FINEID - S4-1

Implementation Profile 1

for Finnish Electronic ID Card

v4.2

Digital and Population Data Services Agency (DVV)
Certification Authority Services
P.O. Box 123
FIN-00531 Helsinki
Finland
http://www.dvv.fi

Authors

Name	Initials	Organization	E-mail
Markku Sievänen	MaSi	Thales DIS Finland Oy	markku.sievanen@thalesgroup.com
Jari Pirinen	JP	DVV	jari.pirinen@dvv.fi

Document history

Version	Date	Editor	Changes	Status
0.9	12.04.2022	MaSi		Initial version.
1.00	10.05.2022	MaSi	• Corrected ASN.1 example of EF.PrKD content. • Reviewed and accepted by DVV. Document version changed from 0.9 to 1.00.	Reviewed and accepted by DVV.
1.01	14.09.2022	MaSi	• Corrected ASN.1 example of EF.PrKD content: privateECKey had been encoded as privateRSAKey and vice versa.	
1.02	25.10.2022	MaSi	• Explained the new encoding method of serialNumber-field in EF.CIAInfo file.	
1.10	14.02.2025	MaSi	• Added description of ATR and ATS bytes for MAV5.1 chip platform.	
1.11	21.03.2025	MaSi	• Corrected ATR bytes for MAV5.1 chip platform ○ Value of T0 byte corrected from ‘FF’ to ‘7F’.	
1.12	20.05.2025	MaSi	• Updated major version from v4.0 to v4.1. • Paragraph 8.1.2 EF.DIR ○ Added used providerId value for MAV5.1 chip platform.	

1.20	29.08.2025	MaSi	• Updated major version from v4.1 to v4.2.
			• Paragraph 4.1. PIN/PUK-code settings
			  ○ Updated used label values for PUK.
			• Added paragraph 4.6. Card activation schemes
			• Paragraph 8.1.2 EF.DIR
			  ○ Added used providerId value for MAV5.1 chip platform and new activation code scheme.
			• Paragraph 8.1.5. EF.AOD
			  ○ Updated value notation example based on new PUK label.	

1.21	02.10.2025	MaSi	• Some typos corrected
			• Paragraph 4.6.1. Old card activation scheme
			  ○ Diagram of PIN states corrected
			• Paragraph 4.6.2. New card activation scheme
			  ○ Length of “activation PIN” changed from 6 digits to 7 digits
			• Paragraph 8.1.3. EF.CIAInfo
			  ○ Removed all SHA-1 based algorithms
			  ○ Added algorithm ckm-ecdh1-derive
			• Copyright corrected from "© Digi- ja väestötietovirasto 2022" to "© Digi- ja väestötietovirasto 2025"	

Table of contents

1. Introduction

This document describes an implementation profile of the FINEID S1 specification version 4.2. This implementation profile is for Finnish Electronic ID Cards and for other smart cards containing Citizen Certificates issued by Digital and Population Data Services Agency (DVV).

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, ISO/IEC 7816-15 AMENDMENT 1, ,ISO/IEC 7816-15 AMENDMENT 2, ISO/IEC 7816-1 TECHNICAL CORRIGENDUM 1, 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.

Full names for the FINEID specifications are listed below:

- FINEID S1 - Electronic Identity Application, v4.0
- FINEID S2 – DVV CA-model and certificate contents, v5.0
- FINEID S4-1 - Implementation Profile 1 for Finnish Electronic ID Card, v4.2
- FINEID S4-2 - Implementation Profile 2 for Organizational Usage, v3.0
- FINEID S5 – Directory Specification, v3.0

FINEID specifications are available at

https://dvv.fi/en/fineid-specifications

The PKCS standards are available at

- **http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/public-key-cryptography-standards.htm**

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 multi-application 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. Dual interface

If IC Card supports dual interface (contact and contactless), the contactless communication is protected by Password Authenticated Connection Establishment (PACE) and secure messaging. PACE is a security feature that require the terminal and the card to share a simple secret in order to authenticate each other. PACE is based on the Diffie-Hellman protocol for strong mutual authentication and is specified in documents: Technical Guideline TR-03110 (BSI) - Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token: Part 1 Version 2.20 - 26. February 2015, Part 2 Version 2.21 - 21. December 2016, Part 3 Version 2.21 - 21. December 2016, Part 4 Version 2.21 - 21. December 2016.

The password that is used for PACE can be the MRZ (Machine Readable Zone), Global PIN or Card Access Number (CAN). In FINEID context all these passwords are used.

3.2. PACE

In FINEID context the supported algorithms for PACE are:

- id-PACE-ECDH-GM-AES-CBC-CMAC-256 with BrainpoolP384r1
- id-PACE-ECDH-CAM-AES-CBC-CMAC-256 with BrainpoolP384r1

EF.CardAccess file contains security information for PACE. Below is presented a content of this file using ASN.1 format:

```
SET { -- SecurityInfos
 SEQUENCE { -- Security Info
  OBJECT IDENTIFIER '0 4 0 127 0 7 2 2 4 2 4' -- PACEInfo, id-PACE-ECDH-GM-AES-CBC-CMAC-256
  INTEGER 2 -- version: 2
  INTEGER 16 -- parameterId : BrainpoolP384r1
   }
 SEQUENCE { -- Security Info
  OBJECT IDENTIFIER '0 4 0 127 0 7 2 2 4 6 4' -- PACEInfo, id-PACE-ECDH-CAM-AES-CBC-CMAC-256
  INTEGER 2 -- version: 2
  INTEGER 16 -- parameterId : BrainpoolP384r1
   }
  }
```

3.3. ATR/ATS bytes

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

3.3.1. ATR bytes

Following table describes typical ATR (Answer To Reset) bytes of the card (according ISO 7816-3 and 7816-4), when contact communication is used.

MAV5.0 chip platform:

Character	Value (hex)	Comment
TS	3B	Initial character: direct convention
T0	7F	Format character: '7' indicates that TA1, TB1 and TC1 are present, 'F' indicates the number of historical characters (15).
TA1	96	Fi = 512 (clock rate conversion integer), Di = 32 (baud rate adjustment integer), f(max.)=5 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.
T2	31	Card service data (tag 3, length 1)
T3	B8	Application selection: by full DF name. DOs available: in EF.DIR and in EF.ATR. EF.DIR and AF.ATR access services: by the READ BINARY command (transparent structure). Card with MF.
T4	65	Pre-issuing data (tag 6, length 5).

