ISO/IEC JTC 1/SC 17

Date: 2002-04-10

ISO/IEC FCD 7816-15

ISO/IEC JTC 1/SC 17/WG 4

Secretariat: AFNOR

# Information technology — Identification cards — Integrated circuit(s) cards with contacts — Part 15: Cryptographic information application

Technologies de l'information — Cartes d'identification — Cartes à circuit(s) intégré(s) à contacts — Partie 15: Application "Information cryptographique"

**Warning**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (40) Enquiry
Document language: E

C:\users\andy\My Documents\ISOs\ISO-IEC_FCD_7816-15.doc STD Version 2.0

# Contents

Page

# Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 7816 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 7816-15 was prepared by Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 17, Working Group WG4.

ISO/IEC 7816 consists of the following parts, under the general title Information technology — Identification cards — Integrated circuit(s) cards with contacts:

- Part 1: Physical characteristics
- Part 2: Dimensions and location of the contacts
- Part 3: Electronic signals and transmission protocols
- Part 4: Interindustry commands for interchange
- Part 5: Registration procedures for application providers
- Part 6: Interindustry data elements for interchange
- Part 7: Interindustry commands for Structured Card Query Language (SCQL)
- Part 8: Interindustry commands for a cryptographic toolbox
- Part 9: Interindustry commands for card and file management
- Part 10: Electronic signals and answer to reset for synchronous cards
- Part 11: Personal verification through biometric methods
- Part 12: USB electrical interface and operating procedures
- Part 13: Registration of integrated circuit manufacturers
- Part 15: Cryptographic information application

## Introduction

Integrated circuit cards with cryptographic functions can be used for secure identification of users of information systems as well as for other core security services such as non-repudiation with digital signatures and distribution of enciphering keys for confidentiality. The objective of this International Standard is to provide a framework for such services based on available international standards. A main goal has been to provide a solution that may be used in large-scale systems with several issuers of compatible cards, providing for international interchange. It is flexible enough to allow for many different environments, while still preserving the requirements for interoperability.

A number of data structures have been provided to manage private keys and key fragments, to support a public key certificate infrastructure and flexible management of user and entity authentication.

This part of ISO/IEC 7816 is based on PKCS #15 v1.1 (see the Bibliography). The relationship between these documents is as follows:

— A common core is identical in both documents;

— Those components of PKCS #15 which do not relate to IC cards have been removed;

— This part of ISO/IEC 7816 includes enhancements to meet specific IC card requirements.

# Information technology — Identification cards — Integrated circuit(s) cards with contacts — Part 15: Cryptographic information application

## 1 Scope

This part of ISO/IEC 7816 specifies an application in a card. This application contains information on cryptographic functionality. This part of ISO/IEC 7816 defines a common syntax and format for the cryptographic information and mechanisms to share this information whenever appropriate.

The objectives of this part of ISO/IEC 7816 are to:

— facilitate interoperability among components running on various platforms (platform neutral);

— enable applications in the outside world to take advantage of products and components from multiple manufacturers (vendor neutral);

— enable the use of advances in technology without rewriting application-level software (application neutral); and

— maintain consistency with existing, related standards while expanding upon them only where necessary and practical.

It supports the following capabilities:

— storage of multiple instances of cryptographic information in a card;

— use of the cryptographic information;

— retrieval of the cryptographic information, a key factor for this is the notion of ”Directory Files,” which provides a layer of indirection between objects on the card and the actual format of these objects;

— cross-referencing of the cryptographic information with DOs defined in ISO/IEC 7816 when appropriate;

— different authentication mechanisms; and

— multiple cryptographic algorithms (the suitability of these is outside the scope of this part of ISO/IEC 7816).

This International Standard does not cover the internal implementation within the card and/or the outside world. It shall not be mandatory for implementations complying with this International Standard to support all options described.

In case of discrepancies between ASN.1 definitions in the body of the text and the module in Annex A, Annex A takes precedence.

## 2 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this part of ISO/IEC 7816. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 7816 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 7812-1:2000, Information Technology – Identification Cards – Identification of Issuers – Part1: Numbering System

ISO/IEC 7816-4:1995, Information Technology – Identification Cards – Integrated Circuit(s) cards with contacts – Part 4: Interindustry commands for interchange

ISO/IEC 7816-5:1994, Information Technology – Identification Cards – Integrated Circuit(s) cards with contacts – Part 5: Numbering system and registration procedure for application identifiers

ISO/IEC 7816-6:1996, Information Technology – Identification Cards – Integrated Circuit(s) cards with contacts – Part 6: Interindustry data elements

ISO/IEC 7816-8:1999, Information Technology – Identification Cards – Integrated Circuit(s) cards with contacts – Part 8: Security related interindustry commands

ISO/IEC 7816-9:2000, Information Technology – Identification Cards – Integrated Circuit(s) cards with contacts – Part 9: Security attributes and additional interindustry commands

ISO/IEC 8583-2:1998, Financial transaction card originated messages – Interchange message specifications – Part 2: Application and registration procedures for Institution Identification Codes (IIC)

ISO/IEC 8824-1:1998 | ITU-T Recommendation X.680 (1997), Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation

ISO/IEC 8824-2:1998 | ITU-T Recommendation X.681 (1997), Information technology – Abstract Syntax Notation One (ASN.1): Information object specification

ISO/IEC 8824-3:1998 | ITU-T Recommendation X.682 (1997), Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification

ISO/IEC 8824-4:1998 | ITU-T Recommendation X.683 (1997), Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications

ISO/IEC 8825-1:1998 | ITU-T Recommendation X.690 (1997), Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

ISO/IEC 8825-2:1998 | ITU-T Recommendation X.691 (1997), Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)

ISO/IEC 9564-1:1996, Banking – Personal Identification Number management and security – Part 1: PIN protection principles and techniques

ISO/IEC 9594-2:1998 | ITU-T Recommendation X.501 (1997), Information technology – Open Systems Interconnection – The Directory: Models

ISO/IEC 9594-6:1998 | ITU-T Recommendation X.520 (1997), Information technology – Open Systems Interconnection – The Directory: Selected attribute types

ISO/IEC 9594-8:1998 | ITU-T Recommendation X.509 (1997), Information technology – Open Systems Interconnection – The Directory: Authentication framework

## 3 Terms and definitions

For the purposes of this part of ISO/IEC 7816, the following terms and definitions apply.

3.1
absolute path
path that starts with the file identifier '3F00'

3.2
application
data structure, data elements and program modules needed for a specific functionality to be satisfied

[ISO/IEC 7816-9:2000]

3.3
application identifier
data element that identifies an application in a card

NOTE – Adapted from ISO/IEC 7816-5:1994

3.4
application provider
entity that provides those components of an application on a card required to perform the respective application

[ISO/IEC 7816-5:1994]

**3.5**
**authentication information object**
**cryptographic information object that provides information about authentication related data, e.g. a password**

**3.6**
**authentication object directory file**
**elementary file containing authentication information objects**

**3.7**
**binary coded decimal**
**number representation where a number is expressed as a sequence of decimal digits and each decimal digit is encoded as a four bit binary number**

**3.8**
**cardholder**
**person to whom the card was issued**

**3.9**
**card issuer**
**organization or entity that issues cards**

**3.10**
**certificate directory file**
**elementary file containing certificate information objects**

**3.11**
**certificate information object**
**cryptographic information object that provides information about a certificate**

**3.12**
**command**
**message that initiates an action and solicits a response from the card**

**3.13**
**cryptographic information application**
**application in a card that contains information on cryptographic information objects, other security data elements and their intended use**

**3.14**
**cryptographic information object**
**structured information contained in a CIA, which describes a cryptographic data element, e.g. a public key or a certificate**

**3.15**
**data container information object**
**cryptographic information object that provides information about a data container, e.g. a file**

**3.16**
**data container object directory file**
**elementary file containing data container information objects**

**3.17**
**dedicated file**
**file containing file control information, and, optionally, memory available for allocation, and which may be the parent of elementary files and/or dedicated files**

**[ISO/IEC 7816-4:1995]**

**3.18**
**directory (DIR) file**
**elementary file containing a list of applications supported by the card, and optional related data elements**

NOTE – Adapted from ISO/IEC 7816-5:1994

**3.19**
**elementary file**
**set of data units or records that share the same file identifier, and which cannot be the parent of another file**

[ISO/IEC 7816-4:1995]

**3.20**
**file identifier**
**2-byte binary value used to address a file**

[ISO/IEC 7816-4:1995]

**3.21**
**function**
**process accomplished by one or more commands and resultant actions**

**3.22**
**master file**
**mandatory unique dedicated file representing the root of the file structure**

[ISO/IEC 7816-4:1995]

NOTE – The MF has file identifier '3F00'.

**3.23**
**message**
**string of bytes transmitted by the interface device to the card or vice-versa, excluding transmission-oriented characters**

[ISO/IEC 7816-4:1995]

**3.24**
**object directory file**
**mandatory elementary file containing information about other CIA directory files**

**3.25**
**password**
**data that may be required by the application to be presented to the card by its user**

[ ISO/IEC 7816-4:1995]

**3.26**
**path**
**concatenation of file identifiers without delimitation**

NOTE – Adapted from ISO/IEC 7816-4:1995.

**3.27**
**private key directory file**
**elementary file containing private key information objects**

**3.28**
**private key information object**
**cryptographic information object that provides information about a private key**

**3.29**
**provider**
**authority who has or who obtained the right to create a dedicated file in the card**

[ISO/IEC 7816-4:1995]

**3.30**
**public key directory file**
**elementary file containing public key information objects**

**3.31**
**public key information object**
**cryptographic information object that provides information about a public key**

**3.32**
**record**
**string of bytes that can be handled as a whole by the card and referenced by a record number or by a record identifier**

**[ISO/IEC 7816-4:1995]**

**3.33**
**relative path**
**path that starts with the file identifier of the current DF**

**3.34**
**secret key directory file**
**elementary file containing secret key information objects**

**3.35**
**secret key information object**
**cryptographic information object that provides information about a secret key**

**3.36**
**template**
**value field of a constructed data object, defined to give a logical grouping of data objects**

**[ISO/IEC 7816-6:1996]**

# 4 Symbols and abbreviated terms

## 4.1 Symbols

**DF.x** **Dedicated file x, where x is the acronym of the file**

**EF.x** **Elementary file x, where x is the acronym of the file**

**'0' – '9' and 'A' – 'F'** **Hexadecimal digits**

## 4.2 Abbreviated terms

**For the purposes of this part of ISO/IEC 7816, the following abbreviations apply:**

**AID** **application identifier**

**AOD** **authentication object directory**

**BCD** **binary-coded decimal**

**CD** **certificate directory**

**CDE** **cryptographic data element**

**CIA** **cryptographic information**

**CIO** **cryptographic information object**

**DCOD** **data container object directory**

**DF** **dedicated file**

**DH** **Diffie-Hellman**

**DSA** **Digital Signature Algorithm**

**EC** **Elliptic-Curve**

**EF** **elementary file**

IDO interindustry data object, as defined in ISO/IEC 7816-6

IFD interface device

KEA Key Exchange Algorithm

MF master file

OD object directory

PKCS public-key cryptography standard

PrKD private key directory

PuKD public key directory

RSA Rivest-Shamir-Adleman

SKD secret key directory

SPKI Simple Public Key Infrastructure

URL uniform resource locator

UTC coordinated universal time

WTLS Wireless Application Protocol transport layer security

# 5 Conventions

This part of ISO/IEC 7816 presents ASN.1 notation in the bold Helvetica typeface. When ASN.1 types and values are referenced in normal text, they are differentiated from normal text by presenting them in the bold Helvetica typeface. The names of commands, typically referenced when specifying information exchanges between cards and IFDs, are differentiated from normal text by displaying them in `Courier`.

If the items in a list are numbered (as opposed to using "–" or letters), then the items shall be considered steps in a procedure.

# 6 Cryptographic information objects

## 6.1 Introduction

This part of ISO/IEC 7816 provides:

— descriptions of objects describing cryptographic information contained in the card;

— descriptions of the intended use of this information;

— ways to retrieve this information (when appropriate);

— an abstract syntax for the information which provides the basis for encodings; and

— an object model for the information.

The information, which also may include access control information, is described in the form of CIOs.

## 6.2 CIO classes

This part of ISO/IEC 7816 defines four classes of CIOs:

— cryptographic key information objects;

— certificate information objects;

— data container information objects; and

— authentication information objects.

The logical structure of these CIOs is shown in Figure 1. The object class of key information objects has three subclasses: private key, secret key, and public key information objects. CIOs inherit attributes from higher-level classes, and may be instantiated on cards.

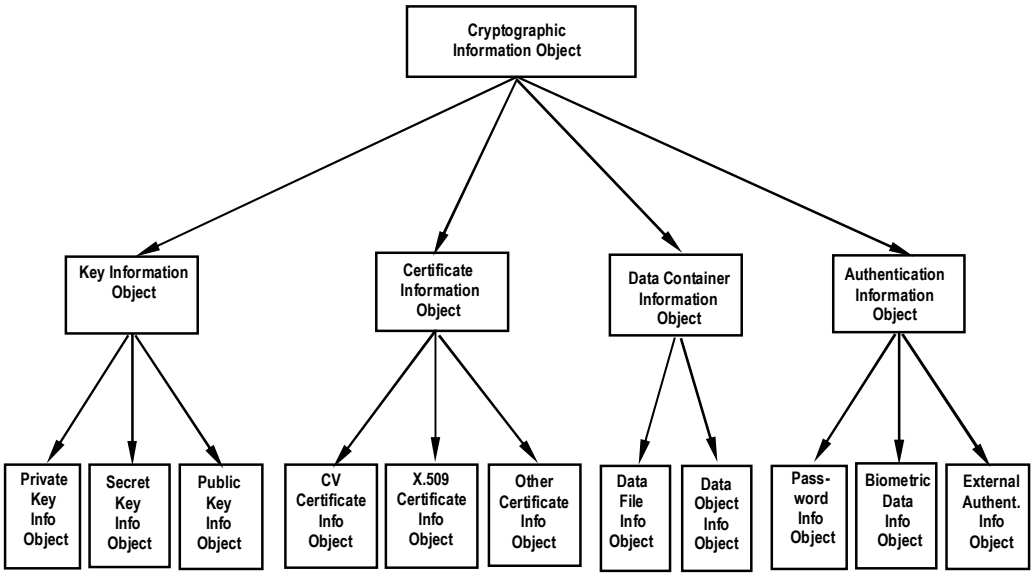



**Figure 1 – CIO class hierarchy**

### 6.3 Attributes

All CIOs have a number of attributes. Type specific attributes are always present. Group specific attributes and attributes common to all CIOs may be inherited as shown in Figure 2. Attributes are defined in clause 8.

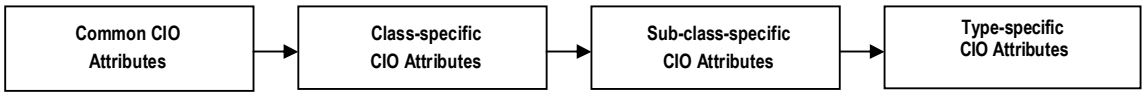



**Figure 2 – Attribute inheritance concept**

### 6.4 Access restrictions

CDEs can be private, meaning that they are protected against unauthorized access, or public. Access (read, write, etc.) to private CDEs is described by Authentication Information Objects (which also includes Authentication Procedures). Conditional access (from a cardholder's perspective) is achieved with knowledge-based user information, biometric user information, or cryptographic means. Public CDEs are not protected from read-access.

## 7 CIO files

### 7.1 Overview

A CIO is contained in an elementary file, and refers in general to a CDE; a CIO may in some cases contain the CDE directly. A dedicated file (DF.CIA) contains CIO elementary files. Certain CIO files may be present under other dedicated files, in which case they are referenced to from the DF.CIA.

### 7.2 IC card requirements

Cards shall comply with the appropriate parts of ISO/IEC 7816, when using:

— hierarchic logical file systems;

— direct or indirect application selection;

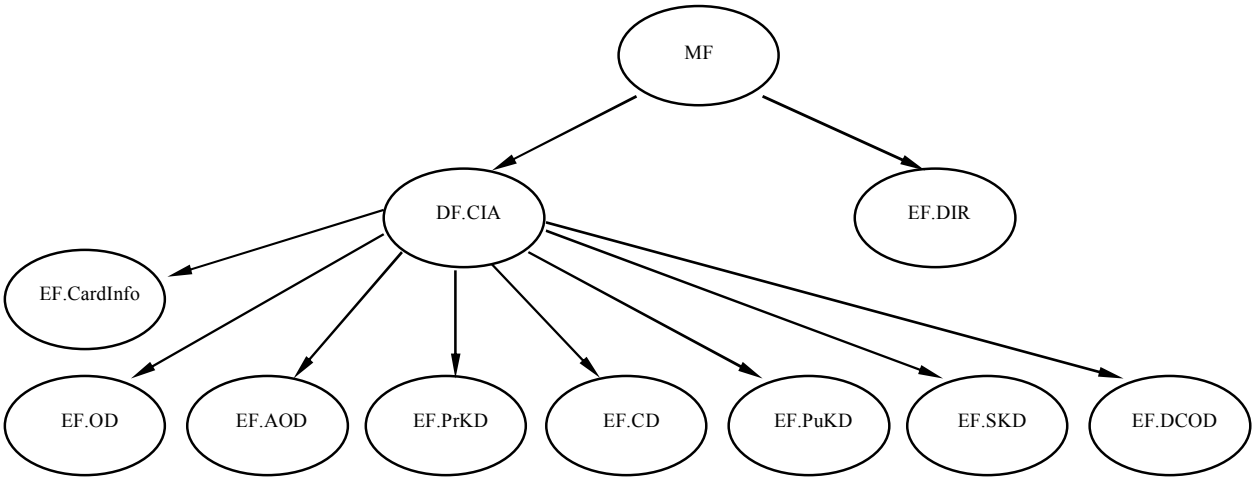
— access control mechanisms;

— read operations; and

— cryptographic operations.

