necessary:

ICC Commands	Meaning
SELECT FILE <EF.C.CH.DS> → ← OK	Selection of EF.C.CH.DS
READ Binary → ← Certificate C.CH.DS	Reading Certificate C.CH.DS
SELECT FILE <EF.PK.CA.DS> → ← OK	Selection of EF.PKCA.DS
READ Binary → ← PK.CA.DS-Info	Reading PK.CA.DS-Info

Fig. 17: commands for reading certificates and public keys

15.2 Selecting and Reading of Data Files

CLA	As defined in ISO/IEC 7816-4 and -8
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 respective certificate file or public key file (see Annex C)
Le	Empty

Tab. 29: SELECT FILE command to select a file containing a certificate or a PK

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

Tab. 30: SELECT FILE response

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'B0' = READ BINARY
P1, P2	'0000' or offset
Lc	Empty
Data field	Empty
Le	'xx' = Length of data to be read ('00' = Read until end-of-file)

Tab. 31: READ BINARY command to read a certificate

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

Tab. 32: READ BINARY response

The public keys of certification authorities must be read with the same commands, but in this case the FID of EF.PK.CA.DS has to be used.

16 Verifying a Digital Signature

16.1 Command Overview

To verify a digital signature (this function is optional) the following data elements are necessary:

- the hash value
- the public key of the signatory
- the digital signature

As chipcards with today's technology cannot interpret X509-certificates, this function is only of value, when the PK of the signatory is already present in the chipcard (this is not the normal situation).

When a reversible digital signature algorithm is used, the chipcard must know the applied DSI format to find the hash value.

The command sequence for the verification of a digital signature is shown in Fig. 18.

ICC Commands	Meaning
PSO: HASH <hash related data> → ← OK	Transfer of the hash value or hashing the last round
MSE SET <key ref> → ← OK	Setting the Key Reference of PK.USER.DS
VERIFY SIGNATURE <digital signature> → ← OK	Signature Verification

Fig. 18: commands to verify a signature

16.2 Delivering a Hash-Value

To deliver the hash value to the card the command PSO: HASH is used.

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	Length of subsequent data field
Data field	- If data shall be hashed: '90' - L - Intermediate hash result (see annex A) '80' - L - Data to be hashed - If the final hash value shall be transmitted: '90' - L - Hash value
Le	Empty

Tab. 37: PSO: HASH command

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

Tab. 38: PSO: HASH response

16.3 Selecting the PK to Verify a Signature

Before the command PSO: VERIFY SIGNATURE can be used, the PK of the signatory has to be selected via the command MSE.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'81' (= SET for verification)
P2	'B6' (= DST)
Lc	'xx' = Length of subsequent data field
Data field	'83 0x xx ... xx' (DO KeyRef of the user's PK, tag '83')
Le	Empty

Tab. 33: MANAGE SECURITY ENVIRONMENT command

Data field	Empty
SW1-SW2	'9000' (= PK set) or '6xxx' (= PK not available or other error condition)

Tab. 34: MANAGE SECURITY ENVIRONMENT response

16.4 Signatur-Verification

Then the command PSO: VERIFY DS follows:

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'2A' = PERFORM SECURITY OPERATION: VERIFY DIGITAL SIGNATURE
P1	'00'
P2	'A8' (= data field contains DOs relevant for verification)

Lc	'xx' = Length of subsequent data field
Data field	'9E'-L-Digital signature (DSI format to be detected by the card)
Le	Empty

Tab. 35: PSO: VERIFY DS command

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

Tab. 36: PSO: VERIFY DIGITAL SIGNATURE response

17 Signature-Protocol

17.1 Command-Sequences

For each signature produced (*created*) a protocol record is written.

The chipcard should write

- the current number registered by the signature counter (to be incremented by the card),
- the signed hash value,
- date and time and
- when present the terminal ID

into a cyclic file. It must be possible to add to this information text from outside the card, for example the ID of the signed document. These data could, for instance be delivered in the DO „Plain Value“ within the command PSO: COMPUTE DS. This service, however, is not yet available. So if the protocol file is present in the chipcard, the protocol record must be produced completely outside the card as specified in chapter 10.11 and then be written into the protocol file.

Before this, however, the last protocol entry must be read to get the latest signature counter reading. Fig. 19 shows the command sequence for writing a signature protocol entry.

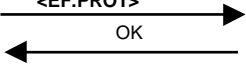
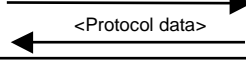
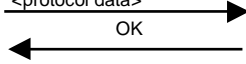
ICC Commands	Meaning
Select File <EF.PROT> 	Selection of protocol-file
Read Record 	Reading last protocol entry
Append Record <protocol data> 	Writing new protocol entry

Fig. 19: Commands for signature-protocol

17.2 Selecting the Protocol-File

First the File EFPROT 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 EFPROT, see annex C
Le	Empty

Tab. 37: SELECT FILE command to select the protocol-file

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

Tab. 38: SELECT FILE response

17.3 Reading and Writing of a protocol record

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. 39: READ RECORD command to read a protocol-record

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

Tab. 40: 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 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. 41: APPEND RECORD command to write a protocol record

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

Tab. 42: APPEND RECORD response

18 Interaction with Customer Terminals

18.1 Security Functions and Command Overview

If the signature card is applied at a customer terminal, after application opening an entity authorisation must be processed, before further services of the chipcard can be used. The aim of the authorisation is

- to prove the access right to read the display messages and
- to agree on session keys.

The following steps have to be taken:

Step 1: exchange of the AUT-certificates and their verification (if necessary)
 Step 2: running the authentication procedure and agreement on session keys for secure messaging

Step 3: reading the display message (already done with secure messaging) and, if necessary, changing the display message

The agreed session keys are deleted when the application is closed.

If a chipcard cannot be authenticated the terminal shall terminate the communication with this card, at least with the SigG application or the SigG function in this card.

If the terminal cannot be authenticated the chipcard must assure that

- no further commands are processed within this application or
- only those commands are processed which are allowed at the current security status.

If the mutual authentication is successful the terminal can proceed with the communication with the chipcard, whereby

- each command and each response is protected by a cryptographic checksum and
- certain data fields such as „Verification Data“ are transferred as cryptograms.

In the following text only those commands for the signature card are described, which are used within the context of authentication procedure including certification verification and output and change of the display message. The other commands are the same as those which are used at a private or company terminal – with the difference, however, that the commands at a customer terminal are protected by secure messaging (see Annex D).

ICC Commands	Meaning
SELECT FILE <EF.C.ICC.AUT> → ← OK	Selection of EF.C.ICC.AUT
READ Binary → ← Certificate C.ICC.AUT	Reading of Certificate C.ICC.AUT (handed over to be checked by the customer terminal)
MSE: SET <key ref> → ← OK	Setting the Key Reference of PK.CA.CS_AUT
VERIFY CERTIFICATE <Certificate C.IFD.AUT> → ← OK	Verification of terminal certificate and temporary storing of - PK.IFD.AUT with Key reference and OID (opt) - Certificate Holder Authorisation CHA

Fig. 20: Changing and Checking the Certificates

ICC Commands	Meaning
MSE <key refs> → ← OK	Setting the key Reference of SK.ICC.AUT and PK.IFD.AUT
INT. Authenticate <Challenge> → ← <Authentication data>	Authentication of the chipcard by the terminal
GET CHALLENGE → ← Challenge	Getting a random no.
EXT. AUTHENTICATE <Authentication data> → ← OK	Authentication of the terminal by the chipcard

Fig. 21: Authentication procedure with session key agreement

ICC Commands	Meaning
Select File <EF.DM> → ← OK	Selection of EF.DM
READ BINARY → ← Display Message	Reading Display-Message
UPDATE BINARY <Display Message> → ← OK	Changing Display-Message if required by the user

Fig. 22: Read and Change Display Message (if necessary)

18.2 Read Chipcard Certificate

Chipcard authentication by the terminal is done by first reading the ICC authentication certificate C.ICC.AUT via the commands SELECT FILE and READ BINARY (it may be necessary to read the certificate C.CA.AUT as well).