T5	B0	Family name.
T6	85	Product name.
T7	05	OS version.
T8	00	Program version.
T9	11	Chip ID.
T10	12	Country code and subsequent data (tag 1, length 2). A country indicator consists of a country code (three quartets with values from '0' to '9'), see ISO 3166-1(21) followed by subsequent data (at least one quartet, odd number of quartets). Country code = FIN=246 = 0x246 = 0010 0100 0110b Subsequent data = 0000 b
T11	24	
T12	60	
T13	82	Status indicator (tag 8, length 2)
T14	90	SW1
T15	00	SW2

MAV5.1 chip platform:

Character	Value (hex)	Comment
TS	3B	Initial character: direct convention
T0	7F	Format character: '7' indicates that TA1, TB1 and TC1 are present, 'F' indicates the number of historical characters (15).
TA1	96	Fi = 512 (clock rate conversion integer), Di = 32 (baud rate adjustment integer), f(max.)=5 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.
T2	31	Card service data (tag 3, length 1)
T3	B8	Application selection: by full DF name. DOs available: in EF.DIR and in EF.ATR. EF.DIR and AF.ATR access services: by the READ BINARY command (transparent structure). Card with MF.
T4	65	Pre-issuing data (tag 6, length 5).
T5	B0	Family name.
T6	85	Product name.
T7	05	OS version.
T8	10	Program version.

T9	24	Chip ID.
T10	12	Country code and subsequent data (tag 1, length 2). A country indicator consists of a country code (three quartets with values from '0' to '9'), see ISO 3166-1(21) followed by subsequent data (at least one quartet, odd number of quartets). Country code = FIN=246 = 0x246 = 0010 0100 0110b Subsequent data = 0000 b
T11	24	
T12	60	
T13	82	Status indicator (tag 8, length 2)
T14	90	SW1
T15	00	SW2

3.3.2. ATS bytes

When contactless communication and Type A are used, typical ATS (Answer To Select) returned during anticollision process is the following:

MAV5.0 chip platform:

Character	Value (hex)			Comment
TL	14			Total length of the ATS (including TL).
T0	78			TA1 & TB1 & TC are present, Maximum frame size (called FSCI, codes FSC) = 256 bytes.
TA1	77			106, 212, 424 and 847 kilobits supported; reception and transmission may use different bit rates.
TB1	95			The most significant half-byte b8 to b5 is called FWI and codes FWT = 154 ms, the least significant half byte b4 to b1 is called SFGI and codes a multiplier value used to define the SFGT.
TC1	02			CID supported, NAD not supported.
HC1	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.
HC2		31		C-TLV Card service data, length 1 byte
HC3			B8	Application selection: by full DF name; DOs available: in EF.ATR/INFO; EF.DIR and EF.ATR/INFO access services: by the read binary command (transparent structure); Card with MF.
HC4		65		C-TLV Pre-issuing data, 5 bytes
HC5			B0	Hardmask identification
HC6			85	Hardmask identification
HC7			05	Hardmask identification
HC8			00	Hardmask identification
HC9			11	Hardmask identification
HC10		12		Country code and subsequent data (tag 1, length 2). A country indicator consists of a country code (three quartets with values from '0' to '9'), see ISO 3166-1(21) followed by subsequent data (at least one quartet, odd number of quartets). Country code = FIN=246 = 0x246 = 0010 0100 0110b Subsequent data = 0000 b
HC11			24	
HC12			60	
HC13		82		Status indicator (tag 8, length 2)
HC14			90	SW1
HC15			00	SW2
CRC1	XX			CRC1
CRC2	XX			CRC2

MAV5.1 chip platform:

Character	Value (hex)			Comment
TL	14			Total length of the ATS (including TL).
T0	78			TA1 & TB1 & TC are present, Maximum frame size (called FSCI, codes FSC) = 256 bytes.
TA1	77			106, 212, 424 and 847 kilobits supported; reception and transmission may use different bit rates.
TB1	95			The most significant half-byte b8 to b5 is called FWI and codes FWT = 154 ms, the least significant half byte b4 to b1 is called SFGI and codes a multiplier value used to define the SFGT.
TC1	02			CID supported, NAD not supported.
HC1	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.
HC2		31		C-TLV Card service data, length 1 byte
HC3			B8	Application selection: by full DF name; DOs available: in EF.ATR/INFO; EF.DIR and EF.ATR/INFO access services: by the read binary command (transparent structure); Card with MF.
HC4		65		C-TLV Pre-issuing data, 5 bytes
HC5			B0	Hardmask identification
HC6			85	Hardmask identification
HC7			05	Hardmask identification
HC8			10	Hardmask identification
HC9			24	Hardmask identification
HC10		12		Country code and subsequent data (tag 1, length 2). A country indicator consists of a country code (three quartets with values from '0' to '9'), see ISO 3166-1(21) followed by subsequent data (at least one quartet, odd number of quartets). Country code = FIN=246 = 0x246 = 0010 0100 0110b Subsequent data = 0000 b
HC11			24	
HC12			60	
HC13		82		Status indicator (tag 8, length 2)
HC14			90	SW1
HC15			00	SW2
CRC1	XX			CRC1
CRC2	XX			CRC2

4. FINEID application functionality

4.1. PIN/PUK-code settings

PIN/PUK 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 1	'authentication key' Possibly used in other applications also.	'perustunnusluku' (FIN) 'basic PIN' (ENG) 'grund PIN' (SWE)	Global. The userConsent element shall be used for the corresponding private key i.e. user interaction is required for each private key signing operation.
PIN 2	'signature key ECC' and 'signature key RSA' PIN 2 shall not be used for any other services or applications.	'allekirjoitustunnusluku' (FIN) 'signature PIN' (ENG) 'signatur PIN' (SWE)	Local. The userConsent element shall be used for the corresponding private key i.e. user interaction is required for each private key signing operation.
PUK	Unblocking PIN 1 and PIN 2.	'avaustunnusluku' (FIN) 'unblocking PIN' (ENG) 'avblockera PIN' (SWE)	Local.

