INTERNATIONAL STANDARD

ISO/IEC 7816-15

Second edition
2016-05-15

Identification cards — Integrated circuit cards —

Part 15: **Cryptographic information application**

Cartes d'identification — Cartes à circuit intégré à contacts —

Partie 15: Application des informations cryptographiques

Reference number
ISO/IEC 7816-15:2016(E)

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

This second edition cancels and replaces the first edition (ISO/IEC 7816-15:2004), which has been technically revised. It also incorporates the Amendments ISO/IEC 7816-15:2004/Amd. 1:2007 and ISO/IEC 7816-15:2004/Amd. 2:2008 and the Technical Corrigendum ISO/IEC 7816-15:2004/Cor. 1:2004.

ISO/IEC 7816 consists of the following parts, under the general title *Identification cards — Integrated circuit cards*:

— *Part 1: Cards with contacts — Physical characteristics*

— *Part 2: Cards with contacts — Dimensions and location of the contacts*

— *Part 3: Cards with contacts — Electrical interface and transmission protocols*

— *Part 4: Organization, security and commands for interchange*

— *Part 5: Registration of application providers*

— *Part 6: Interindustry data elements for interchange*

— *Part 7: Interindustry commands for Structured Card Query Language (SCQL)*

— *Part 8: Commands and mechanisms for security operations*

— *Part 9: Commands for card management*

— *Part 10: Electronic signals and answer to reset for synchronous cards*

— *Part 11: Personal verification through biometric methods*

— *Part 12: Cards with contacts — USB electrical interface and operating procedures*

— *Part 13: Commands for application management in a multi-application environment*

— *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 part of ISO/IEC 7816 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 Reference [9]). 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.

Identification cards — Integrated circuit cards — 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 other parts of ISO/IEC 7816 when appropriate;

— different authentication mechanisms;

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

This part of ISO/IEC 7816 does not cover the internal implementation within the card and/or the outside world. It is not 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 documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 9564-1, *Final services — Personal Identification Number (PIN) management and security — Part 1: Basic principles and requirements for PINs in card-based system*

ISO/IEC 7816 (all parts), *Identification cards — Integrated circuit cards with contacts*

ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-2, *Information technology — Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8824-3, *Information technology — Abstract Syntax Notation One (ASN.1): Constraint specification*

ISO/IEC 8824-4, *Information technology — Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*

ISO/IEC 8825-1, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 9594-8, *Information technology — Open Systems Interconnection — The Directory — Part 8: Public-key and attribute certificate frameworks*

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

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

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

3.2
application
data structures, data elements and program modules needed for performing a specific functionality

[SOURCE: ISO/IEC 7816-4:2013, 3.3, modified]

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

Note 1 to entry: Adapted from ISO/IEC 7816-4.

3.4
application provider
entity providing the components required for performing an application in the card

[SOURCE: ISO/IEC 7816-4:2013, 3.7, modified]

3.5
authentication information object
cryptographic information object that provides information about authentication related data

EXAMPLE 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

EXAMPLE A public key or a certificate.

3.15
data container information object
cryptographic information object that provides information about a data container

EXAMPLE A file.

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

3.17
dedicated file
structure containing file control information, and, optionally, memory available for allocation

[SOURCE: ISO/IEC 7816-4:2013, 3.19]

3.18
Directory
DIR file
optional elementary file containing a list of applications supported by the card and optional related data elements

[SOURCE: ISO/IEC 7816-4:2013, 3.22, modified]

3.19
elementary file
set of data units or records or data objects sharing the same file identifier and the same security attribute(s)

[SOURCE: ISO/IEC 7816-4:2013, 3.23, modified]

3.20
file identifier
data element (two bytes) used to address a file

[SOURCE: ISO/IEC 7816-4:2013, 3.27]

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

3.22
master file
unique dedicated file representing the root in a card using a hierarchy of dedicated files

[SOURCE: ISO/IEC 7816-4:2013, 3.33, modified]

Note 1 to entry: 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

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 for authentication purpose

[SOURCE: ISO/IEC 7816-4:2013, 3.37]

3.26
path
concatenation of file identifiers without delimitation

[SOURCE: ISO/IEC 7816-4:2013, 3.38]

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

[SOURCE: ISO/IEC 7816-4:2013, 3.41]

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 referenced and handled by the card within an elementary file of record structure

[SOURCE: ISO/IEC 7816-4:2013, 3.43]

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
set of data objects forming the value field of a constructed data object

Note 1 to entry: Adapted from ISO/IEC 7816-6.

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’ to ‘9’ and ‘A’ to ‘F’	hexadecimal digits

4.2 Abbreviated terms

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

AID	application identifier

AOD	authentication object directory
BCD	binary-coded decimal
CD	certificate directory
CDE	cryptographic data element
CIA	cryptographic information application
CIO	cryptographic information object
C-RP	command –response pair
CV	card-verifiable
DCOD	data container object directory
DDO	discretionary data object
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
UCS	universal multiple-octet coded character set (see ISO/IEC 10646)
URL	uniform resource locator
UTC	coordinated universal time
UTF-8	UCS transformation format 8

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 General

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 as follows:

— cryptographic key information objects;

— certificate information objects;

— data container information objects;

— authentication information objects.

The logical structure of these CIOs is shown in Figure 1. The object class of cryptographic 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.

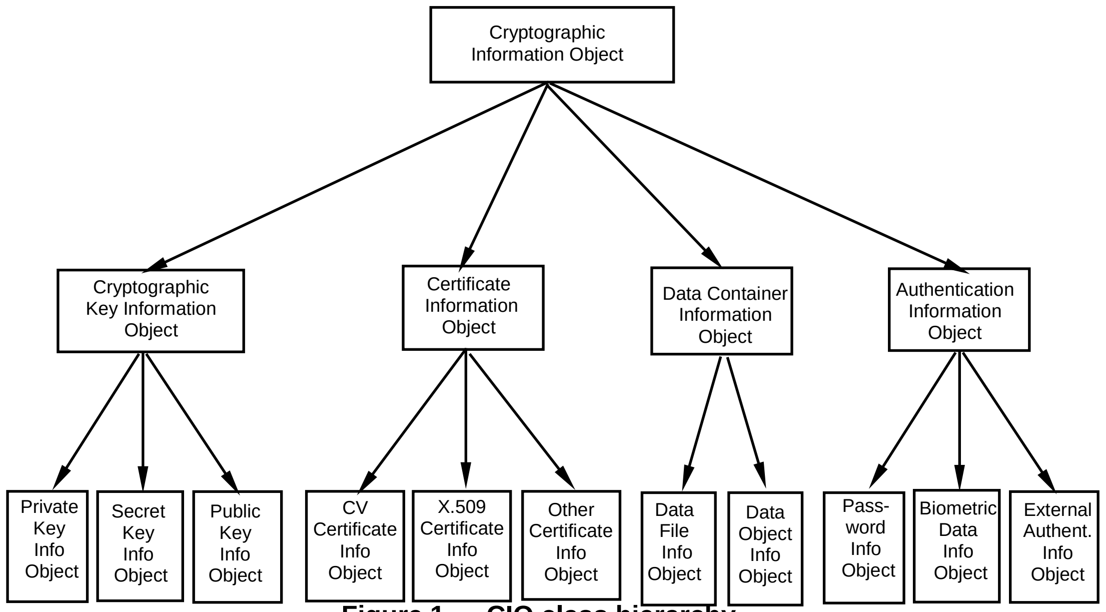

Cryptographic Information Object
Cryptographic Key Information Object
Certificate Information Object
Data Container Information Object
Authentication Information Object
Private Key Info Object
Secret Key Info Object
Public Key Info Object
CV Certificate Info Object
X.509 Certificate Info Object
Other Certificate Info Object
Data File Info Object
Data Object Info Object
Pass-word Info Object
Biometric Data Info Object
External Authent. Info Object

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.

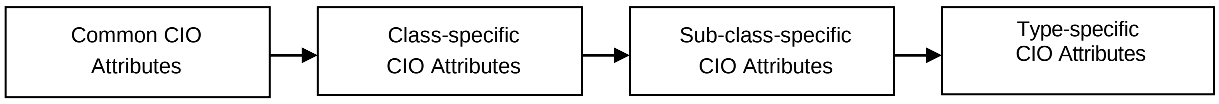

Common CIO Attributes
Class-specific CIO Attributes
Sub-class-specific CIO Attributes
Type-specific CIO Attributes

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,

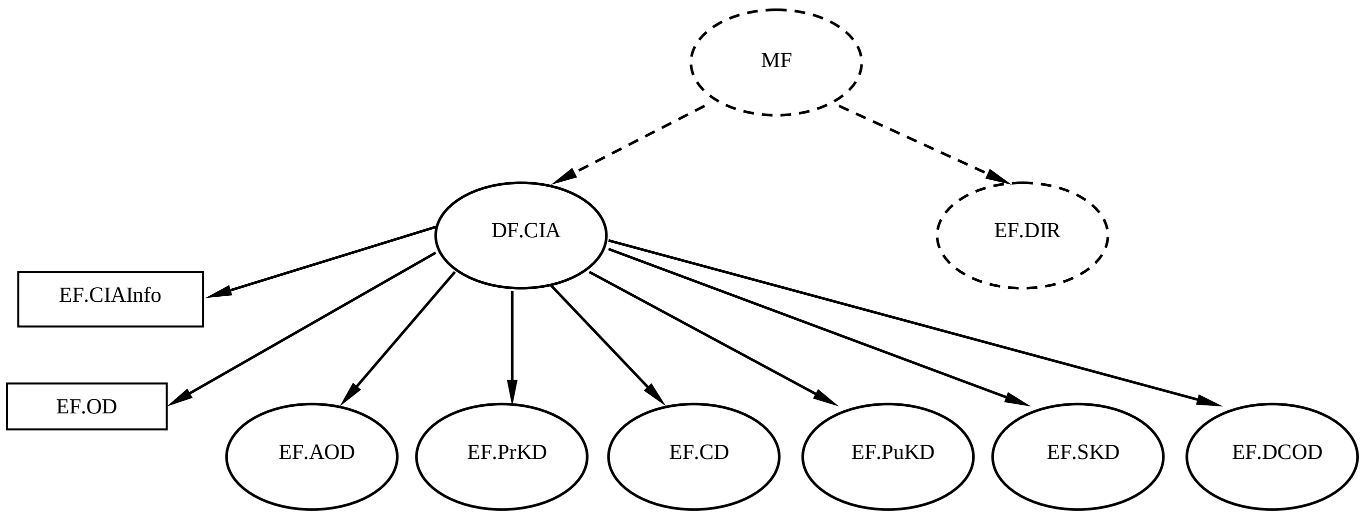
— access control mechanisms,

— 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 1 For the purpose of this part of ISO/IEC 7816, EF.DIR is only needed on cards that do not support application selection using AID as DF name as defined in ISO/IEC 7816-4 or when multiple CIAs reside on a single card.

NOTE 2 Square element files are mandatory for this part of ISO/IEC 7816 (see Table 1). MF may not be seen at the interface (see ISO/IEC 7816-4).

Figure 3 — Example contents of DF.CIA

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

7.4 EF.DIR

This file (file identifier: '2F00') shall, if present, contain one or several application templates as defined in ISO/IEC 7816-4. The application template (tag '61') for a CIA shall at least contain the following IDOs:

— application identifier (tag '4F'), value defined in 7.5.5;

— path (tag '51'), value supplied by application provider; if **Path** is missing, it denotes a virtual DF.CIA the CIO files of which are hosted by an application, either implicitly known or defined by the CIODDO (tag '73').

— CIODDO (tag '73') conditional, with **odfPath** and **ciaInfoPath** reference CIOs; it shall be present if **Path** is missing.

Other IDOs from ISO/IEC 7816-4 may, at the application provider's discretion, be present as well. In particular, it is recommended that application providers include both the "Discretionary data objects" data object (tag '73') and the "Application label" data object (tag '50'). The application label shall contain a UTF-8 encoded label for the application, chosen by the application provider. 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,
        ciaInfoPath     [0] Path OPTIONAL,
        aid             [APPLICATION 15] OCTET STRING (SIZE(1..16)),
                        (CONSTRAINED BY {-- Must be an AID in accordance with ISO/IEC 7816-4--})
                        OPTIONAL,
       securityFileOrObject      SET OF SecurityFileOrObject OPTIONAL,
        ... -- For future extensions
        } -- Context tag 1 is historical and shall not be used
```

NOTE 1 PKCS #15 uses this tag.

NOTE 2 In accordance with ISO/IEC 7816-4, and when present in an application template, the tag **[APPLICATION 19]** ('73') replaces the **CIODDO SEQUENCE** ('30') tag, due to implicit tagging. See D.8 for an example.

NOTE 3 When located under a DF or an ADF other than DF.CIA, the by-default FileID and Short EF Identifier of EF.OD and EF.CIAInfo are respectively as defined in Table 1 unless specified otherwise by the discretionary data object CIODDO or explicit file ID definition is preferred.

```
      SecurityFileOrObject ::= SEQUENCE {
            label                       Label OPTIONAL,
            communicationMode           CommunicationMode OPTIONAL,
            fileOrObjectPath            Path,
            cioSecurityId               INTEGER      OPTIONAL,
            protocol                    OBJECT IDENTIFIER OPTIONAL,
            index                       [0] INTEGER (0..cia-ub-index) OPTIONAL,
            precondition                [1]INTEGER (0..cia-ub-index) OPTIONAL,

      ... -- For future extensions

      }
```

The **providerId** component, if present, shall contain an object identifier uniquely identifying the CIA provider. The **odfPath** and **ciaInfoPath** components shall, if present, contain paths to elementary files EF.OD and EF.CIAInfo respectively or to constructed data objects nesting these files' respective contents. This provides a way for issuers to use non-standard file identifiers for these files or, alternatively, data objects tags without sacrificing interoperability. If data objects are employed, they shall be constructed without control parameter template (see ISO/IEC 7816-4). It also provides card issuers with the opportunity to share CIAInfo files between CIAs, when several CIAs reside on one card. The **aid** component shall, if present, indicate the application to which this CIA applies. The **securityFileOrObject** optional extension of **CIODDO** container refers to security protocol descriptors, i.e. OIDs nested in a logical data structure, of which the execution priority shall be ordered according the **index** value of the **precondition** attribute. **securityFileOrObject** components are

- **fileOrObjectPath**: path to the file or data object nesting a set of OIDs and further protocol relevant parameter denoting security protocols,
- **index**: attribute index assigned to **SecurityFileOrObject**,
- **precondition**: cross-reference to the index of the **SecurityFileOrObject** protocol(s) of which the execution is mandated prior to the current **SecurityFileOrObject** ones,
- **label**: attribute that may serve to identify the security file or object,
- **communicationMode**: physical interface as defined in 8.2.8 and to which the protocols listed in **fileOrObjectPath** apply. If the required communication mode cannot be fulfilled by the IFD, the **SecurityFileOrObject** may be not accessible by the IFD,
- **cioSecurityId**: identifier used as cross-reference to the authentication object required for the preleminary protocol to be executed by the IFD, and
- **protocol**: determines which of the possibly multiple protocols defined in **fileOrObjectPath** referenced structures is to be used.

In case the ICC wants to enforce a high level of privacy, **CIODDO** may only contain **SecurityFileOrObject** structure to indicate to the IFD the preliminary protocol(s) to perform. If multiple **SecurityFileOrObject** containers are present, which do not refer to each other as precondition, they are meant as alternatives. For examples of EF.DIR content with privacy-enabling features, see Table C.1; the table provides EF.DIR description for the file system topology on Figure C.5 and shows how **securityFileOrObject** component is implemented to ensure data minimization principle through access control according to ISO/IEC 7816-4.

The use of a EF.DIR file will simplify application selection when several CIAs reside on one card. Its use is described in ISO/IEC 7816-4.

If within the application template for a CIA, one or more nested application templates (tag '61') are present, they may contain the application identifier (tag '4F'). Each application template corresponds to an application to which this CIA applies.

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
CIAInfo	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 CIAInfo EF

The CIAInfo 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;

— card characteristics (e.g. read only).

The following information may be found:

— CIA serial number;

— manufacturer identification;

— card label;

— allocated security environments;

— file structures;

- supported algorithms;
- issuer identification;
- holder authentication;
- time of last update.

7.5.3 EF.OD

The object directory file (EF.OD) is an elementary file, unless specified otherwise by CIODDO, which may contain references to other CIO EFs. Figure 4 shows the relationship between EF.OD and other CIO EFs (for a reason of simplicity, only one referenced file of each type is shown). The ASN.1 syntax for the contents of EF.OD is described in 8.3.

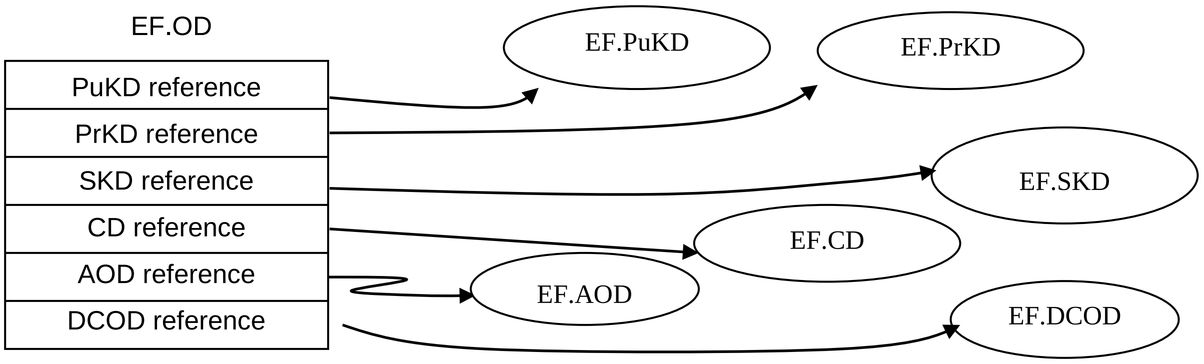


Figure 4 — Indirect retrieval of CIOs using EF.OD

7.5.4 CIO directory files

Each CIO directory file contains CIOs of a certain kind:

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

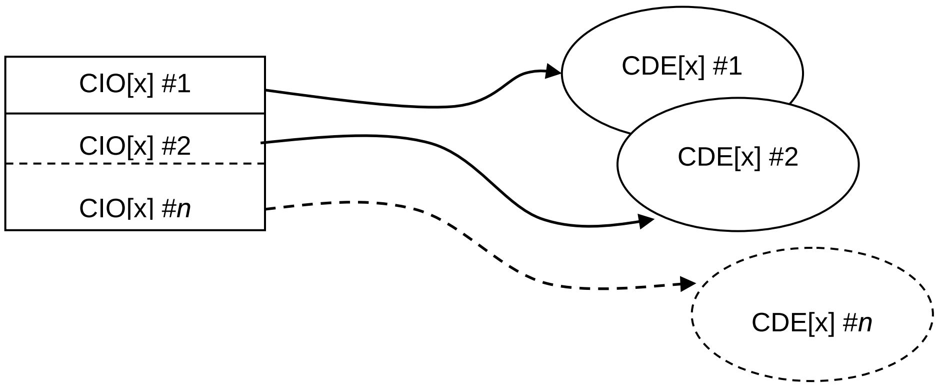
NOTE 2 CIO directory files may also be found in other DFs in the card.

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

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 Letter *x* stands for the kind of the information object which holds the information.

Figure 5 — Indirect retrieval of CDEs using CIOs

7.5.5 DF.CIA selection

The AID of a DF.CIA consists of two fields, the standard identifier E8 28 BD 08 0F (mandatory), optionally followed by either

— a one-byte index in the range ‘00’ to ‘7F’, followed by a proprietary application identifier extension (PIX), or

— a (possibly right-truncated) AID (e.g. of an application using this CIA) using a registered application provider identifier from registration category 'A' or 'D' (see ISO/IEC 7816-4).

The length of the AID shall not exceed 16 bytes. The format of the AID is therefore:

<--------------------------------------- Application Identifier (AID) --------------------------------------->

E8	28BD080F	Index (00-7F)	Proprietary application identifier extension (PIX)

or

E8	28BD080F	Ax/Dx	Remainder of category 'A' or 'D' AID

<------- 5 bytes -------><------------------------------- up to 11 bytes ---------------------------------->

Figure 6 — AID formats

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 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 [see ISO/IEC 8824 (all parts)] 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.

NOTE In this part of ISO/IEC 7816, BIT STRING type can be declared as extensible with extension markers provided related list is always numbered with consecutive numbers and it is recommended accordingly to enforce consecutive positions of extended bits.

8.2 Basic ASN.1 defined types

8.2.1 Identifier

```
Identifier ::= OCTET STRING (SIZE (0..cia-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 ::= CHOICE {
      uniqueByteRef       INTEGER (0..cia-ub-reference),
      multiByteRef        [1] OCTET STRING(SIZE(4..20))
}
```

This type is used for generic reference purposes; for a multiple-byte reference or a reference value exceeding 255, multiByteRef may be used.

8.2.3 Label

```
Label ::= UTF8String (SIZE(0..cia-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                  |
            pgp2KeyId                        |
            openPGPKeyId                     |
            certificateHolderReference,
            ... -- For future extensions
            }

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 nine members in the set of identifiers for private keys and certificates, **KeyIdentifiers**.