The verification of the certificate C.ICC.AUT is not within the scope of this specification.

18.3 Authentication of the Customer Terminal

Verification of the customer terminal certificate by the chipcard is done by first setting the PK to PK.CA.CS_AUT. In the chipcard the following keys are available

- PK.DEPCA.CS_AUT (mandatory) and
- PK.CA.CS_AUT of that certification authority, which has certified the ICC-certificate (optional).

18.3.1 One-step Process

If the customer terminal has a certificate of the same certification authority as the chipcard or is that PK.CA.CS_AUT present in the chipcard, which is necessary to verify the certificate C.IFD.AUT, then the chipcard can verify this certificate directly.

Thus when proceeding in one step, the PK has always to be set to PK.CA.CS_AUT.

Note: If PK.IFD.AUT is already present in the chipcard the command VERIFY CERTIFICATE can be omitted.

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'22' = MANAGE SECURITY ENVIRONMENT
P1	'81' (= SET for verification)
P2	'B6' (= DST)
Lc	'0A' = Length of subsequent data field
Data Field	'83 08 xx ... xx' (KeyRef of PK.CA.CS_AUT, consisting of CAR, see annex B)
Le	Empty

Tab. 43: MANAGE SECURITY ENVIRONMENT command

Data field	Empty
SW1-SW2	'9000' (= PK set) or '6xxx' (= PK not available or other error condition)

Tab. 44: MANAGE SECURITY ENVIRONMENT response

If the MSE command returns an error message two-step processing has to be applied.

If the status bytes returned are '9000' the command VERIFY CERTIFICATE may follow. Data and the use of the chaining function are independent of the supported PK algorithm. If the verification of the certificate was successful the chipcard must (temporarily) store the following data elements:

- PK.IFD.AUT with key reference (= CHR) and OID
- CHA (is needed for the verification of access rights).

Note: The command VERIFY CERTIFICATE may only be processed successfully, if a PK.CA key was set with the command MSE and if the certificate was correct, i.e. if a different PK was set the command VERIFY CERTIFICATE returns an error.

- a) Certificate for signature with message recovery

The certificate is transferred to the chipcard with command chaining.

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. 45: PSO: VERIFY CERTIFICATE command

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

Tab. 46: 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. 47: PSO:VERIFY CERTIFICATE command

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

Tab. 48: PSO: VERIFY CERTIFICATE response

b) Certificate for signature with message recovery

The certificate is transferred to the chipcard with command chaining.

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	'5F4E'-L-Certificate content (see annex B)
Le	Empty

Tab. 49: PSO: VERIFY CERTIFICATE command

Data field	Empty
------------	-------

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

Tab. 50: 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. 51: PSO: VERIFY CERTIFICATE command

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

Tab. 52: PSO: VERIFY CERTIFICATE response

18.3.2 Two step Process

In a two step process the following steps must be taken:

- Set the PK of the root authority (PK.DEPCA.CS_AUT) via MSE (KeyRef = "DEPCA" || Extension)
- Verification of the root authority certificate C.CA.AUT and store (temporarily) the PK.CA.AUT under the respective Key Reference
- Set the PK of the CA, which has produced the certificate C.IFD.AUT via MSE
- Verification of the certificate C.IFD.AUT and store (temporarily) the needed certification data elements.

18.4 Selecting the Keys

Before processing any authorisation commands the necessary keys must be set via the MSE command. The command MSE checks, whether the keys named (qualified with purpose of use and if needed further parameters) are available. Then the keys are set. Following commands check, whether the keys set here can be used.

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)
Lc	'11' = Length of subsequent data field
Data Field	'83 0C xx ... xx' (KeyRef of PK.IFD.AUT, i.e. ICCSN.IFD) '84 01 81' (KeyRef of SK.ICC.AUT)
Le	Empty

Tab. 53: MANAGE SECURITY ENVIRONMENT command

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

Tab. 54: MSE response

18.5 Chipcard-Authentication

Chipcards are authenticated with the command INTERNAL AUTHENTICATE. Before using the keys which were set with the command MSE it must be checked whether they are permissible, i.e. the private key SK.ICC.AUT and the public key PK.IFD.AUT must be selected (see also in Annex D the notes before the relevant table with the data elements of the command INTERNAL AUTHENTICATE).

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'

Tab. 55: INT. AUTHENTICATE command

Data field	Authentication data, see annex D
SW1-SW2	'9000' or specific status bytes

Tab. 56: INT. AUTHENTICATE response

18.6 Customer Terminal-Authentication

The necessary commands to authenticate a customer terminal are:

- GET CHALLENGE
- EXTERNAL AUTHENTICATE

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

Tab. 57: GET CHALLENGE command

Data field	Challenge (8 bytes)
SW1-SW2	'9000' or specific status bytes

Tab. 58: GET CHALLENGE response

Note: The returned challenge is only valid for the next command.

After GET CHALLENGE the command EXTERNAL AUTHENTICATE follows. Before using the keys which were set via the command MSE it must be checked whether they are permissible (see also in Annex D the notes before the relevant table with the data elements of the command INTERNAL AUTHENTICATE).

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, see annex D
Le	Empty

Tab. 59: EXT. AUTHENTICATE command

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

Tab. 60: EXT. AUTHENTICATE response

18.7 Display a Message

After the mutual authentication has been processed successfully, 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 sensible, although no interaction with the chipcard 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 DO 00' '8E'-L-'xx ... xx' (= PV-DO with FID of EFDM CC-DO)
Le	'00'

Tab. 61: SELECT FILE command to select the Display-Message-file

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

Tab. 62: 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. 63: 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. 64: 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.

18.8 Changing 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. 65: 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. 66: UPDATE BINARY response

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

19 MF-Selection

If the MF is selected after the digital signature (see Fig. 2), this is done with the following command:

CLA	As defined in ISO/IEC 7816-4 and -8
INS	'A4' = SELECT FILE
P1	'00'
P2	'0C' = No FCI to be returned
Lc	'02'
Data field	'3F00' (= FID of MF)
Le	Empty

Tab. 67: SELECT FILE command to select the MF

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

Tab. 68: SELECT FILE response

20 Use of the signature function simultaneously with other applications

The signature application can be started at any time in a separate logical channel, i.e. if several applications in the chipcard need the signature function and all these applications are to be continued after processing the digital signature, then the card must support the „Logical Channel“ mechanism.

CLA	As defined in ISO/IEC 7816-4 and –8
INS	'70' = MANAGE CHANNEL
P1	'00' = Open logical channel
P2	'01' or '02' or '03' = Channel no.
Lc	Empty
Data field	Empty
Le	Empty

Tab. 69: MANAGE CHANNEL command to open channel 1, 2 or 3

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

Tab. 70: MANAGE CHANNEL response

CLA	As defined in ISO/IEC 7816-4 and –8
INS	'70' = MANAGE CHANNEL
P1	'80' = Close logical channel
P2	'01' or '02' or '03' = Channel no.
Lc	Empty
Data field	Empty
Le	Empty

Tab. 71: MANAGE CHANNEL command to close channel 1, 2 or 3

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

Tab. 72: MANAGE CHANNEL response

21 GET RESPONSE

Chipcards with transmission protocol must additionally support the ISO/IEC 7816-4-command GET RESPONSE, to get the responses of case

4 commands (data in command, data in response) from the card (e.g. in the command INTERNAL AUTHENTICATE). The length of the data is returned in the Status-Bytes SW1-SW2 = '61xx'.