PIN/PUK policy settings shall be set according to the table below:

Setting	PIN 1	PIN 2	PUK
Coding	ASCII-numeric	ASCII-numeric	ASCII-numeric
Min. length	4	6	8
Stored length	12	12	12
Try limit	5	5	5
Change Condition	PIN 1	PIN 2	NEV
Unblock Condition	PUK	PUK	NEV
Unblock Counter	No limit	No limit	NEV
Usage Counter	No limit	No limit	No limit
Non-repudiation	Yes	Yes	No

4.2. Private ELC keys

The FINEID application shall contain two private ELC key as specified in the table below.

Private key number	Key label	X.509 key usage	Domain Parameters	Key length (in bits)
1	'todentamisavain' (FIN) 'authentication key' (ENG) 'autentisering nyckel' (SWE)	digitalSignature + keyAgreement	secp384r1	384
2	'allekirjoitusavain ECC' (FIN) 'signature key ECC' (ENG) 'signatur nyckel ECC' (SWE)	nonRepudiation	secp384r1	384

4.3. Private RSA keys

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

Private key number	Key label	X.509 key usage	Key length (in bits)	Public exponent
3	'allekirjoitusavain RSA' (FIN) 'signature key RSA' (ENG) 'signatur nyckel RSA' (SWE)	nonRepudiation	3072	65537 (F4)

4.4. 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	'todentamisvarmenne' (FIN) 'authentication cert.' (ENG) 'autentisering certifikat' (SWE)	digitalSignature + keyAgreement
2	'allekirjoitusvarmenne ECC' (FIN) 'signature certificate ECC' (ENG) 'signatur certifikat ECC' (SWE)	nonRepudiation
3	'allekirjoitusvarmenne RSA' (FIN) 'signature certificate RSA' (ENG) 'signatur certifikat RSA' (SWE)	nonRepudiation

End entity certificates are described in the FINEID S2 specification.

4.5. CA certificates

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

Root CA Certificate label (CN field of corresponding certificate)	Signed by	Key length
'DVV Gov. Root CA - G3 ECC' (FIN) 'DVV Gov. Root CA - G3 ECC' (ENG) 'DVV Gov. Root CA - G3 ECC' (SWE)	Self-signed	384 bits
'DVV Gov. Root CA - G3 RSA' (FIN) 'DVV Gov. Root CA - G3 RSA' (ENG) 'DVV Gov. Root CA - G3 RSA' (SWE)	Self-signed	4096 bits

Intermediate CA Certificate label (CN field of corresponding certificate)	Signed by	Key length
'DVV Citizen Certificates - G4E' (FIN) 'DVV Citizen Certificates - G4E' (ENG) 'DVV Citizen Certificates - G4E' (SWE)	'DVV Gov. Root CA - G3 ECC'	384 bits
'DVV Citizen Certificates - G4R' (FIN) 'DVV Citizen Certificates - G4R' (ENG) 'DVV Citizen Certificates - G4R' (SWE)	'DVV Gov. Root CA - G3 RSA'	4096 bits

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

4.6. Card activation schemes

4.6.1. Old card activation scheme

When personalized card leaves Card Manufacturer:

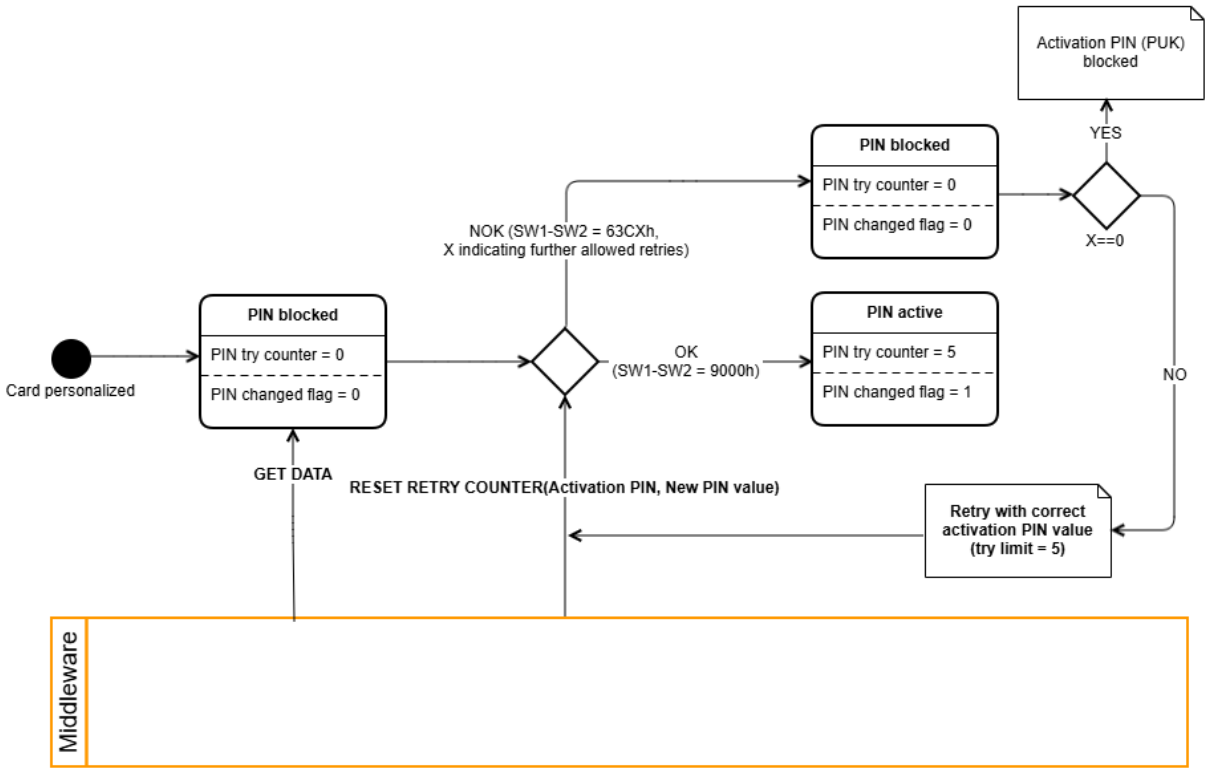
- PIN1 and PIN2 have been blocked
- PUK (the same for PIN1 and PIN2) has been delivered as “activation PIN” to card holder
- PUK (“activation PIN”) can be re-ordered
- Change PIN before first use has been activated
- PSO:CDS (Compute Digital Signature) command not available before PIN(s) have been changed

When card is taken in use by card holder (card activation) using middleware and activation PIN:

- PIN1 and PIN2 are unblocked and new values are set using RESET RETRY COUNTER command
  - PIN changed flag is set to 1 for both PINs
  - PSO:CDS command will be available

In case if PIN(s) has/have been blocked again and card holder has lost the original activation PIN letter PUK (“activation PIN”) can be re-ordered.

