FINEID - S4-1

Implementation Profile 1

for Finnish Electronic ID Card

v 2.1A

Population Register Centre (VRK)
Certification Authority Services
P.O. Box 70
FIN-00581 Helsinki
Finland
http://www.fineid.fi

Authors

Name	Initials	Organization	E-mail
Antti Partanen	AP	VRK	antti.partanen@vrk.intermin.fi
Markku Sievänen	MaSi	Setec Oy	markku.sievanen@setec.com

Document history

Version	Date	Editor	Changes	Status
2.1A	21.10.2004	AP	Additional note on the encoding of EF.CIAInfo, value notation example of EF.CIAInfo corrected, example of ATR bytes updated, minor editorial corrections	Accepted
2.1	15.3.2004	AP	Optional elementary files EF.CD #4 (for useful certs) and EF(Private EmptyArea) implemented. Typical file sizes for Public and Private EmptyAreas increased, references to DF.Sandbox removed, references to ISO/IEC FDIS 7816-15 changed to ISO/IEC 7816-15, editorial corrections and additions	Accepted
2.0	12.8.2003	AP	Editorial changes, minor corrections	Accepted
1.2	11.6.2003	MaSi	First edition	Draft

Table of contents

1. Introduction 1
1.1. About FINEID specifications in general 1
2. FINEID S4-1 2
3. IC Card requirements 2
3.1. ATR bytes 2
4. FINEID application functionality 3
4.1. Private RSA keys 3
4.2. Card holder certificates 3
4.3. CA certificates 4
5. FINEID application 4
6. FINEID object classes 4
7. FINEID file relationships 5
8. FINEID file structure 7
File access methods and conditions 8
8.1. MF 8
Description 8
Access conditions (informative) 8
8.2. DF.ESIGN 8
Description 8
Access conditions 8
8.3. EF.CIAInfo 9
Description 9
Access conditions 9
Value notation example 9
8.4. EF.OD 10
Description 10
Access conditions 10
Value notation example 10
8.5. EF.AOD 11
Description 11
Access conditions 12
PIN-code settings 12
Value notation example 12
8.6. Authentication Object #1 (PIN 1) 13
Description 13
Access conditions 13
8.7. Authentication Object #2 (PIN 2) 14
Description 14

Access conditions....14
8.8. EF.PrKD....14
Description....14
Access conditions....14
Private key labels....14
Value notation example....14
8.9. Private RSA Key #1....16
Description....16
Access conditions....16
8.10. Private RSA key #2....16
Description....16
Access conditions....17
8.11. EF.CD #1....17
Description....17
Access conditions....17
Certificate label....17
Value notation example....17
8.12. Certificate #1....18
Description....18
Access conditions....19
8.13. Certificate #2....19
Description....19
Access conditions....19
8.14. EF.CD #2....19
Description....19
Access conditions....19
8.15. EF.CD #3 (trusted certs)....19
Description....19
Access conditions....20
Value notation example....20
8.16. CA Certificate #1....21
Description....21
Access conditions....21
8.17. CA Certificate #2....21
Description....21
Access conditions....21
8.18. EF.CD #4 (useful certs)....21
Description....21
Access conditions....22
8.19. EF.DCOD....22
Description....22
Access conditions....22

8.20. EF(UnusedSpace) 22
Description 22
Access conditions 23
Value notation example 23
8.21. EF(Public EmptyArea) 24
Description 24
Access conditions 24
Example 24
8.22. EF(Private EmptyArea) 24
Description 24
Access conditions 25
Example 25
9. Certificates 25

1. Introduction

This document describes an implementation profile of the FINEID S1 specification version 2.1. This implementation profile is for Finnish Electronic ID Cards and for other smart cards containing Citizen Qualified Certificates issued by Population Register Centre (VRK).

1.1. About FINEID specifications in general

The FINEID specifications are publicly available documents describing how to implement a public key infrastructure (PKI) using smart cards.

There is a straight correlation between the FINEID specifications, ISO/IEC 7816-15, IETF RFC 3280 (PKIX Certificate and CRL profile), and the PKCS standards. FINEID S1 specifies the framework for the content of an Electronic ID card. FINEID S2 describes the content of certificates. FINEID S4-1 and S4-2 are profiling documents. These documents specify the file and directory format for storing security-related information in smart cards (security tokens). The corresponding documents are listed in the table below.

FINEID document	FINEID comments	Based on
FINEID S1	Framework for the Electronic ID application in the smart card	ISO/IEC 7816-4 and ISO/IEC 7816-8
FINEID S2	CA-model and content of certificates published and administrated by Population Register Centre (VRK)	IETF RFC 3280 and ETSI TS 101862 Qualified certificate profile
FINEID S4-1	Implementation profile 1 for Finnish Electronic ID Card	ISO/IEC 7816-15, PKCS#15 v1.1, FINEID S1 and FINEID S2
FINEID S4-2	Implementation profile 2 for Organizational Usage	FINEID S4-1
FINEID S5	Directory specification	IETF RFC 2256, LDAPv2 and LDAPv3

FINEID S4-1 contains an implementation profile specifying how the FINEID S1 specification should be put into practice in FINEID context. FINEID S4-1 is mainly based on ISO/IEC 7816-15. However, because of ISO/IEC 7816-15 doesn't specify the free space management of the EID application, FINEID S4-1 uses EF(UnusedSpace) file defined in PKCS#15 v1.1 to solve this problem. Also the specification "Application Interface for smart cards used as Secure Signature Creation Devices, Part 1 – Basic requirements, CEN/ISSS, CWA 14890-1:2004(E), Version 1 Release 9 rev2" has influenced to this document.