CLA	As defined in ISO/IEC 7816-4 and –8
INS	'C0' = GET RESPONSE
P1	'00'
P2	'00'
Lc	Empty
Data field	Empty
Le	'xx' = Length of data to be retrieved

Tab. 73: GET RESPONSE command

Data field	Data (Le bytes)
SW1-SW2	'9000' or specific status bytes

Tab. 74: GET RESPONSE response

22 CLA-Byte and Status Bytes

22.1 CLA-Byte

According to ISO/IEC 7816-4 and 7816-8 the CLA byte is normally coded as '00'.

CLA coding	Meaning
'00'	No SM, channel 0, only command or last command of a command chain
'08'	As before, but with SM
'0C'	As before, but SM with header authentication
'1x'	Further commands of a command chain follow

Tab. 75: CLA-Coding

22.2 Status Bytes

Under normal conditions of use the status bytes always have the value '9000'. In the following commands other values are permissible in normal processing. The values are given in the description of the respective command.

- READ BINARY
- INTERNAL AUTHENTICATE
- VERIFY
- CHANGE REFERENCE DATA

- RESET RETRY COUNTER

While an application is developed other status bytes may occur when an erroneous command is sent to the chipcard. The coding of these errors shall be ISO conform, they should at least be described unambiguously in the manuals.

If the transmission protocol T=0 is used, with some commands the coding of the status bytes may be '6Cxx' (e.g. with READ BINARY, when Le = '00' was sent to the card). In this case the command must be repeated with the length given in SW2.

Annex A (normative)

DS Formats

References

H. Dobertin, A. Bosselaers, B. Preneel:
RIPEMD-160: A strengthened version of
RIPEMD, Fast Software Encryption, Cambridge
Workshop 1996, LNCS, Band 1039, S. 71-82,
Springer-Verlag 1996 (Final version under
<ftp.esat.kuleuven.ac.be/pub/COSIC/bosselaer/ripemd>)

NIST: FIPS Publication 186:
Digital Signature Standard (DSS), Mai 1994

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 Draft Version 4, Standard Spec-
ifications for Public Key Cryptography,
16. Juni 1998

ANSI X 9.62 Working Draft,
Public Key Cryptography for the Financial
Services Industry: The Elliptic Curve Digital
Signature Algorithm,
17. November 1997

ANSI X 9.63, Working Draft, Version 2.0,
Public Key Cryptography for the Financial
Services Industry: Elliptic Curve Key Agreement
and Key Transport Schemes,
5. Juli 1998

ISO/IEC 9797-1: 1998
Information technology – Security techniques –
Message authentication codes, Part 1: Mecha-
nism 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

Abbreviations

DSA = Digital Signature Algorithm
ELC = Elliptic Curve
RSA = Sig-Alg. von Rivest, Shamir, Adleman

1 Hash-Input-Formats

When hashing is performed partly inside and
partly outside the chipcard, 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 8 bytes counter (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
- nth 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 chipcard 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 chipcard.

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

Public keys of root certification authorities for different security services (signature function according to

SigG, ...) are stored in the file EFPK.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 must contain at least one public key of the top certification authority (RegTP) as a 'security anchor' (e.g PK from 1998 and maybe PK from 1999).

The coding of the respective data fields complies with those in a certificate according to ISO/IEC

9594-8 (ITU X.509). Below the syntax provided for up till now and a coding example are given.

Note: The structure described here for the public keys corresponds to the current technical situation at the RegTP. In case changes have to be made when SigI comes into force, they will be included in this chapter when the specification is revised.

PublicRootKeyInfo ::= SET OF PublicRootKey

```
PublicRootKey ::= SEQUENCE
{
    subject                Name
    serialNumber           CertificateSerialNumber OPTIONAL
    subjectPublicKeyInfo   SubjectPublicKeyInfo
}
```

Name ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET OF AttributeTypeAndValue

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

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType

CertificateSerialNumber ::= INTEGER

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

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

The ASN.1 type „Name“ is coded according to ITU-T X.520. The key owner (Subject) is unambiguously identified by this object. Additionally various RelativeDistinguishedName (RDN) can be used. The following RDN are mandatory (The values comply with the RegTP key for 1998):

1. Country (C):
attributeType: OID {2 5 4 6}
attributeValue: PRINTABLE STRING 'DE'
2. Common Name (CN) + Name Distinguisher

```

Name Distinguisher:
attributeType:  OID {0 2 262 1 10 7 20}
attributeValue:  PRINTABLE STRING '1'

Common Name (CN):
attributeType:  OID {2 5 4 3}
attributeValue:  TELETEX STRING 'ROOTCA-1 1:PN'

```

The object subjectPublicKey is defined as a BIT STRING and again contains, according to X.509, for the algorithm RSA the following type (DER coded):

```

SEQUENCE
{
    modulus      INTEGER
    exponent     INTEGER
}

```

Coding example:

```

31 xx          -- SET OF PublicRootKey
 30 xx          -- SEQUENCE
                -- Subject Name:
 30 xx          -- SEQUENCE OF (RDN)
 31 0B          -- SET OF (AttributeValueAsser.)
 30 09          -- SEQUENCE
    06 03 55 04 06 -- OID Country
    13 02 44 45   -- PRINTABLE STRING 'DE'
 31 24          -- SET OF (AttributeValueAsser.)
 30 0C          -- SEQUENCE
    06 07 02 82 06 01 0A 07 14 -- OID Name Distinguisher
    13 01 31      -- PRINTABLE STRING '1'
 30 14          -- SEQUENCE
    06 03 55 04 03 -- OID Common Name
    14 0D 52 4F 4F 54 43 41 -- TELETEX STRING
                        2D 31 20 31 3A 50 4E -- 'ROOTCA-1 1:PN'
 30 xx          -- SubjectPublicKeyInfo:
                -- SEQUENCE
                -- algorithm:
 30 xx          -- SEQUENCE
    06 xx xx xx xx ... -- OID Algorithm
    05 00          -- Null (Parameter)
                -- subjectPublicKey:
 03 xx          -- BIT STRING
 00 30 xx        -- SEQUENCE
                -- modulus:
                02 xx xx xx ... -- INTEGER
                -- exponent:
                02 xx xx xx ... -- INTEGER

```

5 Security Environments and Headerlists

Chipcards which function only with one security environment, because they support only one signature algorithm and one DSI format, use SE #0 implicitly. Chipcards which support more than one signature algorithm or more than one RSA.DSI formats, there can be described in the signature template of the SSD file which algorithm complies with which security environment. SE #1 is default.

SEs can be stored in the chipcard in a file e.g. in SE templates. In the control reference template for digital signature the format mechanism can be controlled with the help of a header list (see Fig. A.1).

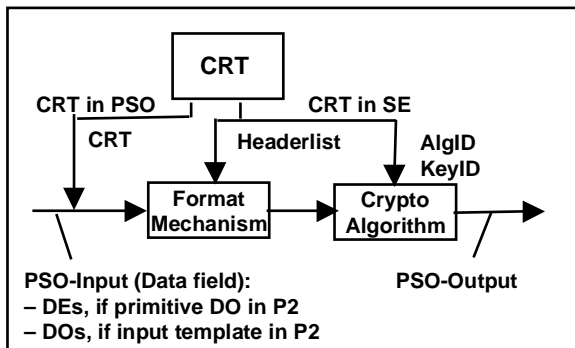


Fig. A.1: CRTs and their use (example)

The Figs. A.2 and A.3 show CRTs with their headerlist for the RSA.DSI-variants ISO/IEC 9796-2 and PKCS #1.