— **issuerAndSerialNumber**: The value of this type shall be a **SEQUENCE** consisting 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 an ISO/IEC 9594-8 certificate which contains the public key associated with the private key. This identifier can be used for certificate chain traversals.

— **subjectKeyHash**: 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**: An **OCTET STRING** that contains a SHA-1 hash of the issuer's name as it appears in the certificate.

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

— **subjectNameHash**: An **OCTET STRING** that contains a SHA-1 hash of the subject's name as it appears in the certificate.

— **pgp2KeyId**: An **OCTET STRING (SIZE(8))** that contains a PGP2 key identifier.

NOTE PGP 2 key identifiers are defined in IETF RFC 4880 (see Reference [4]).

— **openPGPKeyId**: An **OCTET STRING (SIZE (8))** that contains an OpenPGP key identifier.

NOTE OpenPGP key identifiers are defined in IETF RFC 4880 (see Reference [4]).

— **certificateHolderReference**: An OCTET STRING that denotes the holder of an ISO/IEC 7816-8 card verifiable certificate and that is used as subject key identifier to reference the public key of the certificate holder.

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 {
        efidOrTagChoice CHOICE {
```

```
efidOrPath OCTET STRING,
tagRef     [0]  SEQUENCE {
           tag OCTET STRING,
           efidOrPath OCTET STRING OPTIONAL
                  },
appFileRef [1]  SEQUENCE       {
           aid [APPLICATION 15] OCTET STRING,
           efidOrpath OCTET STRING
},
appTagRef [2]  SEQUENCE{
           aid [APPLICATION 15] OCTET STRING,
           tag OCTET STRING,
           efidOrPath OCTET STRING OPTIONAL
}
           },
index             INTEGER (0..cia-ub-index) OPTIONAL,
length            [0] INTEGER (0..cia-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 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 **CIAInfo** file as well (see 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. **aid** and **tag** are used for referencing from CIA of logical data structures located in application context.

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 five 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 and consists of an even number of bytes, it references a file either by an absolute or relative path (i.e. concatenation of file identifiers), and

— longer than two bytes and consists of an odd number of bytes, it references a qualified path (see ISO/IEC 7816-4).

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. In the **urlWithDigest** case, assuming that the CIO card is protected against unauthorized data modifications, the **digest** component will protect the externally protected object against unauthorized modifications too.

NOTE The URL syntax is defined in IETF RFC 3986 (see Reference [3]).

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' or '00' 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 7.5.4).

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..cia-ub-userConsent) OPTIONAL,
        accessControlRules  SEQUENCE SIZE (1..MAX) OF AccessControlRule OPTIONAL,
        currentLCS          LifeCycleStatus 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,
        communicationMode            CommunicationMode OPTIONAL,
        lifeCycleStatus              LifeCycleStatus OPTIONAL,
        verifLimitDates            RangeOfDate  OPTIONAL,

        ... -- For future extensions
        }

AccessMode ::= BIT STRING {
        read       (0),
        update     (1),
        execute    (2),
        delete     (3),
        attribute  (4),
        pso_cds    (5),
        pso_verif  (6),
        pso_dec    (7),
        pso_enc    (8),
        int_auth   (9),
        ext_auth   (10)

        }
```

```
CommunicationMode::= BIT STRING {
   contact (0),
   contactLess (1),
   usb (2),
   nfc (3),
   contactC6 (4)

}

LifeCycleStatus::= ENUMERATED {creation(0), init(1), op-activated(2), op-deactivated(3), termination(4),
proprietary(5),…}

RangeOfDate ::= SEQUENCE {
      startDate             GeneralizedTime   OPTIONAL,
      endDate               [0] GeneralizedTime     OPTIONAL
}


SecurityCondition ::= CHOICE {
            always                  NULL,
            authId                  Identifier,
            authReference           AuthReference,
            not                     [0] SecurityCondition,
            and                     [1] SEQUENCE SIZE (2..cia-ub-securityConditions) OF SecurityCondition,
            or                      [2] SEQUENCE SIZE (2..cia-ub-securityConditions) OF SecurityCondition,
            ... -- For future extensions
            }
AuthReference ::= SEQUENCE {
            authMethod     AuthMethod,
            seIdentifier   Reference OPTIONAL
            }
AuthMethod ::= BIT STRING {secureMessaging(0), extAuthentication(1), userAuthentication(2), always(3)}
```

The **label** is purely for display purposes (man-machine interface), for example, when a user has several certificates for one key pair (e.g. “bank certificate”, “e-mail certificate”).

The **flags** component 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** component 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 8.9).

The **userConsent** component 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). A value of **1** means that a new authentication is required before each access.

The **accessControlRules** component 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 component is not present, access control rules will have to be deduced by other means. The **authReference** option allows for a closer coupling with other parts of ISO/IEC 7816, through the reference to Security Environments and identification of the class of authentication method (**authMethod**).

The **AccessMode** component gives information of access mode to the object or its attribute. **read**, **update**, **execute**, and **delete** are access mode for the object itself and **attribute** is for its attribute change, for example, resetting key retry counter.

Other access mode attributes are intended for the completion of the **execute** access mode meaning. Those further attributes are to be set along with **execute** attribute to describe the action. **pso_cds** is for PERFORM SECURITY OPERATION (PSO) COMPUTE DIGITAL SIGNATURE command, **pso_verify** for PSO VERIFY CERTIFICATE command, **pso_dec** for PSO DECIPHER command, **pso_enc** for PSO ENCIPHER command, **int_auth** for INTERNAL AUTHENTICATE command, and **ext_auth** for EXTERNAL AUTHENTICATE command.

The **CommunicationMode** component indicates for which physical interface the accessControlRule applies. The **nfc** option relates to Near Field Communication protocol (see ISO/IEC 18092); the **usb** option relates to Universal Serial Bus protocol (see ISO/IEC 7816-12); the **contactC6** option is provided for legacy cards using contact C6 for programming power required to write or to erase the internal non-volatile memory (see ISO/IEC 7816-3); the **contactLess** option relates to proximity cards supporting ISO/IEC 14443; the **contact** option relates to cards with contacts (see ISO/IEC 7816-3). These options are according the transport type descriptor as per ISO/IEC 7816-4:2012, 7.4.12.1.

The **LifeCycleStatus** component indicates the life cycle status to which the AccessControlRule applies for any file or data object (see ISO/IEC 7816-4:2012, 7.4.12 for definitions).

- **creation**: Creation state
- **init**: Initialisation state
- **op-activated**: Operational (activated)
- **op-deactivated**: Operational (deactivated)
- **termination**: Termination state
- **proprietary**: Proprietary state

The **verifLimitDates** component is optional and provides the time interval during which the **AccessControlRule** to which it is attached is allowed for verification; outside the limit dates indicated by either both or one of **startDate** and **endDate** attributes, the **AccessControlRule** cannot be verified anymore. To achieve the fit between **verifLimitDates** and the actual date, i.e. date of the day, when **AccessControlRule** verification occurs, a time-stamped token may, as example, be delivered to the card to check the date of the day. As application examples: (1) an IFD presents its certificate with extension denoting a range of dates; if the verification limit dates indicated in Access Control rules (**startDate**, **endDate**) fit the range presented in the certificate, the IFD can authenticate to the card and access the authorized services accordingly; (2) A cryptographic object lifespan can be set to a determined interval of time, between **startDate** and **endDate**, so that, once the validity period after card issuance is elapsed, the cryptographic object is locked and may require a specific activation to resume. Each time this object is requested by the outside world, a timestamp may be presented to the ICC; (3) A quality control procedure on batches of cards protected by a password based mechanism (of which the password should only be known to the card owner) may resort to a temporary password for quality control purposes; such a password can be valid only during the interval between **startDate** and **endDate**.

- **startDate** and **endDate** components shall be encoded as a DO with a tag of [UNIVERSAL 24] class (ASN.1 encoding rules, ISO/IEC 8825-1), i.e. DO'18', and nested within a SEQUENCE. See D.9 for **GeneralizedTime** encoding guidelines conformant to ISO/IEC 8825-1.

NOTE 1 When the **accessControlRules** component and the **authID** component both are present, information in the **accessControlRule** component 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 7816-4).

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.

NOTE 4 When present, **currentLCS** component may reflect the **LifeCycleStatus** of the CIO.

NOTE 5 If implemented by the card, **verifLimitDates** attribute may require recording by the card of the first trusted time reference delived to the card, i.e. timestamp for the current date from a trusted party, which may be updated each time a more recent subsequent timestamp is presented to the card, or rejected otherwise.

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** component 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 8.2.15).

The **usage** component (**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 **keyUsage** flags for public keys, CIO flags for public keys, and CIO flags for private keys, use Table 2.

Table 2 — Mapping between CIO key usage flags and ISO/IEC 9594-8 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 1 Implementations should verify that all key usage flags for a particular key pair is consistent.		
NOTE 2 Only those ISO/IEC 9594-8 key usage flags that are relevant for this part of ISO/IEC 7816 have been given a mapping.		

The **native** component identifies whether the cryptographic algorithms associated with the key are implemented in the card hardware.

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**;

— **cardGenerated** indicates that the key was randomly generated on the card.

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

The **keyReference** component 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 and ISO/IEC 7816-8).

NOTE The value of the **keyReference** component is intended for use in key reference DOs (ISO/IEC 7816-4) and any values, also negative values, are conceivable.

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

The **algReference** component identifies algorithms the key may be used with by referencing **supportedAlgorithm** values from the EF.CIAInfo file.

8.2.10 CommonPrivateKeyAttributes

```
CommonPrivateKeyAttributes ::= SEQUENCE {
        name            Name OPTIONAL,
        keyIdentifiers  [0] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
        generalName     [1] GeneralNames OPTIONAL,
        keyUsageConstraints     [2] KeyUsageConstraints OPTIONAL,
    ... -- For future extensions
    }
(CONSTRAINED BY {-- if keyUsageConstraints present, CommonObjectAttributes.userConsent should be set--})
```

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

Values of the **keyIdentifiers** component 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 8.2.4).

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

```
KeyUsageConstraints ::= SEQUENCE {
        keyUsageConstraintsFlag   BIT STRING {
        immediateUsage  (0)


            },
        refOID     OBJECT IDENTIFIER OPTIONAL,
        …-- For future extensions
       }
```

CommonPrivateKeyAttributes.keyUsageConstraintsFlag: this component indicates whether

— the interface device is not allowed to send a C-RP between the preparation phase and the usage phase on the logical channel used for key usage, i.e **immediateUsage** set to 1, or

— that the interface device may send one or several C-RP(s) between the preparation phase and the usage phase, on arbitrary logical channels, i.e. **immediateUsage** not set.

The OID reference **refOID,** if present, shall provide details for key usage constraints. If both **userConsent** attribute (see 8.2.8) and **keyUsageConstraints** component are implemented, **userConsent** shall be correlated with **keyUsageConstraints** to set the number of times an application may access the key object without explicit consent from the user.

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** components of the **CommonPublicKeyAttributes** type shall be the same as for the corresponding components of the **CommonPrivateKeyAttributes**.

The **trustedUsage** component indicates one or more purposes for which the public key is trusted by the cardholder (see 8.2.15).

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

8.2.12 CommonSecretKeyAttributes

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

The optional **keyLen** component 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** component.

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,
        validity          [4] Validity OPTIONAL,
        ... -- For future extensions
        } -- 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,
        ... – For future extensions
        } (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** component. This requirement simplifies 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** component.

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

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

The **certHash** component is useful from a security perspective when a certificate is stored externally 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** component indicates one or more purposes for which the certified public key is trusted by the cardholder. Object identifiers for the **extKeyUsage** component may be defined by any organization with a need. For actual usage, the intersection of the indicated usage in this component and the **keyUsage** extension (if present) in the certificate itself should be taken. If the **trustedUsage** component 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** component and follow the cross-reference (**commonKeyAttributes.iD**) to an appropriate certificate.

The **identifiers** component 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.

The **validity** component provides information about the certificate's validity period.

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** component is intended to contain the name or the application to which the data container object in question “belongs”.

The **applicationOID** component is intended to contain 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 components has to be present in a value of type **CommonDataContainerObjectAttributes**.

The **iD** component 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** component, when present, shall contain a value of a “key reference” object (see ISO/IEC 7816-4), which is the way to reference these keys in Security Environments.

The **seIdentifier** component, when present, identifies the security environment to which the authentication object belongs.

8.2.19 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 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}

PublicKeys  ::=   PathOrObjects {PublicKeyChoice}

SecretKeys  ::=   PathOrObjects {SecretKeyChoice}

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’ or '00' 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**).

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

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

A **usefulCertificates** component references certificate information objects describing certificates that do not belong in either a **trustedCertificates** or **certificates** component. 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                Path,
        modulusLength        INTEGER, -- modulus length in bits, e.g. 1024
        keyInfo              KeyInfo {NULL, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the components 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 shall be known. The length shall be expressed in bits, e.g. 1 024.

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

8.4.3 Private elliptic curve key attributes

```
PrivateECKeyAttributes ::= SEQUENCE {
        value     Path,
        keyInfo   KeyInfo {Parameters, PublicKeyOperations} OPTIONAL,
          ... -- For future extensions
        }
```

The interpretation of the components 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 component in 8.4.2.

NOTE 1 For **Parameters** definition, this part of ISO/IEC 7816 references the module from ANSI X9.68:1998.

NOTE 2 The OID referencing ANSI X9.62 is updated in this part of ISO/IEC 7816 (see Annex A) to solve the conflict identified on **{iso(1) member-body(2) us(840) ansi-x962(10045) module(4) 1}** that incidentally referred to both ECDSA signatures with SHA1 and to ANSI X9.62 module.

8.4.4 Private Diffie-Hellman key attributes

```
PrivateDHKeyAttributes ::= SEQUENCE {
        value      Path,
        keyInfo    KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the components 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 component in 8.4.2.

8.4.5 Private DSA key attributes

```
PrivateDSAKeyAttributes ::= SEQUENCE {
        value      Path,
        keyInfo    KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -– For future extensions
        }
```

The interpretation of the components 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 component in 8.4.2.

8.4.6 Private KEA key attributes

```
PrivateKEAKeyAttributes ::= SEQUENCE {
        value      Path,
        keyInfo    KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }
```

The interpretation of the components 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 component in 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 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. Must contain a public RSA key.
        ... – For future extensions
        }

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

The interpretation of the components 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 shall 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 shall be known. The length shall be expressed in bits, e.g. 1 024.

— **PublicRSAKeyAttributes.keyInfo:** See corresponding component in 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,
        spki       SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. Must contain a public elliptic-curve key
        ... – For future extensions
        }
```

The interpretation of the components 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 component in 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,
        spki      SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. Must contain a public D-H key.
        ... – For future extensions
        }

DHPublicNumber ::= INTEGER
```

The interpretation of the components 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 component in 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,
        spki      SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. Must contain a public DSA key.
        ... – For future extensions
        }

DSAPublicKey ::= INTEGER
```

The interpretation of the components 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 component in 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,
        spki      SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. Must contain a public KEA key.
        ... – For future extensions
        }

KEAPublicKey ::= INTEGER
```

The interpretation of the components 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 component in 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 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 component 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 GenericSecretKey type

This type is intended to contain information specific to a secret key of any kind. See 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
        }
```

```
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 components 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 components is the same as for the corresponding components in ISO/IEC 9594. The values of these components shall be exactly the same as for the corresponding components 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 components 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 components shall be exactly the same as for the corresponding components in the attribute certificate itself. They may be stored explicitly for easier lookup.

— **X509AttributeCertificateAttributes.attrTypes:** This optional component 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 Reference [5]).

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

The interpretation of the component 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 4880 (see Reference [4]).

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

The interpretation of the component 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 Reference [10]).

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

The interpretation of the component 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 [1]domain certificate attributes

NOTE X9.68 domain certificates are defined in ANSI X9.68:2-2001 (see Reference [12]) but not available anymore from the ANSI catalog, and may become obsolete in the future.

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

The interpretation of the component shall be as follows.

— **X9-68CertificateAttributes.value**: The value shall be a **ReferencedValue** identifying either a file containing a DER or PER (see ISO/IEC 8825-2) encoded 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. Their main use is in public-key based card authentication methods.

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

The interpretation of the component shall be as follows.

[1] ANSI X9.68 is not available in the ANSI catalog. This data type is only for backward compatibility and may become obsolete in a future edition.

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

— **CVCertificateAttributes.certificationAuthorityReference**: The value of this component shall be exactly the same as for the corresponding component in the card verifiable certificate.

8.7.9 Generic certificate attributes

This type is intended to contain information specific to a certificate of any kind. See 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 ISO/IEC 7816-4.

```
ISO7816DOAttributes ::= ObjectValue {CIO-OPAQUE.&Type}
     (CONSTRAINED BY {-- All such data container objects shall be defined in accordance with ISO/IEC 7816-4 --})
```

Each **iso7816DO** entry in an EF.DCOD will therefore reference a file, which shall conform to ISO/IEC 7816-4. 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** (see ISO/IEC 9594-6).

```
OidDOAttributes ::= SEQUENCE {
        id          CIO-OPAQUE.&id ({AllowedOidDOs}),
        value       CIO-OPAQUE.&Type ({AllowedOidDOs}{@id})
        }
AllowedOidDOs CIO-OPAQUE ::= {...}
```

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},
        internal              [3] AuthenticationObject{InternalAuthObjectAttributes},
        ... -- 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

8.9.2.1 General

```
PasswordAttributes ::= SEQUENCE {
        pwdFlags                  PasswordFlags,
        pwdType                   PasswordType,
        minLength                 INTEGER (cia-lb-minPasswordLength..cia-ub-minPasswordLength),
        storedLength              INTEGER (0..cia-ub-storedPasswordLength),
        maxLength                 INTEGER OPTIONAL,
        pwdReference              [0] Reference DEFAULT 0,
        padChar                   OCTET STRING (SIZE(1)) OPTIONAL,
        lastPasswordChange        GeneralizedTime OPTIONAL,
        path                      Path OPTIONAL,
        verifDataHistoryLength       [1]    INTEGER    (0..cia-ub-storedVerifDataValueNumber)    OPTIONAL,
       cioSecurityId             [2] INTEGER     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),
        resetRetryCounter1        (12),
        resetRetryCounter2        (13),
        context-dependent         (14),
        multiStepProtocol         (15)
        } (CONSTRAINED BY { -- 'unblockingPassword' and 'soPassword' cannot both be set, and 'context-
dependent' supersedes both 'integrity-protected' and 'confidentiality-protected' when set-- })

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

The interpretation of these types shall be as follows.

— **PasswordAttributes.pwdFlags**: This component 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,
- can be reset by means of a RESET RETRY COUNTER command with P1 = '00' (resetRetryCounter1 and resetRetryCounter2 are not set), P1 = '01' (only resetRetryCounter2 is set), P1 = '02' (only resetRetryCounter1 is set) or P1 = '03' (both bits are set), with (**resetRetryCounter1**, **resetRetryCounter2**),

  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** (see ISO/IEC 7816-4, 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 and 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**) except if **context-dependent** attribute is set which overrides both **integrity-protected** and **confidentiality-protected**,
- shall be presented to the card encrypted (**confidentiality-protected**) except if **context-dependent** attribute is set which overrides both **integrity-protected** and **confidentiality-protected**,
- is **context-dependent**, meaning it may be presented either secured (confidentiality and/or integrity) or in plain text depending on the context, e.g. physical interface, trusted or non-trusted environment, reader or PinPad capabilities; if set, this attributes supersedes both **integrity-protected** and **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**), and
- is **multiStepProtocol** enabled when to be implicitly verified by a multi-step authentication protocol. The protocol parameters are determined by an optional OID given in **SecurityFileOrObject.protocol** and/or may be either obtained with GET DATA command after activation of the ICC or with READ BINARY command from a file specified in **SecurityFileOrObject.fileOrObjectPath**.

- **PasswordAttributes.pwdType**: This component 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 contains an ASCII (see Reference [2]) 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');
  - **iso9564-1** (encoding in accordance with ISO 9564-1).

- **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 component 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 '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 component is intended to be used in applications that require 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) '**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.
- **PasswordAttributes.verifDataHistoryLength**: Maximum number of verification data values that shall be recorded and maintained; its range is within the closed interval [0, 8].
- **PasswordAttributes.cioSecurityId:** identifier used as cross-reference back to the **SecurityFileOrObject** containing a protocol descriptor verifying and requiring this password for the execution of a preliminary protocol by the IFD.

8.9.2.2 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** component, 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 stored biometric reference data with presented biometric verification data, contains information about the stored biometric reference data ("template").