Full names for the FINEID specifications are listed below:

- FINEID S1 - Electronic Identity Application, v2.1
- FINEID S2 – VRK (PRC) CA-model and certificate contents, v2.1
- FINEID S4-1 - Implementation Profile 1 for Finnish Electronic ID Card, v2.1A
- FINEID S4-2 - Implementation Profile 2 for Organizational Usage, v2.1A
- FINEID S5 – Directory Specification, v2.1

FINEID specifications are available at

- **http://www.fineid.fi**

The PKCS standards are available at

- **http://www.rsasecurity.com/rsalabs**

IETF PKIX documentation and other IETF RFC's are available at

- **http://www.ietf.org/rfc**

2. FINEID S4-1

FINEID S1 specifies the contents of an Electronic ID application. However, there are many options and features that are left for the EID application issuer to decide (number of private keys in the EID application, key lengths etc). This document contains an implementation profile complementing those options.

It should be noticed that the FINEID S1 and S4-1 documents specify the requirements for the EID application only. In a multiapplication smart card there may potentially exist several other applications in addition to the EID application.

3. IC Card requirements

The requirements for the EID application command interface are specified in FINEID S1.

Also other cards or microchips with the capability of having a FINEID application could be supported after checking other security elements of the token.

3.1. ATR bytes

FINEID application is designed to be platform independent and therefore following ATR bytes are listed here only as an example.

Following table describes typical ATR bytes of the card (according ISO 7816-3 and 7816-4).

Character	Value (hex)	Comment
TS	3B	Initial character: direct convention
T0	7B	Format character: '7' indicates that TA1, TB1 and TC1 are present, 'B' indicates the number of historical characters (11).
TA1	94	FI = 9, DI = 4, indicating max. 57 600 bit/sec at clock rate of 3,57 MHz.
TB1	00	VPP not required.
TC1	00	Indicates the amount of extra quardtime required.
T1	80	Historical characters, max.15 bytes (T1-T15): T1 = Category Indicator, 80 = status information, if present, is contained in an optional COMPACT-TLV data object. In this case no status info is provided.
T2	62	Pre-issuing data (tag 6, length 2): card operating system version.

T3	01	Card operating system minor version.
T4	51	Card operating system major version.
T5	56	Card issuer's data (tag 5, length 6)
T6	46	'F'
T7	69	'i'
T8	6E	'n'
T9	45	'E'
T10	49	'I'
T11	44	'D'

Note 1: Two bytes in pre-issuing data contain card operating system (COS) version numbers, not FINEID application version numbers.

Note 2: Interface characters and COS version numbers are token dependent.

Note 3: Other applications on the card might store additional or different historical bytes.

4. FINEID application functionality

4.1. Private RSA keys

The FINEID application shall contain two private RSA keys as specified in the table below.

Private key number	**Key label**	**X.509 key usage**	**Key length (in bits)**	**Public exponent**
1	'todentamis- ja salausavain' (FIN) 'auth. and encipherment key' (ENG) 'aut. och kryptering nyckel' (SWE)	digitalSignature + keyEncipherment + dataEncipherment	1024	65537 (F4)
2	'allekirjoitusavain' (FIN) 'signature key' (ENG) 'signatur nyckel' (SWE)	nonRepudiation	1024	3 (F0)

4.2. Card holder certificates

The following card holder certificates shall be stored into the FINEID application.

Corresponding user private key number	**Certificate label**	**X.509 key usage**
1	'todentamis- ja salausvarmenne' (FIN) 'auth. and encipherment cert.' (ENG) 'aut. och kryptering certifikat' (SWE)	digitalSignature + keyEncipherment + dataEncipherment
2	'allekirjoitusvarmenne' (FIN) 'signature certificate' (ENG) 'signatur certifikat' (SWE)	nonRepudiation

End entity certificates are described in the FINEID S2 specification.

4.3. CA certificates

Two CA certificates (keylength 2048 bit) shall be stored into the FINEID application. These can be used as starting points of trust for the card holder.

Root CA Certificate label	Signed by
'VRK Gov. Root CA' (FIN) 'VRK Gov. Root CA' (ENG) 'VRK Gov. Root CA' (SWE)	Self-signed

Intermediate CA Certificate label	Signed by
' VRK Gov. CA for Citizen Qualified Certificates' (FIN) ' VRK Gov. CA for Citizen Qualified Certificates' (ENG) ' VRK Gov. CA for Citizen Qualified Certificates' (SWE)	'VRK Gov. Root CA'

The contents of Root and CA certificates are described in the FINEID S2 specification.

5. FINEID application

The FINEID application is selected using following Application Identifier (AID):

```
A0 00 00 00 63 50 4B 43 53 2D 31 35
```

Because multiapplication smart cards contain multiple independent applications, FINEID application must be selected before further card access. When smart card reset occurs, FINEID application might not be the default smart card application. For security reasons, application selection resets also PIN –codes verification status.

6. FINEID object classes

This document defines four general classes of objects (check ISO/IEC 7816-15 for additional information):

- Key Information Objects,
- Certificate Information Objects,
- Data Container Information Objects and
- Authentication Information Objects.

All these object classes have sub-classes, e.g. Private Key Information is a sub-class of the Key Information Object. Objects can be private, meaning that they are protected against unauthorized access, or public. In FINEID application, access to private objects is defined by Access Conditions. Conditional access is usually achieved with PINs. Public objects are not protected from read-access.