Below has been illustrated the PIN states during card activation.



PIN information (including PIN try counter and PIN changed flag) can be retrieved from the card using GET DATA command, see FINEID - S1 specification for details.

4.6.2. New card activation scheme

When personalized card leaves Card Manufacturer:

- PIN1 and PIN2 have been set to the same random value (7 digits); this is delivered as “activation PIN” to card holder
- PUK (the same for PIN1 and PIN2) can be ordered on demand
- Change PIN before first use has been activated
- PSO:CDS (Compute Digital Signature) command not available before PIN(s) have been changed

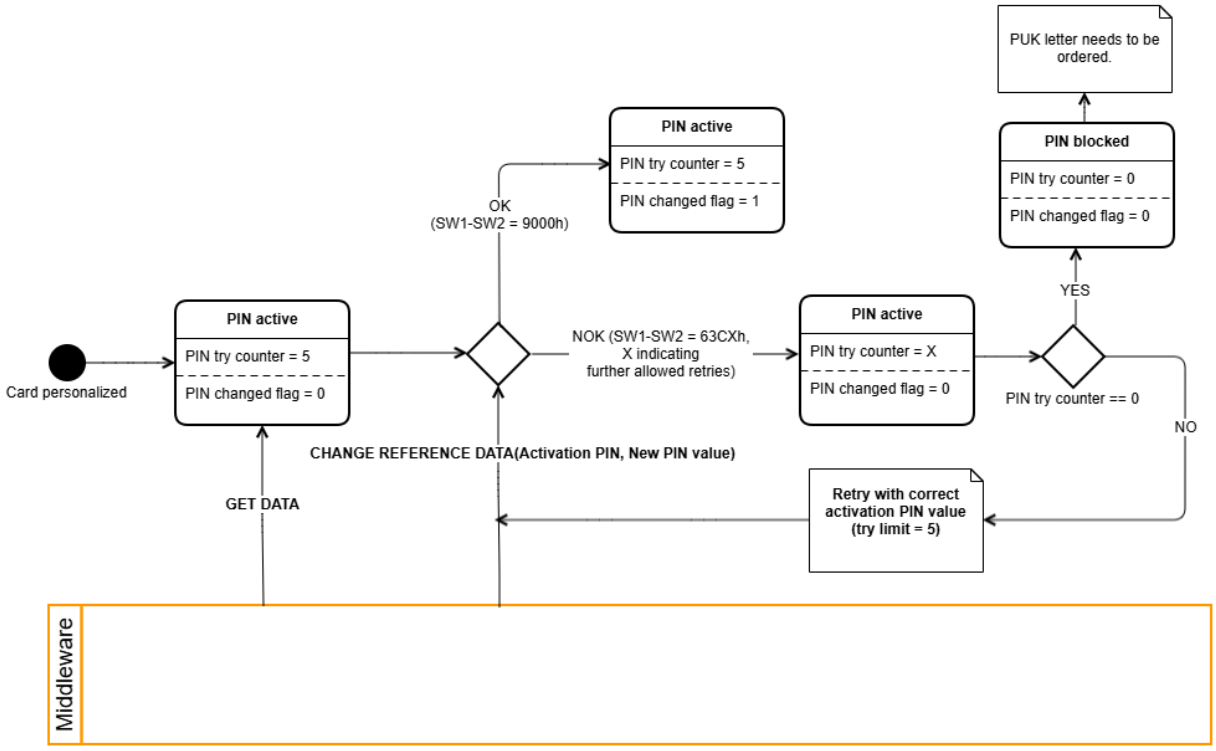
When card is taken in use by card holder (card activation) using middleware and activation PIN

- PIN1 and PIN2 are changed using activation PIN and CHANGE REFERENCE DATA command
  - PIN changed flag is set to 1 for both PINs
  - PSO:CDS command will be available

When PIN(s) is/are blocked:

- PUK (“unblocking PIN”) needs be ordered
- PIN1 and/or PIN2 can be unblocked and new values set using RESET RETRY COUNTER command
  - PIN changed flag is set to 1 for PIN(s) if not already set earlier

Below has been illustrated the PIN states during card activation.



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.

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.