```
BiometricAttributes ::= CHOICE {
        biometricTemplateAttributes     BiometricTemplateAttributes,
        bit                             [APPLICATION 96] BiometricInformationTemplate,
        bitGroup                        [APPLICATION 97] BiometricInformationTemplateGroup
        }
BiometricInformationTemplate ::= CHOICE {
  biometricInformationTemplate OCTET STRING,
  -- Shall contain an ISO/IEC 7816-11 Biometric Information Template value
    SEQUENCE {
      maxBITLength INTEGER OPTIONAL,
      biometricInformationTemplate OCTET STRING,
      -- Shall contain an ISO/IEC 7816-11 BiometricInformationTemplate value
 }
BiometricInformationTemplateGroup ::= OCTET STRING
-- Shall contain an ISO/IEC 7816-11 Biometric Information Template group template value
BiometricTemplateAttributes ::= 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,
          chained          [1] SEQUENCE SIZE (2..cia-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
          }
```

If present along with a **biometricInformationTemplate** value, the **BiometricInformationTemplate.maxBITLength** attribute determines the maximum size in byte unit allocated to a further Biometric Information Template for enrollment during post-issuance, i.e. size of DO'7F60' (see ISO/IEC 7816-11).

The BiometricAttributes **type provides for two distinct ways to present information about stored biometric reference data** as follows**:**

— through information specific to this part of ISO/IEC 7816 (the **biometricTemplateAttributes** component);

— through information defined in ISO/IEC 7816-11 (the **bit** and **bitGroup** components) which may conform to ISO/IEC 19785-3 TLV-encoded CBEFF Patron Format for use with smartcards or other tokens (see ISO/IEC 19785-3) and for which the ASN.1 object identifier is **{iso registration-authority cbeff(19785) biometric-organization(0) jtc1-sc37(257) patron-format(1) tlv-encoded(5)}.**

  NOTE BiometricTemplateAttributes attributes list is not defined according to ISO/IEC 7816-11 and likely to be deprecated in future revisions of this part of ISO/IEC 7816.

The semantics of the components of the BiometricTemplateAttributes type is as follows.

— **BiometricAttributes.bioFlags**: Same as for **PasswordAttributes.pwdFlags** but replace “password” with “biometrical reference data”.

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

— **BiometricAttributes.bioType**: This component determines the type of biometrical information stored in the card, e.g. the right pointer finger. The "**chained**" component means that more than one biometric feature has to be presented within the same verification process, possibly by using chained commands, in order for authentication to succeed.

— **BiometricAttributes.bioReference, BiometricAttributes.lastChange**, and **BiometricAttributes.path**: As for corresponding components in **PasswordAttributes**, but replace “password” with “biometrical reference data”.

NOTE Attention drawn on consistency issue: The value of **biometricTemplateAttribute** may reflect the biometric reference data of the CIO.

8.9.4 Authentication objects for external and internal authentication

NOTE This subclause describes ways to authenticate to the card or to perform internal or mutual authentication including internal authentication implicitly specified by external authentication.

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

InternalAuthObjectAttributes ::= SEQUENCE {
       cioSecurityId     INTEGER  OPTIONAL,
       authKeyAttributes       AuthKeyAttributes,


... -- For future extensions
}


AuthKeyAttributes ::= SEQUENCE {
        derivedKey       BOOLEAN DEFAULT TRUE,
        authKeyId        Identifier,
        ... -- For future extensions
        }
CertBasedAuthenticationAttributes ::= SEQUENCE {
        cha              OCTET STRING,
        cioSecurityId    INTEGER    OPTIONAL,
        … -- For future extensions
        }
```

The interpretation of these types shall be as follows.

— **AuthKeyAttributes.derivedKey**: This component 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 component specifies the identifier (**CommonKeyAttribute.iD**) of the authentication key as, for instance, described in an EF.SKD.

— **CertBasedAuthenticationAttributes.cioSecurityId:** This component specifies an identifier used as cross-reference to the authentication object to be verified and required for the preleminary protocol to be executed by the IFD.

— **CertBasedAuthenticationAttributes.cha**: This component specifies the certificate holder authorization or certificate holder authorization template coding information on the basis of object identifiers as presented in a card-verifiable certificate (see ISO/IEC 7816-6). 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.

For symmetric role authentication, the **authKeyAttributes** choice should be selected, whereas for asymmetric role authentication, the **certBasedAttributes** choice should be selected.

8.10 Cryptographic information file, EF.CIAInfo

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

```
CIAInfo ::= 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 5646
        profileIndication       [6] SEQUENCE OF ProfileIndication OPTIONAL,
        ... -- For future extensions
        } (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 7816-4 --}) OPTIONAL,
        ... -- For future extensions
        }

RecordInfo ::= SEQUENCE {
        oDRecordLength      [0] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        prKDRecordLength    [1] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        puKDRecordLength    [2] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        sKDRecordLength     [3] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        cDRecordLength      [4] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        dCODRecordLength    [5] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        aODRecordLength     [6] INTEGER (0..cia-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.CIAInfo shall contain one DER-encoded value of type **CIAInfo**.

The interpretation of the **CIAInfo** type shall be as follows.

— **CIAInfo.version:** This component shall be set to **v2** for this edition of this part of ISO/IEC 7816. Future editions may use other values. A CIAInfo 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.

— **CIAInfo.serialNumber:** This component shall contain the CIA's unique serial number, as chosen by the application provider.

— **CIAInfo.manufacturerID**: This optional component shall, when present, contain identifying information about the card manufacturer, UTF-8 encoded.

— **CIAInfo.label**: This optional component shall, when present, contain identifying information about the application.

— **CIAInfo.cardflags**: This component 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.

— **CIAInfo.seInfo**: This optional component 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 7816-4. The **aid** component indicates the (card) application for which the security environment is applicable.

— **CIAInfo.recordInfo**: This optional component 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 component is present, they shall be linear record files; otherwise, they shall be transparent files);

  — 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).

— **CIAInfo.supportedAlgorithms**: The intent of this optional component is to indicate cryptographic algorithms, associated parameters, operations and algorithm input formats supported by the card. The **reference** component of **AlgorithmInfo** is a unique reference that is used for cross-reference purposes from PrKDs and PuKDs. Values for the **algorithm** component are for private use. Values of the **supportedOperations** component (**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** component indicates the object identifier for the algorithm. The **algRef** component 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** component may be chosen from, and interpreted as, mechanism numbers in PKCS #11 (see Reference [8]).

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

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

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

— **CIAInfo.preferredLanguage:** The preferred language of the cardholder, encoded in accordance with IETF RFC 5646.

— **CIAInfo.profileIndication**: This optional component 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 Annex 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 1}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

informationFramework, authenticationFramework, certificateExtensions, attributeCertificateDefinitions
        FROM UsefulDefinitions {joint-iso-itu-t(2) ds(5) module(1) usefulDefinitions(0) 7}

AttributeCertificate
      FROM AttributeCertificateDefinitions attributeCertificateDefinitions

Name
        FROM InformationFramework informationFramework

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


GeneralName, GeneralNames, KeyUsage
        FROM CertificateExtensions certificateExtensions

BiometricInformationTemplate , BiometricInformationTemplateGroup
     FROM CBEFF-SMARTCARD-BIDO{iso standard 19785 modules(0) types-for-smartcard(8)}

ALGORITHM-IDENTIFIER
    FROM PKCS-5 {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-5(5) modules(16) pkcs-5(1)}


ECPoint, Parameters
        FROM ANSI-X9-62 {iso(1) member-body(2) us(840) ansi-x962(10045) module(5) 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

cia-ub-identifier                      INTEGER ::= 255
cia-ub-reference                       INTEGER ::= 255
cia-ub-index                           INTEGER ::= 65535
cia-ub-label                           INTEGER ::= cia-ub-identifier
cia-lb-minPasswordLength               INTEGER ::= 4
cia-ub-minPasswordLength               INTEGER ::= 8
cia-ub-storedPasswordLength            INTEGER ::= 64
cia-ub-recordLength                    INTEGER ::= 16383
cia-ub-userConsent                     INTEGER ::= 32767
cia-ub-securityConditions              INTEGER ::= 255
cia-ub-biometricTypes                  INTEGER ::= 127
cia-ub-storedVerifDataValueNumber      INTEGER ::=8

-- A.2 Basic types

-- A.2.1

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

-- A.2.2
```

```
Reference ::= CHOICE {
        uniqueByteRef    INTEGER(0..cia-ub-reference),
        multiByteRef     [1] OCTET STRING(SIZE(4..20))

}

-- A.2.3

Label ::= UTF8String (SIZE(0..cia-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                   |
            pgp2KeyId                         |
            openPGPKeyId                      |
            certificateHolderReference,
            ... – For future extensions
            }

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}

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

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

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

pgp2KeyId KEY-IDENTIFIER ::=
            {SYNTAX OCTET STRING (SIZE(8)) IDENTIFIED BY 8}

openPGPKeyId KEY-IDENTIFIER ::=
            {SYNTAX OCTET STRING (SIZE(8)) IDENTIFIED BY 9}

certificateHolderReference  KEY-IDENTIFIER ::=
              {SYNTAX OCTET STRING IDENTIFIED BY 10}

-- 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 {{DigestAlgorithms}} ::= {
            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 {{DigestAlgorithms}} DEFAULT alg-id-sha1,
            digest         OCTET STRING (SIZE(8..128))

}

DigestAlgorithms ALGORITHM-IDENTIFIER ::= {
            {NULL IDENTIFIED BY sha-1},
            ... -- For future extensions
}

-- NOTE    DigestAlgorithms is defined as an open type that is constrained in this part of ISO/IEC 7816 to a
--unique identifier, i.e. sha-1 valuating to the default value NULL, but it may be extended to further algorithm
--identifiers,for sha-2.

Path ::= SEQUENCE {
            efidOrTagChoice CHOICE {
              efidOrPath OCTET STRING,
              tagRef         [0]  SEQUENCE {
                             tag OCTET STRING,
                             efidOrPath OCTET STRING OPTIONAL
                             },
              appFileRef [1]       SEQUENCE  {
                     aid [APPLICATION 15] OCTET STRING,
                     efidOrpath OCTET STRING
            },
              appTagRef [2]       SEQUENCE{
                     aid [APPLICATION 15] OCTET STRING,
                     tag OCTET STRING,
                     efidOrPath OCTET STRING OPTIONAL
            }
                     },
            index          INTEGER (0..cia-ub-index) OPTIONAL,
            length         [0] INTEGER (0..cia-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..cia-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,
            communicationMode                   CommunicationMode OPTIONAL,
            lifeCycleStatus                 LifeCycleStatus OPTIONAL,
            verifLimitDates                RangeOfDate  OPTIONAL,

            ... -- For future extensions
            }

AccessMode ::= BIT STRING {
            read       (0),
            update     (1),
            execute    (2),
            delete     (3),
            attribute (4),
            pso_cds  (5),
            pso_verif (6),
            pso_dec  (7),
            pso_enc   (8),
            int_auth (9),
            ext_auth (10)
            }



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

AuthReference ::= SEQUENCE {
            authMethod       AuthMethod,
            seIdentifier     INTEGER OPTIONAL
            }
```

```
AuthMethod ::= BIT STRING {secureMessaging(0), extAuthentication(1), userAuthentication(2), always(3)}

CommunicationMode::= BIT STRING {
            contact         (0),
            contactLess     (1),
            usb             (2),
            nfc             (3),
            contactC6       (4)
            }

LifeCycleStatus::= ENUMERATED {creation(0), init(1), op-activated(2), op-deactivated(3), termination(4),
proprietary(5),…}

RangeOfDate ::= SEQUENCE {
      startDate           GeneralizedTime   OPTIONAL,
      endDate             [0] GeneralizedTime     OPTIONAL,
}



-- 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,
             keyUsageConstraints         [2] KeyUsageConstraints OPTIONAL,
            ... -- For future extensions
            }
(CONSTRAINED BY {-- if keyUsageConstraints present, CommonObjectAttributes.userConsent should be set--})

KeyUsageConstraints ::= SEQUENCE {
               keyUsageConstraintsFlag        BIT STRING {
```

```
            immediateUsage    (0)

               },
             refOID         OBJECT IDENTIFIER 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,
        validity          [4] Validity 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,
        seIdentifier    [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}

PublicKeys  ::=     PathOrObjects {PublicKeyChoice}

SecretKeys  ::=     PathOrObjects {SecretKeyChoice}

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                   Path,
        modulusLength           INTEGER, -- modulus length in bits, e.g. 1024
        keyInfo                 KeyInfo {NULL, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.3

PrivateECKeyAttributes ::= SEQUENCE {
        value     Path,
        keyInfo   KeyInfo {Parameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.4

PrivateDHKeyAttributes ::= SEQUENCE {
        value     Path,
        keyInfo   KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -- For future extensions
        }

-- A.4.5

PrivateDSAKeyAttributes ::= SEQUENCE {
        value     Path,
        keyInfo   KeyInfo {DomainParameters, PublicKeyOperations} OPTIONAL,
        ... -– For future extensions
        }

-- A.4.6

PrivateKEAKeyAttributes ::= SEQUENCE {
        value     Path,
        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. 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,
        spki    SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. 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,
        spki    SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. 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,
        spki       SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. 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,
        spki       SubjectPublicKeyInfo, -- See ISO/IEC 9594-8. 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
```

```
            }

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},
            certificationAuthorityReference    OCTET STRING OPTIONAL,
            ... -- 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 7816-4 --})

-- A.8.4

OidDOAttributes ::= SEQUENCE {
            id          CIO-OPAQUE.&id ({AllowedOidDOs}),
            value       CIO-OPAQUE.&Type ({AllowedOidDOs}{@id})
            }
AllowedOidDOs CIO-OPAQUE ::= {...}

-- 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},
            internal              [3] AuthenticationObject{InternalAuthObjectAttributes},
            ... -- For future extensions
            }

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

-- A.9.2

PasswordAttributes ::= SEQUENCE {
            pwdFlags                    PasswordFlags,
            pwdType                     PasswordType,
            minLength                   INTEGER (cia-lb-minPasswordLength..cia-ub-minPasswordLength),
            storedLength                INTEGER (0..cia-ub-storedPasswordLength),
            maxLength                   INTEGER OPTIONAL,
            pwdReference                [0] Reference DEFAULT 0,
            padChar                     OCTET STRING (SIZE(1)) OPTIONAL,
            lastPasswordChange          GeneralizedTime OPTIONAL,
            path                        Path OPTIONAL,
            verifDataHistoryLength      [1] INTEGER (0..cia-ub-storedVerifDataValueNumber) OPTIONAL,
            cioSecurityId               [2] INTEGER 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),
            resetRetryCounter1          (12),
            resetRetryCounter2          (13),
            context-dependent           (14),
            multiStepProtocol           (15)

            } (CONSTRAINED BY { -- 'unblockingPassword' and 'soPassword' cannot both be set, and 'context-
dependent' supersedes both 'integrity-protected' and 'confidentiality-protected' when set-- })

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

```
-- A.9.3

BiometricAttributes ::= CHOICE {
        biometricTemplateAttributes     BiometricTemplateAttributes,
        bit                             [APPLICATION 96] BiometricInformationTemplate,
        bitGroup                        [APPLICATION 97] BiometricInformationTemplateGroup
        }

BiometricInformationTemplate ::= CHOICE {
                biometricInformationTemplate    OCTET STRING,
                --Shall contain an ISO/IEC 7816-11 Biometric Information Template value
                SEQUENCE {
                        maxBITLength            INTEGER OPTIONAL,
                        biometricInformationTemplate OCTET STRING,
                        --Shall contain an ISO/IEC 7816-11 BiometricInformationTemplate value
                }
          }

BiometricInformationTemplateGroup ::= OCTET STRING
-- Shall contain an ISO/IEC 7816-11 Biometric Information Template group template value

BiometricTemplateAttributes ::= 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,
        chained   [1] SEQUENCE SIZE (2..cia-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
        }

InternalAuthObjectAttributes ::= SEQUENCE {
      cioSecurityId     INTEGER  OPTIONAL,
     authKeyAttributes       AuthKeyAttributes,

... -- For future extensions
} (CONSTRAINED BY {-- at least one out of protocol and description attributes shall be present --})

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

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

-- A.10 Cryptographic and card information

CIAInfo ::= 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 5646
        profileIndication           [6] SEQUENCE OF ProfileIndication OPTIONAL,
        ... -- For future extensions
        } (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 7816-4 --}) OPTIONAL,
        ... -- For future extensions
        }

RecordInfo ::= SEQUENCE {
        oDRecordLength      [0] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        prKDRecordLength    [1] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        puKDRecordLength    [2] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        sKDRecordLength     [3] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        cDRecordLength      [4] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        dCODRecordLength    [5] INTEGER (0..cia-ub-recordLength) OPTIONAL,
        aODRecordLength     [6] INTEGER (0..cia-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
        derive-key         (8)   -- H/W key derivation
        }

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

AlgorithmSet CIO-ALGORITHM ::= {
        cia-alg-null,
        ... -- See PKCS #11 for possible values for the &id component (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,
    ciaInfoPath [0] Path OPTIONAL,
    aid                 [APPLICATION 15] OCTET STRING
                        (CONSTRAINED BY {-- Must be an AID in accordance with ISO/IEC 7816-4--}) OPTIONAL,
    securityFileOrObject   SET OF SecurityFileOrObject OPTIONAL,

    ... -- For future extensions
    } -- Context tag 1 is historical and shall not be used

SecurityFileOrObject ::= SEQUENCE {
         label                      Label OPTIONAL,
    communicationMode           CommunicationMode OPTIONAL,
         fileOrObjectPath           Path,
    protocol                    OBJECT IDENTIFIER OPTIONAL,
    cioSecurityId               INTEGER OPTIONAL,
    index                              [0] INTEGER (0..cia-ub-index) OPTIONAL,
    precondition                       [1] INTEGER (0..cia-ub-index) OPTIONAL,
    ... -- For future extensions
}