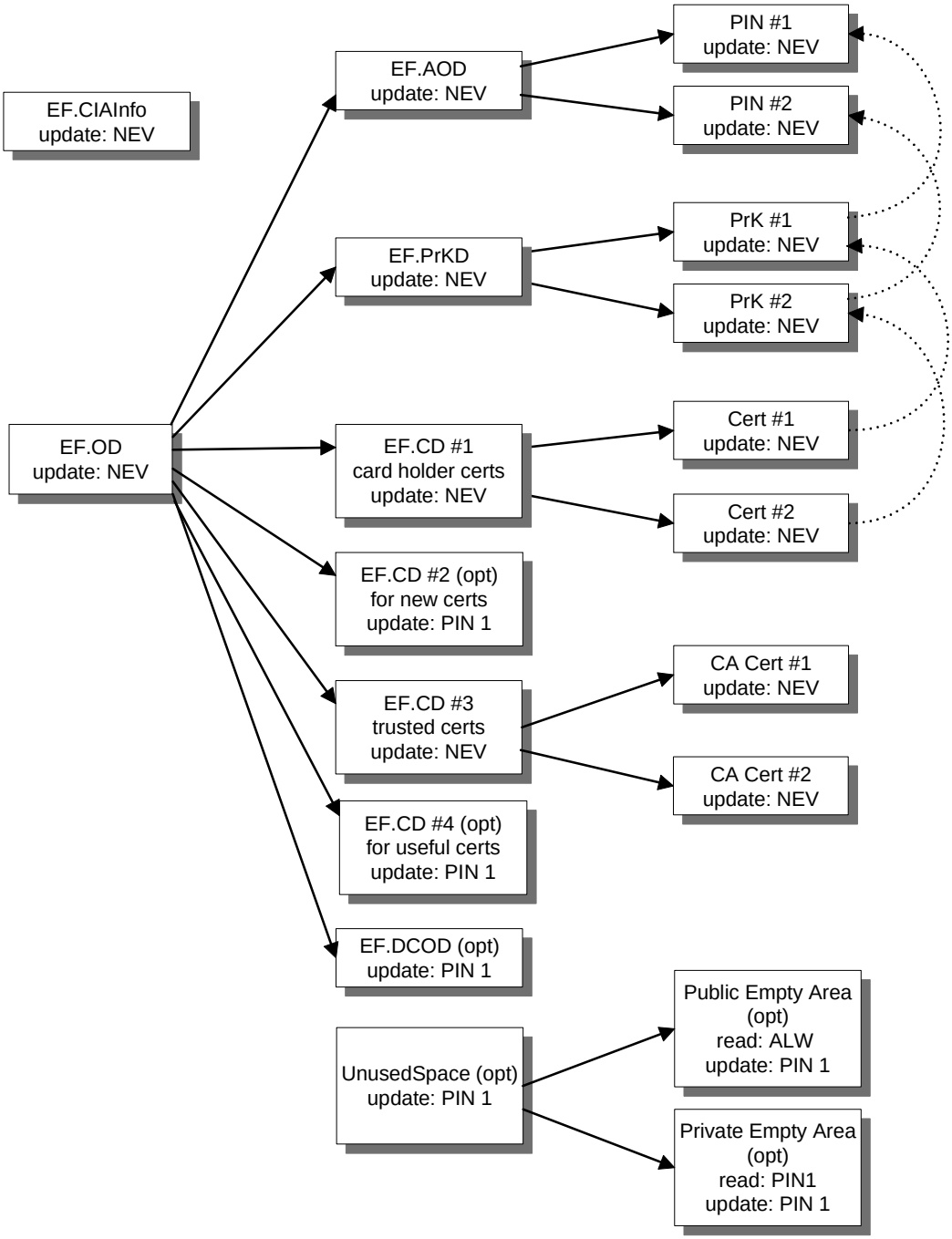
Any number of 'FF' octets may occur before, between or after the values of these objects without any meaning (i.e. as padding for unused space or deleted values).

7. FINEID file relationships

Abbreviations for file names:

OD	Object Directory
EF	Elementary File
PrKD	Private Key Directory
DF	Directory File
CD	Certificate Directory
Cert	Certificate
AOD	Authentication Object Directory
CA	Certification Authority
DCOD	Data Container Object Directory
PIN	Authentication Object
PrK	Private Key

The following figure shows the relationship between certain files (EF.OD, EF.PrKD, EF.CDs, EF.DCOD and EF.AOD) in the FINEID Application. EF.OD points to other EFs. Dashed arrows are explained below. The optional files are marked with "(opt)" keyword.



EF.PrKD contains cross-reference pointers to authentication objects (PINs) used to protect access to the keys. This is indicated by arrows between PINs and PrKs.

A certificate (#1 & #2) contains a public key whose corresponding private key also resides on the card, so the certificate contains the same identifier as the corresponding private key. This is indicated by arrows between Certs and PrKs.

8. FINEID file structure

The file structure of the FINEID application is described in the figure below. It is based on the ISO/IEC 7816-15 and PKCS#15 specification. Notice that the FINEID application must be selected prior to being able to access this file structure. The FINEID application may potentially exist in a multiapplication smart card or other interoperable token.