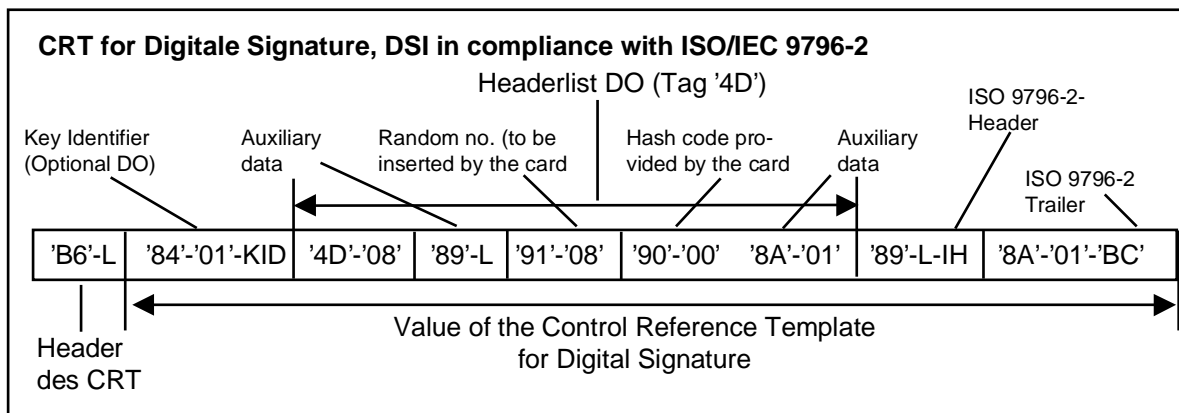


Fig. A.2: CRT for DSI according to ISO/IEC 9796-2

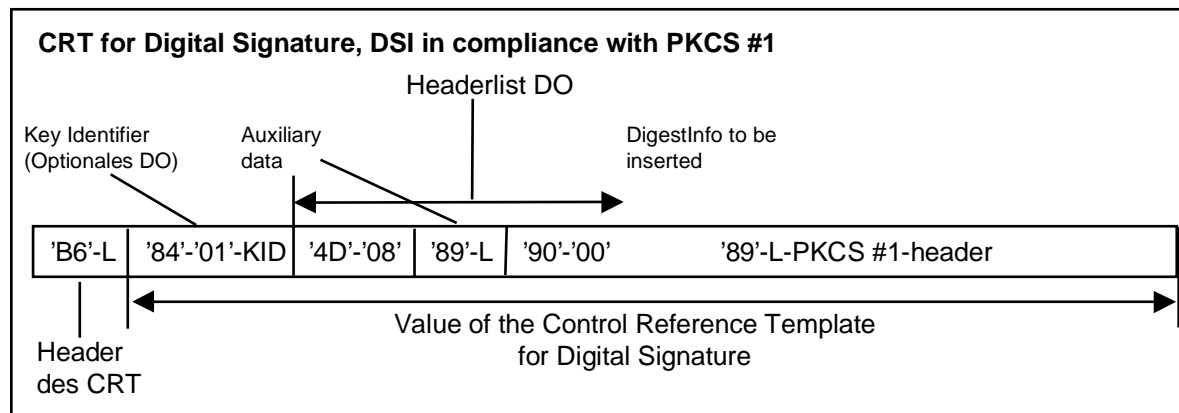


Fig. A.3: CRT for DSI according to PKCS #1 with transfer of the DigestInfo to the card

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 chipcard 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 chipcard 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.

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.2: Structure of the Certification Authority Reference (Authority Key Identifier)

The extension has the following structure:

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

Table B.3: Structure of the extension for key referencing

The Service Indicator (Key usage) has the value 0 = 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.4: 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) was issued.

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

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

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.

In the context of this specification the certificate holder must be authorised to read the display message. The security attributes of EF.DM can be used to assign the security conditions (SC) to the relevant allowed access modes (AM) (see Table C.3).

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 SigG) (6 B)	Role ID (1B)
-------------------------------	-----------------

Table B.6: Structure of Certificate Holder Authorisation for the SigG-Application

The table below shows CHA Role Identifiers relevant to the SigG-application.

CHA Role ID	Meaning	Relevant for C.CA.AUT	Relevant for C.ICC.AUT	Relevant for C.IFD.AUT
'00'	No access right to data	X	X	
'01'	CHA Role ID for proving the access right of an IFD (Read access to EF.DM)			X
'02'	CHA Role ID for proving the access right of a CA (e.g. read/write access to certificate files and EF.DM)			X

Table B.7: CHA Role ID coding for the SigG-Application

The use of CHA Role ID '02' will be described in the 'DIN Specification for the Personalisation of the SigG Application on Chipcards according to SigG/SigV'.

1.6 Object Identifier for Signature Algorithm of the Certificate Holder

The Object Identifiers are described in Annex E.

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 (z.B. 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, $y = g \exp(x) \bmod p$ (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

$$Y^2 = X^3 + aX + b \text{ in } F \text{ with } a, b \in F.$$

Let k be the smallest number with $p \leq 2^{8k}$. L.F is a number $\geq k$.

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

$$Y^2 + XY = X^3 + aX^2 + b \text{ with } a, b \in F.$$

Let k be the smallest number with $m \leq 8k$. $L.F$ is a number $\geq k$.

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, polynomial 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 k_1 of pentanomial polynome $X^m+X^{k_3}+X^{k_2}+X^{k_1}+1$ (length $L.m$)
- Tag '8A': value k_2 of pentanomial polynome $X^m+X^{k_3}+X^{k_2}+X^{k_1}+1$ (length $L.m$)
- Tag '8B': value k_3 of pentanomial polynome $X^m+X^{k_3}+X^{k_2}+X^{k_1}+1$ (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 defined (?eingetragen) 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 chipard 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.ICC or C.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 separately, 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

The following figures **show** the key management, when one step and two step procedure of certification verification applies. The treatment of other data elements, which are present in the certificate (i.e. CHA), is omitted in these figures.

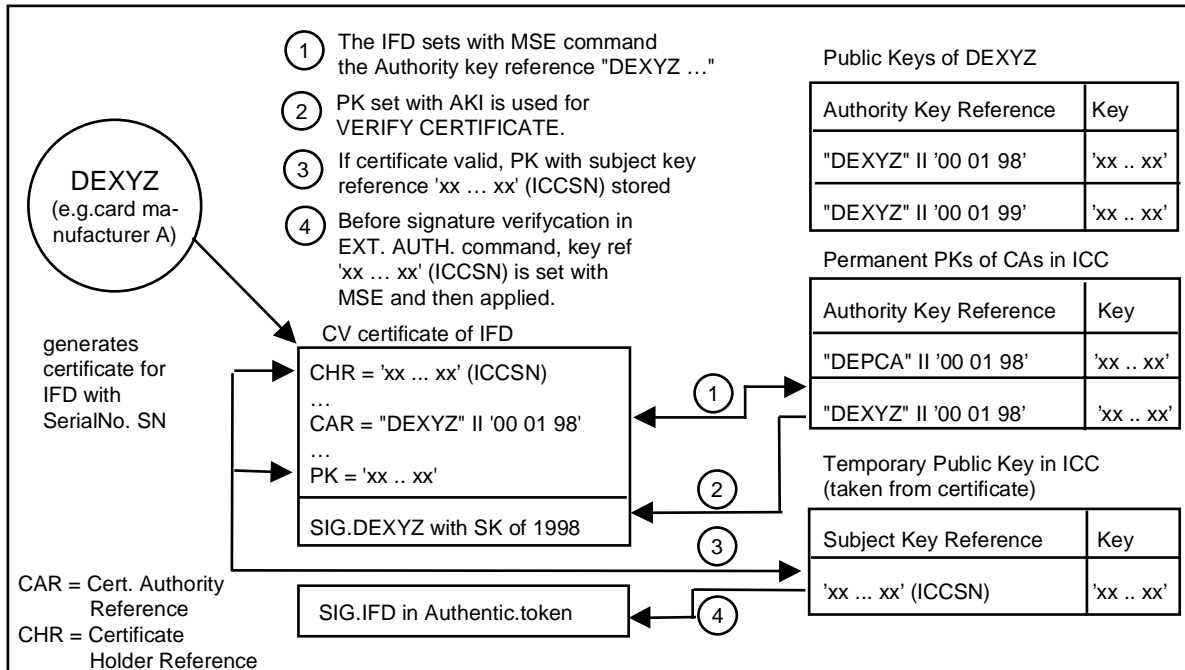


Fig. B.1: One Step Procedure (Precondition: PK from DEXYZ present in the chipcard)