END
```

Annex B
(informative)

CIA example for cards with digital signature and authentication functionality

B.1 General

This Annex 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 provider'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 provider's discretion. There is no requirement for host-side applications to handle these keys.

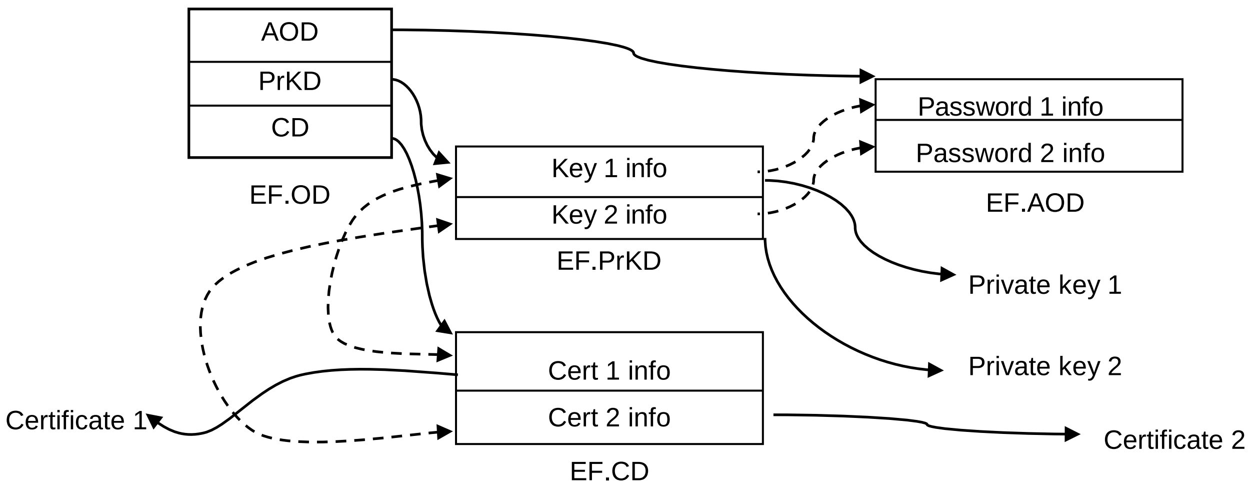
— Certificates: For each private key, at least one corresponding certificate should be stored in the card. The certificates shall be of type **X509Certificate**. If an application provider stores CA certificates on a card which supports the ISO/IEC 7816-4 logical file organization, and which has suitable 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 the application provider (or not modifiable at all). This implies usage of the **trustedCertificates** choice in the **CIOChoice** type.

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

— 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 four 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.



Note: Dashed arrows indicate cross-references.

Figure B.1 — File relationships in DF.CIA

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 Delete: External authentication
EF.CIAInfo	Read: Always Update: User authentication or External authentication or Never Append: Never
EF.OD	Read: Always Update: External authentication Append: External authentication
EF.AOD	Read: Always Update: Never Append: User authentication or External authentication
EF.PrKD, EF.PuKD, EF.SKD, EF.CD and EF.DCOD	Read: Always or User authentication Update: User authentication or External authentication or Never Append: User authentication or External authentication or Never
EF.CD containing references to trusted certificates	Read: Always Update: External authentication or Never Append: External authentication or Never
Other EFs in DF.CIA	Read: Always or User authentication Update: User authentication or External authentication or Never Append: User authentication or External authentication or Never
NOTE 1 External authentication is described in ISO/IEC 7816-4. NOTE 2 External authentication should include secure messaging as described in ISO/IEC 7816-4.	

If an application provider wants to protect a CIO directory file with an authentication object, then by default, the first authentication object in EF.AOD shall be used. For further protection measures, see CIODDO extensions in 7.4.

Annex C
(informative)

Example topologies

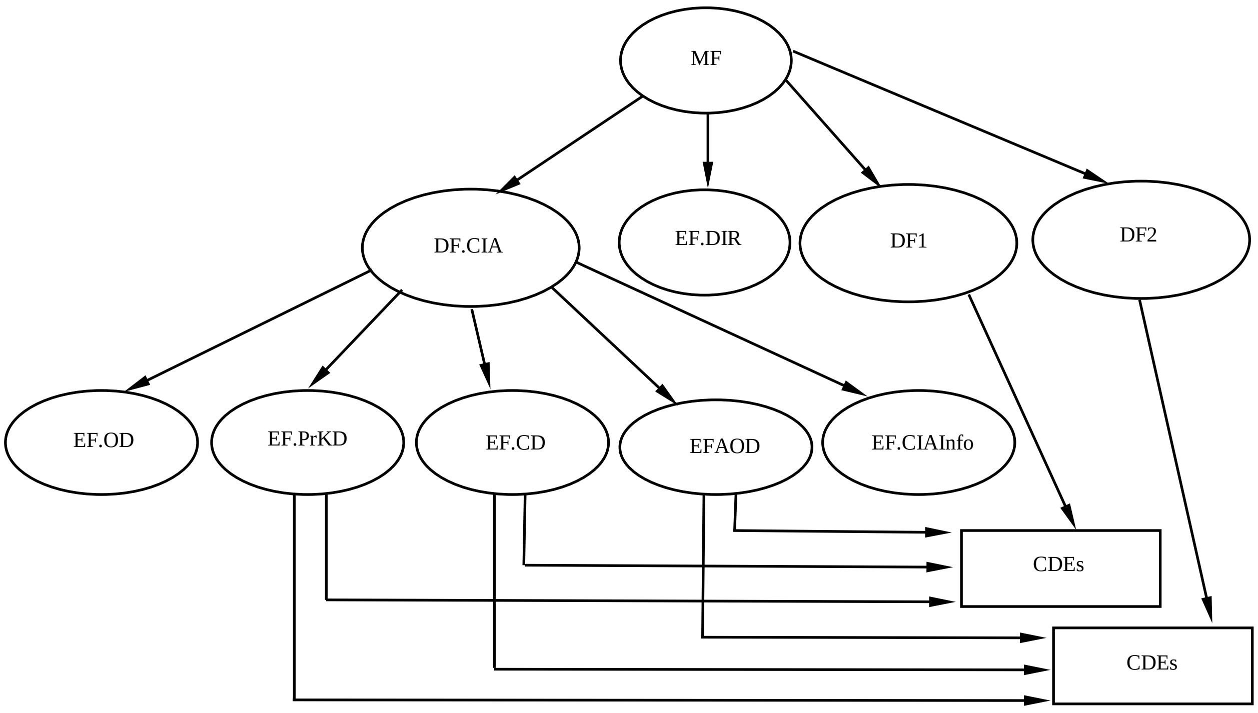


Figure C.1 — Example with three applications (cryptographic data elements are stored outside the CIA)

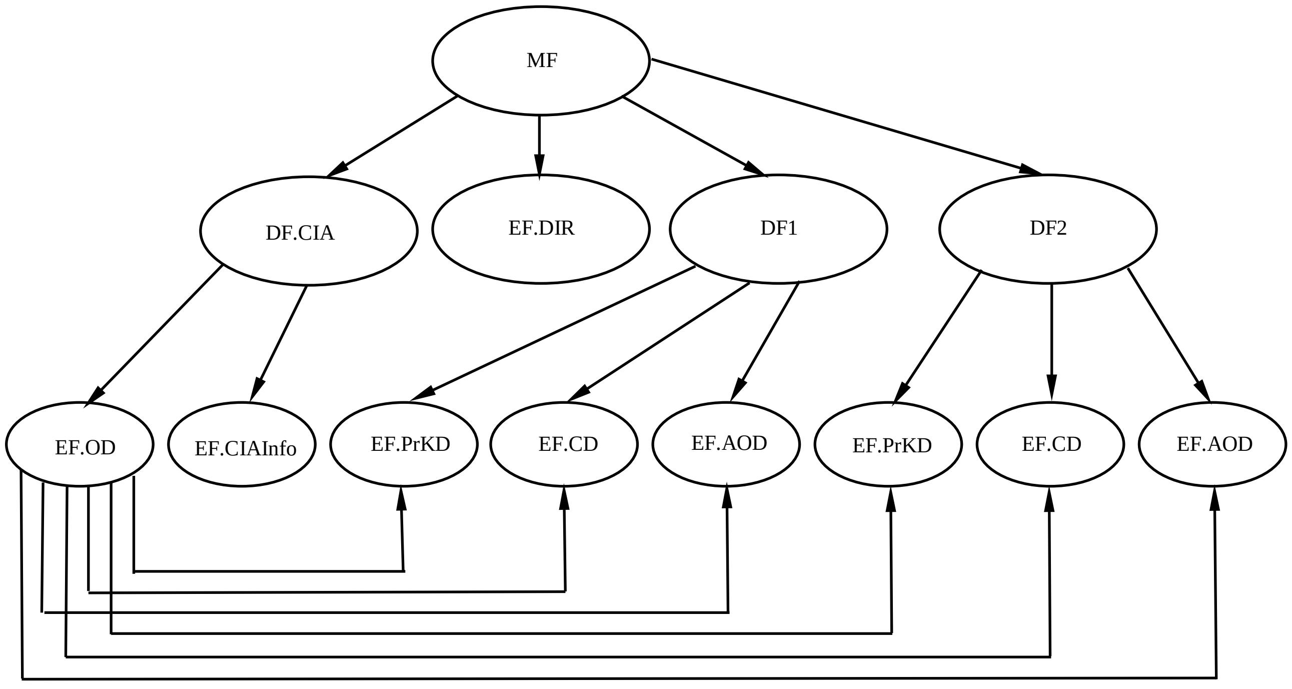


Figure C.2 — Example with three applications (only EF.OD and EF.CIAInfo in DF.CIA)

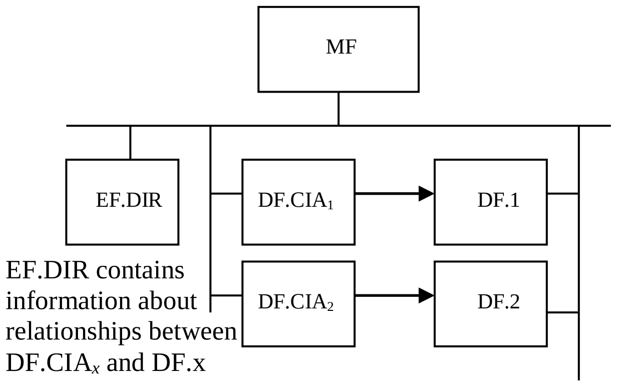

MF
EF.DIR
DF.CIA1
DF.1
DF.CIA2
DF.2
EF.DIR contains information about relationships between DF.CIAx and DF.x

Figure C.3 — Example of usage of EF.DIR

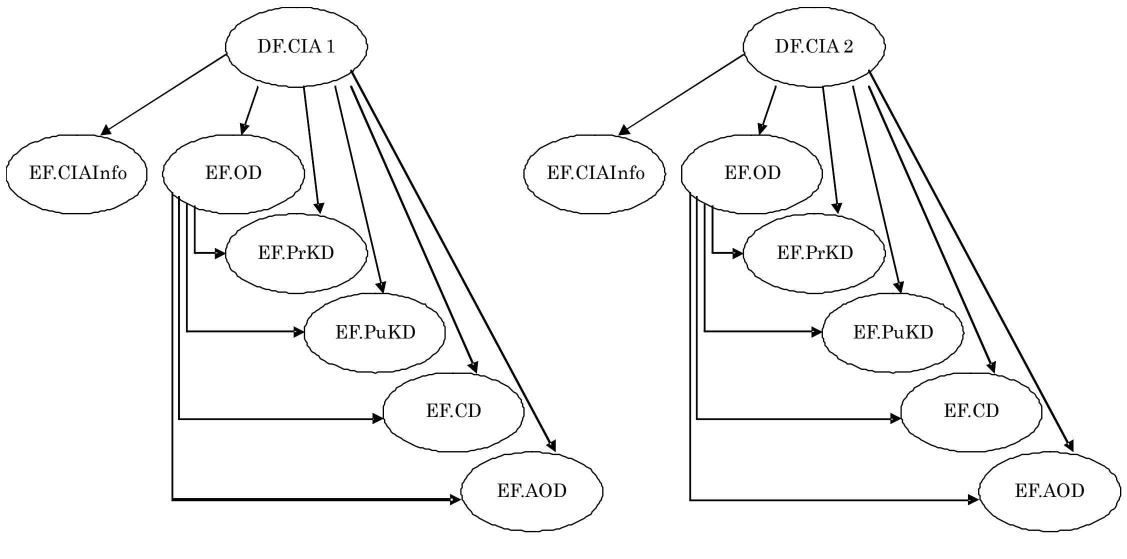


Figure C.4 — Example with two applications on the card with no EF.DIR seen at the interface

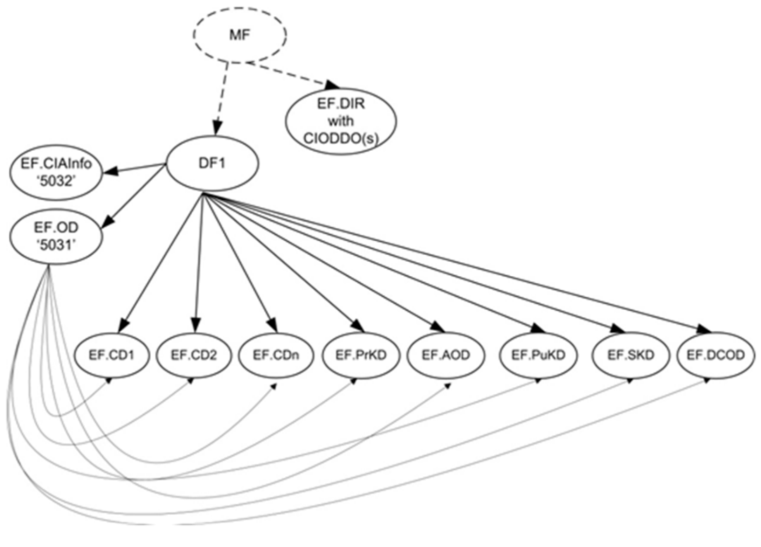


Figure C.5 — Application DF hosting DF.CIA files

Table C.1 — EF.DIR description with privacy enabling features (see topology on Figure C.5)

1st generation	2nd generation	3rd generation	4th generation	5th generation	6th generation	Description
61-L						
	4F-L-E8 28 BD 08 0F (mandatory)					Standard identifier of a DF.CIA denoting the presence of a DF.CIA application. The PIX is not encoded to prevent traceability or identification of the bearer through some AID specifics. The DF.CIA is virtual with its CIO files hosted in an application, i.e. the SELECT on DF.CIA is executed on this application; the response to this SELECT is out of scope of this part of ISO/IEC 7816.
	51-L-Path (conditional)					At application provider's discretion, Path is missing if selection by DF name is employed. Not recommended because the Path may deliver identifiable card topology.
	50-L-label (optional)					At application provider's discretion, the label is intended for IFD display and not for implementation or interpretation purposes. If used, label shall be neutral and specific to a group of cards, e.g. national cards.
	73-L-					**CIODDO** discretionary data object.
		30-L				**CIODDO** SEQUENCE.
			4F-L (conditional)			AID of the application to which the CIA referenced under this template applies. Its use is constrained by: either DO'4F' or DO'5F50' should be present in CIODDO. DO'4F' may allow for traceability or group

						identification; accordingly, it is preferable to use DO’5F50’.
			30-L			SET of **SecurityFileOrObject**
				30-L		**SecurityFileOrObject** SEQUENCE
					0C-L (optional)	UTF8String coded label denoting as example the **SecurityFileOrObject** purpose, e.g. “CardAccess”.
					03-L (optional)	BIT STRING coded **communicationMode** describing the physical interface which the protocols listed in **SecurityFileOrObject** apply.
					30-L (mandatory)	**Path** SEQUENCE pointing to the file or object hosting the security protocols opening access to CIA contents.
					02-L (optional)	INTEGER coded **cioSecurityId** cross-reference to the authentication object required for the preliminary protocol to be executed by the IFD.
					06-L (optional)	May be employed to determine which of the possibly multiple protocols defined in **fileOrObjectPath** referenced structures is to be used.
					A0-L (optional)	Recommended attribute **index** assigned to **SecurityFileOrObject.** May be skipped if implicitly known.
					A1-L (optional)	Recommended cross-reference to the index of the **SecurityFileOrObject** protocol(s) of which the execution is mandated prior to the current **SecurityFileOrObject** ones.

	5F50-L (optional)					interindustry data element nested under ‘61’ template and serving as uniform resource locator (URL) as defined in IETF RFC 1738 and IETF RFC 2396 (see ISO/IEC 7816-4, 12.2.4, Table 122). It points to part of the software required in the interface device to communicate with the application in the card. It can lead the terminal to look up the AID of the application to which the CIA referenced under this template applies. This DO’5F50’ may be used to prevent exposing to the outside world any application AID likely to allow for traceability or group identification. DO’5F50’ should be mutually exclusive with DO’4F’.

NOTE 1 The main attention is brought on data minimization principle. The most important information for discoverability are hosted in CIODDO component. The IFD can access DF.CIA contents only once the security measures required by **SecurityFileOrObject** components are met.

NOTE 2 DO‘73‘ may contain ciaInfoPath and efodPath in case explicit file ID definition is preferred.

Annex D
(informative)

Examples of CIO values and their encodings

D.1 General

Each subclause of this Annex, e.g. D.x refers to one of the EFs defined in the card file structure. Each subclause comprises three sections as follows.

— 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.

— 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 part of ISO/IEC 7816.

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 7816-4, 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 context-specific 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 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.CIAInfo

D.3.1 ASN.1 value notation

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

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

```
CIAInfo 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 {
                        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 {
                        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 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 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 },
                aid 'FAB123456789'H
                }
        }
```

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

```
  ApplicationTemplate SET: tag = [APPLICATION 1] constructed; length = 53
    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 = 12
      DDOTemplate OpenType
        providerId OBJECT IDENTIFIER: tag = [UNIVERSAL 6] primitive; length = 10
        { 1 2 840 113549 1 15 4 1 }
        aid OCTET STRING: tag = [APPLICATION 15] primitive; length = 6
```

D.8.3 Hexadecimal DER-encoding of ApplicationTemplate

```
61 33
      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 14
            06 0A
                  2A 86 48 86 F7 0D 01 0F 04 01

            4F 06

                  FA B1 23 45 67 89
```

D.9 GeneralizedTime encoding guidelines

The following rules are given as guidelines conformant to ISO 8825-1 and applying to GeneralizedTime Type encoding.

a) The value of type GeneralizedTime is a character string of type VisibleString with lexical restrictions as a string of the form "YYYYMMDDhh[mm[ss[(.|,)ffff]]]" standing for a local time, with four digits for the year (YYYY), two for the month (MM), two for the day (DD) and two for the hour (hh), followed by two digits for the minutes (mm) and two for the seconds (ss) if required, then a dot or a comma, and a number indicating the fractions of second, the maximum precision depending on the application (see ISO/IEC 8601), e.g. if the expiration date is August 10th 2011 at 7:30PM, the value field of GeneralizedTime DO is "201108101930".

b) A string conforming to (a) and followed by the letter "Z" would denote a UTC time (Greenwich meridian time or Greenwich Main Time), e.g. if expiration date according to Greenwich Main Time is 8AM on August 10th 2011, the value field of GeneralizedTime DO is "201108100800Z".

c) A string conforming to (a) and followed by a string "(+|-)hh[mm]" entails that the GeneralizedTime is the difference between the string of (a) and the second string, whereby giving the time lag if required, e.g. if expiration date local time is 8AM on August 10th 2011 and co-ordinated universal time is 12 noon on August 10th 2011, the value field of GeneralizedTime DO is "201108100600-0400".

The following range of values is valid for (a) : YYYY (0000 to 9999), MM (01 to 12), DD (01 to 31), hh (00 to 23), mm (00 to 59), ss (00 to 59).

The string value "YYYYMMDDhh[mm[ss[(.|,)ffff]]]" shall be encoded as ASCII to Hex then encapsulated in DO '18' for GeneralizedTime.

Annex E
(informative)

Examples of the use of the cryptographic information application

E.1 General

The purpose of this informative Annex is to provide practical examples of the use of the cryptographic information application. By providing sample program code for each example, programmers can see the programmatic connection between high-level ASN.1 representations and low-level BER representations, and thus, create more efficient and more compact software that uses the cryptographic information application.

Each clause in the Annex is a free-standing example and consists of four paragraphs.

1) Description of the example.

2) A specification of the example described in paragraph (1) in commented ISO/IEC 7816-15 ASN.1 constructs, using the formal value notation defined in ISO/IEC 8824-1.

3) Annotated code in the ISO/IEC 9899 TC2 C programming language for BER encoding and decoding according to the ASN.1 specification of paragraph (2).

4) BER encoding of the example as produced by the encoder of paragraph (3). Two examples also include graphic representations of the BER at the end of the Annex.

5) The source code provided in paragraph (3) was compiled and run to generate the output shown in paragraph (4).

A transcription of the ASN.1 encoding of the cryptographic information application listed in Annex A was used for all examples. A free, publically-available ASN.1 compiler was used to generate the BER encoders and decoders from this ASN.1.

E.2 Encoding of a private key

E.2.1 Cryptographic information application example description

This is an example of an ISO/IEC 7816-15 RSA private key.

E.2.2 ASN.1 encoding of an RSA private key

```
privateKeys objects { -- SEQUENCE OF --
            privateRSAKey { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '4b455931'H  -- "KEY1" --,
                  flags '80'H,
                  authId '41444d'H  -- "ADM" --,
                  userConsent 1
               },
               classAttributes { -- SEQUENCE --
                  iD '9b'H,
                  usage '2040'H,
                  native TRUE,
                  accessFlags '98'H,
                  keyReference 10
               },
               subClassAttributes { -- SEQUENCE --
                  keyIdentifiers { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        idType 5,
                        idValue '3132333435363738'H  -- "12345678" --
                     }
                  }
               },
               typeAttributes { -- SEQUENCE --
                  value indirect path { -- SEQUENCE --
                     efidOrPath '3f004041'H
                  },
                  modulusLength 1024
               }
            }
         }
```

E.2.3 Code encoding and decoding from the ASN.1

```
/*
** Encoding of a Private Key as a Data Object in EF.OD
*/
void Part15PrivateKey(const char       *label,
                      unsigned char    objectFlags,
                      unsigned char    *authId,
                      unsigned int     authIdLength,
                      unsigned int     userConsent,
                      unsigned char    native,
                      unsigned char    *iD, unsigned int iDLength,
                      unsigned short   usageFlags,
                      unsigned char    accessFlags,
                      unsigned int     keyReference,
                      unsigned int     identifierType,
                      unsigned char    *externalIdentifier,
                      unsigned char    *path, unsigned int pathLength,
                      unsigned int     modulusLength
                     )
{
   unsigned int l;
   CIOChoice *cio;
   PrivateKeyChoice *prk, **prkp;
   CredentialIdentifier *crid, **cridp;

   PrivateKeyObject_PrivateRSAKeyAttributes pattr  = { 0 };
   CommonObjectAttributes commonObjAttr            = { 0 };
   CommonKeyAttributes commonKeyAttr               = { 0 };
   CommonPrivateKeyAttributes commonPrivateKeyAttr = { 0 };
   PrivateRSAKeyAttributes privateRSAKeyAttr       = { 0 };
   Path pathOctets                                 = { 0 } ;
   AsnOcts issuerHash                              = { 0 };

   char commonObjectFlags[1] = { 0 };
   AsnBits commonFlagsAsnBits = { 3, commonObjectFlags };

   char keyUsage[2] = { 0 };
   AsnBits keyUsageAsnBits = { 10, keyUsage };

   char keyAccessFlags[1] = { 0 };
   AsnBits keyAccessFlagsAsnBits = { 5, keyAccessFlags };

   /*
```

```
** Section 8.3 The CIOChoice type
**
** "EF.OD shall contain the concatenation of 0, 1, or more DER-encoded CIOChoice values."
*/
cio = (CIOChoice *)calloc(1, sizeof(PrivateKeyChoice));

cio->choiceId = CIOCHOICE_PRIVATEKEYS;

/*
** "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."
**
** PathOrObjects{PrivateKeyChoice}
*/
cio->a.privateKeys = (PrivateKeys *)calloc(1, sizeof(PrivateKeys));
cio->a.privateKeys->choiceId = PATHOROBJECTS_PRIVATEKEYCHOICE_OBJECTS;
cio->a.privateKeys->a.objects = AsnListNew(sizeof (void*));

/*
** Section 8.4.1 PrivateKeyChoice
**
** "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."
*/
prkp = (PrivateKeyChoice **)AsnListAppend(cio->a.privateKeys->a.objects);

*prkp = prk = calloc(1, sizeof(PrivateKeyChoice));

prk->choiceId = PRIVATEKEYCHOICE_PRIVATERSAKEY;

prk->a.privateRSAKey = &pattr;

pattr.commonObjectAttributes = &commonObjAttr;
pattr.classAttributes        = &commonKeyAttr;
pattr.subClassAttributes     = &commonPrivateKeyAttr;
pattr.typeAttributes         = &privateRSAKeyAttr;

/*
** Section 8.2.8 CommonObjectAttributes
**
** "This type is a container for attributes common to all CIOs."
*/
commonObjAttr.label.octs = _strdup(label);
commonObjAttr.label.octetLen = strlen(label);

commonObjectFlags[0] = objectFlags;
commonObjAttr.flags = commonFlagsAsnBits;

commonObjAttr.authId.octetLen=authIdLength;
commonObjAttr.authId.octs = authId;

commonObjAttr.userConsent = &userConsent;

/*
** Section 8.2.9 CommonKeyAttributes
**
** "The iD field shall be unique for each key information object, except when a public
```