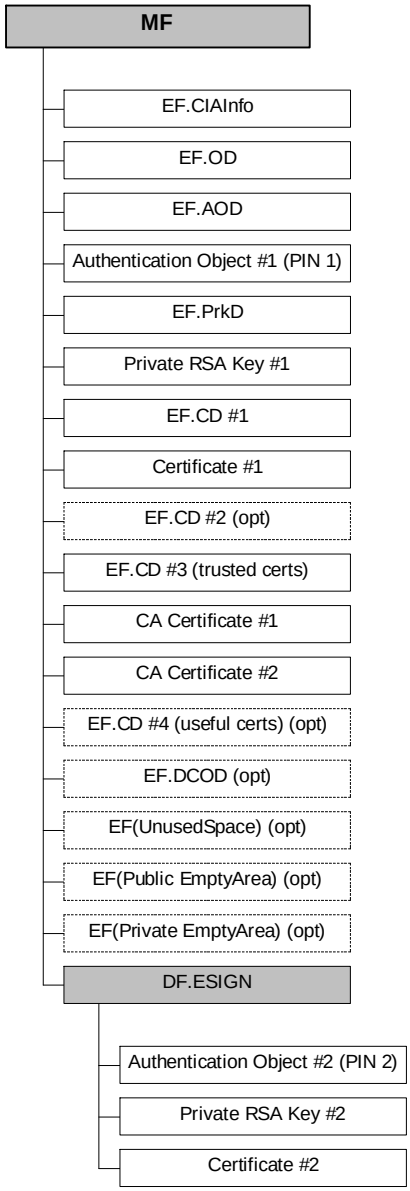


Figure 1. FINEID S4-1 file structure (optional files are marked using dashed borders)

File access methods and conditions

File type	Access method	Meaning
DF	Create	Allows new files both EFs and DFs to be created in the DF.
	Delete	Allows files in the DF to be deleted.
EF	Read	It is allowed to read the file's contents.
	Update	It is allowed to update the file's contents.
	Append	It is allowed to append information to the file.
	Compute checksum	* The contents of the file can be used when computing a checksum.
	Compute signature	* The contents of the file can be used when computing a signature.
	Verify checksum	* The contents of the file can be used when verifying a checksum.
	Verify signature	* The contents of the file can be used when verifying a signature.
	Encipher	* The contents of the file can be used in enciphering operation.
	Decipher	* The contents of the file can be used in deciphering operation.

"*" indicates that the access method is only relevant for files containing keys (in this case, Private RSA Key #1 & #2).

Each access method can have the following conditions:

Type	Meaning
NEV	The operation is never allowed, not even after cardholder verification.
ALW	The operation is always allowed, without cardholder verification.
CHV (PIN X)	The operation is allowed after a successful cardholder verification.
SYS	The operation is allowed after a system key presentation, typically available only to the FINEID application issuer, e.g. ´EXTERNAL AUTHENTICATE`.

8.1. MF

Description

The MF represents the root of the FINEID application file structure. The MF access conditions are chosen by the FINEID application issuer.

Access conditions (informative)

MF access method	Access condition
Create	NEV
Delete	NEV

8.2. DF.ESIGN

Description

The DF.ESIGN represents the subdirectory of the FINEID application, where the Authentication Object #2 (PIN 2), Private RSA key #2 and Certificate #2 objects are stored. The AID of this directory is `A0 00 00 01 67 45 53 49 47 4E`.

Access conditions

DF access method	Access condition
Create	NEV
Delete	NEV

8.3. EF.CIAInfo

Description

The CIAInfo file contains generic information about the application as such and it's capabilities. This information includes the serial number, algorithms implemented etc.

Notice: the AlgorithmInfo type is coded according to the PKCS#15 v1.1, meaning that the algId field (named as objId in ISO/IEC 7816-15) is defined as optional, and therefore not included in this specification.

The label shall be set according to the table below:

Application label
'HENKILOKORTTI' (FIN) 'IDENTITY CARD' (ENG) 'IDENTITETSKORT' (SWE)

The preferredLanguage shall be set according to the table below:

Preferred language
'fi' (FIN) 'en' (ENG) 'sv' (SWE)

Access conditions

Access methods	Access condition
Read	ALW
Update	NEV

Value notation example

```
{
   version          v2,
   serialNumber     169752222515401241, -- an example
   manufacturerID   "VRK-FINEID",
   label            "IDENTITY CARD",
   cardflags       {authRequired},
   supportedAlgorithms { -- SEQUENCE OF AlgorithmInfo
      {
         reference     '00´H,
         algorithm     '01'H, -- CKM_RSA_PKCS from PKCS #11
         parameters    NULL,
         supportedOperations{compute-signature, decipher}
      }
      {
         reference     '01´H,
         algorithm     '03''H, -- CKM_RSA_X_509 from PKCS #11
         parameters    NULL,
         supportedOperations{compute-signature, decipher}
      }
      {
         reference     '02´H,
         algorithm     '06'H, -- CKM_SHA_1_RSA_PKCS from PKCS #11
```

```
            parameters   NULL,
            supportedOperations{compute-signature}
         }
         {
            reference    '03´H,
            algorithm    '00'H, -- CKM_RSA_PKCS_KEY_PAIR_GEN from PKCS #11
            parameters   NULL,
            supportedOperations{generate-key}
         }
      }
      preferredLanguage   "en",
   }
```

8.4. EF.OD

Description

The Object Directory (OD) file is a transparent elementary file, which contains pointers to other elementary files (PrKDs, CDs, AODs, DCODs) of the FINEID application. The information is presented in ASN.1 syntax according to ISO/IEC 7816-15.