### 7.3 Card file structure

A typical card supporting this part of ISO/IEC 7816 will have the following layout:




NOTE – For the purpose of this part of ISO/IEC 7816, EF.DIR is only needed on cards which do not support direct application selection as defined in ISO/IEC 7816-5:1994 or when multiple CIAs reside on a single card.

Figure 3 – Example contents of DF.CIA

Other possible topologies are discussed in Annex C. The contents and purpose of each file and directory is described below.

## 7.4 EF.DIR

This file under the MF (file identifier: '2F00') shall, if present, contain one or several application templates as defined in ISO/IEC 7816-5:1994. The application template (tag '61') for a CIA shall at least contain the following IDOs:

— Application Identifier (tag '4F'), value defined in this part of ISO/IEC 7816

— Path (tag '51'), value supplied by application issuer

Other IDOs from ISO/IEC 7816-5:1994 may, at the application issuer's discretion, be present as well. In particular, it is recommended that application issuers include both the "Discretionary data objects" data object (tag '73') and the "Application label" data object (tag '50'). The application label shall contain an UTF-8 encoded label for the application, chosen by the card issuer. The "Discretionary data objects" data object shall, if present, contain a DER-encoded (ISO/IEC 8825-1:1998) value of the ASN.1 type CIODDO:

```
CIODDO ::= SEQUENCE {
    providerId          OBJECT IDENTIFIER OPTIONAL,
    odfPath             Path OPTIONAL,
    cardInfoPath  [0] Path OPTIONAL,
    aid                 [APPLICATION 15] OCTET STRING (SIZE(1..16)),
                        (CONSTRAINED BY {-- Must be an AID in accordance with ISO/IEC 7816-5:1994--}) OPTIONAL,
    ... -- For future extensions
    } -- Context tag 1 is historical and shall not be used
```

NOTE: PKCS #15 uses this tag.

The providerId field, if present, shall contain an object identifier uniquely identifying the CIA provider. The odfPath and CardInfoPath fields shall, if present, contain paths to elementary files EF.OD and EF.CardInfo respectively. This provides a way for issuers to use non-standard file identifiers for these files without sacrificing interoperability. It also provides card issuers with the opportunity to share CardInfo files between CIAs, when several CIAs reside on one card. The aid field shall, if present, indicate the application to which this CIA applies.

The use of a DIR file will simplify application selection when several CIAs reside on one card. Its use is described in ISO/IEC 7816-5:1994.

## 7.5 Contents of DF.CIA

### 7.5.1 Overview

Table 1 lists elementary (mandatory and optional) files in the DF.CIA, along with their reserved file identifiers. File types (linear record or transparent) are indicated in the last column.

Table 1 – Elementary files in DF.CIA

| File | Mandatory | (Default) File Identifier | Short EF identifier | File type |
|---|---|---|---|---|
| CardInfo | X | '5032' | '12' | Transparent |
| OD | X | '5031' | '11' | Linear record or transparent |
| AODs | | | | Linear record or transparent |
| PrKDs | | | | Linear record or transparent |
| PuKDs | | | | Linear record or transparent |
| SKDs | | | | Linear record or transparent |
| CDs | | | | Linear record or transparent |
| DCODs | | | | Linear record or transparent |
| - | | '5033' | | Reserved for historical reasons |

### 7.5.2 EF.OD

The Object Directory file (EF.OD) is an elementary file, which may contain references to other CIO EFs. Figure 4 shows the relationship between EF.OD and other CIO EFs (for reasons of simplicity, only one referenced file of each type is shown). The ASN.1 syntax for the contents of EF.OD is described in clause 8.3.

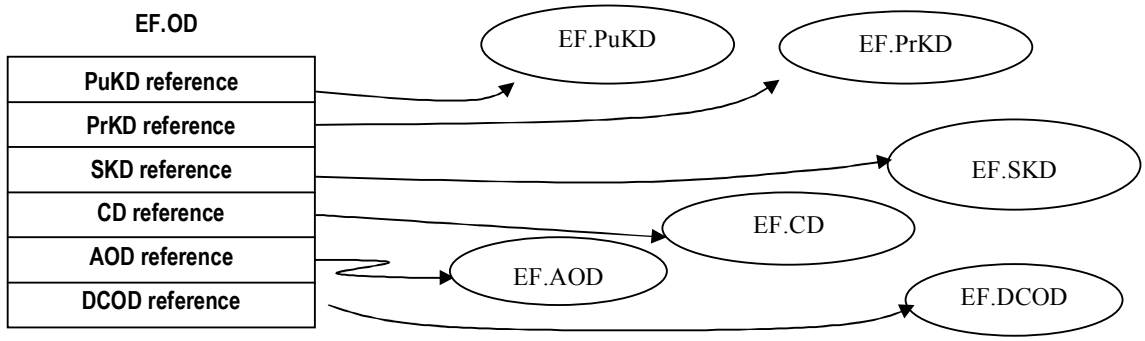



Figure 4 – Indirect retrieval of CIOs using EF.OD

### 7.5.3 CIO Directory files

Each CIO directory file contains CIOs of a certain kind:

- private key directory files contains private key information objects;
- public key directory files contains public key information objects;
- secret key directory files contains secret key information objects;
- certificate directory files contains certificate information objects;
- data container object directory files contains data container information objects; and
- authentication object directory files contains authentication information objects.

Multiple CIO directory files of the same kind may be present in a DF.CIA.

The object directory file EF.OD is unique and contains references to CIO directory files.

NOTE 1 – If a CIO directory file of a certain kind exists in a DF.CIA, it will usually not be empty.

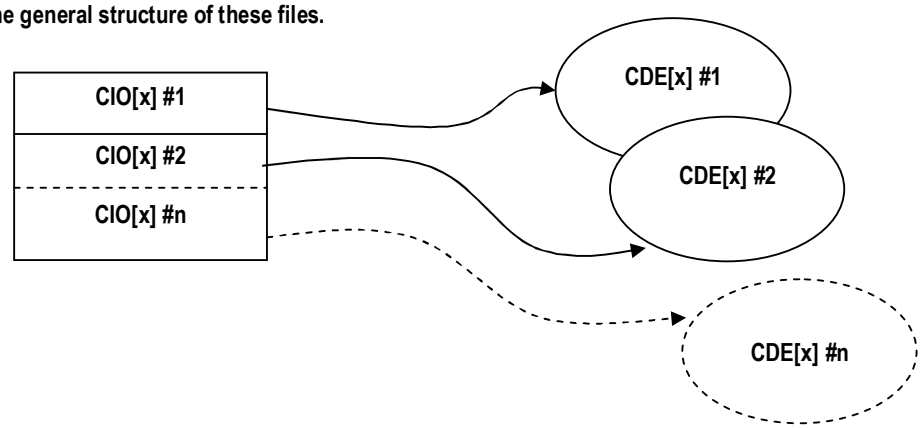
NOTE 2 – CIO directory files may also be found in other DFs in the card.

NOTE 3 – CIOs may be stored directly in an EF.OD (without any indirection), or in CIO directory files. CDEs may likewise be stored directly in CIOs or be referenced by them.

NOTE 4 – The use of indirection simplifies personalization, allows for more flexible access rules, and is recommended.

When CIOs reference CDEs that are logically linked (e.g. a private key CIO and a corresponding public key CIO) the CDEs shall have the same CIO identifier.

Figure 5 describes the general structure of these files.




NOTE – In the figure, the letter x stands for the kind of the information object which holds the information

Figure 5 – Indirect retrieval of CDEs using CIOs

### 7.5.4 The CardInfo EF

The CardInfo EF shall contain information about the card and its capabilities, pertaining to the use of CIOs. The following information shall always be present:

- version number; and
- card characteristics.

The following information may be found:

- card serial number;
- manufacturer identification;
- card label;
- allocated security environments;
- file structures;
- supported algorithms;
- issuer identification;
- holder authentication; and
- time of last update.

### 7.5.5 DF.CIA selection

The AID of a DF.CIA consists of three fields:

- the standard identifier E8 28 BD 08 0F (mandatory);
- an optional, 1-byte index in the range '00' to '7F'; and
- an optional, proprietary application identifier extension (PIX).

The index must be present whenever a PIX is present. The length of the AID must not exceed 16 bytes. The format of the AID is therefore (see further ISO/IEC 7816-5:2002):

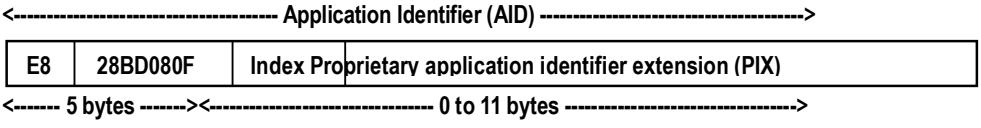



Figure 6 – AID format

DF.CIA may be selected using its AID by cards supporting direct application selection.

NOTE – For historical reasons, DF.CIA may be selected using the AID: A0 00 00 00 63 50 4B 43 53 2D 31 35.

If direct application selection is not possible, an EF.DIR file with contents as specified in clause 7.4 shall be used.

When several DF.CIAs reside on one card, they may be distinguished by information in the application template in EF.DIR. It is recommended that the application label (tag '50') also be present to simplify the man-machine interface (e.g. vendor name in short form).

# 8 Information syntax in ASN.1

## 8.1 Guidelines and encoding conventions

This part of ISO/IEC 7816 uses ASN.1 to describe CIOs. When stored in a card, DER-encoding of CIO values are assumed. Annex A contains a complete specification in ASN.1 of all CIOs; the text of this clause is explanatory only.

The contents of a CIO directory file is the concatenation of 0, 1, or more DER-encoded values of the same type, see e.g. Annex D.

## 8.2 Basic ASN.1 defined types

### 8.2.1 Identifier

```
Identifier ::= OCTET STRING (SIZE (0..cti-ub-identifier))
```

The Identifier type is used as a CIO identifier. For cross-reference purposes, two or more CIOs may have the same Identifier value. One example of this is a private key and one or more corresponding certificates.

### 8.2.2 Reference

```
Reference ::= INTEGER (0..cti-ub-reference)
```

This type is used for generic reference purposes.

### 8.2.3 Label

```
Label ::= UTF8String (SIZE(0..cti-ub-label))
```

This type is used for all labels (i.e. user assigned object names).

### 8.2.4 CredentialIdentifier

```
CredentialIdentifier {KEY-IDENTIFIER : IdentifierSet} ::= SEQUENCE {
        idType  KEY-IDENTIFIER.&id ({IdentifierSet}),
        idValue KEY-IDENTIFIER.&Value ({IdentifierSet}{@idType})
        }

KeyIdentifiers KEY-IDENTIFIER ::= {
        issuerAndSerialNumber           |
        issuerAndSerialNumberHash       |
        subjectKeyId                            |
        subjectKeyHash                  |
        issuerKeyHash                           |
        issuerNameHash                  |
        subjectNameHash,
        ...
        }

KEY-IDENTIFIER ::= CLASS {
        &id  INTEGER UNIQUE,
        &Value
        } WITH SYNTAX {
        SYNTAX &Value IDENTIFIED BY &id
        }
```

The CredentialIdentifier type is used to identify a particular key or certificate. There are currently seven members in the set of identifiers for private keys and certificates, KeyIdentifiers:

— issuerAndSerialNumber: The value of this type shall be a sequence of the issuer's distinguished name and the serial number of a certificate which contains the public key associated with the private key.

— issuerAndSerialNumberHash: As for issuerAndSerialNumber, but the value is an OCTET STRING which contains a SHA-1 hash value of this information in order to preserve space.

— subjectKeyId: The value of this type shall be an OCTET STRING with the same value as the subjectKeyIdentifier certificate extension in a ISO/IEC 9594-8:1998 certificate which contains the public key associated with the private key. This identifier can be used for certificate chain traversals.

— subjectPublicKeyHash: An OCTET STRING which contains the SHA-1 hash of the public key associated with the private key.

— issuerKeyHash: An OCTET STRING which contains the SHA-1 hash of the public key used to sign the requested certificate.

— issuerNameHash: A SHA-1 hash of the issuer's name as it appears in the certificate.

NOTE – This identifier may, in conjunction with the subjectNameHash identifier also be used for certificate chain construction.

— subjectNameHash: A SHA-1 hash of the subject's name as it appears in the certificate.

### 8.2.5 ReferencedValue and Path

```
ReferencedValue ::= CHOICE {
        path        Path,
        url         URL
        } -- The syntax of the object is determined by the context

URL ::= CHOICE {
        url         CHOICE {printable PrintableString, ia5 IA5String},
        urlWithDigest [3] SEQUENCE {
            url             IA5String,
            digest  DigestInfoWithDefault
            }
        }

Path ::= SEQUENCE {
        efidOrPath OCTET STRING,
        index           INTEGER (0..cti-ub-index) OPTIONAL,
        length          [0] INTEGER (0..cti-ub-index) OPTIONAL
        }( WITH COMPONENTS {..., index PRESENT, length PRESENT}|
           WITH COMPONENTS {..., index ABSENT, length ABSENT})
```

A ReferencedValue is a reference to a CIO value of some kind. This can either be some external reference (captured by the url choice) or a reference to a file on the card (the path identifier). The syntax of the value is determined by the context.

In the path case, identifiers index and length may specify a specific location within the file. If the file is a linear record file, index, when present, shall specify the record number (in the ISO/IEC 7816-4:1995 definition) and length can be set to 0 (if the card's operating system allows an $L_e$ parameter equal to '00' in a 'READ RECORD' command). Lengths of fixed records may be found in the CardInfo file as well (see clause 8.10). If the file is a transparent file, index, when present, shall specify an offset within the file, and length – the length of the segment (index would then become parameter $P_1$ and/or $P_2$ and length – the parameter $L_e$ in a 'READ BINARY' command). By using index and length, several objects may be stored within the same transparent file.

NOTE – From the above follows that a length of 0 indicates that the file pointed to by efidOrPath is a linear record file.

When efidOrPath is:

— empty, no file is referenced by it;

— one byte long, it references a short EF identifier in the most significant 5 bits (bits b3, b2 and b1 shall be set to 0);

— two bytes long, it references a file by its file identifier;

— longer than two bytes, it references a file either by an absolute or relative path (i.e. concatenation of file identifiers);

— longer than two bytes and consists of an odd number of bytes, it references a qualified path (see ISO/IEC 7816-5:2002).

NOTE – A short EF identifier should only be used if there is a unique related DF to which the referred file belongs.

In the url case, the URL may either be a simple URL or a URL in combination with a cryptographic hash of the object stored at the given location. Assuming that the CIO card is integrity-protected, the digest will protect the externally protected object as well.

NOTE – The URL syntax is defined in IETF RFC 2396 (see the Bibliography).

### 8.2.6 ObjectValue

```
ObjectValue { Type } ::= CHOICE {
        indirect    ReferencedValue,
        direct      [0] Type,
        ... -- For future extensions
        }
```

An object value of type ObjectValue type shall, unless otherwise mentioned, be stored by indirect reference (i.e. by pointing to another location where the actual value resides).

### 8.2.7 PathOrObjects

```
PathOrObjects {ObjectType} ::= CHOICE {
        path        Path,
        objects     [0] SEQUENCE OF ObjectType,
        ... -- For future extensions
        }
```

The PathOrObjects type is used to reference sequences of objects residing either within the OD or in another file. If the path alternative is used the referenced file shall contain the concatenation of 0, 1 or more DER-encoded values of the given type. Any number of 'FF' octets may occur before, between or after the values without any meaning (i.e. as padding for unused space or deleted values). The path alternative is strongly recommended (see Note 4 in clause 7.5.3).

### 8.2.8 CommonObjectAttributes

NOTE – This type is a container for attributes common to all CIOs.

```
CommonObjectAttributes ::= SEQUENCE {
        label               Label OPTIONAL,
        flags               CommonObjectFlags OPTIONAL,
        authId              Identifier OPTIONAL,
        userConsent         INTEGER (1..cti-ub-userConsent) OPTIONAL,
        accessControlRules      SEQUENCE SIZE (1..MAX) OF AccessControlRule OPTIONAL,
        ...
        } (CONSTRAINED BY {-- authId should be present if flags.private is set.
        -- It shall equal an authID in one authentication object in the AOD -- })

CommonObjectFlags ::= BIT STRING {
        private             (0),
        modifiable  (1),
        internal            (2)
        } -- Bit (2) is present for historical reasons and shall not be used

AccessControlRule ::= SEQUENCE {
        accessMode              AccessMode,
        securityCondition  SecurityCondition,
        ... -- For future extensions
        }

AccessMode ::= BIT STRING {
        read        (0),
        update      (1),
        execute     (2),
        delete      (3)
        }

SecurityCondition ::= CHOICE {
        always                  NULL,
        authId                  Identifier,
        authReference           AuthReference,
        not                     [0] SecurityCondition,
        and             [1] SEQUENCE SIZE (2..cti-ub-securityConditions) OF SecurityCondition,
        or                      [2] SEQUENCE SIZE (2..cti-ub-securityConditions) OF SecurityCondition,
        ... -- For future extensions
        }
```

```
AuthReference ::= SEQUENCE {
        authMethod      AuthMethod,
        seldentifier Reference OPTIONAL
        }

AuthMethod ::= BIT STRING {secureMessaging(0), extAuthentication(1), userAuthentication(2)}
```

The label is purely for display purposes (man-machine interface), for example when a user have several certificates for one key pair (e.g. "Bank certificate", "E-mail certificate").

The flags field indicates whether the particular object is private or not, and whether it is of type read-only or not. A private object may only be accessed after proper authentication (e.g. password verification). If an object is marked as modifiable, it should be possible to update the value of the object. If an object is both private and modifiable, updating is only allowed after successful authentication, however.

The authId field gives, in the case of a private object, a cross-reference back to the authentication object used to protect this object (For a description of authentication objects, see clause 8.9).