```
   **  key information object and its corresponding private key 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 ..."
   */
   commonKeyAttr.iD.octs = iD;
   commonKeyAttr.iD.octetLen = iDLength;

   keyUsage[0] = (unsigned char)(usageFlags>>8);
   keyUsage[1] = (unsigned char)(usageFlags);
   commonKeyAttr.usage = keyUsageAsnBits;

   keyAccessFlags[0] = accessFlags;
   commonKeyAttr.accessFlags= keyAccessFlagsAsnBits;

   commonKeyAttr.native = &native;

   commonKeyAttr.keyReference = &keyReference;

   /*
   ** Section 8.2.10 CommonPrivateKeyAttributes
   **
   ** "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."
   */
   commonPrivateKeyAttr.keyIdentifiers =
             (CommonPrivateKeyAttributesSeqOf *)AsnListNew(sizeof (void*));

   cridp = (CredentialIdentifier **)AsnListAppend(commonPrivateKeyAttr.keyIdentifiers);

   *cridp = crid = (CredentialIdentifier *)calloc(1, sizeof(CredentialIdentifier));

   issuerHash.octs = _strdup(externalIdentifier);
   issuerHash.octetLen = strlen(externalIdentifier);

   crid->idType = identifierType;
   crid->idValue.value = &issuerHash;
   SetAnyTypeByInt(&(crid->idValue), identifierType);

   /*
   ** Section 8.4.2 Private RSA Key Attributes
   **
   ** "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."
   */
   privateRSAKeyAttr.value = (ObjectValue *)calloc(1, sizeof(ObjectValue));
   privateRSAKeyAttr.value->choiceId = OBJECTVALUE_INDIRECT;
   privateRSAKeyAttr.value->a.indirect =
                      (ReferencedValue *)calloc(1, sizeof(ReferencedValue));

   privateRSAKeyAttr.value->a.indirect->choiceId = REFERENCEDVALUE_PATH;

   pathOctets.efidOrPath.octs = (char *)calloc(1, pathLength);
   memcpy(pathOctets.efidOrPath.octs, path, pathLength);
   pathOctets.efidOrPath.octetLen = pathLength;
   privateRSAKeyAttr.value->a.indirect->a.path = &pathOctets;
   privateRSAKeyAttr.modulusLength = modulusLength;

   /*
   ** Print the Private Key Data Object
   */
   PrintCIOChoice(stdout, cio, 3);

   /*
   ** BER Encode the Private Key Data Object
   */
   BERLength = BEncCIOChoiceContent(gb, cio);
}

/*
** Decoding of a Private Key as a Data Object in EF.OD
*/
PrivateKeyObject_PrivateRSAKeyAttributes *PrivateRSAKey(unsigned char *BER, unsigned int BERLength)
{
SBuf b;
   GenBuf *gb;
   unsigned int bytesDecoded = 0;
   ENV_TYPE env;
   CIOChoice *cio;
   AsnTag tagId0;
   AsnLen elmtLen0;

   if(setjmp(env)!= 0) exit(0);

   cio = calloc(1, sizeof(CIOChoice));

   SBufInstallData(&b, BER, BERLength);
   SBuftoGenBuf(&b, &gb);

   tagId0   = BDecTag(gb, &bytesDecoded, env);
   elmtLen0 = BDecLen(gb, &bytesDecoded, env);

   /*
   ** Decode the RSA Private Key Data Object
   */
   BDecCIOChoiceContent(gb, tagId0, elmtLen0, cio, &bytesDecoded, env);

   return ((PrivateKeyChoice *)(cio->a.privateKeys->a.objects->first->data))->a.privateRSAKey;
}
```

E.2.4 BER encoding

```
<EF_OD>
0xa0,0x51,0xa0,0x4f,0x30,0x4d,0x30,0x12,0x0c,0x04,0x4b,0x45,0x59,0x31,0x03,0x02,
0x05,0x80,0x04,0x03,0x41,0x44,0x4d,0x02,0x01,0x01,0x30,0x12,0x04,0x01,0x9b,0x03,
0x03,0x06,0x20,0x40,0x01,0x01,0xff,0x03,0x02,0x03,0x98,0x02,0x01,0x0a,0xa0,0x13,
0x30,0x11,0xa0,0x0f,0x30,0x0d,0x02,0x01,0x05,0x04,0x08,0x31,0x32,0x33,0x34,0x35,
0x36,0x37,0x38,0xa1,0x0e,0x30,0x0c,0x30,0x06,0x04,0x04,0x3f,0x00,0x40,0x41,0x02,
0x02,0x04,0x00
</EF_OD>
```

Table E.1 is a diagrammatic representation of this BER encoding.

Table E.1 — EF.PrKD of RSA private key

												Data type
A0	51	CIOChoice: Private key data object										
		A0	4F	PrivateKeyChoice: Private RSA Key								
				30	4D	Private RSA Key object						
						30	12	Common object Attribute				
								0C	04	label	4B, 45, 59, 31	UTF8String
								03	02	flags	05, 80	BIT STRING
								04	03	auth Id	41, 44, 44	OCTET STRING
								02	01	userConsent	01	INTEGER
						30	12	Common Key Attrribute				
								04	01	iD	9B	OCTET STRING
								03	03	usage	06, 20, 40	BIT STRING
								01	01	native	FF	BOOLEAN
								03	02	accessFlags	03, 98	BIT STRING
								02	01	keyReference	0A	INTEGER

Table E.1 (*continued*)

	A0	13	Common Private Key Attribute										
			30	11	Sequence								
					A0	0F	keyIdentifier						
							30	0D	Sequence				
									02	01	idType	05	INTEGER
									60	08	idValue	31, 32, 33, 34, 35, 36, 37, 38	OpenType
	A1	0e	Private RSA key attribute										
			30	0C	Sequence								
					30	06	Path						
							04	04	efidOrPath			3F, 00, 40, 41	OCTET STRING
					02	02	modulusLength					04, 00	INTEGER

E.3 Encoding of a protected data container

E.3.1 Cryptographic information application example description

A data container object with two security conditions, one for READ and one for UPDATE. The data in the data container is a BER-TLV. The secret key SK-1 shall be verified in order to change password AO-1.

E.3.2 ASN.1 encoding of the protected data container object

```
dataContainerObjects objects { -- SEQUENCE OF --
         iso7816DO { -- SEQUENCE --
             commonObjectAttributes { -- SEQUENCE --
               label '444f2d31'H  -- "DO-1" --,
               flags '40'H,
               accessControlRules { -- SEQUENCE OF --
                 { -- SEQUENCE --
                   accessMode '80'H,
                   securityCondition or { -- SEQUENCE OF --
                       authId '414f2d31'H  -- "AO-1" --,
                       authId '414f2d32'H  -- "AO-2" --
                     }
                 },
                 { -- SEQUENCE --
                   accessMode '40'H,
                   securityCondition and { -- SEQUENCE OF --
                       authId '414f2d31'H  -- "AO-1" --,
                       authId '414f2d32'H  -- "AO-2" --
                     }
                 }
```

```
                }
              },
              classAttributes { -- SEQUENCE --

              },
              typeAttributes direct '80020102'H
            }
        }

authObjects objects { -- SEQUENCE OF --
         pwd { -- SEQUENCE --
             commonObjectAttributes { -- SEQUENCE --
               label '414f2d31'H  -- "AO-1" --,
               flags '40'H
             },
             classAttributes { -- SEQUENCE --
               authId '414f2d31'H  -- "AO-1" --,
               authReference 1,
               seIdentifier 2
             },
             typeAttributes { -- SEQUENCE --
               pwdFlags '0400'H,
               pwdType 1,
               minLength 4,
               storedLength 12,
               maxLength 8,
               padChar 'ff'H  -- " " --,
               path { -- SEQUENCE --
                 efidOrPath '3f004045'H
               }
}
           }
       }

secretKeys objects { -- SEQUENCE OF --
         genericSecretKey { -- SEQUENCE --
             commonObjectAttributes { -- SEQUENCE --
               label '534b2d31'H  -- "SK-1" --,
               flags '40'H,
               authId '414f2d31'H  -- "AO-1" --
             },
             classAttributes { -- SEQUENCE --
               iD '534b2d31'H  -- "SK-1" --,
               usage '0200'H,
               native TRUE,
               accessFlags '10'H,
               keyReference 10
             },
             subClassAttributes { -- SEQUENCE --
               keyLen 64
             },
             typeAttributes { -- SEQUENCE --
               keyType {2 8},
               keyAttr '58'H
             }
           }
       }
```

E.3.3 Code from the ASN.1 for encoding and decoding BER

```
/*
** Encoding of a Protected Data Object
*/
```

```
void DataObject(unsigned char *label,
                unsigned char  objectFlags,
                unsigned char *password1,
                unsigned char *password2
               )
{
   CIOChoice *cio;
   DataContainerObjectChoice *dco, **dcop;
   AccessControlRule *acr, **acrp;
   SecurityCondition *sc, **scp;

   SecurityCondition securityCondition1;
   SecurityCondition securityCondition2;
   AsnOcts authId1;
   AsnOcts authId2;

   CommonObjectAttributes commonObjectAttr                                  = { 0 };
   CommonDataContainerObjectAttributes commonDataContainerObjectAttributes   = { 0 };
   ISO7816DOAttributes iso7816DOAttributes                                  = { 0 };

   DataContainerObject_ISO7816DOAttributes pattr                            = { 0 };

   CredentialIdentifier credentialIdentifier           = { 0 };
   Path pathOctets                                     = { 0 };
   AsnOcts dataObjectValue1                            = { sizeof(doValue1), doValue1 };

char commonObjectFlags[1]                           = { 0 };
   AsnBits commonFlagsAsnBits                          = { 2, commonObjectFlags };

   char acessControlRuleFlags1[1]                      = { 0 };
   AsnBits acessControlRuleAsnBits1                    = { 4, acessControlRuleFlags1 };
   char acessControlRuleFlags2[1]                      = { 0 };
   AsnBits acessControlRuleAsnBits2                    = { 4, acessControlRuleFlags2 };

   char usageFlagBits[2]                               = { 0 };
   AsnBits usageFlagsAsnBits                           = { 10, usageFlagBits };

   authId1.octetLen = strlen(password1);
   authId1.octs = strdup(password1);

   authId2.octetLen = strlen(password2);
   authId2.octs = strdup(password2);

   /*
   ** Data Object Choice
   */
   cio = (CIOChoice *)calloc(1, sizeof(DataContainerObjectChoice));
   cio->choiceId = CIOCHOICE_DATACONTAINEROBJECTS;
   cio->a.dataContainerObjects = (DataContainerObjects *)calloc(1, sizeof(DataContainerObjects));
   cio->a.dataContainerObjects->a.objects = AsnListNew(sizeof (void*));
   cio->a.dataContainerObjects->choiceId = PATHOROBJECTS_DATACONTAINEROBJECTCHOICE_OBJECTS;

   dcop = (DataContainerObjectChoice **)AsnListAppend(cio->a.dataContainerObjects->a.objects);

   *dcop = dco = calloc(1, sizeof(DataContainerObjectChoice));

   dco->choiceId = DATACONTAINEROBJECTCHOICE_ISO7816DO;

   dco->a.iso7816DO = &pattr;
   pattr.commonObjectAttributes = &commonObjectAttr;
   pattr.classAttributes        = &commonDataContainerObjectAttributes;
   pattr.subClassAttributes     = NULL;
   pattr.typeAttributes         = &iso7816DOAttributes;

   /*
   ** Common Object Attributes
   */
   commonObjectAttr.label.octetLen = strlen(label);
   commonObjectAttr.label.octs = label;

   commonObjectFlags[0] = objectFlags;
   commonObjectAttr.flags = commonFlagsAsnBits;

   commonObjectAttr.accessControlRules = AsnListNew(sizeof (void*));

   acrp = (AccessControlRule **)AsnListAppend(commonObjectAttr.accessControlRules);
   *acrp = acr = calloc(1, sizeof(AccessControlRule));
   acessControlRuleFlags1[0] = (unsigned char)(READ_FLAG);
   acr->accessMode = acessControlRuleAsnBits1;
   securityCondition1.choiceId = SECURITYCONDITION_SEACOS_OR;
   securityCondition1.a.seacos_or = AsnListNew(sizeof (void*));
   scp = (SecurityCondition **)AsnListAppend(securityCondition1.a.seacos_or);
   *scp = sc = calloc(1, sizeof(SecurityCondition));
   sc->choiceId = SECURITYCONDITION_AUTHID;
   sc->a.authId = &authId1;
scp = (SecurityCondition **)AsnListAppend(securityCondition1.a.seacos_or);
   *scp = sc = calloc(1, sizeof(SecurityCondition));
   sc->choiceId = SECURITYCONDITION_AUTHID;
   sc->a.authId = &authId2;
   acr->securityCondition = &securityCondition1;

   acrp = (AccessControlRule **)AsnListAppend(commonObjectAttr.accessControlRules);
   *acrp = acr = calloc(1, sizeof(AccessControlRule));
   acessControlRuleFlags2[0] = (unsigned char)(UPDATE_FLAG);
   acr->accessMode = acessControlRuleAsnBits2;
   securityCondition2.choiceId = SECURITYCONDITION_SEACOS_AND;
   securityCondition2.a.seacos_and = AsnListNew(sizeof (void*));
   scp = (SecurityCondition **)AsnListAppend(securityCondition2.a.seacos_and);
   *scp = sc = calloc(1, sizeof(SecurityCondition));
   sc->choiceId = SECURITYCONDITION_AUTHID;
   sc->a.authId = &authId1;
   scp = (SecurityCondition **)AsnListAppend(securityCondition2.a.seacos_and);
   *scp = sc = calloc(1, sizeof(SecurityCondition));
   sc->choiceId = SECURITYCONDITION_AUTHID;
   sc->a.authId = &authId2;
   acr->securityCondition = &securityCondition2;

   /*
   ** Common Data Container Object Attributes
   */

   /*
   ** ISO/IEC 7816 Data Object Attributes
   */
   iso7816DOAttributes.choiceId = OBJECTVALUE_DIRECT;

   iso7816DOAttributes.a.direct.value = &dataObjectValue1;
   SetAnyTypeByInt(&(iso7816DOAttributes.a.direct), issuerKeyHash);
   /*
   ** Print the Data Object
   */
   PrintCIOChoice(stdout, cio, 3);

   /*
   ** BER Encode the Data Object
   */
   BERLength += BEncCIOChoiceContent(gb,cio);
}

void Password(const char    *label,
                      unsigned char    objectFlags,
                      unsigned char    *authId,
                      unsigned int     authReference,
                      unsigned int     seReference,
                      unsigned char    *iD,
                      unsigned char    *path, unsigned int pathLength,
                      unsigned short   pwdFlags,
                      unsigned int     pwdType,
                      unsigned int     minimumLength,
                      unsigned int     storedLength,
                      unsigned int     maximumLength,
                      unsigned char    paddingCharacter
                     )
{
CIOChoice *cio;
   AuthenticationObjectChoice *auth, **authp;

   AuthenticationObject_PasswordAttributes pattr                      = { 0 };

   CommonObjectAttributes commonObjAttr                               = { 0 };
   CommonAuthenticationObjectAttributes commonAuthenticationObjectAttr = { 0 };
   PasswordAttributes passwordAttributes                              = { 0 };

   Path pathOctets                                                    = { 0 };
   AsnOcts padChar                                                    = { 0 };

   char commonObjectFlags[1]                                          = { 0 };
   AsnBits commonFlagsAsnBits                                         = { 2, commonObjectFlags };
   char passwordFlags[2]                                              = { 0 };
   AsnBits passwordFlagsBits                                          = { 12, passwordFlags };

   /*
   ** Authentication Object Choice
   */
   cio = (CIOChoice *)calloc(1, sizeof(AuthenticationObjectChoice));
   cio->choiceId = CIOCHOICE_AUTHOBJECTS;
   cio->a.authObjects = (AuthObjects *)calloc(1, sizeof(AuthObjects));
   cio->a.authObjects->choiceId = PATHOROBJECTS_AUTHENTICATIONOBJECTCHOICE_OBJECTS;
   cio->a.authObjects->a.objects = AsnListNew(sizeof (void*));

   authp = (AuthenticationObjectChoice **)AsnListAppend(cio->a.authObjects->a.objects);

   *authp = auth = calloc(1, sizeof(AuthenticationObjectChoice));

   auth->choiceId = AUTHENTICATIONOBJECTCHOICE_PWD;

   auth->a.pwd = &pattr;
   pattr.commonObjectAttributes = &commonObjAttr;
   pattr.classAttributes        = &commonAuthenticationObjectAttr;
   pattr.subClassAttributes     = (AsnNull *)NULL;
   pattr.typeAttributes         = &passwordAttributes;

   /*
   ** Common Object Attributes
   */
   commonObjAttr.label.octs = _strdup(label);
   commonObjAttr.label.octetLen = strlen(label);

   commonObjectFlags[0] = objectFlags;
   commonObjAttr.flags = commonFlagsAsnBits;

   /*
   ** Common Authentication Object Attributes
   */
   commonAuthenticationObjectAttr.authId.octs = iD;
   commonAuthenticationObjectAttr.authId.octetLen = strlen(iD);

   commonAuthenticationObjectAttr.authReference = &authReference;
   commonAuthenticationObjectAttr.seIdentifier  = &seReference;

   /*
   ** Password Attributes
   */
passwordFlags[0] = (unsigned char)(pwdFlags>>8);
   passwordFlags[1] = (unsigned char)(pwdFlags);
   passwordAttributes.pwdFlags = passwordFlagsBits;

   passwordAttributes.pwdType      = pwdType;
   passwordAttributes.minLength    = minimumLength;
   passwordAttributes.storedLength = storedLength;
   passwordAttributes.maxLength    = (AsnInt *)calloc(1, sizeof(AsnInt));
   *passwordAttributes.maxLength   = maximumLength;

   padChar.octetLen = 1;
   padChar.octs = (char *)calloc(1,1);
   padChar.octs[0] = paddingCharacter;
   passwordAttributes.padChar = padChar;

   pathOctets.efidOrPath.octs = path;
   pathOctets.efidOrPath.octetLen = pathLength;
   passwordAttributes.path = &pathOctets;

   /*
   ** Print the Authentication Data Object
   */
   fprintf(stdout, "\n\n");
   PrintCIOChoice(stdout, cio, 3);

   /*
   ** BER Encode the Authentication Data Object
   */
   BERLength += BEncCIOChoiceContent(gb,cio);
}

void SecretKey(const char    *label,
                     unsigned char    objectFlags,
                     unsigned char    *authId,
                     unsigned short   usageFlags,
                     unsigned int     keyReference,
                     unsigned char    *iD,
                     unsigned int     keyLength
                    )
{
   CIOChoice *cio;
   SecretKeyChoice *sk, **skp;

   SecretKeyObject_GenericKeyAttributes pattr      = { 0 };

   CommonObjectAttributes commonObjAttr            = { 0 };
   CommonKeyAttributes commonKeyAttr               = { 0 };
   CommonSecretKeyAttributes commonSecretKeyAttr   = { 0 };
   GenericKeyAttributes genericKeyAttr             = { 0 };

   Path pathOctets                                 = { 0 };
   AsnOcts keyOidOcts                              = { 0 };
   AsnOcts keyAttrOcts                             = { 0 };
   AsnOid keyOid                                   = { 0 };
   char commonObjectFlags[1]                       = { 0 };
   AsnBits commonFlagsAsnBits                      = { 2, commonObjectFlags };
   char keyUsage[2]                                = { 0 };
   AsnBits keyUsageAsnBits                         = { 9, keyUsage };
   char keyNativeAsnBool                           = FALSE;
   char keyAccessFlags[1]                          = { 0 };
   AsnBits keyAccessFlagsAsnBits                   = { 4, keyAccessFlags };
   /*
   ** Secret Key Choice
   */
   cio = (CIOChoice *)calloc(1, sizeof(SecretKeyChoice));
   cio->choiceId = CIOCHOICE_SECRETKEYS;
   cio->a.secretKeys = (SecretKeys *)calloc(1, sizeof(SecretKeys));
   cio->a.secretKeys->choiceId = PATHOROBJECTS_SECRETKEYCHOICE_OBJECTS;
   cio->a.secretKeys->a.objects = AsnListNew(sizeof (void*));

   skp = (SecretKeyChoice **)AsnListAppend(cio->a.secretKeys->a.objects);

   *skp = sk = calloc(1, sizeof(SecretKeyChoice));

   sk->choiceId = SECRETKEYCHOICE_GENERICSECRETKEY;

   sk->a.genericSecretKey = &pattr;

   pattr.commonObjectAttributes = &commonObjAttr;
   pattr.classAttributes        = &commonKeyAttr;
   pattr.subClassAttributes     = &commonSecretKeyAttr;
   pattr.typeAttributes         = &genericKeyAttr;

   /*
   ** Common Object Attributes
   */
   commonObjAttr.label.octs = label;
   commonObjAttr.label.octetLen = strlen(label);

   commonObjAttr.authId.octetLen = strlen(authId);
   commonObjAttr.authId.octs = authId;

   commonObjectFlags[0] = objectFlags;
   commonObjAttr.flags = commonFlagsAsnBits;
   /*
   ** Common Key Attributes
   */
   commonKeyAttr.iD.octs = iD;
   commonKeyAttr.iD.octetLen = strlen(iD);

   keyUsage[0] = (unsigned char)(usageFlags>>8);
   keyUsage[1] = (unsigned char)(usageFlags);
   commonKeyAttr.usage = keyUsageAsnBits;

   keyNativeAsnBool = TRUE;
   commonKeyAttr.native = &keyNativeAsnBool;

   keyAccessFlags[0] = NEVEREXTRACTABLE_FLAG;
   commonKeyAttr.accessFlags= keyAccessFlagsAsnBits;

   commonKeyAttr.keyReference = &keyReference;