An off-card application using the FINEID application shall use this file to determine how to perform security services with the card.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

Value notation example

```
{
   authObjects : path : {
      path `3F004401´H -- AOD
   },
   privateKeys : path : {
      path `3F004402´H -- PrkD
   },
   certificates: path : {
      path `3F004403´H -- CD #1 for card holder certs
   },
   certificates: path : {
      path `3F004404´H -- CD #2 for new card holder certs
   },
   trustedCertificates : path : {
      path `3F004405´H -- CD #3 for trusted CA certs
   },
   dataContainerObjects : path : {
      path '3F004406'H -- DCOD for (new) data objects
   },
   usefulCertificates : path : {
      path `3F004407´H -- CD #4 for useful certs
```

```
   },
}
```

In DER encoding the outermost `SEQUENCE OF` is omitted.

```
A8 08
   30 06
      04 04 3F 00 44 01
A0 08
   30 06
      04 04 3F 00 44 02
A4 08
   30 06
      04 04 3F 00 44 03
A4 08
   30 06
      04 04 3F 00 44 04
A5 08
   30 06
      04 04 3F 00 44 05
A7 08
   30 06
      04 04 3F 00 44 06
A6 08
   30 06
      04 04 3F 00 44 07
```

As can be seen, the OD file simply consists of seven records.

Note: Pointers to CD #2, DCOD and CD #4 are optional.

8.5. EF.AOD

Description

This elementary file (Authentication Object Directory) contains generic authentication object attributes such as allowed characters, PIN length, PIN padding character, etc. It also contains the pointers to the authentication objects themselves (in the case of PINs, pointers to the DF in which the PIN file resides). The authentication objects are used to control access to other objects such as keys. The contents of this file is according to ISO/IEC 7816-15.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

PIN-code settings

- PIN-codes shall contain only numbers (**ASCII-numeric** PIN encoding)
- Minimum PIN length for PIN 1 shall be 4 characters and minimum PIN length for PIN 2 shall be 6 characters
- Maximum PIN length shall be 8 characters
- Card shall support PIN changing
- Card shall support unblocking PIN
- PIN labels and the global/local flag shall be set according to the table below

PIN number	PIN used by	PIN label	Local or global PIN
PIN 1	'authentication and encipherment key' Possibly used in other applications also.	'perustunnusluku' (FIN) 'basic PIN' (ENG) 'grund PIN' (SWE)	Local PIN
PIN 2	'signature key' PIN 2 shall not be used for any other services or applications.	'allekirjoitustunnusluku' (FIN) 'signature PIN' (ENG) 'signatur PIN' (SWE)	Local PIN The userConsent element shall be used for the corresponding private key i.e. user interaction is required for each private key operation.

Value notation example

```
{
   pwd : {
      commonObjectAttributes { -- CommonObjectAttributes
         label "basic PIN",
         flags {private, modifiable}
      },
      classAttributes { -- CommonAuthenticationObjectAttributes
         authID '01'H -- cross referenced from PrKD
      },
   typeAttributes { -- PasswordAttributes
      pwdFlags {local, initialized, needs-padding},
      pwdType ascii-numeric,
      minLength 4,
      storedLength 8,
      pwdReference '81'H ,
      padChar '00'H,
      path {
         path '3F00'H -- PIN is in MF
      }
   },
   pwd : {
      commonObjectAttributes { -- CommonObjectAttributes
```

```
        label "signature PIN",
        flags {private, modifiable}
      },
      classAttributes { -- CommonAuthenticationObjectAttributes
        authID '02'H -- cross referenced from PrKD
      },
   typeAttributes { -- PasswordAttributes
      pwdFlags {local, initialized, needs-padding},
      pwdType ascii-numeric,
      minLength 6,
      storedLength 8,
      pwdReference '82'H ,
      padChar '00'H,
      path {
        path '3F005016'H -- PIN in ESIGN DF
      }
   }
}
```

In DER encoding the outermost `SEQUENCE OF` is omitted.

The contents of the actual PIN files are card specific.

8.6. Authentication Object #1 (PIN 1)

Description

The Authentication Object #1 file contains the PIN 1 code (basic PIN) of the FINEID application.

Rules for PIN 1:

- After three consecutive false PIN presentations the PIN is blocked
- After ten consecutive false PUK presentations the PIN is unusable forever
- The PIN can be unblocked an infinite amount of times
- PIN verification status is dropped to state 'not verified' automatically after the FINEID application is deselected

Access conditions

Access method	Access condition
Read	NEV
Update	NEV

8.7. Authentication Object #2 (PIN 2)

Description

The Authentication Object #2 file contains the FINEID application PIN 2 code. PIN 2 is used as an access condition to protect the usage of the private RSA non-repudiation **'signature key'** of the FINEID application.

Rules for PIN 2:

- After three consecutive false PIN presentations the PIN is blocked
- After ten consecutive false PUK presentations the PIN is unusable forever
- The PIN can be unblocked an infinite amount of times
- PIN verification status is dropped to state 'not verified' automatically after the FINEID application is deselected
- PIN verification status is dropped to state 'not verified' automatically by the card after each RSA transformation performed with the non-repudiation **'signature key'**

Access conditions

Access method	Access condition
Read	NEV
Update	NEV

8.8. EF.PrKD

Description

This transparent elementary file (Private Key Directory) contains general key attributes such as labels, intended usage, identifiers etc. When applicable, it contains cross-reference pointers to authentication objects used to protect access to the keys. It also contains the pointers to the keys themselves.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

Private key labels

The Private key labels shall be set according to the table in chapter **4.1. Private RSA keys**.