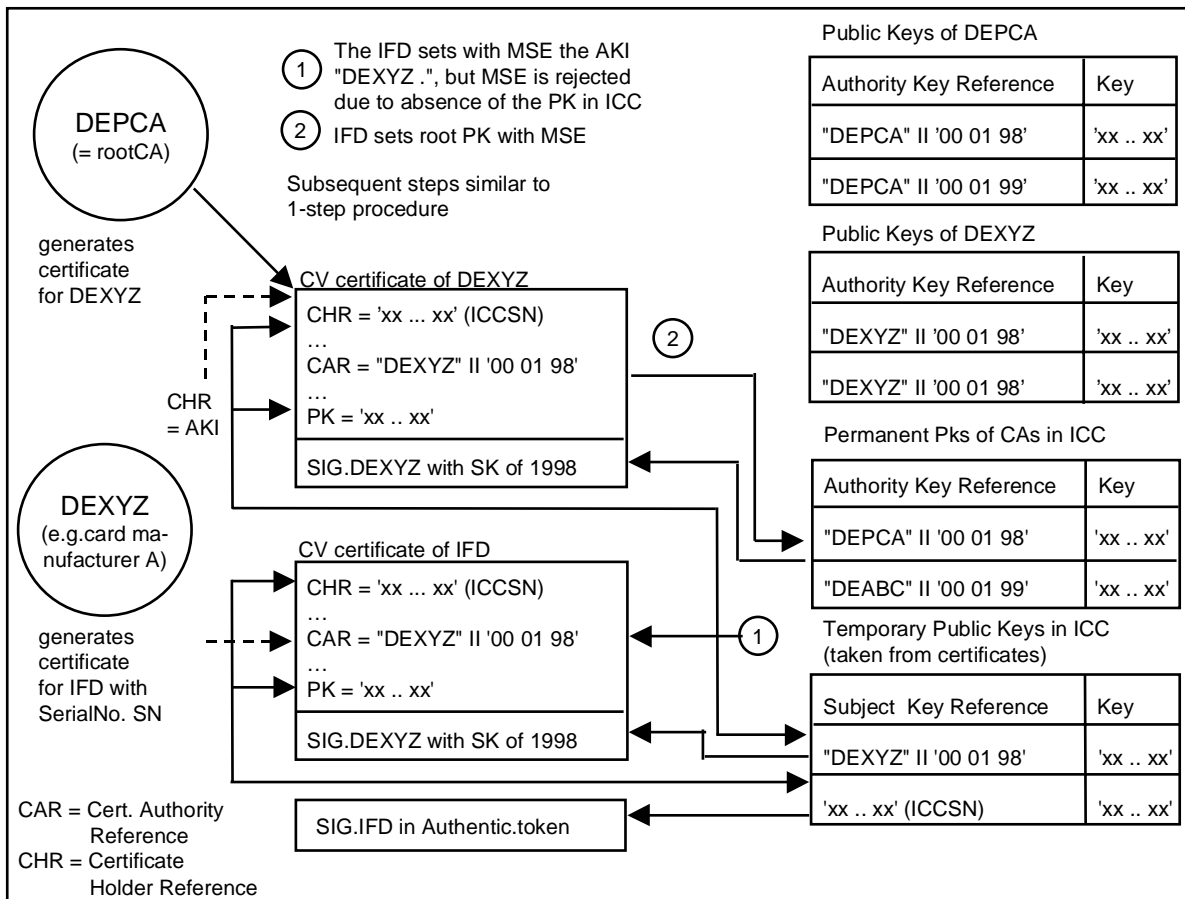


Fig. B.2: Two Step Procedure (customer terminal has certificate from DEXYZ and not from DEABC)

Annex C (normative)

SigG Files

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.

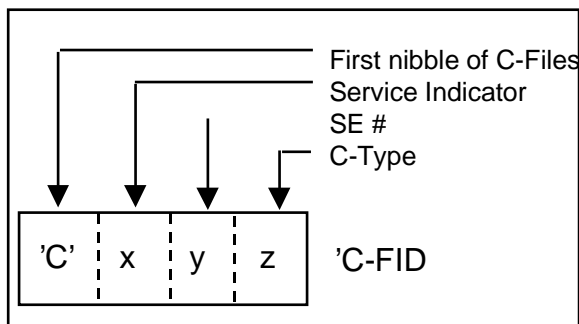


Fig. C.1: Generic structure of a Certificate FID

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

Value	Service
'0'	Digital Signature according to SigG
'1'	Entity Authentication
'2'	Key Encipherment
'3'	Data Encipherment
'4'	Key Agreement
	Other values RFU

Table C.1: Values for Service Indicator

The SE# (Value=0, if chipcard does not support SE number) 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

2 SigG-Files with file characteristics and access rights

The following table shows the SigG 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.

File	FID	Structure	Size (length of data)	Access Condition	Presence in case 1*	Presence in case 2*
EF.GDO (Global Data Objects)	'2F02'	transparent	64 Bytes	Read: always Update: never	mandatory	mandatory
EF.SSD (Security Service Descriptors)	'1F00'	transparent	256 bytes or card specific length	Read: always Update: never	optional	optional
EF.C.ICC.AUT (AUT Certificate of ICC)	'C100'	transparent	256 bytes or length of C.ICC.AUT	Read: always Update: never (or IFD authentication (RoleID = '02') and SM)	optional**)	mandatory
EF.C.CA.AUT (AUT Certificate of CA)	'C108'	transparent	256 bytes or length of C.CA.AUT	Read: always Update: never (or IFD authentication (RoleID = '02') and SM)	optional**)	mandatory
EF.DM (Display message)	'D000'	transparent	8 Bytes	Read: IFD authentication (RoleID = '01' or '02') in combination with SM or user auth. Update: user authentic.	not used	mandatory
EF.C.CH.DS (DS Certificate of Cardholder issued by CA of CH)	'C00x' (x = 0 - 7, see tab.C.2)	transparent	2k bytes or length of C.CH.DS	Read: user authentication Update: never (or IFD authentication (RoleID = '02') and SM)	mandatory***)	mandatory***)
EF.C.CA.DS (DS Certificate of CA issued by RCA)	'C00x' (x = 8 - E, see tab.C.2)	transparent	1k bytes or length of C.CA.DS	Read: always Update: never (or IFD authentication (RoleID = '02') and SM)	optional	optional
EF.PK.RCA.DS (Public Key(s) of RCA)	'B000'	transparent	512 bytes or length of stored PKs	Read: always Update: never (or IFD authentication (RoleID = '02') and SM)	mandatory (if no RCA self-certif. present)	mandatory
EF.PK.CA.DS (Public Key(s) of CAs)	'B001'	transparent	512 bytes or length of stored PKs	Read: always Update: never (or IFD authentication (RoleID = '02') and SM)	optional	optional
EF.PROT (Signature log)	'A000'	cyclic	20 Records, each 53 bytes	Read: user authentic. Update: user authentic.	optional	optional

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

- * = Case 1: Use only at a private or company terminal
Case 2: Use additionally at customer terminals
- ** = The certificates C.ICC.AUT and C.CA.AUT can also be used for personalisation
- *** = When, e.g. for memory reasons, the certificate related to the signature key is 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).