   /*
   ** Common Secret Key Attributes
   */
   commonSecretKeyAttr.keyLen = (AsnInt *)calloc(1, sizeof(AsnInt));

   *commonSecretKeyAttr.keyLen = keyLength;

   /*
** Generic Secret Key Type
   */
   keyOidOcts.octetLen = 1;
   keyOidOcts.octs = (char *)calloc(1,1);
   keyOidOcts.octs[0] = 88;
   genericKeyAttr.keyType = keyOidOcts;

   SetAnyTypeByInt(&genericKeyAttr.keyAttr, subjectKeyId);
   keyAttrOcts.octetLen = 1;
   keyAttrOcts.octs = (char *)calloc(1,1);
   keyAttrOcts.octs[0] = 88;
   genericKeyAttr.keyAttr.value = &keyAttrOcts;

   /*
   ** Print the Secret Key Data Object
   */
   fprintf(stdout, "\n\n");
   PrintCIOChoice(stdout, cio, 3);

   /*
   ** BER Encode the Secret Key Data Object
   */
   BERLength += BEncCIOChoiceContent(gb,cio);
}

/*
** Finding the Authentication Object that Protects a Data Object
*/
void Access_Condition_for_Data_Object(unsigned char *DOBER, unsigned int DOBERLength,
                                      unsigned char *AOBER, unsigned int AOBERLength)
{
   ENV_TYPE env;
   CIOChoice *cioDO, *cioAO;
   SBuf dob, aob;
   GenBuf *dogb, *aogb;
   unsigned int bytesDecoded = 0;
   AsnTag tagId0;
   AsnLen elmtLen0;
   DataContainerObjectChoice *dataObject;
   AuthenticationObjectChoice *authenticationObject;
   AuthenticationObject_PasswordAttributes* pwd;
   AccessControlRule *accessControlRule;
   SecurityCondition *securityCondition, *securityConditionAuthID;
   AsnOcts *authId1;

   if(setjmp(env)!=0) exit(0);

   /*
   ** Retrieve the access rule for reading from the Data Object
   */
   cioDO = (CIOChoice *)calloc(1, sizeof(DataContainerObjectChoice));

   SBufInstallData(&dob, DOBER, DOBERLength);
   SBuftoGenBuf(&dob, &dogb);

   tagId0 = BDecTag(dogb, &bytesDecoded, env);
   elmtLen0 = BDecLen(dogb, &bytesDecoded, env);

   BDecCIOChoiceContent(dogb, tagId0, elmtLen0, cioDO, &bytesDecoded, env);
dataObject = (DataContainerObjectChoice *)(cioDO->a.dataContainerObjects->a.objects->first->data);

   FOR_EACH_LIST_ELMT(accessControlRule,              dataObject->a.iso7816DO->commonObjectAttributes-
>accessControlRules)
   {
      if(accessControlRule->accessMode.bits && READ_FLAG)
         break;
   }

   if(accessControlRule == NULL)
      exit(0);

   securityCondition = accessControlRule->securityCondition;

   if(securityCondition->choiceId != SECURITYCONDITION_SEACOS_OR)
      exit(0);

   securityConditionAuthID = (SecurityCondition *)(securityCondition->a.seacos_or->first->data);

   if(securityConditionAuthID->choiceId != SECURITYCONDITION_AUTHID)
      exit(0);

   authId1 = (AsnOcts *)securityConditionAuthID->a.authId;

   /*
   ** Find the Authentication Object associated with the first term in the OR
   */
   cioAO = (CIOChoice *)calloc(1, sizeof(AuthenticationObjectChoice));

   SBufInstallData(&aob, AOBER, AOBERLength);
   SBuftoGenBuf(&aob, &aogb);

   tagId0 = BDecTag(aogb, &bytesDecoded, env);
   elmtLen0 = BDecLen(aogb, &bytesDecoded, env);
   BDecCIOChoiceContent(aogb, tagId0, elmtLen0, cioAO, &bytesDecoded, env);

   FOR_EACH_LIST_ELMT(authenticationObject, cioAO->a.dataContainerObjects->a.objects)
   {
      if(authenticationObject->choiceId == AUTHENTICATIONOBJECTCHOICE_PWD)
      {
         pwd = authenticationObject->a.pwd;
         if((authId1->octetLen == pwd->commonObjectAttributes->label.octetLen) &&
            memcmp(authId1->octs, pwd->commonObjectAttributes->label.octs, authId1->octetLen) == 0)
         {
            /*
            ** Found the AO that goes with the first term in the OR
            ** condition associated with the READ access mode of the DO
            */
            break;
         }
      }
   }
}
```

E.3.4 BER encoding

```
<EF_DO>
0xa7,0x46,0xa0,0x44,0xa0,0x42,0x30,0x34,0x0c,0x04,0x44,0x4f,0x2d,0x31,0x03,
0x02,0x06,0x40,0x30,0x28,0x30,0x12,0x03,0x02,0x04,0x80,0xa2,0x0c,0x04,0x04,
0x41,0x4f,0x2d,0x31,0x04,0x04,0x41,0x4f,0x2d,0x32,0x30,0x12,0x03,0x02,0x04,
0x40,0xa1,0x0c,0x04,0x04,0x41,0x4f,0x2d,0x31,0x04,0x04,0x41,0x4f,0x2d,0x32,
0x30,0x00,0xa1,0x08,0xa0,0x06,0x60,0x04,0x80,0x02,0x01,0x02
</EF_DO>

<EF_AO>
0xa8,0x3e,0xa0,0x3c,0x30,0x3a,0x30,0x0a,0x0c,0x04,0x41,0x4f,0x2d,0x31,0x03,0x02,
0x06,0x40,0x30,0x0c,0x04,0x04,0x41,0x4f,0x2d,0x31,0x02,0x01,0x01,0x80,0x01,0x02,
0xa1,0x1e,0x30,0x1c,0x03,0x03,0x04,0x04,0x00,0x0a,0x01,0x01,0x02,0x01,0x04,0x02,
0x01,0x0c,0x02,0x01,0x08,0x04,0x01,0xff,0x30,0x06,0x04,0x04,0x3f,0x00,0x40,0x45
</EF_AO>

<EF_SK>
0xa3,0x3b,0xa0,0x39,0xaf,0x37,0x30,0x10,0x0c,0x04,0x53,0x4b,0x2d,0x31,0x03,0x02,
0x06,0x40,0x04,0x04,0x41,0x4f,0x2d,0x31,0x30,0x14,0x04,0x04,0x53,0x4b,0x2d,0x31,
0x03,0x02,0x02,0x00,0x01,0x01,0xff,0x03,0x02,0x04,0x10,0x02,0x01,0x0a,0xa0,0x05,
0x30,0x03,0x02,0x01,0x40,0xa1,0x06,0x30,0x04,0x06,0x01,0x58,0x58
</EF_SK>
```

E.4 Encoding of a certificate

E.4.1 Cryptographic information application example description

This is a description of an X.509 certificate.

E.4.2 ASN.1 Encoding of an X.509 certificate

```
certificates objects { -- SEQUENCE OF --
         x509Certificate { -- SEQUENCE --
             commonObjectAttributes { -- SEQUENCE --
               label '436572746966696361746520312031'H  -- "Certificate 1" --,
               flags '40'H,
               authId '17'H,
               userConsent 5
             },
             classAttributes { -- SEQUENCE --
               iD '41444d'H  -- "ADM" --,
               authority FALSE,
               identifier { -- SEQUENCE --
                 idType 0,
                 idValue '3132333435363738'H  -- "12345678" --
               },
               certHash { -- SEQUENCE --
                 hashAlg { -- SEQUENCE --
                   algorithm {0 17 34 51},
                   parameters '332211'H
                 },
                 certId { -- SEQUENCE --
                   issuer iPAddress 'c0a82d01'H,
                   serialNumber 13107
                 },
                 hashVal '998877'H
               },
trustedUsage { -- SEQUENCE --
                 keyUsage '2000'H
               },
               identifiers { -- SEQUENCE OF --
                 { -- SEQUENCE --
                   idType 5,
                   idValue '616263'H  -- "abc" --
                 },
                 { -- SEQUENCE --
                   idType 5,
                   idValue '78797a'H  -- "xyz" --
                 }
               }
             },
             typeAttributes { -- SEQUENCE --
               value indirect path { -- SEQUENCE --
                     efidOrPath '3f004042'H  -- "? @B" --
                   },
               subject rdnSequence { -- SEQUENCE OF --
                   { -- SET OF --
                     { -- SEQUENCE --
                       type {1 11 68},
                       value NULL,
                       valuesWithContext { -- SET OF --
                         { -- SEQUENCE --
                           distingAttrValue '5577'H,
                           contextList { -- SET OF --
                             { -- SEQUENCE --
                               contextType {2 40 86},
                               contextValues { -- SET OF --
                                 '876543'H
                               },
                               fallback TRUE
                             }
                           }
                         }
                       }
                     }
                   }
                 },
               issuer rdnSequence { -- SEQUENCE OF --
                   { -- SET OF --
                     { -- SEQUENCE --
                       type {2 5 102},
                       value NULL,
                       valuesWithContext { -- SET OF --
                         { -- SEQUENCE --
                           distingAttrValue '8899'H,
                           contextList { -- SET OF --
                             { -- SEQUENCE --
                               contextType {2 40 86},
                               contextValues { -- SET OF --
                                 '785634'H
                               },
                               fallback TRUE
                             }
                           }
                         }
                       }
                     }
                   }
                 },
               serialNumber 22376
             }
           }
       }
```

E.4.3 Code from the ASN.1 for encoding and decoding BER

```
/*
** Example of Code for Encoding the BER
*/
void X509Certificate(unsigned char    *label,
                     unsigned char    objectFlags,
                     unsigned char    *iD, unsigned int iDLength,
                     unsigned char    authority,
                     unsigned short   usageFlags,
                     unsigned int     externalIdentifierType,
                     unsigned char    *externalIdentifier,
                     unsigned char    *path, unsigned int pathLength,
                     unsigned char    *BER, unsigned int *BERLength    // Output
                    )
{
   unsigned int l;
   SBuf b;
   GenBuf *gb;
   unsigned char buffer[1024];

   CIOChoice *cio;
   CertificateChoice *prk, **prkp;
   AccessControlRule *acr, **acrp;
   SecurityCondition *sc, **scp;
   CredentialIdentifier *cid, **cidp;
   RelativeDistinguishedName *rdn, **rdnp;
   AttributeTypeAndDistinguishedValue *atadv, **atadvp;
   AttributeTypeAndDistinguishedValueSetOfSeq *atadvsos, **atadvsosp;
   Context *atadvsosso, **atadvsossop;
   AsnAny *any, **anyp;

   SecurityCondition securityCondition1;
   SecurityCondition securityCondition2;
   AsnOcts authId11                          = { sizeof(AuthId11), AuthId11 };
   AsnOcts authId12                          = { sizeof(AuthId12), AuthId12 };
   AsnOcts authId21                          = { sizeof(AuthId21), AuthId21 };
   AsnOcts authId22                          = { sizeof(AuthId22), AuthId22 };

   CertificateObject_X509CertificateAttributes pattr   = { 0 };
   CommonObjectAttributes commonObjAttr                = { 0 };
   CommonCertificateAttributes commonCertificateAttr   = { 0 };
   X509CertificateAttributes x509CertificateAttr       = { 0 };

   CredentialIdentifier credentialIdentifier           = { 0 };
   Path pathOctets                                     = { 0 };
   AsnOcts issuerHash                                  = { 0 };
   AsnOcts issuerHash1                                 = { sizeof(identifier1), identifier1 };
   AsnOcts issuerHash2                                 = { sizeof(identifier2), identifier2 };

   AsnOcts asnATADVvalue           = { sizeof(ATADVvalue), ATADVvalue };
   AsnOcts asnATADVdistvalue       = { sizeof(ATADVdistvalue), ATADVdistvalue };
   AsnOcts asnATADVvalueIssuer     = { sizeof(ATADVvalueIssuer ), ATADVvalueIssuer  };
AsnOcts asnATADVdistvalueIssuer = { sizeof(ATADVdistvalueIssuer ), ATADVdistvalueIssuer };

   char commonObjectFlags[1]                           = { 0 };
   AsnBits commonFlagsAsnBits                          = { 2, commonObjectFlags };

   char acessControlRuleFlags1[1]                       = { 0 };
   AsnBits acessControlRuleAsnBits1                     = { 4, acessControlRuleFlags1 };
   char acessControlRuleFlags2[1]                       = { 0 };
   AsnBits acessControlRuleAsnBits2                     = { 4, acessControlRuleFlags2 };

   char usageFlagBits[2]                               = { 0 };
   AsnBits usageFlagsAsnBits                           = { 10, usageFlagBits };

   CertHash certHash = { 0 };
   AlgorithmIdentifier algoId = { 0 };
   AsnOcts parameters = { 0 };

   CertId certId = { 0 };
   GeneralName issuer;

   Usage usage = { 0 };

   AsnAny contextValue1;
   AsnOcts contextValue1Octs = { sizeof(ContextValue1Octs), ContextValue1Octs };
   AsnAny contextValue1Issuer;
   AsnOcts contextValue1OctsIssuer = { sizeof(ContextValue1OctsIssuer), ContextValue1OctsIssuer };

   InitAnyCryptographicInformationFramework();
   InitAnyInformationFramework2();

   SBufInit(&b, buffer, sizeof(buffer));
   SBufResetInWriteRvsMode(&b);
   SBuftoGenBuf(&b, &gb);

   /*
   ** Section 8.3 The CIOChoice type
   **
   ** "EF.OD shall contain the concatenation of 0, 1, or more DER-encoded CIOChoice values."
   **
   **
```

```
    */
    cio = (CIOChoice *)calloc(1, sizeof(CertificateChoice));
    cio->choiceId = CIOCHOICE_CERTIFICATES;

    /*
    ** "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."
    **
    ** PathOrObjects{CertificateChoice}
    */
    cio->a.certificates = (P15Certificates *)calloc(1, sizeof(P15Certificates));
    cio->a.certificates->choiceId = PATHOROBJECTS_CERTIFICATECHOICE_OBJECTS;
    cio->a.certificates->a.objects = AsnListNew(sizeof (void*));

    /*
    ** Section 8.4.1 CertificateChoice
    **
    ** "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."
    */
    prkp = (CertificateChoice **)AsnListAppend(cio->a.certificates->a.objects);

    *prkp = prk = calloc(1, sizeof(CertificateChoice));

    prk->choiceId = CERTIFICATECHOICE_X509CERTIFICATE;

    prk->a.x509Certificate = &pattr;

    pattr.commonObjectAttributes = &commonObjAttr;
    pattr.classAttributes        = &commonCertificateAttr;
    pattr.subClassAttributes     = NULL;
    pattr.typeAttributes         = &x509CertificateAttr;

    /*
    ** Section 8.2.8 CommonObjectAttributes
    **
    ** "This type is a container for attributes common to all CIOs."
    **
    */
    commonObjAttr.label.octs = label;
    commonObjAttr.label.octetLen = strlen(label);

    commonObjectFlags[0] = objectFlags;
    commonObjAttr.flags = commonFlagsAsnBits;

    commonObjAttr.authId.octetLen = sizeof(authId);
    commonObjAttr.authId.octs = authId;

    commonObjAttr.userConsent = &one;

    /*
    ** Section 8.2.15 CommonCertificateAttributes
    **
    ** "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."
   */
   commonCertificateAttr.iD.octetLen = iDLength;
   commonCertificateAttr.iD.octs = iD;

   commonCertificateAttr.authority = &authority;

   issuerHash.octs = externalIdentifier;
   issuerHash.octetLen = strlen(externalIdentifier);
   credentialIdentifier.idValue.value = &issuerHash;
   SetAnyTypeByInt(&(credentialIdentifier.idValue), externalIdentifierType);

   commonCertificateAttr.identifier = &credentialIdentifier;

   /* Cert Hash */
   commonCertificateAttr.certHash = &certHash;
   certHash.hashAlg = &algoId;
   algoId.algorithm.octetLen = sizeof(algo1Id);
algoId.algorithm.octs = algo1Id;

   parameters.octetLen = sizeof(algo1Pm);
   parameters.octs = algo1Pm;
   algoId.parameters.value = &parameters;
   SetAnyTypeByInt(&algoId.parameters, externalIdentifierType);

   /* Cert Identifier */
   certHash.certId = &certId;
   certId.issuer = &issuer;
   issuer.choiceId = GENERALNAME_IPADDRESS;
   issuer.a.iPAddress = (AsnOcts *)calloc(1, sizeof(AsnOcts));
   issuer.a.iPAddress->octetLen = sizeof(issuerIPAddress);
   issuer.a.iPAddress->octs = issuerIPAddress;
   certId.serialNumber = 0x3333;

   /* Cert Hash */
   certHash.hashVal.bitLen = 8*sizeof(hashbits);
   certHash.hashVal.bits = hashbits;

   /* Usage */
   commonCertificateAttr.trustedUsage = &usage;
   usageFlagBits[0] = (unsigned char)(usageFlags>>8);
   usageFlagBits[1] = (unsigned char)(usageFlags);
   usage.keyUsage = usageFlagsAsnBits;

   /* Identifiers */
   commonCertificateAttr.identifiers = AsnListNew(sizeof (void*));
   cidp = (CredentialIdentifier **)AsnListAppend(commonCertificateAttr.identifiers);
   *cidp = cid = calloc(1, sizeof(CredentialIdentifier));
   cid->idType = externalIdentifierType;
   cid->idValue.value = &issuerHash1;
   SetAnyTypeByInt(&(cid->idValue), externalIdentifierType);

   cidp = (CredentialIdentifier **)AsnListAppend(commonCertificateAttr.identifiers);

   *cidp = cid = calloc(1, sizeof(CredentialIdentifier));

   cid->idType = externalIdentifierType;
   cid->idValue.value = &issuerHash2;
```

```
SetAnyTypeByInt(&(cid->idValue), externalIdentifierType);

/*
** Section 8.7.2 X509 Certificate Attributes
**
** "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. 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."
*/
x509CertificateAttr.value = (ObjectValue *)calloc(1, sizeof(ObjectValue));

x509CertificateAttr.value->choiceId = OBJECTVALUE_INDIRECT;
x509CertificateAttr.value->a.indirect = (ReferencedValue *)calloc(1, sizeof(ReferencedValue));
x509CertificateAttr.value->a.indirect->choiceId = REFERENCEDVALUE_PATH;

pathOctets.efidOrPath.octs = (char *)calloc(1, pathLength);
memcpy(pathOctets.efidOrPath.octs, path, pathLength);
pathOctets.efidOrPath.octetLen = pathLength;
x509CertificateAttr.value->a.indirect->a.path = &pathOctets;
/* Subject */
x509CertificateAttr.subject = (Name *)calloc(1, sizeof(Name));
x509CertificateAttr.subject->choiceId = NAME_RDNSEQUENCE;
x509CertificateAttr.subject->a.rdnSequence = AsnListNew(sizeof (void*));
rdnp = (RelativeDistinguishedName **)AsnListAppend(x509CertificateAttr.subject->a.rdnSequence);
*rdnp = rdn = AsnListNew(sizeof (void*));
atadvp = (AttributeTypeAndDistinguishedValue **)AsnListAppend(rdn);
*atadvp=atadv=
  (AttributeTypeAndDistinguishedValue*)
      calloc(1,sizeof(AttributeTypeAndDistinguishedValue));
atadv->type.octetLen = sizeof(ATADVtype);
atadv->type.octs = ATADVtype;

atadv->value.value = &asnATADVvalue;
SetAnyTypeByOid(&(atadv->value), &noRevAvail);

atadv->valuesWithContext = AsnListNew(sizeof (void*));
atadvsosp= (AttributeTypeAndDistinguishedValueSetOfSeq **)AsnListAppend(atadv->valuesWithContext);
*atadvsosp=atadvsos=
   (AttributeTypeAndDistinguishedValueSetOfSeq *)
       calloc(1,sizeof(AttributeTypeAndDistinguishedValueSetOfSeq));

atadvsos->distingAttrValue.value = &asnATADVdistvalue;
SetAnyTypeByInt(&(atadvsos->distingAttrValue), externalIdentifierType);

atadvsos->contextList = AsnListNew(sizeof (void*));
atadvsossop = (Context **)AsnListAppend(atadvsos->contextList);
*atadvsossop = atadvsosso = (Context *)calloc(1, sizeof(Context));
atadvsosso->contextType.octetLen = sizeof(contextTypeIssuer);
atadvsosso->contextType.octs = contextTypeIssuer;
atadvsosso->contextValues = AsnListNew(sizeof (void*));
anyp = (AsnAny **)AsnListAppend(atadvsosso->contextValues);
contextValue1.value = &contextValue1Octs;
SetAnyTypeByInt(&(contextValue1), externalIdentifierType);
```

```
   *anyp = any = &contextValue1;

   atadvsosso->fallback = &True;

   atadv->primaryDistinguished = FALSE;