Value notation example

```
{
   privateRSAKey : {
      commonObjectAttributes { -- CommonObjectAttributes
         label "auth. and encipherment key",
         flags {private},
         authID '01'H
         accessControlRules: { -- SEQUENCE OF AccessControlRule
            { --- AccessControlRule
```

```
            accessMode { execute }
            authId '01'H
         }
      }
   },
   classAttributes { -- CommonKeyAttributes
      iD '45'H,
      usage {decipher, sign, keyDecipher},
      -- native by default true (HW RSA)
      accessFlags {sensitive, alwaysSensitive, neverExtractable,
         cardGenerated},
      keyReference '00'H
   },
   subClassAttributes { -- CommonPrivateKeyAttributes
      keyIdentifiers { -- SEQUENCE OF KeyIdentifier
         {
            idType  4, -- Subject public key hash
            idValue OCTET STRING :
               '11223344556677889900112233445566778899 00'H
               -- Faked value of SHA-1 hash
         }
      }
   },
   typeAttributes { -- PrivateRSAKeyAttributes
      value indirect : path : {
         path '4B01'H
      },
      modulusLength 1024,
   }
},
privateRSAKey : {
   commonObjectAttributes { -- CommonObjectAttributes
      label "signature key",
      flags {private},
      authID '02'H,
      userConsent '01'H   -- user consent required for each
                             private key operation !!!
      accessControlRules: { -- SEQUENCE OF AccessControlRule
         { --- AccessControlRule
             accessMode { execute }
             authId '02'H
         }
      }
   },
   classAttributes { -- CommonKeyAttributes
      iD '46'H,
      usage {nonRepudiation},
      -- native by default true (HW RSA)
      accessFlags {sensitive, alwaysSensitive, neverExtractable,
         cardGenerated},
      keyReference '00'H
```

```
      },
      subClassAttributes { -- CommonPrivateKeyAttributes
         keyIdentifiers { -- SEQUENCE OF PKCS15KeyIdentifier
            {
               idType  4, -- Subject public key hash
               idValue OCTET STRING :
                  '11223344556677889900112233445566778899 00'H
                  -- Faked value of SHA-1 hash
            }
         }
      },
      typeAttributes { -- PrivateRSAKeyAttributes
         value indirect : path : {
            path '3F0050164B02'H
         },
         modulusLength 1024,
      }
   }
}
```

In DER encoding the outermost `SEQUENCE OF` is omitted.

The contents of the actual private key files are completely card specific. Operations possible to perform with keys in these files may either be deduced by looking at the contents of the CIAInfo file or by external knowledge of the card in question.

8.9. Private RSA Key #1

Description

This file contains the private RSA **'auth. and encipherment key'**.

PIN 1 must be verified before RSA transformation can be performed. PIN 1 verification status remains unaffected after the RSA transformation is performed.

Access conditions

Access method	Access condition
Read	NEV
Update	NEV
Get Data (public key)	ALW
Compute Signature, Decipher	CHV (PIN 1)

8.10. Private RSA key #2

Description

This file contains the private RSA **'signature key'**.

PIN 2 must be verified every time before RSA transformation can be performed. PIN 2 verification status is dropped to state 'not verified' automatically by the card after each RSA transformation performed with this key. The userConsent element in PrKD contains

value 1 for this key, i.e. the card holder must manually enter the corresponding PIN for each private key operation.

Access conditions

Access method	Access condition
Read	NEV
Update	NEV
Get Data (public key)	ALW
Compute Signature	CHV (PIN 2)

8.11. EF.CD #1

Description

This transparent elementary file contains attributes and pointers to card holder certificates **'auth. and encipherment cert.'** (Certificate #1) and **'signature certificate'** (Certificate #2). These certificates are intended to be kept unmodified during the validity period of the application. Information in this file contains certificate attributes such as labels, key identifiers, pointers to certificate files etc.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

Certificate label

The certificate label shall be set according to the table in chapter

4.2. Card holder certificates.

Value notation example

```
{
   x509Certificate : {
      commonObjectAttributes { -- CommonObjectAttributes
         label "auth. and encipherment cert.",
         flags {}
         accessControlRules: { -- SEQUENCE OF AccessControlRule
            { --- AccessControlRule
                accessMode { read }
                NULL  --always
            }
         }
  },
      classAttributes { -- CommonCertificateAttributes
         iD '45'H
         -- By default authority FALSE i.e. not CA cert
         requestId { -- KeyIdentifier
            idType  3, -- Issuer and serial number hash
            idValue OCTET STRING :
               '11223344556677889900112233445566778899 00'H
```

```
                -- Faked value of SHA-1 hash
            }
        },
        typeAttributes { -- X509CertificateAttributes
            value indirect : path : {
                path '3F004331'H
            }
        }
    },
    x509Certificate : {
        commonObjectAttributes { -- CommonObjectAttributes
            label "signature certificate",
            flags {}
            accessControlRules: { -- SEQUENCE OF AccessControlRule
                { --- AccessControlRule
                    accessMode { read }
                    NULL  --always
                }
            }
        },
        classAttributes { -- CommonCertificateAttributes
            iD '46'H
            -- By default authority FALSE i.e. not CA cert
            requestId { -- KeyIdentifier
                idType  3, -- Issuer and serial number hash
                idValue OCTET STRING :
                    '1122334455667788990011223344556677889900'H
                    -- Faked value of SHA-1 hash
            }
        },
        typeAttributes { -- X509CertificateAttributes
            value indirect : path : {
                path '3F0050164332'H
            }
        }
    }
}
```

In DER encoding the outermost `SEQUENCE OF` is omitted.

Files 3F00/4331 and 3F00/5016/4332 should contain DER encoded certificate structure in accordance with ISO/IEC 9594-8.

8.12. Certificate #1

Description

This file contains the card holder's **'auth. and encipherment cert.'** containing the public key corresponding to the private RSA **'auth. and encipherment key'** (Private RSA Key #1). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.13. Certificate #2

Description

This file contains the card holder's **'signature certificate'** containing the public key corresponding to the private RSA **'signature key'** (Private RSA Key #2). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.14. EF.CD #2

Description

This transparent elementary file contains attributes and pointers to additional card holder certificates that are written to the application after the centralized personalization. Information in this file contains certificate attributes such as labels, key identifiers, pointers to certificate files etc.