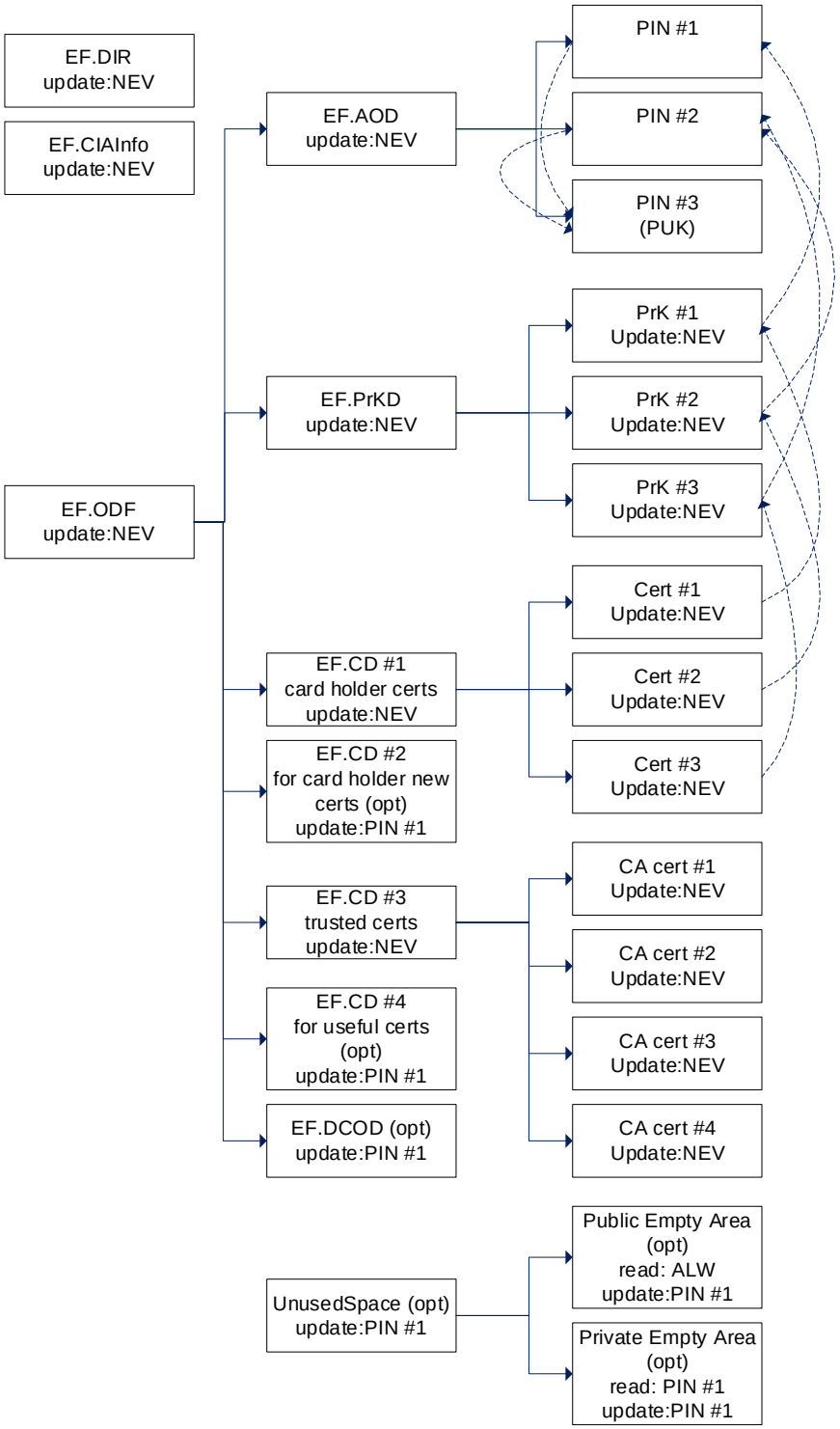
Any number of '00' octets may occur before, between or after the values of these objects without any meaning (i.e. as padding for unused space or deleted values).

7. FINEID file/object 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 & #3) 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.

PIN #3 is used to unblock PIN #1 and PIN#2. This is indicated by arrow between PIN #1 and PIN #3 and arrow between PIN #2 and PIN #3.

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 multi-application smart card or other interoperable token.

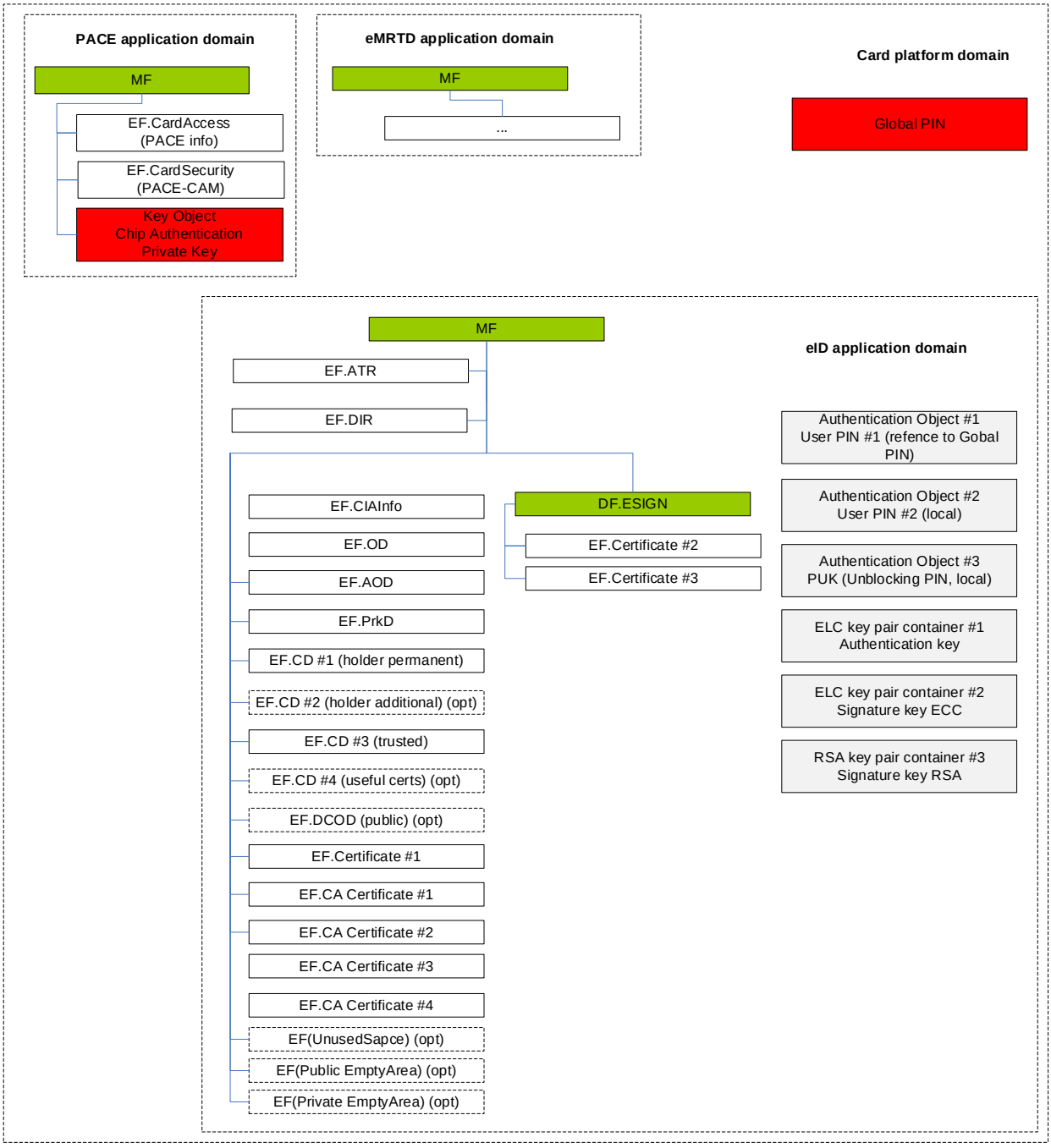


Figure 1. FINEID S4-1 file/object structure (optional files are marked using dashed borders)

File/object 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.
	Compute signature	* The contents of the file can be used when computing a signature.
	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 ELC Key #1, Private ELC Key #2 and Private RSA Key #3).