The userConsent field gives, in the case of a private object (or an object for which access conditions has been specified), the number of times an application may access the object without explicit consent from the user (e.g. a value of 3 indicates that a new authentication will be required before the first, the 4th, the 7th, etc. access). The card may enforce this value, e.g. through the use of "counter objects" (see ISO/IEC 7816-8:1999). A value of 1 means that a new authentication is required before each access.

The accessControlRules field gives an alternative, and more fine-grained, way to inform a host-side application about security conditions for various methods of accessing the object in question. Any Boolean expression in available authentication methods is allowed. If a certain access mode is not allowed, there shall be no access control rule for it (i.e. it is implicit). If this field is not present, access control rules will have to be deduced by other means. The authReference option allows for a closer coupling with ISO/IEC 7816-8:1999 and ISO/IEC 7816-9:2000, through the reference to Security Environments and identification of the class of authentication method (authMethod).

NOTE 1-When the accessControlRules field and the authID field both are present, information in the accessControlRule field takes precedence. This can occur for backwards-compatibility reasons.

NOTE 2 - Since properties related to access control can be deduced e.g. by studying EFs FCI, such information is optional and not necessary when these circumstances applies (see also ISO/IEC 7816-9:2000).

NOTE 3 – The access control information represented in these structures reflects access control rules in the card, but is not necessarily used as such by the card.

### 8.2.9 CommonKeyAttributes

```
CommonKeyAttributes ::= SEQUENCE {
        iD              Identifier,
        usage           KeyUsageFlags,
        native          BOOLEAN DEFAULT TRUE,
        accessFlags     KeyAccessFlags OPTIONAL,
        keyReference    KeyReference OPTIONAL,
        startDate   GeneralizedTime OPTIONAL,
        endDate     [0] GeneralizedTime OPTIONAL,
        algReference    [1] SEQUENCE OF Reference OPTIONAL,
        ... -- For future extensions
        }

KeyUsageFlags ::= BIT STRING {
        encipher                (0),
        decipher                (1),
        sign                    (2),
        signRecover             (3),
        keyEncipher             (4),
        keyDecipher             (5),
        verify                  (6),
        verifyRecover           (7),
        derive                  (8),
        nonRepudiation  (9)
        }

KeyAccessFlags ::= BIT STRING {
        sensitive       (0),
        extractable             (1),
```

```
        alwaysSensitive    (2),
        neverExtractable   (3),
        cardGenerated              (4)
        }

KeyReference ::= INTEGER
```

The iD field shall be unique for each key information object, except when a public key information object and its corresponding private key information object are stored on the same card. In this case, the information objects shall share the same identifier (which may also be shared with one or several certificate information objects, see subclause 8.2.15).

The usage field (encipher, decipher, sign, signRecover, keyEncipher, keyDecipher, verify, verifyRecover, derive and nonRepudiation) signals the possible usage of the key. Actual algorithms and methods used for these operations are implicit and not defined in this part of ISO/IEC 7816. To map between ISO/IEC 9594-8:1998 keyUsage flags for public keys, CIO flags for public keys, and CIO flags for private keys, use the following table:

**Table 2 – Mapping between CIO key usage flags and ISO/IEC 9594-8:1998 key usage flags**

| Key usage flags for public keys in ISO/IEC 9594-8 public key certificates | Corresponding CIO key usage flags for public keys | Corresponding CIO key usage flags for private keys |
|---|---|---|
| DataEncipherment | Encipher | Decipher |
| DigitalSignature, keyCertSign, cRLSign (signature algorithms without message recovery) | Verify | Sign |
| DigitalSignature, keyCertSign, cRLSign (signature algorithms with message recovery) | VerifyRecover | SignRecover |
| KeyAgreement | Derive | Derive |
| KeyEncipherment | KeyEncipher | KeyDecipher |
| NonRepudiation | NonRepudiation | NonRepudiation |
| NOTE – Implementations should verify that all key usage flags for a particular key pair is consistent. | | |

The native field identifies whether the card is able to use the key for hardware computations or not.

The interpretation of the KeyAccessFlags bits shall be as follows:

- sensitive indicates that the key material cannot be revealed in plaintext outside the card;
- if extractable is not set the key material cannot be extracted from the card, even in encrypted form;
- alwaysSensitive indicates that the key has always been sensitive;
- neverExtractable indicates that the key has never been extractable; and
- cardGenerated indicates that the key was randomly generated on the card.

The accessFlags field may be absent in cases where its value can be deduced by other means.

The keyReference field is only applicable for cards with cryptographic capabilities. If present, it contains a card-specific reference to the key in question (for further information see ISO/IEC 7816-4:1995 and ISO/IEC 7816-8:1999).

NOTE – The value of the keyReference field is intended for use in key reference DOs (ISO/IEC 7816-8:1999, Table 3), and any values, also negative values, are conceivable.

The startDate and endDate fields, if present, indicate the period during which the key is valid for use.

The algReference field identifies algorithms the key may be used with by referencing supportedAlgorithm values from the EF.Cardinfo file.

### 8.2.10 CommonPrivateKeyAttributes

```
CommonPrivateKeyAttributes ::= SEQUENCE {
        name                Name OPTIONAL,
        keyIdentifiers      [0] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        generalName         [1] GeneralNames OPTIONAL,
        ... -- For future extensions
        }
```

The name field, when present, names the owner of the key, as specified in a corresponding certificate's subject field.

Values of the keyIdentifiers field can be matched to identifiers from external messages or protocols to select the appropriate key to a given operation. The values can also be transmitted to a receiving party to indicate which key was used. A number of mechanisms for identifying a key are supported (see clause 8.2.4).

The generalName field, when present, provides other ways to identify the owner of the key.

### 8.2.11 CommonPublicKeyAttributes

```
CommonPublicKeyAttributes ::= SEQUENCE {
        name            Name OPTIONAL,
        trustedUsage    [0] Usage OPTIONAL,
        generalName     [1] GeneralNames OPTIONAL,
        keyIdentifiers  [2] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the name, generalName and keyIdentifiers fields of the CommonPublicKeyAttributes type shall be the same as for the corresponding fields of the CommonPrivateKeyAttributes.

The trustedUsage field indicates one or more purposes for which the public key is trusted by the cardholder (See further clause 8.2.15).

NOTE - The exact semantics of "trust" is outside the scope of this standard.

### 8.2.12 CommonSecretKeyAttributes

```
CommonSecretKeyAttributes ::= SEQUENCE {
        keyLen      INTEGER OPTIONAL, -- keylength (in bits)
        ... -- For future extensions
        }
```

The optional keyLen field signals the key length used, in those cases where a particular algorithm can have a varying key length.

### 8.2.13 GenericKeyAttributes

```
GenericKeyAttributes ::= SEQUENCE {
        keyType     CIO-ALGORITHM.&objectIdentifier ({AllowedAlgorithms}),
        keyAttr     CIO-ALGORITHM.&Parameters ({AllowedAlgorithms}{@keyType})
        }

AllowedAlgorithms CIO-ALGORITHM ::= {...}
```

This type is intended to contain information specific to a key of a given kind. The definition of the AllowedAlgorithms information object set is deferred, perhaps to standardized profiles or to protocol implementation conformance statements. The set is required to specify a table constraint on the components of GenericKeyAttributes.

### 8.2.14 KeyInfo

```
KeyInfo {ParameterType, OperationsType} ::= CHOICE {
        paramsAndOps SEQUENCE {
            parameters ParameterType,
            operations      OperationsType OPTIONAL
            },
        reference   Reference -- Historical, not to be used
        }
```

NOTE – PKCS #15 uses the reference option.

This type, which is an optional part of each private and public key type, contains either algorithm-specific details about the parameters of the key and operations supported by the card or a reference to such information. If present, algorithm-specific values override any values referenced by the CommonKeyAttributes.algReference field.

### 8.2.15 CommonCertificateAttributes

```
CommonCertificateAttributes ::= SEQUENCE {
        iD                  Identifier,
```

```
        authority    BOOLEAN DEFAULT FALSE,
        identifier          CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        certHash    [0] CertHash OPTIONAL,
        trustedUsage        [1] Usage OPTIONAL,
        identifiers  [2] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        ...
        } -- Context tag [3] is reserved for historical reasons
```

NOTE – PKCS #15 uses context tag [3].

```
Usage ::= SEQUENCE {
        keyUsage KeyUsage OPTIONAL,
        extKeyUsage SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER OPTIONAL,
        ...
        } (WITH COMPONENTS {..., keyUsage PRESENT} | WITH COMPONENTS {..., extKeyUsage PRESENT})
```

When a public key in a certificate referenced by a certificate information object corresponds to a private key referenced by a private key information object, then the information objects shall share the same value for the iD field. This requirement will simplify searches for a private key corresponding to a particular certificate and vice versa. Multiple certificates for the same key shall share the same value for the iD field.

The authority field indicates whether the certificate is for an authority (e.g. certification authority) or not.

The identifier field is present for historical reasons only, and the identifiers field shall be used instead.

The certHash field is useful from a security perspective when a certificate is stored external to the card (the url choice of ReferencedValue), since it enables a user to verify that no one has tampered with the certificate.

The trustedUsage field indicates one or more purposes for which the certified public key is trusted by the cardholder. Object identifiers for the extKeyUsage field may be defined by any organization with a need. For actual usage, the intersection of the indicated usage in this field, and the keyUsage extension (if present) in the certificate itself should be taken. If the trustedUsage field is absent, all usage is possible.

NOTE 1 – The exact semantics of "trust" is outside the scope of this part of ISO/IEC 7816.

NOTE 2 – To find a cardholder certificate for a specific usage, use the commonKeyAttributes.usage field, and follow the cross-reference (commonKeyAttributes.iD) to an appropriate certificate.

The identifiers field simplifies the search of a particular certificate, when the requester knows (and conveys) some distinguishing information about the requested certificate. This can be used, for example, when a user certificate has to be chosen and sent to a server as part of a user authentication, and the server provides the client with distinguishing information for a particular certificate. Use of the subjectNameHash and issuerNameHash alternatives may also facilitate fast chain building.

### 8.2.16 GenericCertificateAttributes

```
GenericCertificateAttributes ::= SEQUENCE {
        certType    CIO-OPAQUE.&id ({AllowedCertificates}),
        certAttr    CIO-OPAQUE.&Type ({AllowedCertificates}{@certType})
        }

AllowedCertificates CIO-OPAQUE ::= {...}
```

This type is intended to contain information specific to a certificate of any kind. The definition of the AllowedCertificates information object set is deferred, perhaps to standardized profiles or to protocol implementation conformance statements. The set is required to specify a table constraint on the components of GenericCertificateAttributes.

### 8.2.17 CommonDataContainerObjectAttributes

```
CommonDataContainerObjectAttributes ::= SEQUENCE {
        applicationName   Label OPTIONAL,
        applicationOID          OBJECT IDENTIFIER OPTIONAL,
iD                              Identifier OPTIONAL,
        ... -- For future extensions
        } (WITH COMPONENTS {..., applicationName PRESENT}| WITH COMPONENTS {..., applicationOID PRESENT})
```

The applicationName field is intended to contain the name or the registered object identifier for the application to which the data container object in question "belongs". In order to avoid application name collisions, at least the applicationOID alternative is recommended. As indicated in ASN.1, at least one of the fields has to be present in a value of type CommonDataContainerObjectAttributes.

The iD field may be used to associate a certain data container object with some other CIO, e.g. a private key information object.

### 8.2.18 CommonAuthenticationObjectAttributes

```
CommonAuthenticationObjectAttributes ::= SEQUENCE {
        authId              Identifier OPTIONAL,
        authReference       Reference OPTIONAL,
        seIdentifier [0] Reference OPTIONAL,
        ... -- For future extensions
        }
```

The authId shall be a unique identifier. It is used for cross-reference purposes from private CIOs.

The authReference field, when present, shall contain a value of a "key reference" object (see ISO/IEC 7816-8:1999, Table 3), which is the way to reference these keys in Security Environments.

The seIdentifier field, when present, identifiers the security environment to which the authentication object belongs.

### 8.2.19 The CIO type

This type is a template for all kinds of CIOs. It is parameterized with object class attributes, object subclass attributes and object type attributes.

```
CIO {ClassAttributes, SubClassAttributes, TypeAttributes} ::= SEQUENCE {
        commonObjectAttributes  CommonObjectAttributes,
        classAttributes                 ClassAttributes,
        subClassAttributes              [0] SubClassAttributes OPTIONAL,
        typeAttributes                  [1] TypeAttributes
        }
```

## 8.3 The CIOChoice type

```
CIOChoice ::= CHOICE {
        privateKeys                     [0] PrivateKeys,
        publicKeys                      [1] PublicKeys,
        trustedPublicKeys               [2] PublicKeys,
        secretKeys                      [3] SecretKeys,
        certificates            [4] Certificates,
        trustedCertificates             [5] Certificates,
        usefulCertificates      [6] Certificates,
        dataContainerObjects    [7] DataContainerObjects,
        authObjects                     [8] AuthObjects,
        ... -- For future extensions
        }

PrivateKeys ::=   PathOrObjects {PrivateKeyChoice}

SecretKeys  ::=   PathOrObjects {SecretKeyChoice}

PublicKeys  ::=   PathOrObjects {PublicKeyChoice}

Certificates ::=  PathOrObjects {CertificateChoice}

DataContainerObjects ::= PathOrObjects {DataContainerObjectChoice}

AuthObjects ::=   PathOrObjects {AuthenticationObjectChoice}
```

EF.OD shall contain the concatenation of 0, 1 or more DER-encoded CIOChoice values. Any number of 'FF' octets may occur before, between, or after the values without any meaning (i.e. as padding for unused space or delete values). A specific choice may appear more than once in the file (which may be done, for example, to apply different access control rules to separate collections of objects of the same type).

It is expected that an EF.OD entry will usually reference a separate file (the path choice of PathOrObjects) containing CIOs of the indicated type. An entry may, however, hold CIOs directly (the objects choice of PathOrObjects), if the objects and the EF.OD file have the same access control requirements.

The trustedPublicKeys field references public key information objects describing public keys that are trusted by the cardholder for some purpose, such as being the trust point (root) for certificate path processing.

The certificates choice references certificate information objects describing certificates issued to the card or the cardholder.

The trustedCertificates field references certificate information objects describing certificates trusted by the cardholder for their indicated purposes. For instance, CA certificates referenced by this field may be used as trust points (roots) during certificate path processing.

NOTE – To maintain the desired trust in given certificates and/or public keys, their associated CIOs within trustedCertificates and/or trustedPublicKeys fields need appropriate protection against modification (i.e. appropriate access control). This protection must apply to the EF.OD file, any CIO file referenced by the trustedCertificates or trustedPublicKeys fields, and any actual key or certificate file referenced from the individual CIOs.

A usefulCertificates field references certificate information objects describing certificates that does not belong in either a trustedCertificates or certificates field. It may be used to store either end-entity or CA certificates that may be useful, e.g. a certificate for a colleague's encryption key or intermediate CA certificates to simplify certificate path processing.

## 8.4 Private key information objects

### 8.4.1 PrivateKeyChoice

```
PrivateKeyChoice ::= CHOICE {
        privateRSAKey           PrivateKeyObject {PrivateRSAKeyAttributes},
        privateECKey            [0] PrivateKeyObject {PrivateECKeyAttributes},
        privateDHKey            [1] PrivateKeyObject {PrivateDHKeyAttributes},
        privateDSAKey           [2] PrivateKeyObject {PrivateDSAKeyAttributes},
        privateKEAKey           [3] PrivateKeyObject {PrivateKEAKeyAttributes},
        genericPrivateKey [4] PrivateKeyObject {GenericKeyAttributes},
        ... -- For future extensions
        }

PrivateKeyObject {KeyAttributes} ::= CIO {
        CommonKeyAttributes, CommonPrivateKeyAttributes, KeyAttributes}
```

This type contains information pertaining to a private key. Each value consists of attributes common to any object, any key, any private key and attributes particular to the key.

### 8.4.2 Private RSA key attributes

```
PrivateRSAKeyAttributes ::= SEQUENCE {
        value                   ObjectValue {CIO-OPAQUE.&Type},
        modulusLength   INTEGER, -- modulus length in bits, e.g. 1024
        keyInfo                 KeyInfo {NULL, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the fields shall be as follows:

— PrivateRSAKeyAttributes.value: The value shall be a path to a file containing a private RSA key. If there is no need to specify a path to a file, the path value may be set to the empty path.

— PrivateRSAKeyAttributes.modulusLength: On many cards, one needs to format data to be signed prior to sending the data to the card. In order to be able to format the data in a correct manner the length of the key must be known. The length shall be expressed in bits, e.g. 1024.

— PrivateRSAKeyAttributes.keyInfo: Information about parameters that applies to this key and operations the card can carry out with it. The values override any CardInfo.supportedAlgorithms value referenced by the CommonKeyAttributes.algReference field. The field is not needed if the information is available through other means.

### 8.4.3 Private Elliptic Curve key attributes

```
PrivateECKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {Parameters, PublicKeyOperations} OPTIONAL,
          ... -- For future extensions
        }
```

The interpretation of the fields shall be as follows:

— PrivateECKeyAttributes.value: The value shall be a path to a file containing a private elliptic-curve key. If there is no need to specify a path to a file, the path value may be set to the empty path.

— PrivateECKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.4.4 Private Diffie-Hellman key attributes

```
PrivateDHKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the fields shall be as follows:

— PrivateDHKeyAttributes.value: The value shall be a path to a file a private Diffie-Hellman key. If there is no need to specify a path to a file, the path value may be set to the empty path.

— PrivateDHKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.4.5 Private DSA key attributes

```
PrivateDSAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the fields shall be as follows:

— PrivateDSAKeyAttributes.value: The value shall be a path to a file containing a private DSA key. If there is no need to specify a path to a file, the path value may be set to the empty path.

— PrivateDSAKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.4.6 Private KEA key attributes

```
PrivateKEAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the fields shall be as follows:

— PrivateKEAKeyAttributes.value: The value shall be a path to a file containing a private KEA key. If there is no need to specify a path to a file, the path value may be set to the empty path.

— PrivateKEAKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.4.7 Generic Private key information objects

This type is intended to contain information specific to a private key of any kind. See further clause 8.2.13.

## 8.5 Public key information objects

### 8.5.1 PublicKeyChoice

```
PublicKeyChoice ::= CHOICE {
        publicRSAKey            PublicKeyObject {PublicRSAKeyAttributes},
        publicECKey             [0] PublicKeyObject {PublicECKeyAttributes},
        publicDHKey             [1] PublicKeyObject {PublicDHKeyAttributes},
        publicDSAKey            [2] PublicKeyObject {PublicDSAKeyAttributes},
        publicKEAKey            [3] PublicKeyObject {PublicKEAKeyAttributes},
        genericPublicKey  [4] PublicKeyObject{GenericKeyAttributes},
        ... -- For future extensions
        }

PublicKeyObject {KeyAttributes} ::= CIO {
        CommonKeyAttributes, CommonPublicKeyAttributes, KeyAttributes}
```

This type contains information pertaining to a public key. Each value consists of attributes common to any object, any key, any public key and attributes particular to the key.

### 8.5.2 Public RSA key attributes

```
PublicRSAKeyAttributes ::= SEQUENCE {
        value                   ObjectValue {RSAPublicKeyChoice},
        modulusLength   INTEGER, -- modulus length in bits, e.g. 1024
        keyInfo                 KeyInfo {NULL, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

RSAPublicKeyChoice ::= CHOICE {
        raw RSAPublicKey,
        spki[1] SubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public RSA key.
        ...
        }

RSAPublicKey ::= SEQUENCE {
        modulus                 INTEGER,
        publicExponent  INTEGER
        }
```

The interpretation of the fields shall be as follows:

— PublicRSAKeyAttributes.value: The value shall be a path to a file containing either an RSAPublicKeyChoice value or (some card-specific representation of) a public RSA key.

— PublicRSAKeyAttributes.modulusLength: On many cards, one must format data to be encrypted prior to sending the data to the card. In order to be able to format the data in a correct manner the length of the key must be known. The length shall be expressed in bits, e.g. 1024.

— PublicRSAKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.5.3 Public Elliptic Curve key attributes

```
PublicECKeyAttributes ::= SEQUENCE {
        value       ObjectValue {ECPublicKeyChoice},
        keyInfo     KeyInfo {Parameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

ECPublicKeyChoice ::= CHOICE {
        raw ECPoint, -- See ANSI X9.62,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public elliptic-curve key
        ...
        }
```

The interpretation of the fields shall be as follows:

— PublicECKeyAttributes.value: The value shall be a path to a file containing either an ECPublicKeyChoice value or (some card-specific representation of) a public elliptic curve key.

— PublicECKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.5.4 Public Diffie-Hellman key attributes

```
PublicDHKeyAttributes ::= SEQUENCE {
        value       ObjectValue {DHPublicKeyChoice},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

DHPublicKeyChoice ::= CHOICE {
        raw DHPublicNumber,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public D-H key.
        ...
        }

DHPublicNumber ::= INTEGER
```

The interpretation of the fields shall be as follows:

— PublicDHKeyAttributes.value: The value shall be a path to a file containing either a DHPublicKeyChoice value or (some card-specific representation of) a public D-H key.

— PublicDHKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.5.5 Public DSA key attributes

```
PublicDSAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {DSAPublicKeyChoice},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

DSAPublicKeyChoice ::= CHOICE {
        raw DSAPublicKey,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public DSA key.
        ...
        }

DSAPublicKey ::= INTEGER
```

The interpretation of the fields shall be as follows:

— PublicDSAKeyAttributes.value: The value shall be a path to a file containing either a DSAPublicKeyChoice value or (some card-specific representation of) a public DSA key.

— PublicDSAKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.5.6 Public KEA key attributes

```
PublicKEAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {KEAPublicKeyChoice},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

KEAPublicKeyChoice ::= CHOICE {
        raw KEAPublicKey,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public KEA key.
        ...
        }

KEAPublicKey ::= INTEGER
```

The interpretation of the fields shall be as follows:

— PublicKEAKeyAttributes.value: The value shall be a path to a file containing either a KEAPublicKeyChoice value or (some card-specific representation of) a public KEA key.

— PublicKEAKeyAttributes.keyInfo: See corresponding field in clause 8.4.2.

### 8.5.7 Generic public key information objects

This type is intended to contain information specific to a public key of any kind. See further clause 8.2.13.

## 8.6 Secret key information objects

### 8.6.1 SecretKeyChoice

```
SecretKeyChoice ::= CHOICE {
        algIndependentKey       SecretKeyObject {SecretKeyAttributes},
        genericSecretKey  [15] SecretKeyObject {GenericKeyAttributes},
        ... -- For future extensions
        } -- Note: Context tags [0] – [14] are historical and not to be used
```

NOTE – PKCS #15 uses these tags.

```
SecretKeyObject {KeyAttributes} ::= CIO {
         CommonKeyAttributes, CommonSecretKeyAttributes, KeyAttributes}
```

This type contains information pertaining to a secret key. Each value consists of attributes common to any object, any key, any secret key and attributes particular to the key.

### 8.6.2 Algorithm independent key attributes

These objects represent secret keys available for use in various algorithms, or for derivation of other secret keys.

```
SecretKeyAttributes ::= SEQUENCE {
        value       ObjectValue { OCTET STRING },
        ... -- For future extensions
        }
```

The interpretation of the field shall be as follows:

— SecretKeyAttributes.value: The value shall be a path to a file either containing an OCTET STRING or (in the case of a card capable of performing secret-key operations) some card specific representation of the key.

### 8.6.3 The GenericSecretKey type

This type is intended to contain information specific to a secret key of any kind. See further clause 8.2.13.

## 8.7 Certificate information objects

### 8.7.1 CertificateChoice

```
CertificateChoice ::= CHOICE {
        x509Certificate             CertificateObject {X509CertificateAttributes},
        x509AttributeCertificate    [0] CertificateObject {X509AttributeCertificateAttributes},
        spkiCertificate                 [1] CertificateObject {SPKICertificateAttributes},
        pgpCertificate                  [2] CertificateObject {PGPCertificateAttributes},
        wtlsCertificate                 [3] CertificateObject {WTLSCertificateAttributes},
        x9-68Certificate            [4] CertificateObject {X9-68CertificateAttributes},
        cvCertificate                   [5] CertificateObject {CVCertificateAttributes},
        genericCertificateObject    [6] CertificateObject {GenericCertificateAttributes},
        ... -- For future extensions
        } -- Context tag 4 is reserved for forthcoming ANSI X9.68 certificates

CertificateObject {CertAttributes} ::= CIO {
        CommonCertificateAttributes, NULL, CertAttributes}
```

This type contains information pertaining to a certificate. Each value consists of attributes common to any object, any certificate and attributes particular to the certificate.

### 8.7.2 X.509 certificate attributes

```
X509CertificateAttributes ::= SEQUENCE {
        value               ObjectValue { Certificate },
        subject             Name OPTIONAL,
        issuer              [0] Name OPTIONAL,
        serialNumber        CertificateSerialNumber OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the fields shall be as follows:

— X509CertificateAttributes.value: The value shall be a ReferencedValue either identifying a file containing a DER encoded certificate at the given location, or a URL pointing to some location where the certificate can be found.

— X509CertificateAttributes.subject, X509CertificateAttributes.issuer and X509CertificateAttributes.serialNumber: The semantics of these fields is the same as for the corresponding fields in ISO/IEC 9594-8:1998. The values of these fields shall be exactly the same as for the corresponding fields in the certificate itself. The reason for making them optional is to provide some space-efficiency, since they already are present in the certificate itself.

### 8.7.3 X.509 attribute certificate attributes

```
X509AttributeCertificateAttributes ::= SEQUENCE {
        value               ObjectValue { AttributeCertificate },
        issuer              GeneralNames OPTIONAL,
```

```
    serialNumber        CertificateSerialNumber OPTIONAL,
    attrTypes   [0] SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
    ... -- For future extensions
    }
```

The interpretation of the fields shall be as follows:

— X509AttributeCertificateAttributes.value: The value shall be a ReferencedValue identifying either a file containing a DER encoded attribute certificate at the given location, or a URL pointing to some location where the attribute certificate can be found.

— X509AttributeCertificateAttributes.issuer and X509AttributeCertificateAttributes.serialNumber: The values of these fields shall be exactly the same as for the corresponding fields in the attribute certificate itself. They may be stored explicitly for easier lookup.

— X509AttributeCertificateAttributes.attrTypes: This optional field shall, when present, contain a list of object identifiers for the attributes that are present in this attribute certificate. This offers an opportunity for applications to search for a particular attribute certificate without reading and parsing the certificate itself.

### 8.7.4 SPKI certificate attributes

NOTE – SPKI Certificates are defined in IETF RFC 2693 (see the Bibliography).

```
SPKICertificateAttributes ::= SEQUENCE {
    value       ObjectValue { CIO-OPAQUE.&Type },
    ... -- For future extensions
    }
```

The interpretation of the field shall be as follows:

— SPKICertificateAttributes.value: The value shall be a ReferencedValue identifying either a file containing a SPKI certificate at the given location, or a URL pointing to some location where the certificate can be found.

### 8.7.5 PGP (Pretty Good Privacy) certificate attributes

NOTE – PGP Certificates are defined in IETF RFC 2440 (see the Bibliography).

```
PGPCertificateAttributes ::= SEQUENCE {
    value       ObjectValue { CIO-OPAQUE.&Type },
    ... -- For future extensions
    }
```

The interpretation of the field shall be as follows:

— PGPCertificateAttributes.value: The value shall be a ReferencedValue identifying either a file containing a PGP certificate at the given location, or a URL pointing to some location where the certificate can be found.

### 8.7.6 WTLS certificate attributes

NOTE – WTLS Certificates are defined in the "Wireless Transport Layer Security Protocol" specification (see the Bibliography).

```
WTLSCertificateAttributes ::= SEQUENCE {
    value       ObjectValue { CIO-OPAQUE.&Type },
    ... -- For future extensions
    }
```

The interpretation of the field shall be as follows:

— WTLSCertificateAttributes.value: The value shall be a ReferencedValue identifying either a file containing a WTLS encoded certificate at the given location, or a URL pointing to some location where the certificate can be found.

### 8.7.7 ANSI X9.68 domain certificate attributes

NOTE – X9.68 domain certificates are defined in ANSI X9.68:2-2001 (see the Bibliography).

```
X9-68CertificateAttributes ::= SEQUENCE {
    value       ObjectValue { CIO-OPAQUE.&Type },
    ... -- For future extensions
    }
```

The interpretation of the field shall be as follows:

— X9-68CertificateAttributes.value: The value shall be a ReferencedValue identifying either a file containing a DER or PER (ISO/IEC 8825-2:1998) encoded ANSI X9.68:2-2001 domain certificate at the given location, or a URL pointing to some location where the certificate can be found.

#### 8.7.8 Card Verifiable Certificate attributes

NOTE – Card Verifiable Certificates are defined in ISO/IEC 7816-8:1999. Their main use is in public-key based card authentication methods.

```
CVCertificateAttributes ::= SEQUENCE {
        value       ObjectValue { CIO-OPAQUE.&Type},
        ... -- For future extensions
        }
```

The interpretation of the field shall be as follows:

— CVCertificateAttributes.value: The value shall be a ReferencedValue identifying either either a file containing an ISO/IEC 7816-8:1999 Card Verifiable Certificate at the given location, or a URL pointing to some location where the certificate can be found.

#### 8.7.9 Generic certificate attributes

This type is intended to contain information specific to a certificate of any kind. See further clause 8.2.16

### 8.8 Data container information objects

#### 8.8.1 DataContainerObjectChoice

```
DataContainerObjectChoice ::= CHOICE {
        opaqueDO DataContainerObject {OpaqueDOAttributes},
        iso7816DO [0] DataContainerObject {ISO7816DOAttributes},
        oidDO           [1] DataContainerObject {OidDOAttributes},
        ... -- For future extensions
        }

DataContainerObject {DataObjectAttributes} ::= CIO {
        CommonDataContainerObjectAttributes, NULL, DataObjectAttributes}
```

This type contains information pertaining to a data container object. Each value consists of attributes common to any object, any data container object and attributes particular to the data container object.

#### 8.8.2 Opaque data container object attributes

Interpretation of these objects is left to applications accessing them.

```
OpaqueDOAttributes ::= ObjectValue {CIO-OPAQUE.&Type}
```

#### 8.8.3 ISO/IEC 7816 data object attributes

EF.DCOD may contain information about one or several IDOs. These objects shall follow a compatible tag allocation scheme as defined in clause 4.4 of ISO/IEC 7816-6:1996.

```
ISO7816DOAttributes ::= ObjectValue {CIO-OPAQUE.&Type}
    (CONSTRAINED BY {-- All such data container objects shall be defined in accordance with ISO/IEC 7816-6 --})
```

Each iso7816DO entry in an EF.DCOD will therefore reference a file which shall conform to ISO/IEC 7816-6:1996. By using these data container objects, applications enhance interoperability.

When the CDE being referenced is a data object to be retrieved e.g. in a 'GET DATA' command, the direct choice of ObjectValue shall be used, and the CIO-OPAQUE.&Type value shall be the data object's tag.

#### 8.8.4 Data container information objects identified by OBJECT IDENTIFIERS

This type provides a way to store, search, and retrieve data container objects with assigned object identifiers. An example of this type of information is any ASN.1 ATTRIBUTE.

```
OidDOAttributes ::= SEQUENCE {
        id          OBJECT IDENTIFIER,
        value       ObjectValue {CIO-OPAQUE.&Type}
        }
```

## 8.9 Authentication information objects

### 8.9.1 AuthenticationObjectChoice

```
AuthenticationObjectChoice ::= CHOICE {
        pwd                     AuthenticationObject { PasswordAttributes },
        biometricTemplate       [0] AuthenticationObject{ BiometricAttributes},
        authKey                 [1] AuthenticationObject {AuthKeyAttributes},
        external                [2] AuthenticationObject {ExternalAuthObjectAttributes},
        ... -- For future extensions
        }

AuthenticationObject {AuthObjectAttributes} ::= CIO {
        CommonAuthenticationObjectAttributes, NULL, AuthObjectAttributes}
```

This type contains information about a particular authentication method. Each authentication object shall have a distinct CommonAuthenticationObjectAttributes.authID, enabling unambiguous authentication object lookup for private objects.

### 8.9.2 Password attributes

```
PasswordAttributes ::= SEQUENCE {
        pwdFlags   PasswordFlags,
        pwdType    PasswordType,
        minLength INTEGER (cti-lb-minPasswordLength..cti-ub-minPasswordLength),
        storedLength       INTEGER (0..cti-ub-storedPasswordLength),
        maxLength          INTEGER OPTIONAL,
        pwdReference       [0] Reference DEFAULT 0,
        padChar    OCTET STRING (SIZE(1)) OPTIONAL,
        lastPasswordChange      GeneralizedTime OPTIONAL,
        path               Path OPTIONAL,
        ... -- For future extensions
        }

PasswordFlags ::= BIT STRING {
        case-sensitive                  (0),
        local                           (1),
        change-disabled         (2),
        unblock-disabled        (3),
        initialized             (4),
        needs-padding           (5),
        unblockingPassword      (6),
        soPassword                      (7),
        disable-allowed         (8),
        integrity-protected     (9),
        confidentiality-protected (10),
        exchangeRefData         (11)
        } (CONSTRAINED BY { -- 'unblockingPassword' and 'soPassword' cannot both be set -- })

PasswordType ::= ENUMERATED {bcd, ascii-numeric, utf8, ..., half-nibble-bcd, iso9564-1}
```

The interpretation of these types shall be as follows:

— PasswordAttributes.pwdFlags: This field signals whether the password:

  — is case-sensitive, meaning that a user-given password shall not be converted to all-uppercase before presented to the card (see below);

  — is local, meaning that the password is local to the application to which it belongs;

  NOTE – A pwd, which is not "local," is considered "global". A local password may only be used to protect data within a given application. For a local password the lifetime of verification is not guaranteed and it may have to be re-verified on each use. In contrast to this, a successful verification of a global password means that the verification remains in effect until the card has been removed or reset, or until a new verification of the same password fails. An application, which has verified a global password, can assume that the password remains valid, even if other applications verify their own, local passwords, select other DFs, etc.

  — is change-disabled, meaning that it is not possible to change the password;

  — is unblock-disabled, meaning that it is not possible to unblock the password;

  — is initialized, meaning that the password has been initialized;

- needs-padding, meaning that, depending on the length of the given password and the stored length, the password may need to be padded before being presented to the card;
- is an unblockingPassword (ISO/IEC 7816-8:1999 resetting code), meaning that this password may be used for unblocking purposes, i.e. to reset the retry counter of the related authentication object to its initial value;
- is a soPassword, meaning that the password is a Security Officer (administrator) password;

  NOTE – Since passwords are described by CIOs other authentication objects may protect them. This gives a way to specify the password that can be used to unblock (i.e. reset retry counter for) another password - let the authID of a password information object point to an unblocking password authentication object.

- is disable-allowed, meaning that the password might be disabled;
- shall be presented to the card with secure messaging (integrity-protected);
- shall be presented to the card encrypted (confidentiality-protected);
- can be changed by just presenting new reference data to the card or if both old and new reference data needs to be presented. If the bit is set, both old and new reference data shall be presented; otherwise only new reference data needs to be presented (exchangeRefData).

— PasswordAttributes.pwdType: This field determines the type of password:

- bcd (Binary Coded Decimal, each nibble of a byte shall contain one digit of the password);
- ascii-numeric (Each byte of the password contain an ASCII (ANSI X3.4, see the Bibliography) encoded digit);
- utf8 (Each character is encoded in accordance with UTF-8);
- half-nibble-bcd (lower nibble of a byte shall contain one digit of the password, upper nibble shall contain 'F'); or
- iso9564-1 (Encoding in accordance with ISO 9564-1:1996).