Originally this file is empty. New pointers can be added and old ones deleted with card holder's discretion (update access condition is PIN 1). The actual certificates are intended to be stored into the EF(Public EmptyArea) using the EF(UnusedSpace) according to PKCS#15.

This file is optional.

Access conditions

Access method	Access condition
Read	ALW
Update	CHV (PIN 1)

8.15. EF.CD #3 (trusted certs)

Description

This transparent elementary file contains attributes and pointers to trusted CA certificates. These certificates are used as starting points of trust for the card holder (e.g. when verifying other certificates). Originally this file will contain pointers to the following CA certificates:

- 'VRK Gov. Root CA' (self signed)
- **'**VRK Gov. CA for Citizen Qualified Certificates**'** (signed by 'VRK Gov. Root CA')

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

Value notation example

```
{
   x509Certificate : {
      commonObjectAttributes { -- CommonObjectAttributes
         label "VRK Gov. Root CA",
         flags {}
         accessControlRules: { -- SEQUENCE OF AccessControlRule
            { --- AccessControlRule
                accessMode { read }
                NULL  --always
            }
         }
      },
      classAttributes { -- CommonCertificateAttributes
         iD '48'H
         authority TRUE, -- CA certificate
         requestId { -- KeyIdentifier
            idType  2, -- Subject key identifier
            idValue OCTET STRING :
               '11223344556677889900112233445566778899 00'H
               -- Faked value of subjectKeyIdentifier extension
         }
      },
      typeAttributes { -- X509CertificateAttributes
         value indirect : path : {
            path '3F004334'H
         }
      }
   }
   x509Certificate : {
      commonObjectAttributes { -- CommonObjectAttributes
         label "VRK Gov. CA for Citizen Qualified Certificates"
         flags {}
         accessControlRules: { -- SEQUENCE OF AccessControlRule
            { --- AccessControlRule
                accessMode { read }
                NULL  --always
            }
         }
      },
      classAttributes { -- CommonCertificateAttributes
         iD '47'H
         authority TRUE, -- CA certificate
         requestId { -- KeyIdentifier
            idType  2, -- Subject key identifier
            idValue OCTET STRING :
```

```
                'l1223344556677889900l1223344556677889900'H
                -- Faked value of subjectKeyIdentifier extension
            }
        },
        typeAttributes { -- X509CertificateAttributes
            value indirect : path : {
                path '3F004333'H
            }
        }
    }
}
```

In DER encoding the outermost SEQUENCE OF is omitted.

Files 3F00/4333 and 3F00/4334 should contain DER-encoded CA certificates in accordance with ISO/IEC 9594-8.

8.16. CA Certificate #1

Description

This file contains the trusted root CA certificate (keyLength 2048 bits, self-signed). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.17. CA Certificate #2

Description

This file contains the trusted intermediate CA certificate (keyLength 2048 bits, signed by root CA). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.18. EF.CD #4 (useful certs)

Description

This transparent elementary file contains attributes and pointers to additional useful certificates that are written to the application after the centralized personalization. Useful certificates means certificates that does not belong in either to a card holder (EF.CD #1 or

EF.CD #2) or to trusted CA certificates (EF.CD #3). It may be used to store either certificates that may be useful, e.g. a certificate for a colleague’s encryption key or intermediate CA certificates to simplify certificate path processing.

Information in this file contains certificate attributes such as labels, key identifiers, pointers to certificate files etc.

Originally this file is empty. New pointers can be added and old ones deleted with card holder's discretion (update access condition is PIN 1). The actual certificates are intended to be stored into the EF(Public EmptyArea) using the EF(UnusedSpace) according to PKCS#15.

This file is optional.

Access conditions

Access method	Access condition
Read	ALW
Update	CHV (PIN 1)

8.19. EF.DCOD

Description

This transparent elementary file contains attributes and pointers to data objects that are written to the card after the centralized personalization. Originally this file is empty. New pointers can be added and old ones deleted with card holder's discretion (update access condition is PIN 1). When this file is to be updated, the coding shall be according to ISO/IEC 7816-15. The actual data objects are intended to be stored into the EF(Public EmptyArea) or EF(Private EmptyArea) using the EF(UnusedSpace) according to PKCS#15.

This file is optional.

Access conditions

Access method	Access condition
Read	ALW
Update	CHV (PIN 1)

8.20. EF(UnusedSpace)

Description

This transparent elementary file is used to keep track of unused space in empty files of the card. Initially this file will contain pointers to the empty transparent files EF(Public EmptyArea) and EF(Private EmptyArea). The format of the file is otherwise according to PKCS #15 except that the AccessControlRule component is coded according to ISO/IEC 7816-15 (NULL value for SecurityCondition means ALWAYS access).

This file is optional.

Access conditions

Access method	Access condition
Read	ALW
Update	CHV (PIN 1)

Value notation example

```
{
   { -- UnusedSpace (public)
      path : { -- Path
         path '3F00433F'H,
         index 0,
         length 7040 -- size of empty area varies
      }
      authId '01'H
      accessControlRules: { -- SEQUENCE OF AccessControlRule
         { --- AccessControlRule
             accessMode { read }
             NULL  --always
         }
         { --- AccessControlRule
             accessmode { update }
             authId '01'H
         }
      }
   }
   { -- UnusedSpace (private)
      path : { -- Path
         path '3F00433E'H,
         index 0,
         length 3808 -- size of empty area varies
      }
      authId '01'H
      accessControlRules: { -- SEQUENCE OF AccessControlRule
         { --- AccessControlRule
             accessmode { read, update }
             authId '01'H
         }
      }
   }
}
```

In DER encoding the outermost `SEQUENCE OF` is omitted.