Annex D (normative)

Device Authentication, Session Key Agreement and Secure Messaging

1 Device Authentication with RSA

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


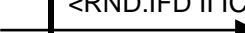

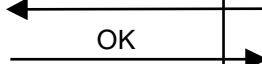
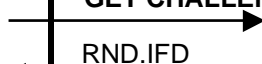
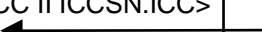

Customer Terminal (IFD)	User Card (ICC)	Actions
GET CHALLENGE 		Get RND
	INT AUTH <RND.IFD ICCSN.IFD>  E.PK.IFD(SIG.ICC('6A' PRND1 K1 h(PRND1 K1 RND.IFD ICCSN.IFD) 'BC')) *) 	Compute AUT-Token
EXT AUTH <E.PK.IFD(SIG.ICC ('6A' PRND1 K1 h(PRND1 K1 RND.IFD ICCSN.IFD) 'BC'))> *) 		Authenticate AUT-Token
	GET CHALLENGE 	Get RND
INT AUTH <RND.ICC ICCSN.ICC>  <E.PK.ICC(SIG.IFD('6A PRND2 K2 h(PRND2 K2 RND.ICC ICCSN.ICC) 'BC'))> **)		Compute AUT-Token
) ENC input = min (SIG, SIG) with SIG* = N.ICC - SIG and N.ICC 0 modulus of ICC **) as before, but N.IFD is used	EXT AUTH <E.PK.ICC(SIG.IFD('6A' PRND2 K2 h(PRND2 K2 RND.ICC ICCSN.ICC) 'BC'))> **) 	Authenticate AUT-Token and Compute Session Keys

Fig. D.1: Diagram of the authentication procedure with RSA according to ISO/IEC 9796-2 (description of exchange and 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

Chipcards, 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 chipcard as well as to the customer terminal. These system parameters are specified in the chipcard 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 DAS signature of the chipcard.

Preconditions:

- Length of p in bytes is $L.p \geq 128$ bytes
- SIG.ICC is a DSA-signature with length = 40 bytes
- h is the Hash-function of the DAS 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
- $KT.A1 = g^a \text{ mod } p$, $KT.A1$ has the length $L.p$
- $KT.B1 = g^b \text{ mod } p$, $KT.B1$ 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: 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.

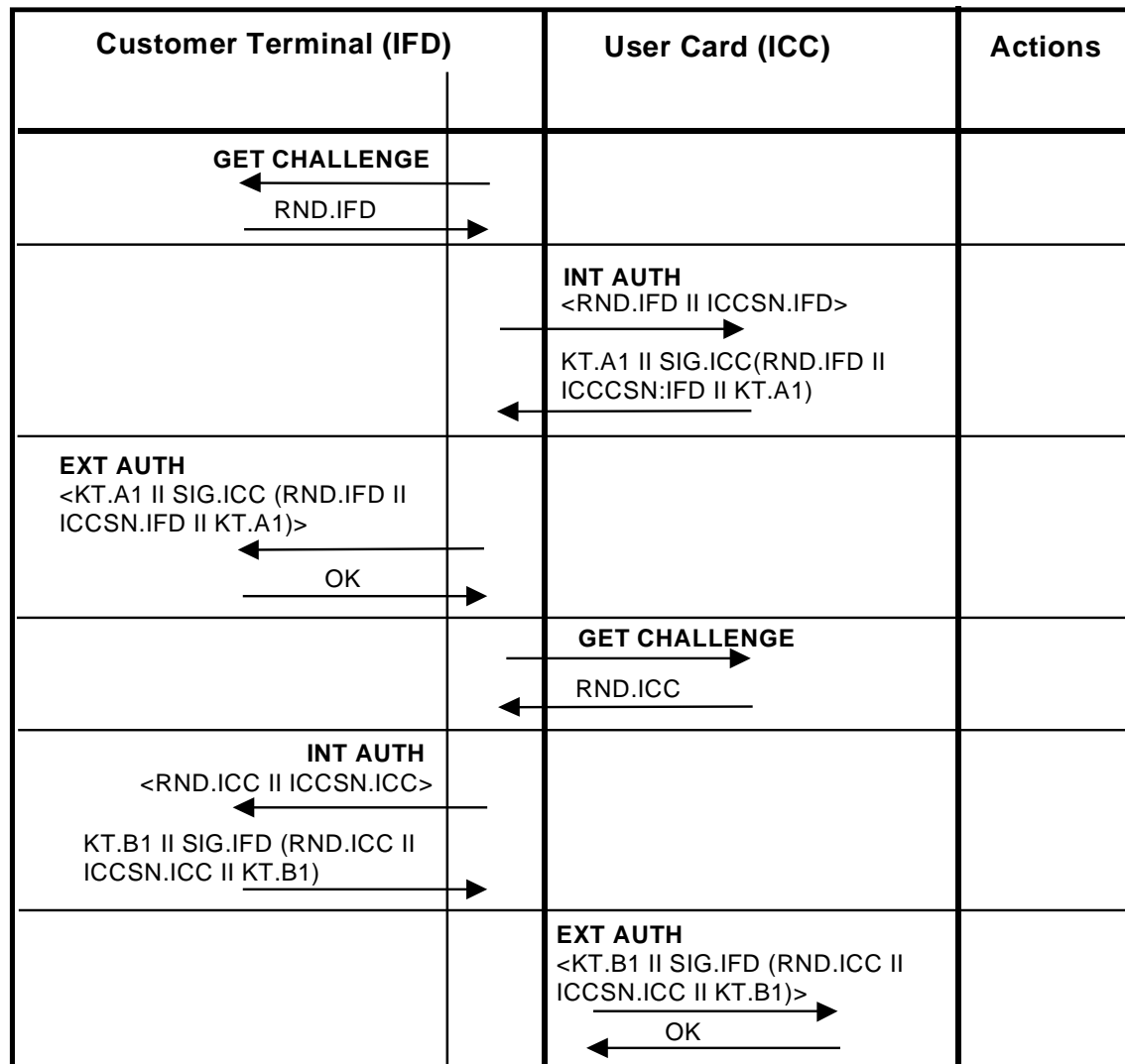


Fig. D.2: Diagram of the authentication procedure with ELC and DSA according to ISO/IEC 11770-3 (without description of exchange and verification of certificates and the use of MSE commands)

Chipcards, 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, 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 chipcard 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 corresponding parameters from the ELC signature of the chipcard.
- 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 $L.F \geq 20$. 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 $2 \cdot L.F \geq 40$.
- SIG.ICC is an ELC-signature of length $L.SIG.ICC \geq 40$ 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 $L.SIG.IFD \geq 40$ 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: 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.

$$K = K1 \oplus K2$$

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

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

The use of the keys K_a and K_b is depicted in figure 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 $KT.B1$.

$$K_{AB} = g^{ra*rb} \text{ mod } p$$

Let $K_{AB}(1)$ be the sequence of bytes, which represents K_{AB} . The length of $K_{AB}(1)$ is $L.p \geq 128$ 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 $2*L.F \geq 40$ 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

$$\text{HASH1} = h.SM(K_{AB}(1) \parallel c) \text{ with } c=1 \text{ and}$$

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.