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.

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

8.1.1.EF.ATR

Description

EF.ATR file is used to give additional information to the inspection system about the capabilities of the chip (e.g. if extended length is supported or not). The following table defines the content of EF.ATR file.

Value	Explanation of values
0x47	**Tag for "Card capabilities" data object**
0x03	**Length:** 3 bytes
0xB4	**Selection method:** - DF selection by full name, path and FID but not by partial name - no implicit DF selection - SFI supported - record numbers and identifiers not supported
0x41	**Data coding byte:** - EFs of BER-TLV structure not supported - Behaviour of write functions: write OR - value 'FF' is not a valid tag (but padding) - data unit size = one byte
0xF3	**Command chaining, length fields and logical channels:** - command chaining supported - extended Lc and Le fields supported - extended length information in EF.ATR - logical channels supported, assigned by card, max. number 4.
0x7F66	**Tag for "Extended length information"**
0x09	**Length:** 9 bytes
0x020203FC	Command APDU size limitation: 1020 bytes (coded as INTEGER)
0x020300FFAA	Response APDU size limitation: 65450 bytes (coded as INTEGER)

Access conditions

Access methods	Access condition
Read	ALW
Update	NEV

8.1.2.EF.DIR

Description

This file under MF contains list of Cryptographic Information Applications found from the card.

Access conditions

Access methods	Access condition
Read	ALW
Update	NEV

Value notation example

ISO 7816-15 Interpretation

```
value ApplicationTemplate ::=
{
  aid 'A000000063504B43532D3135'H,
  label "FINEID S4-1",
  path '3F00'H
  ddo {
    providerId { "1.2.246.517.4.1.9" }
  }
}
```

ASN.1 Format

```
[APPLICATION 1] {
  [APPLICATION 15] { A0 00 00 00 63 50 4B 43 53 2D 31 35 }
  [APPLICATION 16] { "FINEID S4-1" }
  [APPLICATION 17] { 3F 00 }
  [APPLICATION 19] { OBJECT_IDENTIFIER { 1.2.246.517.4.1.9 } }
}
```

Used providerId values:

- MAV5.0 chip platform: `1.2.246.517.4.1.7`
- MAV5.1 chip platform: `1.2.246.517.4.1.8`
- MAV5.1 chip platform and new activation code scheme: `1.2.246.517.4.1.9`

8.1.3. EF.CIAInfo

Description

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

Notice that the encoding of the serialNumber-field (= card number) has been changed when compared to previous profile versions.

Previously the content of serialNumber-field has been encoded as packed BCD (binary-coded decimal), where two digits has encoded into single byte. Before BCD encoding the total length of serialNumber has been added as first digit(s) of value.

Example of previous encoding: original card number value 924600015069205907, add total length 18 -> 18924600015069205907, then encode this as packed BCD -> '18924600015069205907' H and finally encode this as ASN.1 OCTET STRING { 18 92 46 00 01 50 69 20 59 07 }.

Because of the card number can contain also characters in the future, the packed BCD encoding cannot be used any more. The new encoding method is explained in the example below.

Example of new encoding method: original card number value 92460001JA0000001 -> encode this directly as ASN.1 OCTET STRING { 39 32 34 36 30 30 30 31 4A 41 30 30 30 30 30 30 31 }.