— PasswordAttributes.minLength: Minimum length (in characters) of new passwords (if allowed to change).

— PasswordAttributes.storedLength: Stored length on card (in bytes). Used to deduce the number of padding characters needed. Value can be set to 0 and disregarded if pwdFlags indicate that padding is not needed (i.e. no padding characters are sent to the card).

— PasswordAttributes.maxLength: On some cards, passwords are not padded, and there is therefore a need to know the maximum password length (in characters) allowed.

— PasswordAttributes.pwdReference: This field is a card-specific reference to the password. It is anticipated that it can be used as a 'P2' parameter in the ISO/IEC 7816-4:1995 'VERIFY' command, when applicable. If not present, it defaults to the value 0.

— PasswordAttributes.padChar: Padding character to use (usually 'FF' or '00'). Not needed if pwdFlags indicates that padding is not needed for this card. If the PasswordAttributes.pwdType is of type bcd, then padChar should consist of two nibbles of the same value, any nibble could be used as the "padding nibble". E.g., '55' is allowed, meaning padding with '$0101_2$', but '34' is illegal.

— PasswordAttributes.lastPasswordChange: This field is intended to be used in applications that requires knowledge of the date the password last was changed (e.g. to enforce password expiration policies). When the password is not set (or never has been changed) the value shall be (using the value-notation defined in ISO/IEC 8824-1:1998) '000000000000Z'. As another example, a password changed on January 6, 1999 at 1934 (7 34 PM) UTC would have a lastPasswordChange value of '19990106193400Z'.

— PasswordAttributes.path: Path to the DF in which the password resides. The path shall be selected by a host application before doing a password operation, in order to enable a suitable authentication context for the password operation. If not present, a card-holder verification shall always be possible to perform without a prior 'SELECT' operation.

#### 8.9.2.1 Encoding a supplied password

The steps taken by a host-side application to encode a user-supplied password to something presented to the card shall be as follows:

a) Convert the password in accordance with the password type:

   1) If the password is a utf8 password, transform it to UTF-8: x = UTF8(password). Then, if the case-sensitive bit is off, convert x to uppercase: x = NLSUPPERCASE(x) (NLSUPPERCASE = locale dependent uppercase)

2) If the password is a bcd password, verify that each character is a digit and encode the characters as BCD digits: x = BCD(password)

3) If the password is an ascii-numeric or iso9564-1 password, verify that each character is a digit in the current code-page and –if needed– encode the characters as ASCII digits: x = ASCII(password)

4) If the password is a half-nibble-bcd password, verify that each character is a digit and encode the characters as BCD in the lower half of each byte, setting each upper nibble to '$F_{16}$': x = Half-BCD(password)

b) If indicated in the pwdFlags field, pad x to the right with the padding character, padChar, to stored length storedLength: x = PAD(x, padChar, storedLength).

c) If the pwdFlags.integrity-protected or pwdFlags.confidentiality-protected bits are set, apply the appropriate algorithms and keys to the converted and formatted password.

d) Present the password to the card.

EXAMPLE – (ascii-) Numeric password 1234, stored length 8 bytes, and padding character 'FF' gives that the value presented to the card will be '31323334FFFFFFFF'

### 8.9.3 Biometric reference data attributes

This type, only relevant to cards capable of performing authentications by comparing a stored biometric template with a provided biometric reading, contains information about a stored biometric template.

```
BiometricAttributes ::= SEQUENCE {
        bioFlags            BiometricFlags,
        templateId  BiometricTemplateIdentifier,
        bioType             BiometricType,
        bioReference        Reference DEFAULT 0,
        lastChange GeneralizedTime OPTIONAL,
        path                Path OPTIONAL,
        ... -- For future extensions
        }

BiometricTemplateIdentifier ::= CHOICE {
        oid         OBJECT IDENTIFIER,
        issuerId    OCTET STRING,
        ... -- For future extensions
        }

BiometricFlags ::= BIT STRING {
        local                           (1),
        change-disabled         (2),
        unblock-disabled        (3),
        initialized             (4),
        disable-allowed         (8),
        integrity-protected     (9),
        confidentiality-protected  (10)
        }

BiometricType ::= CHOICE {
        fingerPrint  FingerPrintInformation,
        iris                [0] IrisInformation,
        combined  [1] SEQUENCE SIZE (2..cti-ub-biometricTypes) OF BiometricType,
        ... -- For future extensions
        }

FingerPrintInformation ::= SEQUENCE {
        hand        ENUMERATED {left, right},
        finger      ENUMERATED {thumb, pointerFinger, middleFinger, ringFinger, littleFinger},
        }

IrisInformation ::= SEQUENCE {
        eye ENUMERATED {left, right},
        ... -- For future extensions
        }
```

The interpretation of these types shall be as follows:

— BiometricAttributes.bioFlags: Same as for PasswordAttributes.pwdFlags, but replace "password" with "biometrical reference data."

— BiometricAttributes.templateId: This field identifies the data structure that has to be sent to the card.

— BiometricAttributes.bioType: This field determines the type of biometrical information stored in the card, e.g. the right pointer finger.

— BiometricAttributes.bioReference, BiometricAttributes.lastChange, and BiometricAttributes.path: As for corresponding fields in PasswordAttributes, but replace "password" with "biometrical reference data."

### 8.9.4 Authentication objects for external authentication

```
ExternalAuthObjectAttributes ::= CHOICE {
        authKeyAttributes AuthKeyAttributes,
        certBasedAttributes     [0] CertBasedAuthenticationAttributes,
        ... -- For future extensions
        }

AuthKeyAttributes ::= SEQUENCE {
        derivedKeyBOOLEAN DEFAULT TRUE,
        authKeyId  Identifier,
        ... -- For future extensions
        }

CertBasedAuthenticationAttributes ::= SEQUENCE {
        cha OCTET STRING,
        ...
        }
```

The interpretation of these types shall be as follows:

— AuthKeyAttributes.derivedKey: This field specifies whether the authentication key stored in the card is a derived key (i.e. an individual key), a group key, or a master key, used for deriving individual keys.

— AuthKeyAttributes.authKeyId: This field specifies the identifier (CommonKeyAttribute.iD) of the authentication key as described in an EF.SKD.

— CertBasedAuthenticationAttributes.cha: This field specifies the certificate holder authorization as presented in a card-verifiable certificate (see ISO/IEC 7816-8:1999 and ISO/IEC 7816–9:2000). If a card-verifiable certificate containing this value is verified, and the authentication procedure with the corresponding key pair has been successfully completed, then the cha is set as valid, and access to private objects protected within this certificate-holder's authorization granted.

## 8.10 The cryptographic information file, EF.CardInfo

This type contains general information about DF.CIA and the card.

```
CardInfo ::= SEQUENCE {
        version                 INTEGER {v1(0),v2(1)} (v1|v2,...),
        serialNumber            OCTET STRING OPTIONAL,
        manufacturerID      Label OPTIONAL,
        label                   [0] Label OPTIONAL,
        cardflags           CardFlags,
        seInfo                  SEQUENCE OF SecurityEnvironmentInfo OPTIONAL,
        recordInfo          [1] RecordInfo OPTIONAL,
        supportedAlgorithms     [2] SEQUENCE OF AlgorithmInfo OPTIONAL,
        issuerId                [3] Label OPTIONAL,
        holderId                [4] Label OPTIONAL,
        lastUpdate          [5] LastUpdate OPTIONAL,
        preferredLanguage       PrintableString OPTIONAL, -- In accordance with IETF RFC 1766
        profileIndication   [6] SEQUENCE OF ProfileIndication OPTIONAL,
        ...
        } (CONSTRAINED BY { -- Each AlgorithmInfo.reference value shall be unique --})

CardFlags ::= BIT STRING {
        readonly        (0),
        authRequired    (1),
        prnGeneration   (2)
        } -- Bit (3) is reserved for historical reasons
```

```
SecurityEnvironmentInfo ::= SEQUENCE {
        se          INTEGER,
        owner       OBJECT IDENTIFIER OPTIONAL,
        aid         OCTET STRING
                    (CONSTRAINED BY {-- Must be encoded in accordance with ISO/IEC 7816-5 --}) OPTIONAL,
        ... -- For future extensions
        }

RecordInfo ::= SEQUENCE {
        oDRecordLength   [0] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        prKDRecordLength     [1] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        puKDRecordLength     [2] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        sKDRecordLength      [3] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        cDRecordLength   [4] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        dCODRecordLength     [5] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        aODRecordLength      [6] INTEGER (0..cti-ub-recordLength) OPTIONAL
        }

AlgorithmInfo ::= SEQUENCE {
        reference               Reference,
        algorithm               CIO-ALGORITHM.&id({AlgorithmSet}),
        parameters                  CIO-ALGORITHM.&Parameters({AlgorithmSet}{@algorithm}),
        supportedOperations     CIO-ALGORITHM.&Operations({AlgorithmSet}{@algorithm}),
        objId                       CIO-ALGORITHM.&objectIdentifier ({AlgorithmSet}{@algorithm}),
        algRef                      Reference OPTIONAL
        }

LastUpdate ::= CHOICE {
        generalizedTime   GeneralizedTime,
        referencedTime    ReferencedValue,
        ... -- For future extensions
        }(CONSTRAINED BY {-- The referencedValue shall be of type GeneralizedTime --})

ProfileIndication ::= CHOICE {
        profileOID  OBJECT IDENTIFIER,
        profileName       UTF8String,
        ... -- For future extensions
        }
```

**EF.CardInfo shall contain one DER-encoded value of type CardInfo.**

**The interpretation of the CardInfo type shall be as follows:**

- **CardInfo.version: This field shall be set to v2 for this edition of this part of ISO/IEC 7816. Future editions may use other values. A CardInfo value shall not be rejected solely because it has an unknown version number.**

  **NOTE – The version number v1 is used in the equivalent structure in PKCS #15.**

- **CardInfo.serialNumber: This field shall contain the card's unique serial number, for card issued in accordance with ISO/IEC 7812-1:1998 and coded in accordance with ISO/IEC 8583-2:1998.**

- **CardInfo.manufacturerID: This optional field shall, when present, contain identifying information about the card manufacturer (e.g. the card manufacturer), UTF-8 encoded.**

- **CardInfo.label: This optional field shall, when present, contain identifying information about the application.**

- **CardInfo.cardflags: This field contains information about the card per se. Flags include: If the card is read-only, if there are cryptographic functions that require a user to be authenticated, and if the card supports pseudo-random number generation.**

- **CardInfo.seInfo: This optional field is intended to convey information about pre-set security environments on the card, and the owner of these environments. The definition of these environments is currently out of scope for this part of ISO/IEC 7816, see further ISO/IEC 7816-8:1999. The aid field indicates the (card) application for which the security environment is applicable.**

- **CardInfo.recordInfo: This optional field has two purposes:**

  - **to indicate whether the elementary files EF.OD, EF.PrKD, EF.PuKD, EF.SKD, EF.CD, EF.DCOD and EF.AOD are linear record files or transparent files (if the field is present, they shall be linear record files, otherwise they shall be transparent files); and**

- if they are linear record files, whether they are of fixed-length or not (if they are of fixed length, corresponding values in RecordInfo are present and not equal to zero and indicates the record length. If some files are linear record files but not of fixed length, then corresponding values in RecordInfo shall be set to zero.

— CardInfo.supportedAlgorithms: The intent of this optional field is to indicate cryptographic algorithms, associated parameters, operations and algorithm input formats supported by the card. The reference field of AlgorithmInfo is a unique reference that is used for cross-reference purposes from PrKDs and PuKDs. Values for the algorithm field are for private use. Values of the supportedOperations field (compute-checksum, compute-signature, verify-checksum, verify-signature, encipher, decipher, hash and derive-key) identifies operations the card can perform with a particular algorithm. The objId field indicates the object identifier for the algorithm. The algRef field indicates the identifier used by the card for denoting this algorithm (and. which occurs at the card interface as a parameter of, e.g., an "EXTERNAL AUTHENTICATE" command).

NOTE – Values for the algorithm field may be chosen from, and interpreted as, mechanism numbers in PKCS #11 (see the Bibliography).

— CardInfo.issuerId: This optional field shall, when present, contain identifying information about the card issuer (e.g. the card issuer).

— CardInfo.holderId: This optional field shall, when present, contain identifying information about the cardholder (e.g. the cardholder).

— CardInfo.lastUpdate: This optional field shall, when present, contain (or refer to) the date of the last update of files in the CIA. The presence of this field, together with the CardInfo.serialNumber field, will enable host-side applications to quickly find out whether they have to read EF.OD, EF.CD, etc., or if they can used cached copies (if available). The referencedTime alternative of the LastUpdate type is intended for those cases when EF.CardInfo needs to be write-protected.

— CardInfo.preferredLanguage: The preferred language of the cardholder, encoded in accordance with IETF RFC 1766.

— CardInfo.profileIndication: This optional field shall, when present, indicate profiles of this part of ISO/IEC 7816, which the card has been issued in conformance with.

NOTE – It is left to other specifications to define standardized profiles of this part of ISO/IEC 7816.