   /* Issuer */
   x509CertificateAttr.issuer = (Name *)calloc(1, sizeof(Name));
   x509CertificateAttr.issuer->choiceId = NAME_RDNSEQUENCE;
   x509CertificateAttr.issuer->a.rdnSequence = AsnListNew(sizeof (void*));
   rdnp = (RelativeDistinguishedName **)AsnListAppend(x509CertificateAttr.issuer->a.rdnSequence);
   *rdnp = rdn = AsnListNew(sizeof (void*));
   atadvp = (AttributeTypeAndDistinguishedValue **)AsnListAppend(rdn);
   *atadvp=atadv=
        (AttributeTypeAndDistinguishedValue *)
             calloc(1, sizeof(AttributeTypeAndDistinguishedValue));
   atadv->type.octetLen = sizeof(ATADVtypeIssuer);
   atadv->type.octs = ATADVtypeIssuer;

   atadv->value.value = &asnATADVvalueIssuer;
SetAnyTypeByOid(&(atadv->value), &noRevAvail);

   atadv->valuesWithContext = AsnListNew(sizeof (void*));
   atadvsosp= (AttributeTypeAndDistinguishedValueSetOfSeq **)AsnListAppend(atadv->valuesWithContext);
   *atadvsosp=atadvsos=
       (AttributeTypeAndDistinguishedValueSetOfSeq *)
           calloc(1, sizeof(AttributeTypeAndDistinguishedValueSetOfSeq));
   atadvsos->distingAttrValue.value = &asnATADVdistvalueIssuer;
   SetAnyTypeByInt(&(atadvsos->distingAttrValue), externalIdentifierType);

   atadvsos->contextList = AsnListNew(sizeof (void*));
   atadvsossop = (Context **)AsnListAppend(atadvsos->contextList);
   *atadvsossop = atadvsosso = (Context *)calloc(1, sizeof(Context));
   atadvsosso->contextType.octetLen = sizeof(contextTypeIssuer);
   atadvsosso->contextType.octs = contextTypeIssuer;
   atadvsosso->contextValues = AsnListNew(sizeof (void*));
   anyp = (AsnAny **)AsnListAppend(atadvsosso->contextValues);
   contextValue1Issuer.value = &contextValue1OctsIssuer;
   SetAnyTypeByInt(&(contextValue1Issuer), externalIdentifierType);
   *anyp = any = &contextValue1Issuer;

   atadvsosso->fallback = &True;

   atadv->primaryDistinguished = FALSE;

   x509CertificateAttr.serialNumber = &certSerialNumber;

   /*
   ** Print the Certificate Data Object
   */
   PrintCIOChoice(stdout, cio, 3);

   /*
   ** BER Encode the Certificate Data Object
   */
   *BERLength = BEncCIOChoiceContent(gb,cio);
   GenBufResetInReadMode(gb);

   l = 0;
   memcpy(BER, GenBufGetSeg(gb, &l), *BERLength);
```

```
}

/*
** Example of Code for Decoding the BER
*/
Path_to_X509_Certificate(unsigned char *BER, unsigned int BERLength)
{
   SBuf b;
   GenBuf *gb;
   unsigned int bytesDecoded = 0;
   ENV_TYPE env;
   AsnTag tagId0;
   AsnLen elmtLen0;
   CIOChoice *cio;
   CertificateChoice *certificate;
   CertificateObject_X509CertificateAttributes* x509Certificate;
   X509CertificateAttributes* typeAttributes;
   ObjectValue* value;
   unsigned int i, pathLength;
   unsigned char *path;
if(setjmp(env)!= 0) exit(0);

   cio = calloc(1, sizeof(CIOChoice));

   SBufInstallData(&b, BER, BERLength);
   SBuftoGenBuf(&b, &gb);
   tagId0 = BDecTag(gb, &bytesDecoded, env);
   elmtLen0 = BDecLen(gb, &bytesDecoded, env);

   /*
   ** Decode the X.509 Certificate
   */
   BDecCIOChoiceContent(gb, tagId0, elmtLen0, cio, &bytesDecoded, env);

   /*
   ** Find the path to the certificate
   */
   certificate = (CertificateChoice *)(cio->a.certificates->a.objects->first->data);

   x509Certificate = certificate->a.x509Certificate;

   typeAttributes = x509Certificate->typeAttributes;

   value = typeAttributes->value;

   printf("Path to Certificate: ");
   pathLength = value->a.indirect->a.path->efidOrPath.octetLen;
   path = value->a.indirect->a.path->efidOrPath.octs;
   for(i = 0; i < pathLength; i+=2)
      printf("0x%02x%02x ", path[i], path[i+1]);
   printf("\n");
}
```

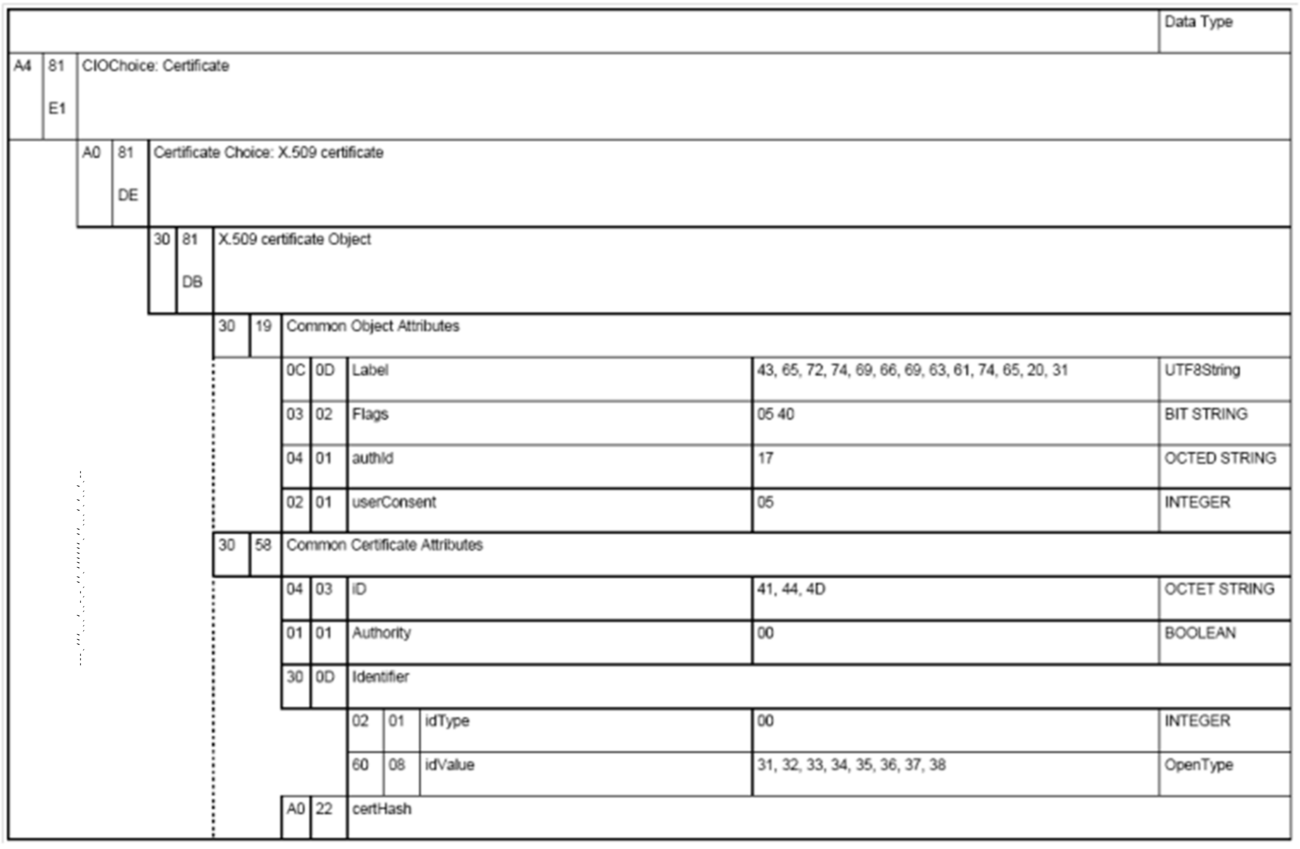
E.4.4 BER encoding

```
<BER>
0xa4,0x81,0xe1,0xa0,0x81,0xde,0x30,0x81,0xdb,0x30,0x19,0x0c,0x0d,0x43,0x65,0x72,
0x74,0x69,0x66,0x69,0x63,0x61,0x74,0x65,0x20,0x31,0x03,0x02,0x06,0x40,0x04,0x01,
0x17,0x02,0x01,0x05,0x30,0x58,0x04,0x03,0x41,0x44,0x4d,0x01,0x01,0x00,0x30,0x0d,
0x02,0x01,0x00,0x60,0x08,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0xa0,0x22,0xa0,
0x0c,0x30,0x0a,0x06,0x03,0x11,0x22,0x33,0x60,0x03,0x33,0x22,0x11,0xa1,0x0c,0x30,
0x0a,0x87,0x04,0xc0,0xa8,0x2d,0x01,0x02,0x02,0x33,0x33,0x03,0x04,0x00,0x99,0x88,
0x77,0xa1,0x05,0x03,0x03,0x06,0x20,0x00,0xa2,0x14,0x30,0x08,0x02,0x01,0x05,0x60,


0x03,0x61,0x62,0x63,0x30,0x08,0x02,0x01,0x05,0x60,0x03,0x78,0x79,0x7a,0xa1,0x64,
0x30,0x62,0x30,0x06,0x04,0x04,0x3f,0x00,0x40,0x42,0x30,0x28,0x31,0x26,0x30,0x24,
0x06,0x02,0x33,0x44,0x60,0x02,0x44,0x33,0x31,0x1a,0x30,0x18,0xa0,0x04,0x60,0x02,
0x55,0x77,0x31,0x10,0x30,0x0e,0x06,0x02,0x78,0x56,0x31,0x05,0x60,0x03,0x87,0x65,
0x43,0x01,0x01,0xff,0xa0,0x2a,0x30,0x28,0x31,0x26,0x30,0x24,0x06,0x02,0x55,0x66,
0x60,0x02,0x66,0x77,0x31,0x1a,0x30,0x18,0xa0,0x04,0x60,0x02,0x88,0x99,0x31,0x10,
0x30,0x0e,0x06,0x02,0x78,0x56,0x31,0x05,0x60,0x03,0x78,0x56,0x34,0x01,0x01,0xff,
0x02,0x02,0x57,0x68
</BER>
```

Table E.2 is a diagrammatic representation of the BER encoding

Table E.2 — EF.CD of X.509 certificate

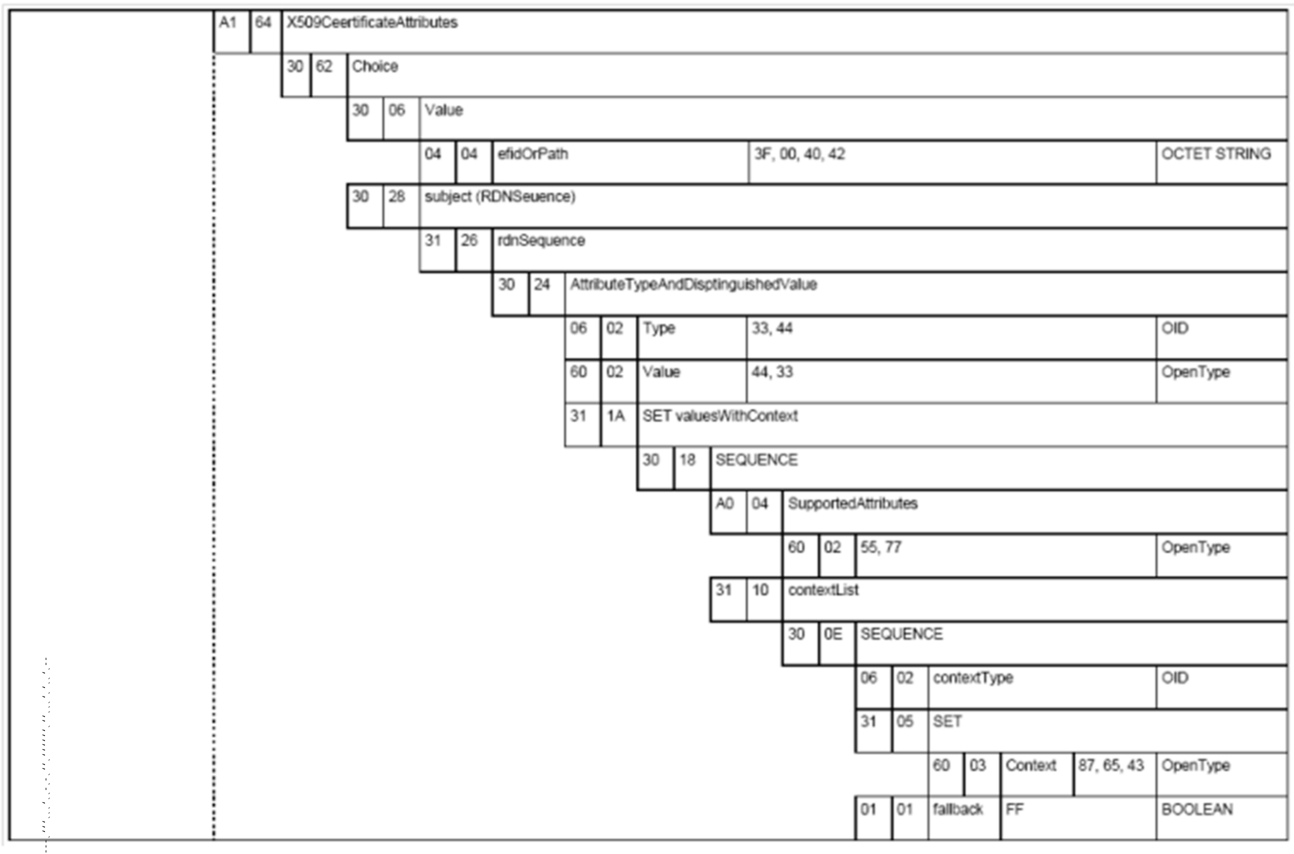


Tag	Length			Data Type
A4	81 E1	CIOChoice: Certificate		
A0	81 DE	Certificate Choice: X.509 certificate		
30	81 DB	X.509 certificate Object		
30	19	Common Object Attributes		
0C	0D	Label	43, 65, 72, 74, 69, 66, 69, 63, 61, 74, 65, 20, 31	UTF8String
03	02	Flags	05 40	BIT STRING
04	01	authId	17	OCTED STRING
02	01	userConsent	05	INTEGER
30	58	Common Certificate Attributes		
04	03	iD	41, 44, 4D	OCTET STRING
01	01	Authority	00	BOOLEAN
30	0D	Identifier		
02	01	idType	00	INTEGER
60	08	idValue	31, 32, 33, 34, 35, 36, 37, 38	OpenType
A0	22	certHash		

Table E.2 (*continued*)

				A0	0C	hashAlg						
						30	0A					
								06	03	Algorithm	11, 22, 33	OID
								60	03	parameters	33, 22, 11	OpenType
				A1	0C	certID						
						30	0A					
								87	04	Issuer(iPAddress)	C0, A8, 2D, 01	OCTET STRING
								02	02	serialNumber	33, 33	INTEGER
				03	04	hashVal					00, 99, 88, 77	INTEGER
		A1	05	trustUsage								
				03	03	keyUsage					06, 20, 00	BIT STRING
		A2	14	identifiers								
				30	08	CredentialIdentifier						
						02	01	idType			05	INTEGER
						60	03	idValue			61, 62, 63	OpenType
				30	08	CredentialIdentifier						
						02	01	idType			05	INTEGER
						60	03	idValue			78, 79, 7A	OpenType

Table E.2 (*continued*)



	A1	64	X509CeertificateAttributes		
	30	62	Choice		
	30	06	Value		
	04	04	efidOrPath	3F, 00, 40, 42	OCTET STRING
	30	28	subject (RDNSeuence)		
	31	26	rdnSequence		
	30	24	AttributeTypeAndDisptinguishedValue		
	06	02	Type	33, 44	OID
	60	02	Value	44, 33	OpenType
	31	1A	SET valuesWithContext		
	30	18	SEQUENCE		
	A0	04	SupportedAttributes		
	60	02		55, 77	OpenType
	31	10	contextList		
	30	0E	SEQUENCE		
	06	02	contextType		OID
	31	05	SET		
	60	03	Context	87, 65, 43	OpenType
	01	01	fallback	FF	BOOLEAN

Table E.2 (*continued*)

		A0	2A																			issuer (RDNSeuence)		
				30	28																	Choice :		
						31	26															SET of AttrivuteTypeAndValue		
								30	24															
										06	02											Type	55, 66	OID
										60	02											Value	66, 77	OpenType
										31	1A											SET valuesWithContext		
												30	18									SEQUENCE		
														A0	04							SupportedAttributes		
																60	02						88, 99	OpenType
														31	10							SET contextList		
																30	0E					SEQIEMCE		
																		06	02			contextType	78, 56	OID
																		31	05			SET Cotext		
																				60	03	Context	78, 56, 34	
																		01	01			fallback	FF	BOOLEAN
		02	02																			serial Number	57, 68	INTEGER

E.5 Encoding of the ESIGN cryptographic information application

E.5.1 Cryptographic information application example description

The encoding is an example of a cryptographic information application for the ESIGN as described in CWA 14890-1:2004, Clause 16.

E.5.2 ASN.1 encoding of the IAS cryptographic information application

```
{ -- SEQUENCE OF --
      privateKeys path { -- SEQUENCE --
               efidOrPath '4001'H
            },
      publicKeys path { -- SEQUENCE --
               efidOrPath '4002'H
            },
      certificates path { -- SEQUENCE --
               efidOrPath '4005'H
            },
      trustedCertificates path { -- SEQUENCE --
               efidOrPath '4004'H
            },
      dataContainerObjects path { -- SEQUENCE --
               efidOrPath '4006'H
            },
      authObjects path { -- SEQUENCE --
               efidOrPath '4003'H
            }
   }
```

```
cardInfo { -- SEQUENCE --
      version 2,
      serialNumber '0102030405060708'H,
      manufacturerID '41434d45'H  -- "ACME" --,
      label '5369676e6174757265204170706c69636174696f6e'H  -- "Signature Application" --,
      cardflags '60'H,
      seInfo { -- SEQUENCE OF --
         { -- SEQUENCE --
            se 1,
            aid 'a000000167455349474e'H
         },
         { -- SEQUENCE --
            se 2,
            aid 'a000000167455349474e'H
         }
      },
      supportedAlgorithms { -- SEQUENCE OF --
         { -- SEQUENCE --
            reference 1,
            algorithm 544,
            parameters ''H  -- "" --,
            supportedOperations '02'H,
            objId {0 1 3 14 3 2 26},
            algRef 16
         },
         { -- SEQUENCE --
            reference 2,
            algorithm -2147483648,
            parameters ''H  -- "" --,
            supportedOperations '40'H,
            objId {0 1 3 36 3 4 3 2 1},
            algRef 17
         },
         { -- SEQUENCE --
            reference 3,
            algorithm 544,
            parameters ''H  -- "" --,
            supportedOperations '40'H,
            objId {0 1 2 72 113 37 1 1 5},
            algRef 18
         },
         { -- SEQUENCE --
            reference 4,
            algorithm -2147483647,
            parameters ''H  -- "" --,
            supportedOperations '50'H,
            objId {0 1 3 36 7 2 1 1},
            algRef 23
         },
         { -- SEQUENCE --
            reference 5,
            algorithm -2147483646,
            parameters ''H  -- "" --,
            supportedOperations '10'H,
            objId {0 1 3 36 3 4 3 2 1}
         }
      },
      issuerId '4d61696e20537472656574204261616e6b'H  -- "Main Street Bank" --,
      holderId '53616c6c7920477265656e'H  -- "Sally Green" --,
      lastUpdate generalizedTime '31393835313130363231303632372e335a'H  -- "19851106210627.3Z" --,
        preferredLanguage '4573706572616e746f'H  -- "Esperanto" --
     }