In the following commands the data in the data field must be transferred as a cryptogram:

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

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

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

The starting value for the SSC is

$SSC = RND.ICC$ (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).

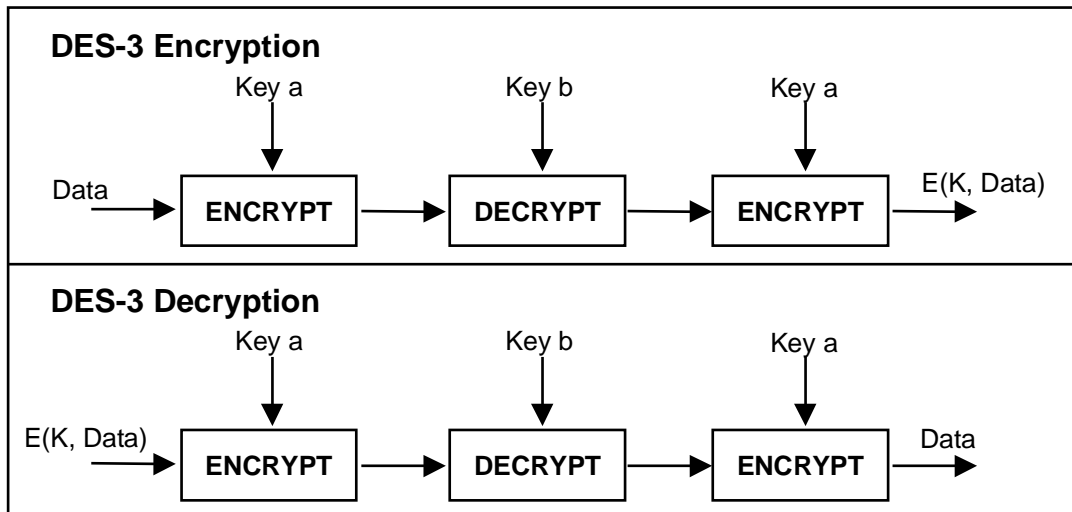


Fig. D.3: DES-3-Encryption/Decryption

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

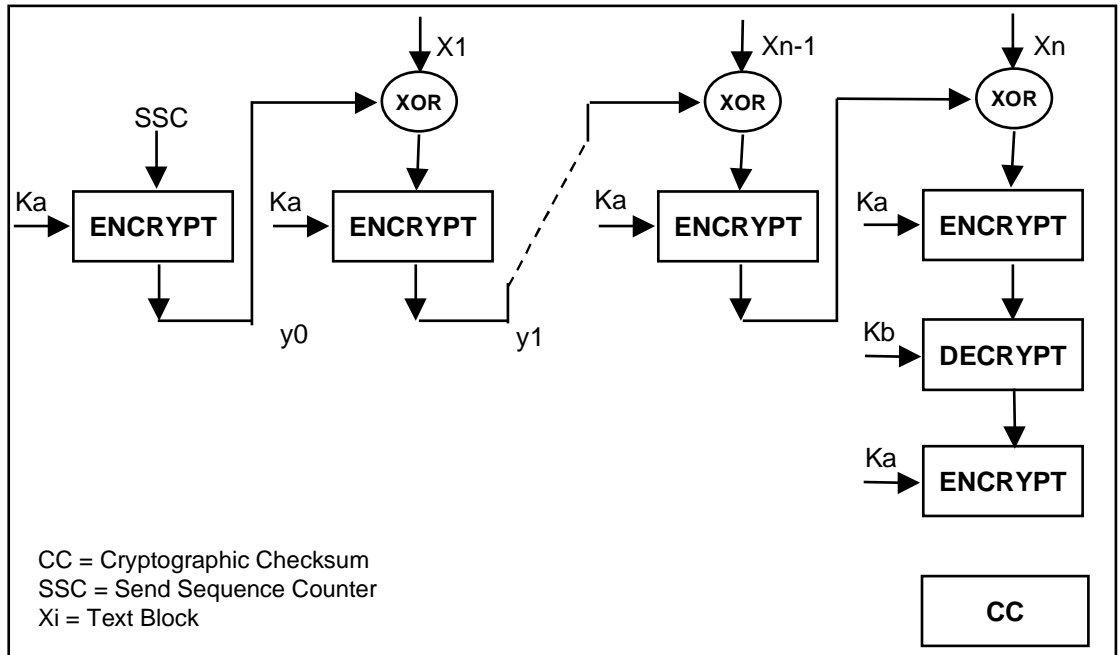


Fig. D.4: Calculation of the retail MAC

Annex E

(normative)

Object Identifiers

1 Construction scheme for Object Identifiers

An object identifier (OID) is a worldwide unique reference to an object, e.g. an algorithm. It consists of a sequence of numbers, each of which represents a node or leaf in the model tree of all objects.

Acc. to [ISO/IEC 8824: 1990] the highest level has three nodes:

- ITU-T/ CCITT (0)
- ISO (1)
- joint ISO-ITU-T/CCITT (2)

The ISO naming scheme is further divided into:

- standard (0)
- registration-authority (1)
- member body (2)
- identified-organization (3)

ISO standards can be referenced directly via the branch (0). In branch (1) object IDs are defined by a registration authority named by ISO. The branch member body (2) is reserved for the national standardisation bodies, i.e. (2) is followed by the numerical country code of the ISO member acc. to ISO 3166, e.g. 840 for the United States (ANSI) or 276 for Germany (DIN). This branch may be extended to companies and authorities with national context, e.g. rsadsi. In branch (3), "identified organisation" authorities, which are accredited with ISO, e.g. industrial bodies such as Open Implementors Workshop (OIW) und TeleTrusT (TTT), can be found. TeleTrusT has received the number '0036' as 'ISO identified organization' from the 'International Code Designator' (ICD), which is, why all object IDs defined by TeleTrusT start with the number { 1 3 36 } = ISO (1) - identified-organization (3) - teletrust (36).

After the authority which names the respective object is defined further branching occurs until the object is reached, which is to be identified. The respective authority can define the structure of its own sub-tree. TeleTrusT decided among others to use (3) for algorithms. The algorithms are subdivided into:

- Encryption Algorithm (1)
- Hash Algorithm (2)
- Signature Algorithm (3)
- Signature Scheme (4).

Figure E.1 displays an example of an OID tree.