# Annex A
# (normative)

# ASN.1 module

This section includes all ASN.1 type, value and information object class definitions contained in this part of ISO/IEC 7816, in the form of the ASN.1 module CryptographicInformationFramework.

```
CryptographicInformationFramework {iso(1) standard(0) 7816 15}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

informationFramework, authenticationFramework, certificateExtensions
        FROM UsefulDefinitions {joint-iso-itu-t(2) ds(5) module(1) usefulDefinitions(0) 3}

Name
        FROM InformationFramework informationFramework

Certificate, AttributeCertificate, CertificateSerialNumber, SubjectPublicKeyInfo, AlgorithmIdentifier
        FROM AuthenticationFramework authenticationFramework

GeneralName, GeneralNames, KeyUsage
        FROM CertificateExtensions certificateExtensions

ECPoint, Parameters
        FROM ANSI-X9-62 {iso(1) member-body(2) us(840) ansi-x962(10045) module(4) 1}

DomainParameters
        FROM ANSI-X9-42 {iso(1) member-body(2) us(840) ansi-x942(10046) module(5) 1};

-- A.1 Upper and lower bounds

cti-ub-identifier                   INTEGER ::= 255
cti-ub-reference                    INTEGER ::= 255
cti-ub-index                        INTEGER ::= 65535
cti-ub-label                        INTEGER ::= cti-ub-identifier
cti-lb-minPasswordLength    INTEGER ::= 4
cti-ub-minPasswordLength    INTEGER ::= 8
cti-ub-storedPasswordLength         INTEGER ::= 64
cti-ub-recordLength         INTEGER ::= 16383
cti-ub-userConsent          INTEGER ::= 15
cti-ub-securityConditions   INTEGER ::= 255
cti-ub-biometricTypes       INTEGER ::= 127

-- A.2 Basic types

-- A.2.1

Identifier ::= OCTET STRING (SIZE (0..cti-ub-identifier))

-- A.2.2

Reference ::= INTEGER (0..cti-ub-reference)

-- A.2.3

Label ::= UTF8String (SIZE(0..cti-ub-label))

-- A.2.4

CredentialIdentifier {KEY-IDENTIFIER : IdentifierSet} ::= SEQUENCE {
        idType  KEY-IDENTIFIER.&id ({IdentifierSet}),
        idValue KEY-IDENTIFIER.&Value ({IdentifierSet}{@idType})
        }

KeyIdentifiers KEY-IDENTIFIER ::= {
        issuerAndSerialNumber       |
        issuerAndSerialNumberHash   |
        subjectKeyId                        |
```

```
        subjectKeyHash                  |
        issuerKeyHash                           |
        issuerNameHash                  |
        subjectNameHash,
        ...
        }

KEY-IDENTIFIER ::= CLASS {
        &id  INTEGER UNIQUE,
        &Value
        } WITH SYNTAX {
        SYNTAX &Value IDENTIFIED BY &id
        }

IssuerAndSerialNumber ::= SEQUENCE {
        issuer Name,
        serialNumber CertificateSerialNumber
        }

issuerAndSerialNumber KEY-IDENTIFIER::=
        {SYNTAX IssuerAndSerialNumber IDENTIFIED BY 1}

subjectKeyId KEY-IDENTIFIER ::=
        {SYNTAX OCTET STRING IDENTIFIED BY 2}
        -- From ISO/IEC 9594-8:1998 certificate extension

issuerAndSerialNumberHash KEY-IDENTIFIER ::=
        {SYNTAX OCTET STRING IDENTIFIED BY 3}
        -- Assumes SHA-1 hash of DER encoding of IssuerAndSerialNumber

subjectKeyHash KEY-IDENTIFIER ::=
        {SYNTAX OCTET STRING IDENTIFIED BY 4}

issuerKeyHash KEY-IDENTIFIER ::=
        {SYNTAX OCTET STRING IDENTIFIED BY 5}

issuerNameHash KEY-IDENTIFIER ::=
        {SYNTAX OCTET STRING IDENTIFIED BY 6}
        -- SHA-1 hash of DER-encoded issuer name

subjectNameHash KEY-IDENTIFIER ::=
        {SYNTAX OCTET STRING IDENTIFIED BY 7}
        -- SHA-1 hash of DER-encoded subject name

-- A.2.5

ReferencedValue ::= CHOICE {
        path        Path,
        url         URL
        } -- The syntax of the object is determined by the context

URL ::= CHOICE {
        url         CHOICE {printable PrintableString, ia5 IA5String},
        urlWithDigest [3] SEQUENCE {
            url             IA5String,
            digest  DigestInfoWithDefault
            }
        }

alg-id-sha1 AlgorithmIdentifier ::= {
        algorithm   id-sha1,
        parameters      SHA1Parameters : NULL
        }

id-sha1 OBJECT IDENTIFIER ::= {iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2) 26 }

SHA1Parameters ::= NULL

DigestInfoWithDefault ::= SEQUENCE {
        digestAlg   AlgorithmIdentifier DEFAULT alg-id-sha1,
        digest              OCTET STRING (SIZE(8..128))

}
```

```
Path ::= SEQUENCE {
        efidOrPath OCTET STRING,
        index           INTEGER (0..cti-ub-index) OPTIONAL,
        length          [0] INTEGER (0..cti-ub-index) OPTIONAL
        }( WITH COMPONENTS {..., index PRESENT, length PRESENT}|
           WITH COMPONENTS {..., index ABSENT, length ABSENT})

-- A.2.6

ObjectValue { Type } ::= CHOICE {
        indirect    ReferencedValue,
        direct      [0] Type
        }

-- A.2.7

PathOrObjects {ObjectType} ::= CHOICE {
        path        Path,
        objects     [0] SEQUENCE OF ObjectType,
        ... -- For future extensions
        }

-- A.2.8

CommonObjectAttributes ::= SEQUENCE {
        label                   Label OPTIONAL,
        flags                   CommonObjectFlags OPTIONAL,
        authId                  Identifier OPTIONAL,
        userConsent             INTEGER (1..cti-ub-userConsent) OPTIONAL,
        accessControlRules      SEQUENCE SIZE (1..MAX) OF AccessControlRule OPTIONAL,
        ... -- For future extensions
        } (CONSTRAINED BY {-- authId should be present if flags.private is set.
        -- It shall equal an authID in one authentication object in the AOD -- })

CommonObjectFlags ::= BIT STRING {
        private         (0),
        modifiable  (1),
        internal        (2)
        } -- Bit (2) is present for historical reasons and shall not be used

AccessControlRule ::= SEQUENCE {
        accessMode          AccessMode,
        securityCondition  SecurityCondition,
        ... -- For future extensions
        }

AccessMode ::= BIT STRING {
        read        (0),
        update      (1),
        execute     (2),
        delete      (3)
        }

SecurityCondition ::= CHOICE {
        always                  NULL,
        authId                  Identifier,
        authReference           AuthReference,
        not                     [0] SecurityCondition,
        and             [1] SEQUENCE SIZE (2..cti-ub-securityConditions) OF SecurityCondition,
        or                      [2] SEQUENCE SIZE (2..cti-ub-securityConditions) OF SecurityCondition,
        ... -- For future extensions
        }

AuthReference ::= SEQUENCE {
        authMethod      AuthMethod,
        seldentifier INTEGER OPTIONAL
        }

AuthMethod ::= BIT STRING {secureMessaging(0), extAuthentication(1), userAuthentication(2)}

-- A.2.9

CommonKeyAttributes ::= SEQUENCE {
```

```
        iD                  Identifier,
        usage               KeyUsageFlags,
        native              BOOLEAN DEFAULT TRUE,
        accessFlags         KeyAccessFlags OPTIONAL,
        keyReference        KeyReference OPTIONAL,
        startDate   GeneralizedTime OPTIONAL,
        endDate     [0] GeneralizedTime OPTIONAL,
        algReference        [1] SEQUENCE OF Reference OPTIONAL,
        ... -- For future extensions
        }

KeyUsageFlags ::= BIT STRING {
        encipher                (0),
        decipher                (1),
        sign                    (2),
        signRecover             (3),
        keyEncipher             (4),
        keyDecipher             (5),
        verify                  (6),
        verifyRecover           (7),
        derive                  (8),
        nonRepudiation  (9)
        }

KeyAccessFlags ::= BIT STRING {
        sensitive       (0),
        extractable             (1),
        alwaysSensitive (2),
        neverExtractable (3),
        cardGenerated           (4)
        }

KeyReference ::= INTEGER

-- A.2.10

CommonPrivateKeyAttributes ::= SEQUENCE {
        name                Name OPTIONAL,
        keyIdentifiers      [0] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        generalName         [1] GeneralNames OPTIONAL,
        ... -- For future extensions
        }

-- A.2.11

CommonPublicKeyAttributes ::= SEQUENCE {
        name                Name OPTIONAL,
        trustedUsage        [0] Usage OPTIONAL,
        generalName         [1] GeneralNames OPTIONAL,
        keyIdentifiers      [2] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        ... -- For future extensions
}

-- A.2.12

CommonSecretKeyAttributes ::= SEQUENCE {
        keyLen      INTEGER OPTIONAL, -- keylength (in bits)
        ... -- For future extensions
        }

-- A.2.13

GenericKeyAttributes ::= SEQUENCE {
        keyType     CIO-ALGORITHM.&objectIdentifier ({AllowedAlgorithms}),
        keyAttr     CIO-ALGORITHM.&Parameters ({AllowedAlgorithms}{@keyType})
        }

AllowedAlgorithms CIO-ALGORITHM ::= {...}

-- A.2.14

KeyInfo {ParameterType, OperationsType} ::= CHOICE {
        paramsAndOps    SEQUENCE {
```

```
            parameters      ParameterType,
            operations      OperationsType OPTIONAL
            },
        reference           Reference -- Historical, not to be used
        }

-- A.2.15

CommonCertificateAttributes ::= SEQUENCE {
        iD                  Identifier,
        authority   BOOLEAN DEFAULT FALSE,
        identifier          CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        certHash    [0] CertHash OPTIONAL,
        trustedUsage        [1] Usage OPTIONAL,
        identifiers  [2] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        ...
        } -- Context tag [3] is reserved for historical reasons

Usage ::= SEQUENCE {
        keyUsage KeyUsage OPTIONAL,
        extKeyUsage SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER OPTIONAL,
        ...
        } (WITH COMPONENTS {..., keyUsage PRESENT} | WITH COMPONENTS {..., extKeyUsage PRESENT})

CertHash ::= SEQUENCE {
        hashAlg     [0] EXPLICIT AlgorithmIdentifier OPTIONAL,
        certId      [1] EXPLICIT CertId OPTIONAL,
        hashVal     BIT STRING
        }(CONSTRAINED BY {-- hashVal is calculated over the whole DER-encoded certificate --})

CertId ::= SEQUENCE {
        issuer              GeneralName,
        serialNumber        CertificateSerialNumber
        }

-- A.2.16

GenericCertificateAttributes ::= SEQUENCE {
        certType    CIO-OPAQUE.&id ({AllowedCertificates}),
        certAttr    CIO-OPAQUE.&Type ({AllowedCertificates}{@certType})
        }

AllowedCertificates CIO-OPAQUE ::= {...}

-- A.2.17

CommonDataContainerObjectAttributes ::= SEQUENCE {
        applicationName   Label OPTIONAL,
        applicationOID          OBJECT IDENTIFIER OPTIONAL,
        iD                      Identifier OPTIONAL,
        ... -- For future extensions
        } (WITH COMPONENTS {..., applicationName PRESENT}| WITH COMPONENTS {..., applicationOID PRESENT})

-- A.2.18

CommonAuthenticationObjectAttributes ::= SEQUENCE {
        authId              Identifier OPTIONAL,
        authReference       Reference OPTIONAL,
        seldentifier [0] Reference OPTIONAL,
        ... -- For future extensions
        }

-- A.2.19

CIO {ClassAttributes, SubClassAttributes, TypeAttributes} ::= SEQUENCE {
        commonObjectAttributes  CommonObjectAttributes,
        classAttributes             ClassAttributes,
        subClassAttributes          [0] SubClassAttributes OPTIONAL,
        typeAttributes              [1] TypeAttributes
        }

-- A.3 CIOs

CIOChoice ::= CHOICE {
```

```
        privateKeys                 [0] PrivateKeys,
        publicKeys                  [1] PublicKeys,
        trustedPublicKeys           [2] PublicKeys,
        secretKeys                  [3] SecretKeys,
        certificates            [4] Certificates,
        trustedCertificates         [5] Certificates,
        usefulCertificates      [6] Certificates,
        dataContainerObjects    [7] DataContainerObjects,
        authObjects                 [8] AuthObjects,
        ... -- For future extensions
        }

PrivateKeys ::=   PathOrObjects {PrivateKeyChoice}

SecretKeys  ::=   PathOrObjects {SecretKeyChoice}

PublicKeys  ::=   PathOrObjects {PublicKeyChoice}

Certificates ::=  PathOrObjects {CertificateChoice}

DataContainerObjects ::= PathOrObjects {DataContainerObjectChoice}

AuthObjects ::=   PathOrObjects {AuthenticationObjectChoice}

-- A.4 Private key information objects

-- A.4.1

PrivateKeyChoice ::= CHOICE {
        privateRSAKey           PrivateKeyObject {PrivateRSAKeyAttributes},
        privateECKey            [0] PrivateKeyObject {PrivateECKeyAttributes},
        privateDHKey            [1] PrivateKeyObject {PrivateDHKeyAttributes},
        privateDSAKey           [2] PrivateKeyObject {PrivateDSAKeyAttributes},
        privateKEAKey           [3] PrivateKeyObject {PrivateKEAKeyAttributes},
        genericPrivateKey [4] PrivateKeyObject {GenericKeyAttributes},
        ... -- For future extensions
        }

PrivateKeyObject {KeyAttributes} ::= CIO {
        CommonKeyAttributes, CommonPrivateKeyAttributes, KeyAttributes}

-- A.4.2

PrivateRSAKeyAttributes ::= SEQUENCE {
        value                   ObjectValue {CIO-OPAQUE.&Type},
        modulusLength   INTEGER, -- modulus length in bits, e.g. 1024
        keyInfo                 KeyInfo {NULL, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.3

PrivateECKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {Parameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.4

PrivateDHKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.5

PrivateDSAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.6
```

```
PrivateKEAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {CIO-OPAQUE.&Type},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.5 Public key information objects

-- A.5.1

PublicKeyChoice ::= CHOICE {
        publicRSAKey            PublicKeyObject {PublicRSAKeyAttributes},
        publicECKey             [0] PublicKeyObject {PublicECKeyAttributes},
        publicDHKey             [1] PublicKeyObject {PublicDHKeyAttributes},
        publicDSAKey            [2] PublicKeyObject {PublicDSAKeyAttributes},
        publicKEAKey            [3] PublicKeyObject {PublicKEAKeyAttributes},
        genericPublicKey  [4] PublicKeyObject{GenericKeyAttributes},
        ... -- For future extensions
        }

PublicKeyObject {KeyAttributes} ::= CIO {
        CommonKeyAttributes, CommonPublicKeyAttributes, KeyAttributes}

-- A.5.2

PublicRSAKeyAttributes ::= SEQUENCE {
        value                   ObjectValue {RSAPublicKeyChoice},
        modulusLength   INTEGER, -- modulus length in bits, e.g. 1024
        keyInfo                 KeyInfo {NULL, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

RSAPublicKeyChoice ::= CHOICE {
        raw RSAPublicKey,
        spki[1] SubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public RSA key.
        ...
        }

RSAPublicKey ::= SEQUENCE {
        modulus                 INTEGER,
        publicExponent  INTEGER
        }

-- A.5.3

PublicECKeyAttributes ::= SEQUENCE {
        value       ObjectValue {ECPublicKeyChoice},
        keyInfo     KeyInfo {Parameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

ECPublicKeyChoice ::= CHOICE {
        raw ECPoint, -- See ANSI X9.62,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public elliptic curve key
        ...
        }

-- A.5.4

PublicDHKeyAttributes ::= SEQUENCE {
        value       ObjectValue {DHPublicKeyChoice},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

DHPublicKeyChoice ::= CHOICE {
        raw DHPublicNumber,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public D-H key.
        ...
        }

DHPublicNumber ::= INTEGER

-- A.5.5
```

```
PublicDSAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {DSAPublicKeyChoice},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

DSAPublicKeyChoice ::= CHOICE {
        raw DSAPublicKey,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public DSA key.
        ...
        }

DSAPublicKey ::= INTEGER

-- A.5.6

PublicKEAKeyAttributes ::= SEQUENCE {
        value       ObjectValue {KEAPublicKeyChoice},
        keyInfo     KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

KEAPublicKeyChoice ::= CHOICE {
        raw KEAPublicKey,
        spkiSubjectPublicKeyInfo, -- See ISO/IEC 9594-8:1998. Must contain a public KEA key.
        ...
        }

KEAPublicKey ::= INTEGER

-- A.6 Secret key information objects

-- A.6.1

SecretKeyChoice ::= CHOICE {
        algIndependentKey       SecretKeyObject {SecretKeyAttributes},
        genericSecretKey  [15] SecretKeyObject {GenericKeyAttributes},
        ... -- For future extensions
        } -- Note: Context tags [0] – [14] historical and not to be used

SecretKeyObject {KeyAttributes} ::= CIO {
         CommonKeyAttributes, CommonSecretKeyAttributes, KeyAttributes}

-- A.6.2

SecretKeyAttributes ::= SEQUENCE {
        value       ObjectValue { OCTET STRING },
        ... -- For future extensions
        }

-- A.7 Certificate information objects

-- A.7.1

CertificateChoice ::= CHOICE {
        x509Certificate             CertificateObject {X509CertificateAttributes},
        x509AttributeCertificate    [0] CertificateObject {X509AttributeCertificateAttributes},
        spkiCertificate                 [1] CertificateObject {SPKICertificateAttributes},
        pgpCertificate                  [2] CertificateObject {PGPCertificateAttributes},
        wtlsCertificate                 [3] CertificateObject {WTLSCertificateAttributes},
        x9-68Certificate            [4] CertificateObject {X9-68CertificateAttributes},
        cvCertificate                   [5] CertificateObject {CVCertificateAttributes},
        genericCertificateObject    [6] CertificateObject {GenericCertificateAttributes},
        ... -- For future extensions
        } -- Context tag 4 is reserved for forthcoming ANSI X9.68 certificates

CertificateObject {CertAttributes} ::= CIO {
        CommonCertificateAttributes, NULL, CertAttributes}

-- A.7.2

X509CertificateAttributes ::= SEQUENCE {
        value               ObjectValue { Certificate },
        subject             Name OPTIONAL,
```

```
        issuer              [0] Name OPTIONAL,
        serialNumber        CertificateSerialNumber OPTIONAL,
        ... -- For future extensions
        }

-- A.7.3

X509AttributeCertificateAttributes ::= SEQUENCE {
        value               ObjectValue { AttributeCertificate },
        issuer              GeneralNames OPTIONAL,
        serialNumber        CertificateSerialNumber OPTIONAL,
        attrTypes   [0] SEQUENCE OF OBJECT IDENTIFIER OPTIONAL,
        ... -- For future extensions
        }

-- A.7.4

SPKICertificateAttributes ::= SEQUENCE {
        value       ObjectValue { CIO-OPAQUE.&Type },
        ... -- For future extensions
        }

-- A.7.5

PGPCertificateAttributes ::= SEQUENCE {
        value       ObjectValue { CIO-OPAQUE.&Type },
        ... -- For future extensions
        }

-- A.7.6

WTLSCertificateAttributes ::= SEQUENCE {
        value       ObjectValue { CIO-OPAQUE.&Type },
        ... -- For future extensions
        }

-- A.7.7

X9-68CertificateAttributes ::= SEQUENCE {
        value       ObjectValue { CIO-OPAQUE.&Type },
        ... -- For future extensions
        }

-- A.7.8

CVCertificateAttributes ::= SEQUENCE {
        value       ObjectValue { CIO-OPAQUE.&Type},
        ... -- For future extensions
        }

-- A.8 Data container information objects

-- A.8.1

DataContainerObjectChoice ::= CHOICE {
        opaqueDO DataContainerObject {OpaqueDOAttributes},
        iso7816DO [0] DataContainerObject {ISO7816DOAttributes},
        oidDO               [1] DataContainerObject {OidDOAttributes},
        ... -- For future extensions
        }

DataContainerObject {DataObjectAttributes} ::= CIO {
        CommonDataContainerObjectAttributes, NULL, DataObjectAttributes}

-- A.8.2

OpaqueDOAttributes ::= ObjectValue {CIO-OPAQUE.&Type}

-- A.8.3

ISO7816DOAttributes ::= ObjectValue {CIO-OPAQUE.&Type}
    (CONSTRAINED BY {-- All such data container objects shall be defined in accordance with ISO/IEC 7816-6 --})

-- A.8.4

OidDOAttributes ::= SEQUENCE {
```

```
        id          OBJECT IDENTIFIER,
        value       ObjectValue {CIO-OPAQUE.&Type}
        }

-- A.9 Authentication information objects

-- A.9.1

AuthenticationObjectChoice ::= CHOICE {
        pwd                     AuthenticationObject { PasswordAttributes },
        biometricTemplate       [0] AuthenticationObject{ BiometricAttributes},
        authKey                 [1] AuthenticationObject {AuthKeyAttributes},
        external                [2] AuthenticationObject {ExternalAuthObjectAttributes},
        ... -- For future extensions
        }

AuthenticationObject {AuthObjectAttributes} ::= CIO {
        CommonAuthenticationObjectAttributes, NULL, AuthObjectAttributes}

-- A.9.2

PasswordAttributes ::= SEQUENCE {
        pwdFlags   PasswordFlags,
        pwdType    PasswordType,
        minLength INTEGER (cti-lb-minPasswordLength..cti-ub-minPasswordLength),
        storedLength    INTEGER (0..cti-ub-storedPasswordLength),
        maxLength       INTEGER OPTIONAL,
        pwdReference    [0] Reference DEFAULT 0,
        padChar    OCTET STRING (SIZE(1)) OPTIONAL,
        lastPasswordChange      GeneralizedTime OPTIONAL,
        path            Path OPTIONAL,
        ... -- For future extensions
        }

PasswordFlags ::= BIT STRING {
        case-sensitive                  (0),
        local                           (1),
        change-disabled         (2),
        unblock-disabled        (3),
        initialized             (4),
        needs-padding           (5),
        unblockingPassword      (6),
        soPassword                      (7),
        disable-allowed         (8),
        integrity-protected     (9),
        confidentiality-protected (10),
        exchangeRefData         (11)
        } (CONSTRAINED BY { -- 'unblockingPassword' and 'soPassword' cannot both be set -- })

PasswordType ::= ENUMERATED {bcd, ascii-numeric, utf8, ..., half-nibble-bcd, iso9564-1}

-- A.9.3

BiometricAttributes ::= SEQUENCE {
        bioFlags        BiometricFlags,
        templateId BiometricTemplateIdentifier,
        bioType         BiometricType,
        bioReference    Reference DEFAULT 0,
        lastChangeGeneralizedTime OPTIONAL,
        path            Path OPTIONAL,
        ... -- For future extensions
        }

BiometricTemplateIdentifier ::= CHOICE {
        oid         OBJECT IDENTIFIER,
        issuerId    OCTET STRING,
        ... -- For future extensions
        }

BiometricFlags ::= BIT STRING {
        local                           (1),
        change-disabled         (2),
```

```
        unblock-disabled        (3),
        initialized             (4),
        disable-allowed         (8),
        integrity-protected     (9),
        confidentiality-protected (10)
        }

BiometricType ::= CHOICE {
        fingerPrint FingerPrintInformation,
        iris            [0] IrisInformation,
        combined  [1] SEQUENCE SIZE (2..cti-ub-biometricTypes) OF BiometricType,
        ... -- For future extensions
        }

FingerPrintInformation ::= SEQUENCE {
        hand    ENUMERATED {left, right},
        finger  ENUMERATED {thumb, pointerFinger, middleFinger, ringFinger, littleFinger}
        }

IrisInformation ::= SEQUENCE {
        eye ENUMERATED {left, right},
        ... -- For future extensions
        }

-- A.9.4

ExternalAuthObjectAttributes ::= CHOICE {
        authKeyAttributes AuthKeyAttributes,
        certBasedAttributes     [0] CertBasedAuthenticationAttributes,
        ... -- For future extensions
        }

AuthKeyAttributes ::= SEQUENCE {
        derivedKey BOOLEAN DEFAULT TRUE,
        authKeyId  Identifier,
        ... -- For future extensions
        }

CertBasedAuthenticationAttributes ::= SEQUENCE {
        cha OCTET STRING,
        ... -- For future extensions
        }

-- A.10 Cryptographic and card information

CardInfo ::= SEQUENCE {
        version                 INTEGER {v1(0),v2(1)} (v1|v2,...),
        serialNumber            OCTET STRING OPTIONAL,
        manufacturerID      Label OPTIONAL,
        label                   [0] Label OPTIONAL,
        cardflags           CardFlags,
        seInfo                  SEQUENCE OF SecurityEnvironmentInfo OPTIONAL,
        recordInfo          [1] RecordInfo OPTIONAL,
        supportedAlgorithms     [2] SEQUENCE OF AlgorithmInfo OPTIONAL,
        issuerId                [3] Label OPTIONAL,
        holderId                [4] Label OPTIONAL,
        lastUpdate          [5] LastUpdate OPTIONAL,
        preferredLanguage       PrintableString OPTIONAL, -- In accordance with IETF RFC 1766
        profileIndication   [6] SEQUENCE OF ProfileIndication OPTIONAL,
        ...
        } (CONSTRAINED BY { -- Each AlgorithmInfo.reference value shall be unique --})

CardFlags ::= BIT STRING {
        readonly        (0),
        authRequired    (1),
        prnGeneration   (2)
        } -- Bit (3) is reserved for historical reasons

SecurityEnvironmentInfo ::= SEQUENCE {
        se      INTEGER,
        owner   OBJECT IDENTIFIER OPTIONAL,
```

```
        aid             OCTET STRING
                        (CONSTRAINED BY {-- Must be encoded in accordance with ISO/IEC 7816-5 --}) OPTIONAL,
        ... -- For future extensions
        }

RecordInfo ::= SEQUENCE {
        oDRecordLength   [0] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        prKDRecordLength        [1] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        puKDRecordLength        [2] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        sKDRecordLength         [3] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        cDRecordLength   [4] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        dCODRecordLength        [5] INTEGER (0..cti-ub-recordLength) OPTIONAL,
        aODRecordLength         [6] INTEGER (0..cti-ub-recordLength) OPTIONAL
        }

AlgorithmInfo ::= SEQUENCE {
        reference               Reference,
        algorithm               CIO-ALGORITHM.&id({AlgorithmSet}),
        parameters                      CIO-ALGORITHM.&Parameters({AlgorithmSet}{@algorithm}),
        supportedOperations     CIO-ALGORITHM.&Operations({AlgorithmSet}{@algorithm}),
        objId                           CIO-ALGORITHM.&objectIdentifier ({AlgorithmSet}{@algorithm}),
        algRef                          Reference OPTIONAL
        }

CIO-ALGORITHM ::= CLASS {
        &id INTEGER UNIQUE,
        &Parameters,
        &Operations Operations,
        &objectIdentifier OBJECT IDENTIFIER OPTIONAL
      } WITH SYNTAX {
        PARAMETERS &Parameters OPERATIONS &Operations ID &id [OID &objectIdentifier]
        }

CIO-OPAQUE ::= TYPE-IDENTIFIER

PublicKeyOperations ::= Operations

Operations ::= BIT STRING {
        compute-checksum        (0), -- H/W computation of checksum
        compute-signature       (1), -- H/W computation of signature
        verify-checksum   (2), -- H/W verification of checksum
        verify-signature  (3), -- H/W verification of signature
        encipher          (4), -- H/W encryption of data
        decipher          (5), -- H/W decryption of data
        hash                    (6), -- H/W hashing
        generate-key            (7)  -- H/W key generation
        }

cti-alg-null  CIO-ALGORITHM ::= {
        PARAMETERS NULL OPERATIONS {{generate-key}} ID -1}

AlgorithmSet CIO-ALGORITHM ::= {
        cti-alg-null,
        ... -- See PKCS #11 (Annex E) for possible values for the &id field (and parameters)
        }

LastUpdate ::= CHOICE {
        generalizedTime   GeneralizedTime,
        referencedTime    ReferencedValue ,
        ... -- For future extensions
        }(CONSTRAINED BY {-- The value for referencedTime shall be of type GeneralizedTime --})

ProfileIndication ::= CHOICE {
        profileOID  OBJECT IDENTIFIER,
        profileName       UTF8String,
        ... -- For future extensions
        }

-- A.11 CIO DDO

CIODDO ::= SEQUENCE {
      providerId              OBJECT IDENTIFIER OPTIONAL,
```

```
        odfPath                 Path OPTIONAL,
        cardInfoPath    [0] Path OPTIONAL,
        aid                     [APPLICATION 15] OCTET STRING
                                (CONSTRAINED BY {-- Must be an AID in accordance with ISO/IEC 7816-5:1994--}) OPTIONAL,
        ... -- For future extensions
        } -- Context tag 1 is historical and shall not be used

END
```

