INTERNATIONAL STANDARD

ISO/IEC 7816-15

First edition
2004-01-15

AMENDMENT 1
2007-11-15

Identification cards — Integrated circuit cards —

Part 15: Cryptographic information application

AMENDMENT 1: Examples of the use of the cryptographic information application

Cartes d'identification — Cartes à circuit intégré —

Partie 15: Application des informations cryptographiques

AMENDEMENT 1: Exemples d'emploi de l'application des informations cryptographiques

Reference number
ISO/IEC 7816-15:2004/Amd.1:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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

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

Amendment 1 to ISO/IEC 7816-15:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

Identification cards — Integrated circuit cards —

Part 15:
Cryptographic information application

AMENDMENT 1: Examples of the use of the cryptographic information application

Insert the following new annex after Annex D.

Annex E
(informative)

Examples of the use of the cryptographic information application

E.1 Introduction

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 above 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 BER 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

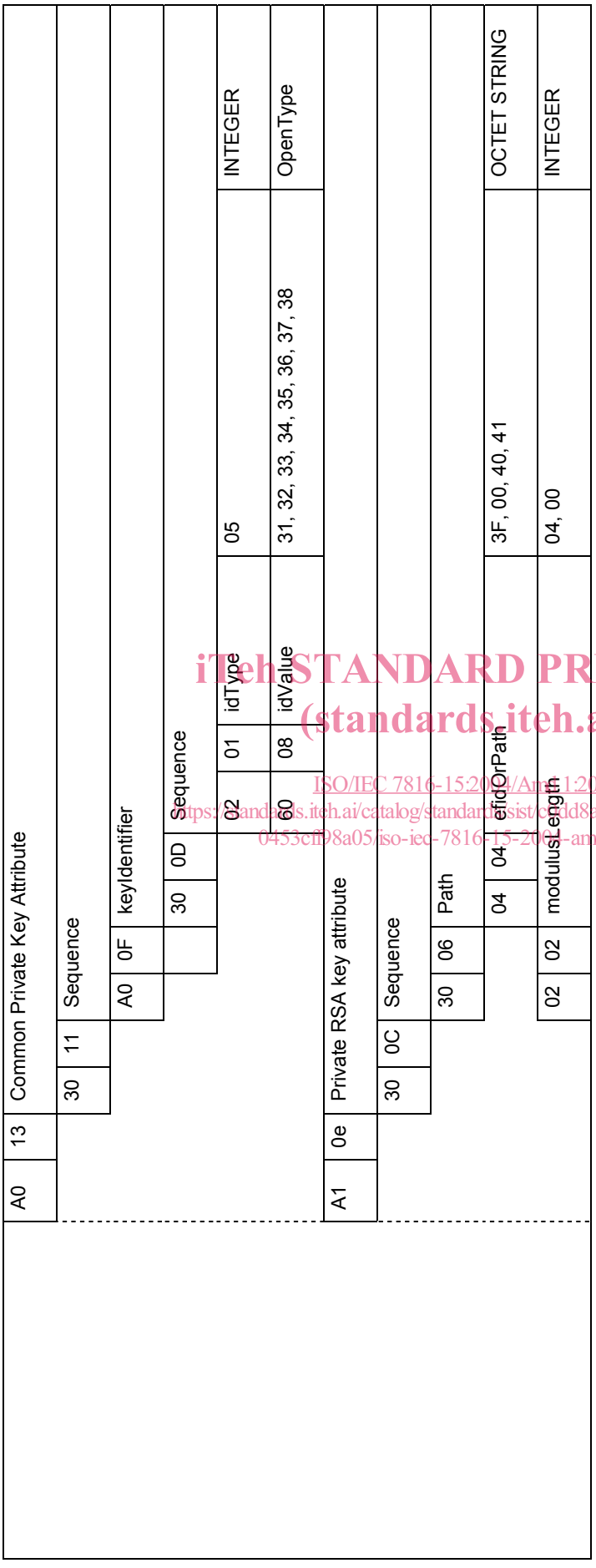

```
<EF_OD>
0xa0,0x51,0xa0,0x4f,0x30,0x4d,0x30,0x12,0x0c,0x04,0x4b,0x45,0x59,0x31,0x03,0x02,
0x05,0x80,0x04,0x03,0x41,0x44,0x4d,0x02,0x01,0x01,0x30,0x12,0x04,0x01,0x9b,0x03,
0x03,0x06,0x20,0x40,0x01,0x01,0xff,0x03,0x02,0x03,0x98,0x02,0x01,0x0a,0xa0,0x13,
0x30,0x11,0xa0,0x0f,0x30,0x0d,0x02,0x01,0x05,0x04,0x08,0x31,0x32,0x33,0x34,0x35,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



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 must 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 };
```