Note: Token might contain only EF(Public EmptyArea) but NOT EF(Private EmptyArea). In this case EF(UnusedSpace) contains pointer only to EF(Public EmptyArea).

8.21. EF(Public EmptyArea)

Description

This transparent elementary file contains empty space for additional certificates or data objects that are not stored into the card during centralized personalization. Pointers in EF(UnusedSpace) keep track of used areas inside this file. This file is intended for public objects, because the access condition for read method is ALW (operation is always allowed, without card holder verification). Originally this file is empty.

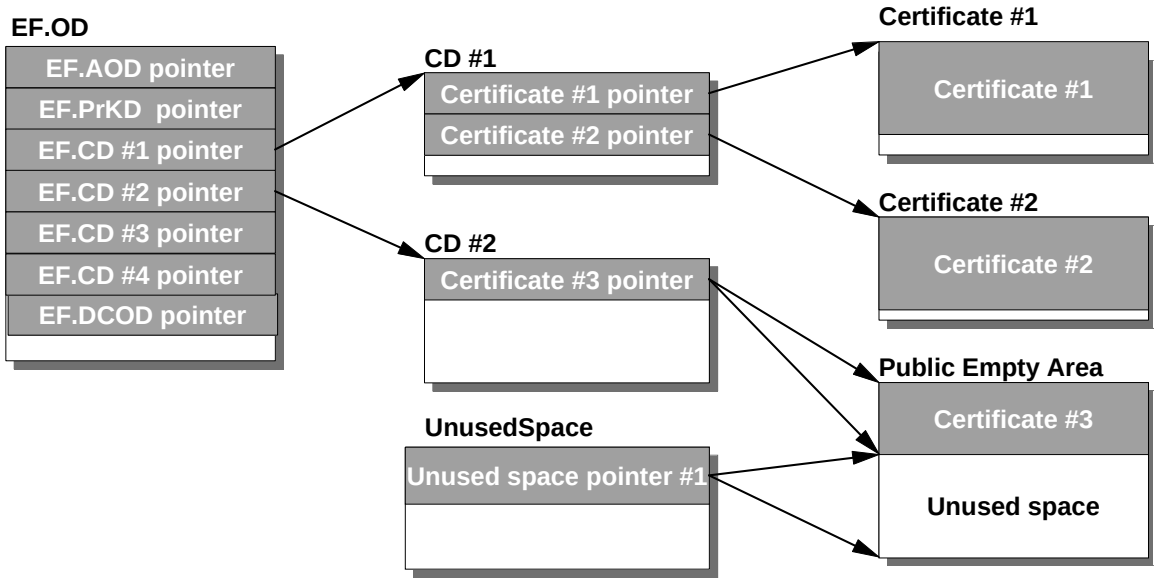
This file is optional.

Access conditions

Access method	Access condition
Read	ALW
Update	CHV (PIN 1)

Example

The figure below describes the content of modified files after adding a new certificate to the card. A pointer to the new certificate #3 is added to the CDF #2. The certificate itself is written to the EF(Public EmptyArea). Unused space pointer in EF(UnusedSpace) is also updated.



8.22. EF(Private EmptyArea)

Description

This transparent elementary file contains empty space for additional private data objects that are not stored into the card during centralized personalization. Pointers in EF(UnusedSpace) keep track of used areas inside this file. This file is intended for private

objects, because the access condition for read method is CHV (PIN 1). Originally this file is empty.

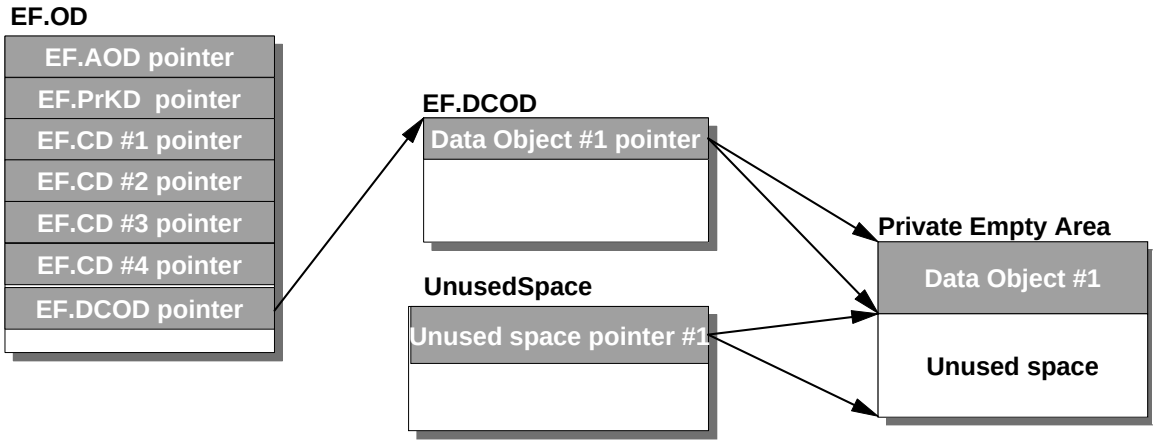
This file is optional.

Access conditions

Access method	Access condition
Read	CHV (PIN 1)
Update	CHV (PIN 1)

Example

The figure below describes the content of modified files after adding a new private data object to the card. A pointer to the new data object is added to the EF.DCOD. The data object itself is written to the EF(Private EmptyArea) (the nature of the new data object is private in this example). Unused space pointer in EF(UnusedSpace) is updated also.



9. Certificates

The contents of the certificates are described in the FINEID S2 specification.