# Annex B
# (informative)

# CIA example for cards with digital signature and authentication functionality

## B.1 Introduction

This section describes an example of the CIA suitable for electronic identification purposes and requirements for it. The example includes requirements both for cards and for host-side applications making use of cards.

## B.2 CIOs

— Private Keys: A CIO card should contain at least two private keys, of which one should be used for digital signature purposes only (key usage flags: any combination of sign, signRecover, and nonRepudiation). At least one of the other keys should be possible to use for client/server authentication and have the value sign and/or decipher set in its key usage flags. Authentication CDEs or encipherment shall protect all private keys. Usage of the signature-only key should require cardholder verification with an authentication CDE used only for this key. The key length shall be sufficient for intended purposes.

  Private key types for this example are: RSA keys, Elliptic Curve keys (this example places no restrictions on the domain parameters other than the ones mentioned above); and DSA keys.

— Secret Keys: CDEs of this type may or may not be present on the card, depending on the application issuer's discretion. There is no requirement for host-side applications to handle these keys.

— Public Keys: CDEs of this type may or may not be present on the card, depending on the application issuer's discretion. There is no requirement for host-side applications to handle these keys.

— Certificates: For each private key at least one corresponding certificate should be stored in the card. The certificates shall be of type X509Certificate. If an application issuer stores CA certificates on a card which supports the ISO/IEC 7816-4:1995 logical file organization, and which has suitable file access mechanisms, then it is recommended that they are stored in a protected file. This file shall be pointed to by a CD file which is only modifiable by the card issuer (or not modifiable at all). This implies usage of the trustedCertificates choice in the CIOChoice type.

— Data container objects: No requirements. CDEs of this type may or may not be present on the card, depending on the application issuer's discretion.

— Authentication objects: At least one authentication CDE shall be present on the card, controlling access to protected CDEs. A separate authentication CDE should be used for the signature-only key, if such a key exist. Any use of the signature-only private key should require a new user authentication. In the case of passwords, any positive verification of one password shall not enable the use of security services associated with another password.

  Passwords shall be at least 4 characters (BCD, UTF-8 or ASCII) long.

  When a password is blocked after consecutive incorrect password verifications, the password may only be unblocked through a resetting code or a special unblocking procedure, defined by the card issuer.

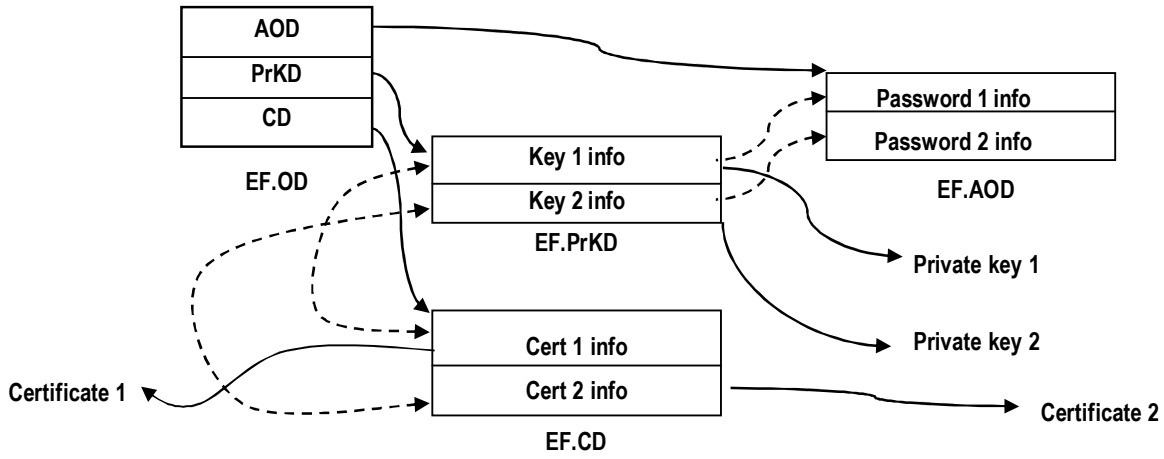



Figure B.1 – File relationships in DF.CIA. Dashed arrows indicate cross-references.

## B.3 Access control

Private keys shall be private objects, and should be marked as sensitive. Files, which contain private keys, should be protected against removal and/or overwriting. The following access conditions shall be set for DF.CIA and elementary files in it.

Table B.1 – Recommended file access conditions

| File | Access Conditions |
|---|---|
| DF.CIA | Create: User authentication or External authentication<br>Delete: External authentication |
| EF.CardInfo | Read: Always<br>Update: User authentication or External authentication or Never<br>Append: Never |
| EF.OD | Read: Always<br>Update: External authentication<br>Append: External authentication |
| EF.AOD | Read: Always<br>Update: Never<br>Append: User authentication or External authentication |
| EF.PrKD, EF.PuKD, EF.SKD, EF.CD and EF.DCOD | Read: Always or User authentication<br>Update: User authentication or External authentication or Never<br>Append: User authentication or External authentication or Never |
| EF.CD containing references to trusted certificates | Read: Always<br>Update: External authentication or Never<br>Append: External authentication or Never |
| Other EFs in DF.CIA | Read: Always or User authentication<br>Update: User authentication or External authentication or Never<br>Append: User authentication or External authentication or Never |
| Note 1 – External authentication is described in ISO/IEC 7816-4:1995<br>Note 2 – External authentication should include secure messaging as described in ISO/IEC 7816-4:1995 | |

NOTE – If an application issuer wants to protect a CIO directory file with an authentication object, then by default the first authentication object in EF.AOD shall be used. Obviously, EF.OD and EF.AOD cannot be protected in this manner.

# Annex C
(informative)

## Example topologies

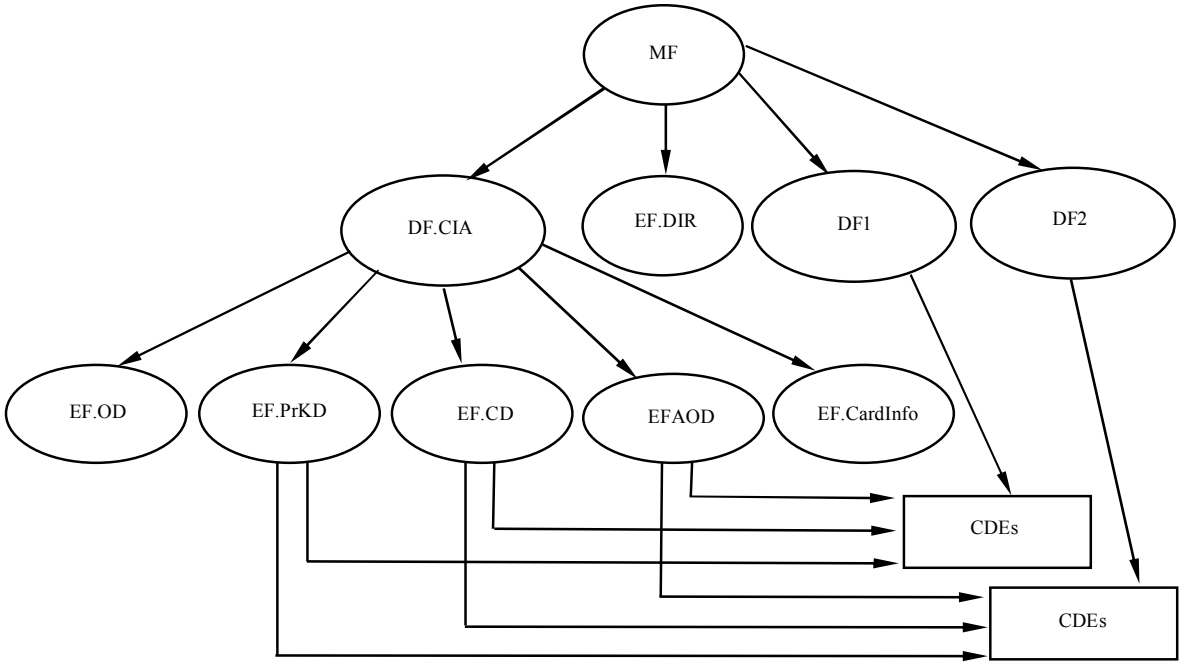



**Figure C.1 – Example with three applications. Cryptographic data elements are stored outside the CIA**

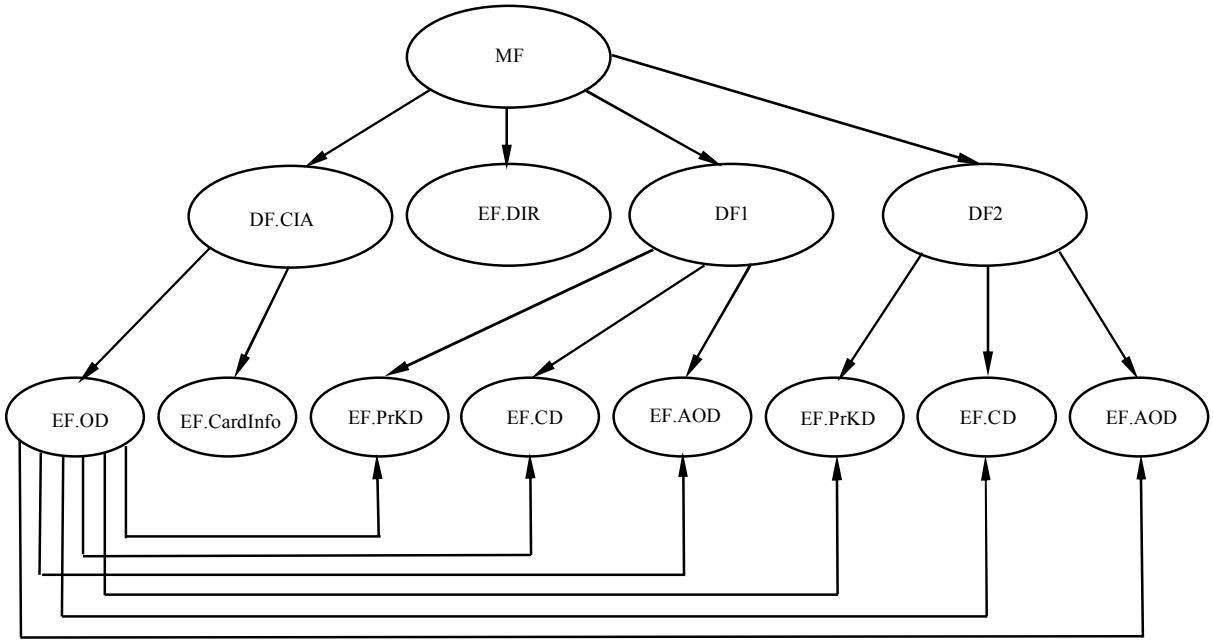



**Figure C.2 – Example with three applications. Only EF.OD and EF.CardInfo in DF.CIA**

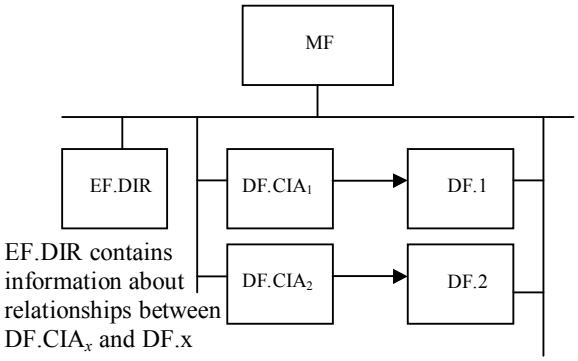



**Figure C.3 – Example of usage of EF.DIR**

# Annex D
# (informative)

# Examples of CIO values and their encodings

## D.1 Introduction

Each sub-clause of this annex, e.g., D.x refers to one of the EFs defined in the card file structure. Each sub-clause is in turn comprised of three sections:

— The first one, e.g. D.x.1, gives the names and sample values of DEs relevant to a DER construction in the ASN.1 value notation defined in ISO/IEC 8824-1:1998.