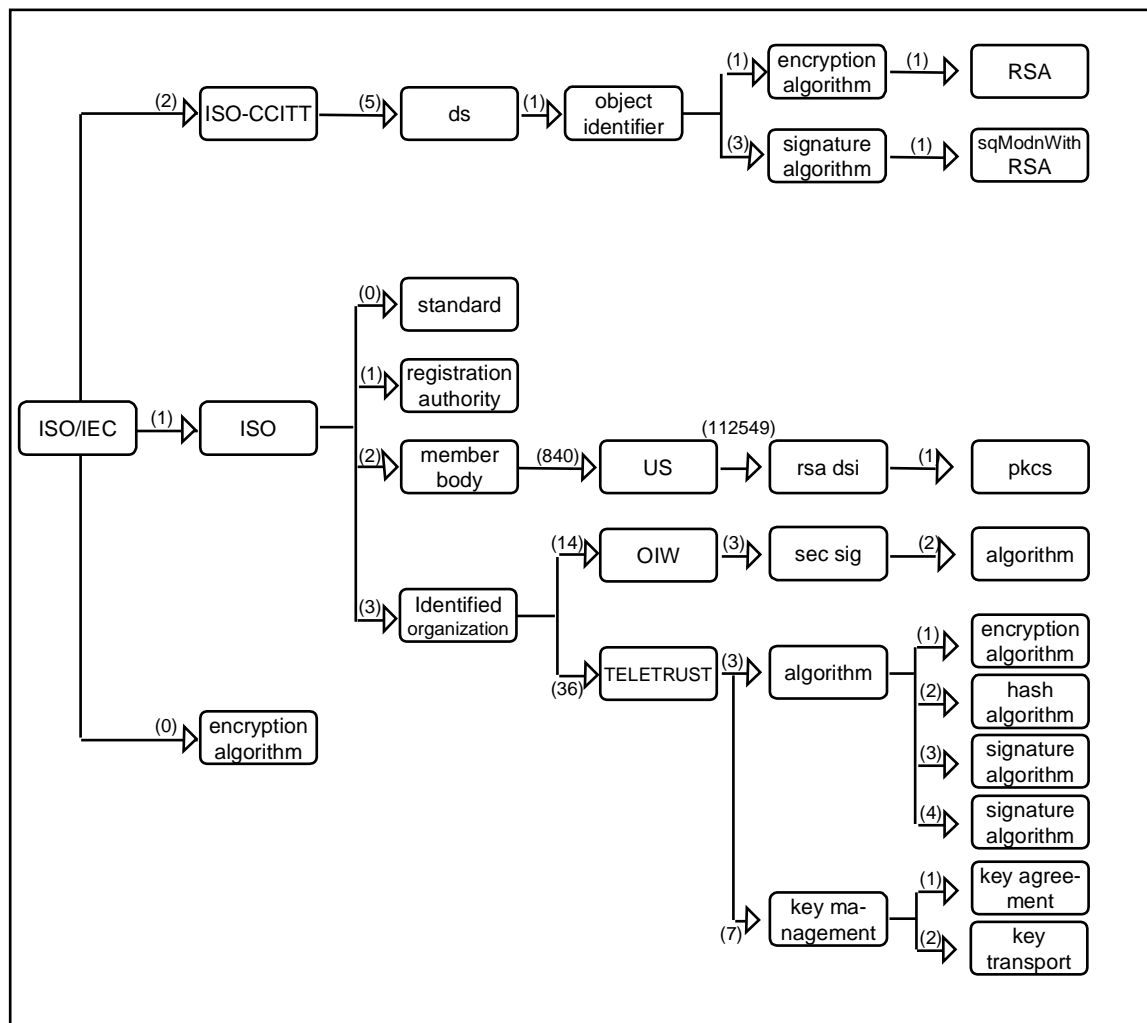


Fig. E.1: Structure of a tree of object identifiers (example)

Object identifiers in certificates have a form different from the sequence numbers described above. The OID acc. to ISO 8825 is calculated as follows:

- the first subidentifier is multiplied by 40 and coded as a 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)

5 Object Identifiers

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

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 padding	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_ISO9796-2WithrsaWithsha1 – 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-Hellman key agreement with DSA and SHA1			x
{1 3 36 7 1 2 2 1}**	TTT	kaDHWithecWithsha1 – Diffie-Hellman key agreement with elliptic curve and SHA1			x

Table E.1: 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 F (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 F.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	INT. AUTHENTICATE
'A2'	External authentication service (sym.)	GET CHALLENGE EXT. AUTHENTICATE
'A3'	External authentication service (asym.)	(MANAGE SE) PSO: VERIFY CERTIF. 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 F.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 encryption 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 content of this DO is application specific.
For the SigG application the following values are defined:
'00': after PIN presentation no limitation
'01'-'0F': the number indicates the amount of signatures possible with one PIN presentation
'10'-'FF': RFU
- 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 default value '00' is not used.
- 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.

3 Algorithmus-Identifizier

In the following table the meaning of AlgIDs for the SigG application is defined. The AlgID should be given in the SSD template (see DO with tag '81'), to inform the world outside the cards, which chipcard supports which algorithms and which DSI formats.

AlgID	Bedeutung
'0x'	No hash-function
'1x'	SHA-1
'2x'	RIPEMD-160
'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 F.2: AlgIDs for hash-functions and signature algorithms

Annex G (informative)

Use of the Security Service Descriptor

The SSD file is a transparent file, which contains a sequence of SSD templates with the description of those security services, which are supported by the chipcard. Thus the terminal can build correctly the command sequences to the card. To make the examples of SSD templates described more readable, the following notation was used:

- All digits and letters represent one hexadecimal digit (halfbyte)
- Tag and length are each followed by a minus
- The value part of a template is set in []
- || means concatenation
- Blanks are for layout, without meaning

In the SSD file only the coding of the DOs is stored. The following tables show coding examples of the SSD file or part of it.

Template	Bedeutung
A0-06- [80-04-00200081]	VERIFY for knowledge based user authentication
A0-09- [80-04-00200082 81-01-01]	VERIFY for biometrical user authentication; AlgID = '01' (= Fingertip Sensor *)
A0-06- [80-04-00240081]	CHANGE RD
A0-06- [80-04-002C0081]	RESET RC *)
A4-11- [80-04-002A9E9A 81-01-01 85-02-C000 86-02-C008]	PSO: COMPUTE DS; AlgID = '01' (= RSA with DSI acc. to ISO/IEC 9796-2); FIDs of the certificate files EF.C.CH.DS and EF.C.CA.DS **)
*) Template is missing, if the service is not supported	
**) DO FID is missing, if the certificate is not on the chipcard	

Table G.1: Complete example of an SSD file

Template	Meaning
A4-17- [80-04-002A90A0 80-04-002A9E9A 81-01-11 85-02-C000 86-02-C008]	PSO: HASH (last round)*); PSO: COMPUTE DS; AlgID = '11' (= SHA-1 and RSA with DSI acc. to ISO/IEC 9796-2); FIDs of the certificate files
*) in case chaining shall be used, then the CLA-Byte in the DO '80' shall be '01' instead of '00' Falls Chaining verwendet werden soll, dann ist im DO '80' als CLA-Byte '01' statt '00' anzugeben	

Table G.2: Example signature template with PSO: HASH and PSO: COMPUTE DS

Template	Meaning
A4-15- [80-04-00227301 80-04-002A9E9A 81-01-01 85-02-C010 86-02-C018]	MSE: RESTORE SE #1 (Default); PSO: COMPUTE DS; AlgID = '01' (= RSA with DSI acc. to ISO/IEC 9796-2); FIDs of the certificate files;
A4-15- [80-04-00227302 80-04-002A9E9A 81-01-02 85-02-C020 86-02-C018]	MSE: RESTORE SE #2; PSO: COMPUTE DS; AlgID = '02' (= RSA with DSI acc. to PKCS #1), FIDs of the certificate files (root-certificate as in SE #1);
A4-15- [80-04-00227303 80-04-002A9E9A 81-01-04 85-02-C030 86-02-C038]	MSE: RESTORE SE #3; PSO: COMPUTE DS; AlgID = '04' (= ELC, parameter in the DS certificate), FIDs of the certificate files;

Table G.3: Example signature templates for RSA (2 different DSI) and ELC

Template	meaning
A1-10- [80-04-00880081 85-02-C100]	INT AUTHENTICATE; used algorithm is given in the certificate either explicit or implicit; FIDs of the certificate file EF.C.ICC.AUT
A3-L- [80-04-002A00xx 80-02-0084 80-02-0082]	PSO: VERIFY CERTIFICATE, GET CHALLENGE, EXT AUTH Cmd;

Table G.4: Example with device authentication

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

Table H.1: ICM-Coding