AuthenticationObjects { -- SEQUENCE OF --
         pwd { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '476c6f62616c2050617373776f7264'H  -- "Global Password" --,
                  flags '40'H,
                  authId '03'H,
                  accessControlRules { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        accessMode '20'H,
                        securityCondition authReference { -- SEQUENCE --
                              authMethod 'c0'H,
                              seIdentifier 2
                           }
                     }
                  }
               },
               classAttributes { -- SEQUENCE --
                  authId '01'H,
                  authReference 1,
                  seIdentifier 2
               },
               typeAttributes { -- SEQUENCE --
                  pwdFlags '08'H,
                  pwdType 0,
                  minLength 4,
                  storedLength 0,
maxLength 8,
                  pwdReference 1,
                  padChar '00'H  -- " " --,
                  path { -- SEQUENCE --
                     efidOrPath ''H  -- "" --
                  }
               }
            },
         pwd { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '5369676e61747572652050617373776f7264'H  -- "Signature Password" --,
                  flags '40'H,
                  authId '00'H,
                  accessControlRules { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        accessMode '20'H,
                        securityCondition authReference { -- SEQUENCE --
                              authMethod 'c0'H,
                              seIdentifier 2
                           }
                     }
                  }
               },
               classAttributes { -- SEQUENCE --
                  authId '02'H,
                  authReference 129,
                  seIdentifier 2
               },
               typeAttributes { -- SEQUENCE --
                  pwdFlags '48'H,
                  pwdType 0,
                  minLength 6,
                  storedLength 0,
                  maxLength 8,
                  pwdReference 129,
                  padChar '00'H  -- " " --,
                  path { -- SEQUENCE --
                     efidOrPath '3f003f01'H
                  }
               }
            },
         pwd { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label
'526573657474696e6720436f646520666f722074686520476c6f62616c2050617373776f7264'H
-- "Resetting Code for the Global Password" --,
                  flags '40'H,
                  authId '03'H,
                  accessControlRules { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        accessMode '20'H,
                        securityCondition authReference { -- SEQUENCE --
                              authMethod 'c0'H,
                              seIdentifier 2
                           }
                     }
                  }
               },
               classAttributes { -- SEQUENCE --
                  authId '03'H,
authReference 129,
                  seIdentifier 2
               },
               typeAttributes { -- SEQUENCE --
                  pwdFlags '4a'H,
                  pwdType 0,
                  minLength 8,
                  storedLength 0,
                  maxLength 8,
                  pwdReference 129,
                  padChar '00'H  -- " " --,
                  path { -- SEQUENCE --
                     efidOrPath ''H  -- "" --
                  }
               }
            }
      }

PrivateKeyObjects { -- SEQUENCE OF --
         privateRSAKey { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '5369676e6174757265204b6579'H  -- "Signature Key" --,
                  flags '00'H,
                  authId '02'H,
                  userConsent 1,
                  accessControlRules { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        accessMode '20'H,
                        securityCondition or { -- SEQUENCE OF --
                              and { -- SEQUENCE OF --
                                    authId '02'H,
                                    authReference { -- SEQUENCE --
                                          authMethod 'a0'H,
                                          seIdentifier 1
                                       }
                                 },
                              and { -- SEQUENCE OF --
                                    authId '01'H,
                                    authReference { -- SEQUENCE --
                                          authMethod ''H,
                                          seIdentifier 2
                                       }
                                 }
                           }
                        }
                     }
                  },
                  classAttributes { -- SEQUENCE --
                     iD '01'H,
                     usage '30'H,
                     native TRUE,
                     accessFlags 'b8'H,
                     keyReference 132,
                     algReference { -- SEQUENCE OF --
                        2,
                        3
                     }
                  },
                  subClassAttributes { -- SEQUENCE –
authReference 129,
                     seIdentifier 2
                  },
                  typeAttributes { -- SEQUENCE --
                     pwdFlags '4a'H,
                     pwdType 0,
                     minLength 8,
                     storedLength 0,
                     maxLength 8,
                     pwdReference 129,
                     padChar '00'H  -- " " --,
                     path { -- SEQUENCE --
                        efidOrPath ''H  -- "" --
                     }
                  }
               }
         }

PrivateKeyObjects { -- SEQUENCE OF --
         privateRSAKey { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '5369676e6174757265204b6579'H  -- "Signature Key" --,
                  flags '00'H,
                  authId '02'H,
                  userConsent 1,
                  accessControlRules { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        accessMode '20'H,
                        securityCondition or { -- SEQUENCE OF --
                              and { -- SEQUENCE OF --
                                    authId '02'H,
                                    authReference { -- SEQUENCE --
                                          authMethod 'a0'H,
                                          seIdentifier 1
                                       }
                                 },
                              and { -- SEQUENCE OF --
                                    authId '01'H,
                                    authReference { -- SEQUENCE --
                                          authMethod ''H,
                                          seIdentifier 2
                                       }
                                 }
                           }
                     }
                  }
               },
               classAttributes { -- SEQUENCE --
                  iD '01'H,
                  usage '30'H,
                  native TRUE,
                  accessFlags 'b8'H,
                  keyReference 132,
                  algReference { -- SEQUENCE OF --
                     2,
                     3
                  }
               },
               subClassAttributes { -- SEQUENCE –
               },
               typeAttributes { -- SEQUENCE --
                  value indirect path { -- SEQUENCE --
                           efidOrPath ''H  -- "" --
                        },
                  modulusLength 1024
               }
            },
         privateRSAKey { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '534b2e4943432e415554'H  -- "SK.ICC.AUT" --,
                  flags '00'H,
                  authId ''H  -- "" --,
                  userConsent 1
               },
               classAttributes { -- SEQUENCE --
                  iD '02'H,
                  usage '50'H,
                  native TRUE,
                  accessFlags 'b0'H,
                  keyReference 17,
                  algReference { -- SEQUENCE OF --
                     4
                  }
               },
               subClassAttributes { -- SEQUENCE --

               },
               typeAttributes { -- SEQUENCE --
                  value indirect path { -- SEQUENCE --
                           efidOrPath ''H  -- "" --
                        },
                  modulusLength 1024
```

```
                }
            }
        }


    PublicKeyOjects { -- SEQUENCE OF --
             publicRSAKey { -- SEQUENCE --
                   commonObjectAttributes { -- SEQUENCE --
                      label '504b2e5243412e43532d415554'H  -- "PK.RCA.CS-AUT" --,
                      flags '40'H,
                      authId ''H  -- "" --
                   },
                   classAttributes { -- SEQUENCE --
                      iD '03'H,
                      usage '01'H,
                      native TRUE,
                      keyReference 11
                   },
                   subClassAttributes { -- SEQUENCE --

                   },
                   typeAttributes { -- SEQUENCE --
                      value indirect path { -- SEQUENCE --
                               efidOrPath ''H  -- "" --
                            },
                      modulusLength 1024
                   }
                }
        }


    CertificateOjects { -- SEQUENCE OF --
             x509Certificate { -- SEQUENCE --
                   commonObjectAttributes { -- SEQUENCE --
                      label '436572746966696361746520666f72205369676e617475726520536572766963 65'H
                                                   -- "Certificate for Signature Service" --
                   },
                   classAttributes { -- SEQUENCE --
                      iD '01'H,
                      authority FALSE
                   },
                   typeAttributes { -- SEQUENCE --
                      value indirect path { -- SEQUENCE --
                               efidOrPath '3f003f01c000'H
                            }
                   }
                },
             x509Certificate { -- SEQUENCE --
                   commonObjectAttributes { -- SEQUENCE --
                      label '434120436572746966696361746520666f72205369676e6174757265205365727669 6365'H
                                                      -- "CA Certificate for Signature Service" --
                   },
                   classAttributes { -- SEQUENCE --
                      iD '01'H,
                      authority TRUE
                   },
                   typeAttributes { -- SEQUENCE --
                      value indirect path { -- SEQUENCE --
                               efidOrPath '3f003f01c608'H
                            }
```

```
               }
            },
         x509Certificate { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '435f43562e4943432e415554'H  -- "C_CV.ICC.AUT" --
               },
               classAttributes { -- SEQUENCE --
                  iD '02'H,
                  authority FALSE
               },
               typeAttributes { -- SEQUENCE --
                  value indirect path { -- SEQUENCE --
                           efidOrPath '3f002f03'H
                        }
               }
            }
      }

DataContainerObjects { -- SEQUENCE OF --
         iso7816DO { -- SEQUENCE --
               commonObjectAttributes { -- SEQUENCE --
                  label '446973706c6179204d657373616765'H  -- "Display Message" --,
                  flags 'c0'H,
                  accessControlRules { -- SEQUENCE OF --
                     { -- SEQUENCE --
                        accessMode '40'H,
                        securityCondition or { -- SEQUENCE OF –
                              and { -- SEQUENCE OF --
                                    authId '01'H,
                                    authReference { -- SEQUENCE --
                                          authMethod '20'H,
                                          seIdentifier 1
                                       }
                                 },
                              and { -- SEQUENCE OF --
                                    authId '01'H,
                                    authReference { -- SEQUENCE --
                                          authMethod 'e0'H,
                                          seIdentifier 2
                                       }
                                 }
                           }
                     },
                     { -- SEQUENCE --
                        accessMode '80'H,
                        securityCondition authReference { -- SEQUENCE --
                              authMethod 'c0'H,
                              seIdentifier 2
                           }
                     }
                  }
               },
               classAttributes { -- SEQUENCE --
                  applicationName 'a000000167455349474e'H
               },
               typeAttributes indirect path { -- SEQUENCE --
                        efidOrPath '3f003f01d000'H
                     }
            }
      }
```

E.5.3 Code from the ASN.1 for encoding a decoding BER

Not provided

E.5.4 BER encoding

```
<ESIGN_EF_OD>
0xa0,0x3c,0x30,0x08,0xa0,0x06,0x30,0x04,0x04,0x02,0x40,0x01,0x30,0x08,0xa1,0x06,
0x30,0x04,0x04,0x02,0x40,0x02,0x30,0x08,0xa4,0x06,0x30,0x04,0x04,0x02,0x40,0x05,
0x30,0x08,0xa5,0x06,0x30,0x04,0x04,0x02,0x40,0x04,0x30,0x08,0xa7,0x06,0x30,0x04,
0x04,0x02,0x40,0x06,0x30,0x08,0xa8,0x06,0x30,0x04,0x04,0x02,0x40,0x03
</ESIGN_EF_OD>

<ESIGN_EF_CardInfo>
0x02,0x01,0x02,0x04,0x08,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x0c,0x04,0x41,
0x43,0x4d,0x45,0x80,0x15,0x53,0x69,0x67,0x6e,0x61,0x74,0x75,0x72,0x65,0x20,0x41,
0x70,0x70,0x6c,0x69,0x63,0x61,0x74,0x69,0x6f,0x6e,0x03,0x02,0x05,0x60,0x30,0x22,
0x30,0x0f,0x02,0x01,0x01,0x04,0x0a,0xa0,0x00,0x00,0x01,0x67,0x45,0x53,0x49,0x47,
0x4e,0x30,0x0f,0x02,0x01,0x02,0x04,0x0a,0xa0,0x00,0x00,0x01,0x67,0x45,0x53,0x49,
0x47,0x4e,0xa2,0x7e,0x30,0x17,0x02,0x01,0x01,0x02,0x02,0x02,0x20,0x60,0x00,0x60,
0x01,0x02,0x06,0x06,0x01,0x03,0x0e,0x03,0x02,0x1a,0x02,0x01,0x10,0x30,0x18,0x02,
0x01,0x02,0x02,0x01,0x00,0x60,0x00,0x60,0x01,0x40,0x06,0x08,0x01,0x03,0x24,0x03,
0x04,0x03,0x02,0x01,0x02,0x01,0x11,0x30,0x19,0x02,0x01,0x03,0x02,0x02,0x02,0x20,
0x60,0x00,0x60,0x01,0x40,0x06,0x08,0x01,0x02,0x48,0x71,0x25,0x01,0x01,0x05,0x02,
0x01,0x12,0x30,0x17,0x02,0x01,0x04,0x02,0x01,0x01,0x60,0x00,0x60,0x01,0x50,0x06,
0x07,0x01,0x03,0x24,0x07,0x02,0x01,0x01,0x02,0x01,0x17,0x30,0x15,0x02,0x01,0x05,
0x02,0x01,0x02,0x60,0x00,0x60,0x01,0x10,0x06,0x08,0x01,0x03,0x24,0x03,0x04,0x03,
0x02,0x01,0x83,0x10,0x4d,0x61,0x69,0x6e,0x20,0x53,0x74,0x72,0x65,0x65,0x74,0x20,
0x42,0x61,0x6e,0x6b,0x84,0x0b,0x53,0x61,0x6c,0x6c,0x79,0x20,0x47,0x72,0x65,0x65,
0x6e,0xa5,0x13,0x18,0x11,0x31,0x39,0x38,0x35,0x31,0x31,0x30,0x36,0x32,0x31,0x30,
0x36,0x32,0x37,0x2e,0x33,0x5a,0x13,0x09,0x45,0x73,0x70,0x65,0x72,0x61,0x6e,0x74,
0x6f
</ESIGN_EF_CardInfo>

<ESIGN_EF_AOD>
0xa0,0x82,0x01,0x1a,0x30,0x50,0x30,0x26,0x0c,0x0f,0x47,0x6c,0x6f,0x62,0x61,0x6c,
0x20,0x50,0x61,0x73,0x73,0x77,0x6f,0x72,0x64,0x03,0x01,0x00,0x04,0x01,0x03,0x30,
0x0d,0x30,0x0b,0x03,0x01,0x00,0x30,0x06,0x03,0x01,0x80,0x02,0x01,0x02,0x30,0x09,
0x04,0x01,0x01,0x02,0x01,0x01,0x80,0x01,0x02,0xa1,0x1b,0x30,0x19,0x03,0x01,0x00,
0x0a,0x01,0x00,0x02,0x01,0x04,0x02,0x01,0x00,0x02,0x01,0x08,0x80,0x01,0x01,0x04,
0x01,0x00,0x30,0x02,0x04,0x00,0x30,0x5a,0x30,0x29,0x0c,0x12,0x53,0x69,0x67,0x6e,
0x61,0x74,0x75,0x72,0x65,0x20,0x50,0x61,0x73,0x73,0x77,0x6f,0x72,0x64,0x03,0x01,
0x00,0x04,0x01,0x00,0x30,0x0d,0x30,0x0b,0x03,0x01,0x00,0x30,0x06,0x03,0x01,0x80,
0x02,0x01,0x02,0x30,0x0a,0x04,0x01,0x02,0x02,0x02,0x00,0x81,0x80,0x01,0x02,0xa1,
0x21,0x30,0x1f,0x03,0x02,0x06,0x40,0x0a,0x01,0x00,0x02,0x01,0x06,0x02,0x01,0x00,
0x02,0x01,0x08,0x80,0x02,0x00,0x81,0x04,0x01,0x00,0x30,0x06,0x04,0x04,0x3f,0x00,
0x3f,0x01,0x30,0x6a,0x30,0x3d,0x0c,0x26,0x52,0x65,0x73,0x65,0x74,0x74,0x69,0x6e,
0x67,0x20,0x43,0x6f,0x64,0x65,0x20,0x66,0x6f,0x72,0x20,0x74,0x68,0x65,0x20,0x47,
0x6c,0x6f,0x62,0x61,0x6c,0x20,0x50,0x61,0x73,0x73,0x77,0x6f,0x72,0x64,0x03,0x01,
0x00,0x04,0x01,0x03,0x30,0x0d,0x30,0x0b,0x03,0x01,0x00,0x30,0x06,0x03,0x01,0x80,
0x02,0x01,0x02,0x30,0x0a,0x04,0x01,0x03,0x02,0x02,0x00,0x81,0x80,0x01,0x02,0xa1,
0x1d,0x30,0x1b,0x03,0x02,0x06,0x40,0x0a,0x01,0x00,0x02,0x01,0x08,0x02,0x01,0x00,
0x02,0x01,0x08,0x80,0x02,0x00,0x81,0x04,0x01,0x00,0x30,0x02,0x04,0x00
</ESIGN_EF_AOD>

<ESIGN_EF_PrKD>
0xa0,0x81,0xa8,0x30,0x67,0x30,0x3b,0x0c,0x0d,0x53,0x69,0x67,0x6e,0x61,0x74,0x75,
0x72,0x65,0x20,0x4b,0x65,0x79,0x03,0x01,0x00,0x04,0x01,0x02,0x02,0x01,0x01,0x30,
0x21,0x30,0x1f,0x03,0x01,0x00,0xa2,0x1a,0xa1,0x0b,0x04,0x01,0x02,0x30,0x06,0x03,
0x01,0x80,0x02,0x01,0x01,0xa1,0x0b,0x04,0x01,0x01,0x30,0x06,0x03,0x01,0x00,0x02,
0x01,0x02,0x30,0x18,0x04,0x01,0x01,0x03,0x01,0x00,0x01,0x01,0xff,0x03,0x01,0x80,
0x02,0x02,0x00,0x84,0xa1,0x06,0x02,0x01,0x02,0x02,0x01,0x03,0xa0,0x02,0x30,0x00,
0xa1,0x0a,0x30,0x08,0x30,0x02,0x04,0x00,0x02,0x02,0x04,0x00,0x30,0x3d,0x30,0x14,
0x0c,0x0a,0x53,0x4b,0x2e,0x49,0x43,0x43,0x2e,0x41,0x55,0x54,0x03,0x01,0x00,0x04,
0x00,0x02,0x01,0x01,0x30,0x15,0x04,0x01,0x02,0x03,0x02,0x06,0x40,0x01,0x01,0xff,
0x03,0x01,0x80,0x02,0x01,0x11,0xa1,0x03,0x02,0x01,0x04,0xa0,0x02,0x30,0x00,0xa1,
0x0a,0x30,0x08,0x30,0x02,0x04,0x00,0x02,0x02,0x04,0x00
</ESIGN_EF_PrKD>

<ESIGN_EF_PuKD>
0xa0,0x36,0x30,0x34,0x30,0x14,0x0c,0x0d,0x50,0x4b,0x2e,0x52,0x43,0x41,0x2e,0x43,
0x53,0x2d,0x41,0x55,0x54,0x03,0x01,0x00,0x04,0x00,0x30,0x0c,0x04,0x01,0x03,0x03,
0x01,0x00,0x01,0x01,0xff,0x02,0x01,0x0b,0xa0,0x02,0x30,0x00,0xa1,0x0a,0x30,0x08,
0x30,0x02,0x04,0x00,0x02,0x02,0x04,0x00
</ESIGN_EF_PuKD>

<ESIGN_EF_CD>
0xa0,0x81,0xa3,0x30,0x3b,0x30,0x23,0x0c,0x21,0x43,0x65,0x72,0x74,0x69,0x66,0x69,
0x63,0x61,0x74,0x65,0x20,0x66,0x6f,0x72,0x20,0x53,0x69,0x67,0x6e,0x61,0x74,0x75,
0x72,0x65,0x20,0x53,0x65,0x72,0x76,0x69,0x63,0x65,0x30,0x06,0x04,0x01,0x01,0x01,
0x01,0x00,0xa1,0x0c,0x30,0x0a,0x30,0x08,0x04,0x06,0x3f,0x00,0x3f,0x01,0xc0,0x00,
0x30,0x3e,0x30,0x26,0x0c,0x24,0x43,0x41,0x20,0x43,0x65,0x72,0x74,0x69,0x66,0x69,
0x63,0x61,0x74,0x65,0x20,0x66,0x6f,0x72,0x20,0x53,0x69,0x67,0x6e,0x61,0x74,0x75,
0x72,0x65,0x20,0x53,0x65,0x72,0x76,0x69,0x63,0x65,0x30,0x06,0x04,0x01,0x01,0x01,
0x01,0xff,0xa1,0x0c,0x30,0x0a,0x30,0x08,0x04,0x06,0x3f,0x00,0x3f,0x01,0xc6,0x08,
0x30,0x24,0x30,0x0e,0x0c,0x0c,0x43,0x5f,0x43,0x56,0x2e,0x49,0x43,0x43,0x2e,0x41,
0x55,0x54,0x30,0x06,0x04,0x01,0x02,0x01,0x01,0x00,0xa1,0x0a,0x30,0x08,0x30,0x06,
0x04,0x04,0x3f,0x00,0x2f,0x03
</ESIGN_EF_CD>

<ESIGN_EF_DO>
0xa0,0x62,0xa0,0x60,0x30,0x44,0x0c,0x0f,0x44,0x69,0x73,0x70,0x6c,0x61,0x79,0x20,
0x4d,0x65,0x73,0x73,0x61,0x67,0x65,0x03,0x01,0x80,0x30,0x2e,0x30,0x1f,0x03,0x01,
0x00,0xa2,0x1a,0xa1,0x0b,0x04,0x01,0x01,0x30,0x06,0x03,0x01,0x00,0x02,0x01,0x01,
0xa1,0x0b,0x04,0x01,0x01,0x30,0x06,0x03,0x01,0x80,0x02,0x01,0x02,0x30,0x0b,0x03,
0x01,0x80,0x30,0x06,0x03,0x01,0x80,0x02,0x01,0x02,0x30,0x0c,0x0c,0x0a,0xa0,0x00,
0x00,0x01,0x67,0x45,0x53,0x49,0x47,0x4e,0xa1,0x0a,0x30,0x08,0x04,0x06,0x3f,0x00,
0x3f,0x01,0xd0,0x00
</ESIGN_EF_DO>
```

Bibliography

[1] A. Phillips, M. Davis, Ed. "*Tags for Identifying Languages, "IETF RFC 5646, September 2009*

[2] ANSI INCITS 4-1986 (R2007), *Information Systems — Coded Character Sets — 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)*

[3] R. Fielding, L. Masinter, "*Uniform Resource Identifier (URI): Generic Syntax," IETF RFC 3986, January 2005*

[4] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer, "OpenPGP Message Format," IETF RFC 4880, November 2007

[5] C. ELLISON, B. FRANTZ, B. LAMPSON, R. RIVEST, B. THOMAS, T. YLONEN, "*SPKI Certificate Theory*", IETF RFC 2693, September 1999

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

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

[8] RSA Laboratories, PKCS #11 v2.30: *Cryptographic Token Interface Standard*

[9] RSA Laboratories, PKCS #15 v1.1: *Cryptographic Token Information Syntax Standard*

[10] WAP Forum, *Wireless Application Protocol — Wireless Transport Layer Security Protocol Specification*, Version 06-Apr-2001

[11] ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*

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

[2] ANSI X9.68 is not available in the ANSI catalog.

ICS 35.240.15

Price based on 117 pages