— The second one, e.g. D.x.2, merges the ASN.1 syntax with the indication constructed/primitive, the sample values and length thereof.

— The third one, e.g. D.x.3, gives the actual hexadecimal DER-coding as read from the file.

The three sections display the values in different formats, in order to show the various ways in which they could appear in specifications based on this standard.

In the first section, simple quotes denote a hexadecimal string, the usual 7816 notation. They are followed by 'H', which is also a common notation, e.g. `'0202'H`. Double quotes denote UTF-8 strings (printable) e.g. `"CIA application"`. Braces denote an object identifier, e.g. {1 2 840 113549 1 15 4 1}, see for instance ISO/IEC 7816-6:1995, Annex B for transcoding into a hexadecimal string.

In the second section, the prefix `0x` denotes a hexadecimal string, a usual programming notation, e.g. `0x3f005015` is equivalent to `'3F005015'` in the usual 7816 notation. A number without this prefix means it is a decimal, e.g. a value of `12` is equivalent to its hexadecimal coding `0x0c` or `'0C'`. Tags are indicated between brackets []. The tag number is given in decimal. The tag class is indicated in the bracket, except for the contextual class, which is the default. The information primitive/constructed gives the value of b6 in the actual tag used in the third section. Each level of indentation indicates a level of encapsulation.

In the third section, hexadecimal coded bytes are separated by spaces, and no quotes are used. Each level of indentation starts with a DO header (Tag-length), except the last level, which is the value of a primitive DO: the indentation rules are the same as in the second section. Thus, all headers of DOs belonging to the same template appear at the same level of indentation.

In order to improve understanding, line numbering is added in clause D.2.

## D.2 EF.OD

### D.2.1 ASN.1 value notation

```
1     privateKeys :
2           path : {
3                 efidOrPath '4401'H
                  },
4     certificates :
5           path : {
6                 efidOrPath '4402'H
                  },
7     dataContainerObjects :
8           path : {
9                 efidOrPath '4403'H
                  },
10    authObjects :
11          path : {
12                efidOrPath '4404'H
                  }
```

### D.2.2 ASN.1 description, tags, lengths and values

```
      CIOChoice CHOICE
1       privateKeys : tag = [0] constructed; length = 6
```

```
        PrivateKeys CHOICE
2         path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
3           efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
              0x4401
    CIOChoice CHOICE
4     certificates : tag = [4] constructed; length = 6
        Certificates CHOICE
5         path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
6           efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
              0x4402
    CIOChoice CHOICE
7     dataContainerObjects : tag = [7] constructed; length = 6
        DataContainerObjects CHOICE
8        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
9           efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
              0x4403
    CIOChoice CHOICE
10    authObjects : tag = [8] constructed; length = 6
        AuthObjects CHOICE
11        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
12          efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
              0x4404
```

### D.2.3 Hexadecimal DER-encoding

```
1   A0 06
2      30 04
3         04 02
             44 01
4   A4 06
5      30 04
6         04 02
             44 02
7   A7 06
8      30 04
9         04 02
             44 03
10  A8 06
11     30 04
12        04 02
             44 04
```

## D.3 EF.CardInfo

### D.3.1 ASN.1 value notation

```
cardInfoExample CardInfo ::= {
        version                 v2,
        serialNumber            '159752222515401240'H,
        manufacturerID    "Acme, Inc.",
        cardflags   {
              prnGeneration
              }
        }
```

### D.3.2 ASN.1 description, tags, lengths and values

```
CardInfo SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 30
  version INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
    1
```

```
  serialNumber OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 9
    0x159752222515401240
  manufacturerID Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 10
    0x41636d652c20496e632e
  cardflags CardFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
    0x0520
```

### D.3.3 Hexadecimal DER-encoding

```
30 1E
      02 01
            01
      04 09
            15 97 52 22 25 15 40 12 40
      0C 0A
            41 63 6D 65 2C 20 49 6E 63 2E
      03 02
            05 20
```

## D.4 EF.PrKD

**In this example, two private keys are referred to. Other key-related files, i.e. EF.PuKD and EF.SKD have the same structure. It is expected that the relevant public keys will also be referenced in EF.PuKD, with the same labels.**

### D.4.1 ASN.1 value notation

```
privateRSAKey : {
      commonObjectAttributes {
            label "KEY1",
            flags { private },
            authId '01'H
                  },
      classAttributes {
            iD '45'H,
            usage { decipher, sign, keyDecipher }
},
      subClassAttributes {
            keyIdentifiers {
                  {
                        idType 4,
                        idValue ParameterString : '4321567890ABCDEF'H
                        }
                  }
            },
      typeAttributes {
            value indirect :
                  path : {
                        efidOrPath '4B01'H
                        },
            modulusLength 1024
            }
      },
privateRSAKey : {
      commonObjectAttributes {
            label "KEY2",
            flags { private },
            authId '02'H
            },
      classAttributes {
            iD '46'H,
            usage { sign, nonRepudiation }
            },
      subClassAttributes {
            keyIdentifiers {
```

```
            {
                idType 4,
                idValue ParameterString : '1234567890ABCDEF'H
                }
            }
            },
        typeAttributes {
            value indirect :
                path : {
                    efidOrPath '4B02'H
                    },
            modulusLength 1024
            }
        }
```

### D.4.2 ASN.1 description, tags, lengths and values

```
PrivateKeyChoice CHOICE
  privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 59
    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
    constructed; length = 13
      label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 4
        0x4b455931
      flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
        0x0780
      authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x01
    classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
    structed; length = 7
      iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x45
      usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 2
        0x0264
    subClassAttributes : tag = [0] constructed; length = 19
      CommonPrivateKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
      length = 17
        keyIdentifiers SEQUENCE OF: tag = [0] constructed; length = 15
          SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 13
            idType INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              4
            idValue OpenType
              0x4321567890abcdef
    typeAttributes : tag = [1] constructed; length = 12
      PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
      length = 10
        value CHOICE
          indirect ReferencedValue CHOICE
            path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
              efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
                0x4b01
        modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
          1024
PrivateKeyChoice CHOICE
  privateRSAKey SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 60
    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
    constructed; length = 13
      label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 4
        0x4b455932
      flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
        0x0780
      authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x02
    classAttributes CommonKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] con-
```

```
      structed; length = 8
        iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x46
        usage KeyUsageFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length = 3
          0x062040
      subClassAttributes : tag = [0] constructed; length = 19
        CommonPrivateKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 17
          keyIdentifiers SEQUENCE OF: tag = [0] constructed; length = 15
            SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 13
              idType INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
                4
              idValue OpenType
                0x1234567890abcdef
      typeAttributes : tag = [1] constructed; length = 12
        PrivateRSAKeyAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 10
          value CHOICE
            indirect ReferencedValue CHOICE
              path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
                efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
                  0x4b02
          modulusLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 2
            1024
```

### D.4.3 Hexadecimal DER-encoding

```
30 3B
     30 0D
          0C 04
               4B 45 59 31
          03 02
               07 80
          04 01
               01
     30 07
          04 01
               45
          03 02
               02 64
     A0 13
          30 11
               A0 0F
                    30 0D
                         02 01
                              04
                         04 08
                              43 21 56 78 90 AB CD EF
     A1 0C
          30 0A
               30 04
                    04 02
                         4B 01
               02 02
                    04 00
30 3C
     30 0D
          0C 04
               4B 45 59 32
          03 02
               07 80
```

```
      04 01
           02
   30 08
      04 01
           46
      03 03
           06 20 40
   A0 13
      30 11
           A0 0F
                30 0D
                     02 01
                          04
                     04 08
                          12 34 56 78 90 AB CD EF
   A1 0C
      30 0A
           30 04
                04 02
                     4B 02
                     02 02
                          04 00
```

## D.5 EF. CD

### D.5.1 ASN.1 value notation

```
x509Certificate : {
        commonObjectAttributes {
                label "CERT1",
                flags { }
                },
        classAttributes {
                iD '45'H
                },
        typeAttributes {
                value indirect :
                        path : {
                                efidOrPath '4331'H
                                }
                }
        },
x509Certificate : {
        commonObjectAttributes {
                label "CERT2",
                flags { }
                },
        classAttributes {
                iD '46'H
          },
typeAttributes {
                value indirect :
                        path : {
                                efidOrPath '4332'H
                                }
                }
        }
```

### D.5.2 ASN.1 description, tags, lengths and values

```
  CertificateChoice CHOICE
    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 27
```

```
      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 10
        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 5
          0x4345525431
        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=1
          0x00
      classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 3
        iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x45
      typeAttributes : tag = [1] constructed; length = 8
        X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 6
          value CHOICE
            indirect ReferencedValue CHOICE
              path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
                efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
                  0x4331
  CertificateChoice CHOICE
    x509Certificate SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 27
      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 10
        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 5
          0x4345525432
        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=1
          0x00
      classAttributes CommonCertificateAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 3
        iD Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x46
      typeAttributes : tag = [1] constructed; length = 8
        X509CertificateAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed;
        length = 6
          value CHOICE
            indirect ReferencedValue CHOICE
              path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 4
                efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
                  0x4332
```

### D.5.3 Hexadecimal DER-encoding

```
30 1B
     30 0A
          0C 05
               43 45 52 54 31
          03 01
               00
     30 03
          04 01
               45
     A1 08
          30 06
               30 04
                    04 02
                         43 31
30 1B
     30 0A
          0C 05
               43 45 52 54 32
          03 01
               00
```

```
30 03
     04 01
          46
A1 08
     30 06
          30 04
               04 02 43 32
```

## D.6 EF.AOD

### D.6.1 ASN.1 value notation

```
pwd : {
    commonObjectAttributes {
        label "PIN1",
        flags { private }
        },
    classAttributes {
        authId '01'H
        },
    typeAttributes {
        pwdFlags { change-disabled, initialized, needs-padding },
        pwdType bcd,
        minLength 4,
        storedLength 8,
        padChar 'FF'H
        }
    },
pwd : {
    commonObjectAttributes {
        label "PIN2",
        flags { private }
        },
    classAttributes {
        authId '02'H
        },
    typeAttributes {
        pwdFlags { change-disabled, initialized, needs-padding },
        pwdType bcd,
        minLength 4,
        storedLength 8,
        padChar 'FF'H,
        path {
            efidOrPath '3F0050150100'H
            }
        }
    }
```

### D.6.2 ASN.1 description, tags, lengths and values

```
AuthenticationObjectChoice CHOICE
  pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 37
    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
    constructed; length = 10
      label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 4
        0x50494e31
      flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
        0x0780
    classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
    tag = [UNIVERSAL 16] constructed; length = 3
      authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x01
    typeAttributes : tag = [1] constructed; length = 18
      PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 16
```

```
        pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
        length = 2
          0x022c
        pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
          0
        minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          4
        storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          8
        padChar OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0xff
AuthenticationObjectChoice CHOICE
  pwd SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 47
    commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
    constructed; length = 10
      label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 4
        0x50494e32
      flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
        0x0780
    classAttributes CommonAuthenticationObjectAttributes SEQUENCE:
    tag = [UNIVERSAL 16] constructed; length = 3
      authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
        0x02
    typeAttributes : tag = [1] constructed; length = 28
      PasswordAttributes SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 26
        pwdFlags PasswordFlags BIT STRING: tag = [UNIVERSAL 3] primitive;
        length = 2
          0x022c
        pwdType PasswordType ENUMERATED: tag = [UNIVERSAL 10] primitive; length=1
          0
        minLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          4
        storedLength INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
          8
        padChar OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0xff
        path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 8
          efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 6
            0x3f0050150100
```

### D.6.3 Hexadecimal DER-encoding

```
30 25
     30 0A
          0C 04
               50 49 4E 31
          03 02
               07 80
     30 03
          04 01
               01
     A1 12
          30 10
               03 02
                    02 2C
               0A 01
                    00
               02 01
                    04
               02 01
                    08
```

```
               04 01
                    FF
30 2F
     30 0A
          0C 04
               50 49 4E 32
          03 02
               07 80
     30 03
          04 01
               02
     A1 1C
          30 1A
               03 02
                    02 2C
               0A 01
                    00
               02 01
                    04
               02 01
                    08
               04 01
                    FF
               30 08
                    04 06
                         3F 00 50 15 01 00
```

## D.7 EF.DCOD

### D.7.1 ASN.1 value notation

```
opaqueDO : {
      commonObjectAttributes {
            label "OBJECT1",
            flags { private, modifiable },
            authId '02'H
            },
      classAttributes {
            applicationName "APP"
            },
      typeAttributes indirect :
            path : {
                  efidOrPath '4431'H,
                  index 64,
                  length 48
                  }
      }
```

### D.7.2 ASN.1 description, tags, lengths and values

```
  DataContainerObjectChoice CHOICE
    opaqueDO SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 39
      commonObjectAttributes CommonObjectAttributes SEQUENCE: tag = [UNIVERSAL 16]
      constructed; length = 16
        label Label UTF8String: tag = [UNIVERSAL 12] primitive; length = 7
          0x4f424a45435431
        flags CommonObjectFlags BIT STRING: tag = [UNIVERSAL 3] primitive; length=2
          0x06c0
        authId Identifier OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 1
          0x02
```

```
    classAttributes CommonDataContainerObjectAttributes SEQUENCE:
    tag = [UNIVERSAL 16] constructed; length = 5
      applicationName Label UTF8String: tag = [UNIVERSAL 12] primitive; length= 3
        0x415050
    typeAttributes : tag = [1] constructed; length = 12
      OpaqueDOAttributes CHOICE
        indirect ReferencedValue CHOICE
          path Path SEQUENCE: tag = [UNIVERSAL 16] constructed; length = 10
            efidOrPath OCTET STRING: tag = [UNIVERSAL 4] primitive; length = 2
              0x4431
            index INTEGER: tag = [UNIVERSAL 2] primitive; length = 1
              64
            length INTEGER: tag = [0] primitive; length = 1
              48
```

### D.7.3 Hexadecimal DER-encoding of DCOD

```
30 27
     30 10
          0C 07
               4F 42 4A 45 43 54 31
          03 02
               06 C0
          04 01
               02
     30 05
          0C 03
               41 50 50
     A1 0C
          30 0A
               04 02
                    44 31
               02 01
                    40
               80 01
                    30
```

## D.8 Application Template (within the EF.DIR)

**In this example, only one application template IDO (i.e. an ApplicationTemplate) is shown.**

### D.8.1 ASN.1 value notation

```
applicationTemplateExample ApplicationTemplate ::= {
      aid     'A000000063504B43532D3135'H,
      label   "RSA DSI",
      path    '3F005015'H,
      ddo {
            providerId { 1 2 840 113549 1 15 4 1 }
            }
      }
```

### D.8.2 ASN.1 description, tags, lengths and values in ApplicationTemplate

```
  ApplicationTemplate SET: tag = [APPLICATION 1] constructed; length = 45
    aid OCTET STRING: tag = [APPLICATION 15] primitive; length = 12
      0xa000000063504b43532d3135
    label UTF8String: tag = [APPLICATION 16] primitive; length = 7
      0x52534120445349
    path OCTET STRING: tag = [APPLICATION 17] primitive; length = 4
      0x3f005015
    ddo : tag = [APPLICATION 19] constructed; length = 14
```

```
DDOTemplate OpenType
  providerId OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 10
  { 1 2 840 113549 1 15 4 1 }
```

### D.8.3 Hexadecimal DER-encoding of ApplicationTemplate

```
61 2D
      4F 0C
            A0 00 00 00 63 50 4B 43 53 2D 31 35
      50 07
            52 53 41 20 44 53 49
      51 04
            3F 00 50 15
      73 0E
            30 0C
                  06 0A
                        2A 86 48 86 F7 0D 01 0F 04 01
```

# Bibliography

H. Alvestrand, "Tags for the Identification of Languages," IETF RFC 1766, March 1995

ANSI X3.4-1986, Information Systems – Coded Character Sets – 7-Bit American National Standard Code for Information Interchange

ANSI X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography

ANSI X9.62-1998, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)

ANSI X9.68:2-2001, Digital Certificates for Mobile/Wireless and High Transaction Volume Financial Systems: Part 2: Domain Certificate Syntax

T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," IETF RFC 2396, August 1998

J. Callas, L. Donnerhacke, H. Finney, R. Thayer, "OpenPGP Message," IETF RFC 2440, November 1998

C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen, "SPKI Certificate Theory," IETF RFC 2693, September 1999

RSA Laboratories, PKCS #11 v2.11: Cryptographic Token Interface Standard

RSA Laboratories, PKCS #15 v1.1: Cryptographic Token Information Syntax Standard

WAP Forum, Wireless Application Protocol – Wireless Transport Layer Security Protocol Specification, Version 06-Apr-2001

| | | |
|---|---|---|
| **Signaturwert** | A9CVlmIGCdRqdNiyKUEaKMyD/vc/mVO/+T0CUkbKhpntOmoZlqU8/y0EwYeUp3juXtp0p7IeC7WHUOytiMvRjg== | |
|  | **Unterzeichner** | Andreas Gregor Fitzek |
| | **Aussteller-Zertifikat** | CN=a-sign-premium-mobile-03,OU=a-sign-premium-mobile-03,O=A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr GmbH,C=AT |
| | **Serien-Nr.** | 658475 |
| | **Methode** | urn:pdfsigfilter:bka.gv.at:binaer:v1.1.0 |
| | **Parameter** | etsi-bka-atrust-1.0:ecdsa-sha256:sha256:sha256:sha1 |
| **Prüfinformation** | Informationen zur Prüfung der elektronischen Signatur und des Ausdrucks finden Sie unter: http://www.signaturpruefung.gv.at | |
| **Datum/Zeit-UTC** | 2012-09-11T06:27:47Z | |