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

ISO 7816-15 Interpretation

```
value CIAInfo ::=
{
  version v2,
  serialNumber '39323436303030314A4130303030303031'H,
  manufacturerID "FINEID",
  label "HENKILOKORTTI",
  cardflags { authRequired, prnGeneration },
  supportedAlgorithms
  {
    {
      reference 0,
      algorithm ckm-rsa-pkcs,
      parameters '0500'H,
      supportedOperations { compute-signature, decipher }
      objId { "1.2.840.113549.1.1.1" }
    },

    {
      reference 2,
      algorithm ckm-sha224-rsa-pkcs,
      parameters '0500'H,
      supportedOperations { compute-signature }
      objId { "1.2.840.113549.1.1.14" }
    },
    {
      reference 3,
      algorithm ckm-sha256-rsa-pkcs,
      parameters '0500'H,
      supportedOperations { compute-signature }
      objId { "1.2.840.113549.1.1.11" }
    },
    {
      reference 4,
      algorithm ckm-sha384-rsa-pkcs,
      parameters '0500'H,
      supportedOperations { compute-signature }
      objId { "1.2.840.113549.1.1.12" }
    },
    {
      reference 5,
      algorithm ckm-sha512-rsa-pkcs,
      parameters '0500'H,
      supportedOperations { compute-signature }
      objId { "1.2.840.113549.1.1.13" }
    },

    {
      reference 7,
      algorithm ckm-sha224-rsa-pkcs-pss,
      parameters
      {
         hashAlgorithm
        {
           algorithm sha-224,
           parameters '0500'H,
         }
```

```
      maskGenAlgorithm
      {
         algorithm pkcs1-MGF,
         parameters
         {
            algorithm sha-224,
            parameters '0500'H,
         }
      }
      saltLength 28
   },
   supportedOperations { compute-signature }
   objId { "1.2.840.113549.1.1.10" }
},
{
   reference 8,
   algorithm ckm-sha256-rsa-pkcs-pss,
   parameters
   {
      hashAlgorithm
     {
        algorithm sha-256,
        parameters '0500'H,
      }
      maskGenAlgorithm
      {
         algorithm pkcs1-MGF,
         parameters
         {
            algorithm sha-256,
            parameters '0500'H,
         }
      }
      saltLength 32
   },
   supportedOperations { compute-signature }
   objId { "1.2.840.113549.1.1.10" }
},
{
   reference 9,
   algorithm ckm-sha384-rsa-pkcs-pss,
   parameters
   {
      hashAlgorithm
     {
        algorithm sha-384,
        parameters '0500'H,
      }
      maskGenAlgorithm
      {
         algorithm pkcs1-MGF,
         parameters
         {
            algorithm sha-384,
            parameters '0500'H,
         }
      }
      saltLength 48
   },

   supportedOperations { compute-signature }
   objId { "1.2.840.113549.1.1.10" }
},
{
   reference 10,
```

```
  algorithm ckm-sha512-rsa-pkcs-pss,
  parameters
  {
     hashAlgorithm
     {
        algorithm sha-512,
        parameters '0500'H,
      }
      maskGenAlgorithm
      {
         algorithm pkcs1-MGF,
         parameters
         {
            algorithm sha-512,
            parameters '0500'H,
         }
      }
      saltLength 64
  },
  supportedOperations { compute-signature }
  objId { "1.2.840.113549.1.1.10" }
},

{
  reference 12,
  algorithm ckm-rsa-pkcs-oaep,
  parameters
  {
     hashAlgorithm
     {
        algorithm sha-224,
        parameters '0500'H,
      }
      maskGenAlgorithm
      {
         algorithm pkcs1-MGF,
         parameters
         {
            algorithm sha-224,
            parameters '0500'H,
         }
      }
  },
  supportedOperations { decipher  }
  objId { "1.2.840.113549.1.1.7" }
},
{
  reference 13,
  algorithm ckm-rsa-pkcs-oaep,
  parameters
  {
     hashAlgorithm
     {
        algorithm sha-256,
        parameters '0500'H,
      }
      maskGenAlgorithm
      {
         algorithm pkcs1-MGF,
         parameters
         {
            algorithm sha-256,
            parameters '0500'H,
         }
      }
```

```
  },
  supportedOperations { decipher  }
  objId { "1.2.840.113549.1.1.7" }
},
{
  reference 14,
  algorithm ckm-rsa-pkcs-oaep,
  parameters
  {
     hashAlgorithm
     {
        algorithm sha-384,
        parameters '0500'H,
      }
     maskGenAlgorithm
     {
        algorithm pkcs1-MGF,
        parameters
        {
           algorithm sha-384,
           parameters '0500'H,
        }
     }
  },
  supportedOperations { decipher  }
  objId { "1.2.840.113549.1.1.7" }
},
{
  reference 15,
  algorithm ckm-rsa-pkcs-oaep,
  parameters
  {
     hashAlgorithm
     {
        algorithm sha-512,
        parameters '0500'H,
      }
     maskGenAlgorithm
     {
        algorithm pkcs1-MGF,
        parameters
        {
           algorithm sha-512,
           parameters '0500'H,
        }
     }
  },
  supportedOperations { decipher  }
  objId { "1.2.840.113549.1.1.7" }
},

{
  reference 17,
  algorithm ckm-ecdsa-sha224,
  parameters '0500'H,
  supportedOperations { compute-signature }
  objId { "1.2.840.10045.4.3.1" }
},
{
  reference 18,
  algorithm ckm-ecdsa-sha256,
  parameters '0500'H,
  supportedOperations { compute-signature }
  objId { "1.2.840.10045.4.3.2" }
},
```

```
    {
      reference 19,
      algorithm ckm-ecdsa-sha384,
      parameters '0500'H,
      supportedOperations { compute-signature }
      objId { "1.2.840.10045.4.3.3" }
    },
    {
      reference 20,
      algorithm ckm-ecdsa-sha512,
      parameters '0500'H,
      supportedOperations { compute-signature }
      objId { "1.2.840.10045.4.3.4" }
    },
    {
      reference 21,
      algorithm ckm-ecdh1-derive,
      parameters '0500'H,
      supportedOperations { derive-key }
      objId { "1.3.132.1.12" }
    },
},
  preferredLanguage "fi"
}
```

ASN.1 Format

```
SEQUENCE {
    INTEGER { 1 }
    OCTET STRING { 39 32 34 36 30 30 30 31 4A 41 30 30 30 30 30 30 31 }
    UTF8STRING { "FINEID" }
    [CONTEXT-SPECIFIC 0] { 48 45 4E 4B 49 4C 4F 4B 4F 52 54 54 49 }
    BIT STRING { 05 60 }
    [CONTEXT-SPECIFIC 2] {
        SEQUENCE {
            INTEGER { 0 }
            INTEGER { 1 }
            NULL { }
            BIT STRING { 02 44 }
            OBJECT IDENTIFIER { 1.2.840.113549.1.1.1 }
        }
        SEQUENCE {
            INTEGER { 2 }
            INTEGER { 70 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.113549.1.1.14 }
        }
        SEQUENCE {
            INTEGER { 3 }
            INTEGER { 64 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.113549.1.1.11 }
        }
        SEQUENCE {
            INTEGER { 4 }
            INTEGER { 65 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.113549.1.1.12 }
        }
        SEQUENCE {
            INTEGER { 5 }
            INTEGER { 66 }
```

```
    NULL { }
    BIT STRING { 06 40 }
    OBJECT IDENTIFIER { 1.2.840.113549.1.1.13 }
}
SEQUENCE {
    INTEGER { 7 }
    INTEGER { 71 }
    SEQUENCE {
       [CONTEXT-SPECIFIC 0] {
            SEQUENCE {
             OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.4 }
             NULL
            }
       }
       [CONTEXT-SPECIFIC 1] {
            SEQUENCE {
             OBJECT IDENTIFIER { 1.2.840.113549.1.1.8 }
             SEQUENCE {
                   OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.4 }
                   NULL
             }
            }
       }
       [CONTEXT-SPECIFIC 2] {
            INTEGER 28
       }
    }
    BIT STRING { 06 40 }
    OBJECT IDENTIFIER { 1.2.840.113549.1.1.10 }
}
SEQUENCE {
    INTEGER { 8 }
    INTEGER { 67 }
    SEQUENCE {
       [CONTEXT-SPECIFIC 0] {
            SEQUENCE {
             OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.1 }
             NULL
            }
       }
       [CONTEXT-SPECIFIC 1] {
            SEQUENCE {
             OBJECT IDENTIFIER { 1.2.840.113549.1.1.8 }
             SEQUENCE {
                   OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.1 }
                   NULL
             }
            }
       }
       [CONTEXT-SPECIFIC 2] {
            INTEGER 32
       }
    }
    BIT STRING { 06 40 }
    OBJECT IDENTIFIER { 1.2.840.113549.1.1.10 }
}
SEQUENCE {
    INTEGER { 9 }
    INTEGER { 68 }
    SEQUENCE {
       [CONTEXT-SPECIFIC 0] {
            SEQUENCE {
             OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.2 }
             NULL
            }
```

```
        }
        [CONTEXT-SPECIFIC 1] {
             SEQUENCE {
              OBJECT IDENTIFIER { 1.2.840.113549.1.1.8 }
              SEQUENCE {
                    OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.2 }
                    NULL
              }
             }
        }
        [CONTEXT-SPECIFIC 2] {
             INTEGER 48
        }
    }
    BIT STRING { 06 40 }
    OBJECT IDENTIFIER { 1.2.840.113549.1.1.10 }
}
SEQUENCE {
    INTEGER { 10 }
    INTEGER { 69 }
    SEQUENCE {
        [CONTEXT-SPECIFIC 0] {
             SEQUENCE {
              OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.3 }
              NULL
             }
        }
        [CONTEXT-SPECIFIC 1] {
             SEQUENCE {
              OBJECT IDENTIFIER { 1.2.840.113549.1.1.8 }
              SEQUENCE {
                    OBJECT IDENTIFIER { 2.16.840.1.101.3.4.2.3 }
                    NULL
              }
             }
        }
        [CONTEXT-SPECIFIC 2] {
             INTEGER 64
        }
    }
    BIT STRING { 06 40 }
    OBJECT IDENTIFIER { 1.2.840.113549.1.1.10 }
}
SEQUENCE {
    INTEGER { 12 }
    INTEGER { 9 }
    SEQUENCE {
        [0] {
             SEQUENCE {
                  OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.4)
                  NULL
             }
        }
        [1] {
             SEQUENCE {
              OBJECT IDENTIFIER (1.2.840.113549.1.1.8)
              SEQUENCE {
                    OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.4)
                    NULL
              }
             }
        }
    }
    BIT STRING { 02 04 }
    OBJECT IDENTIFIER { 1.2.840.113549.1.1.7 }
```

```
}
SEQUENCE {
   INTEGER { 13 }
   INTEGER { 9 }
   SEQUENCE {
      [0] {
           SEQUENCE {
                OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.1)
                NULL
           }
      }
      [1] {
           SEQUENCE {
            OBJECT IDENTIFIER (1.2.840.113549.1.1.8)
            SEQUENCE {
                  OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.1)
                  NULL
            }
           }
      }
   }
   BIT STRING { 02 04 }
   OBJECT IDENTIFIER { 1.2.840.113549.1.1.7 }
}
SEQUENCE {
   INTEGER { 14 }
   INTEGER { 9 }
   SEQUENCE {
      [0] {
           SEQUENCE {
                OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.2)
                NULL
           }
      }
      [1] {
           SEQUENCE {
            OBJECT IDENTIFIER (1.2.840.113549.1.1.8)
            SEQUENCE {
                  OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.2)
                  NULL
            }
           }
      }
   }
   BIT STRING { 02 04 }
   OBJECT IDENTIFIER { 1.2.840.113549.1.1.7 }
}
SEQUENCE {
   INTEGER { 15 }
   INTEGER { 9 }
   SEQUENCE {
      [0] {
           SEQUENCE {
                OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.3)
                NULL
           }
      }
      [1] {
           SEQUENCE {
            OBJECT IDENTIFIER (1.2.840.113549.1.1.8)
            SEQUENCE {
                  OBJECT IDENTIFIER (2.16.840.1.101.3.4.2.3)
                  NULL
            }
           }
```

```
                }
            }
            BIT STRING { 02 04 }
            OBJECT IDENTIFIER { 1.2.840.113549.1.1.7 }
        }
        SEQUENCE {
            INTEGER { 17 }
            INTEGER { 4163 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.10045.4.3.1 }
        }
        SEQUENCE {
            INTEGER { 18 }
            INTEGER { 4164 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.10045.4.3.2 }
        }
        SEQUENCE {
            INTEGER { 19 }
            INTEGER { 4165 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.10045.4.3.3 }
        }
        SEQUENCE {
            INTEGER { 20 }
            INTEGER { 4166 }
            NULL { }
            BIT STRING { 06 40 }
            OBJECT IDENTIFIER { 1.2.840.10045.4.3.4 }
        }
        SEQUENCE {
            INTEGER { 21 }
            INTEGER { 4176 }
            NULL { }
            BIT STRING { 07 00 80 }
            OBJECT IDENTIFIER { 1.3.132.1.12 }
        }
    }
    PrintableString { "fi" }
}
```

8.1.4. 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

ISO 7816-15 Interpretation

```
value CIOChoice ::= authObjects : path :
    {
      efidOrPath '3F004401'H
    }
value CIOChoice ::= privateKeys : path :
    {
      efidOrPath '3F004402'H
    }
value CIOChoice ::= certificates : path :
    {
      efidOrPath '3F004403'H
    }
value CIOChoice ::= certificates : path :
    {
      efidOrPath '3F004404'H
    }
value CIOChoice ::= trustedCertificates : path :
    {
      efidOrPath '3F004405'H
    }
value CIOChoice ::= dataContainerObjects : path :
    {
      efidOrPath '3F004406'H
    }
value CIOChoice ::= usefulCertificates : path :
    {
      efidOrPath '3F004407'H
    }
```

ASN.1 Format

```
[CONTEXT-SPECIFIC 8] {
  SEQUENCE {
    OCTET STRING { 3F 00 44 01 }
  }
}
[CONTEXT-SPECIFIC 0] {
  SEQUENCE {
    OCTET STRING { 3F 00 44 02 }
  }
}
[CONTEXT-SPECIFIC 4] {
  SEQUENCE {
    OCTET STRING { 3F 00 44 03 }
  }
}
[CONTEXT-SPECIFIC 4] {
  SEQUENCE {
    OCTET STRING { 3F 00 44 04 }
  }
}
[CONTEXT-SPECIFIC 5] {
  SEQUENCE {
    OCTET STRING { 3F 00 44 05 }
  }
}
[CONTEXT-SPECIFIC 7] {
  SEQUENCE {
    OCTET STRING { 3F 00 44 06 }
  }
}
[CONTEXT-SPECIFIC 6] {
  SEQUENCE {
    OCTET STRING { 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.

8.1.5. EF.AOD

Description

This elementary file (Authentication Object Directory) contains generic authentication object attributes such as allowed characters, length, padding character, etc. 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

Value notation example

ISO 7816-15 Interpretation

```
value AuthenticationObjectChoice ::= pwd :
  {
    commonObjectAttributes
    {
      label "perustunnusluku",
      flags { private, modifiable }
      authId '03'H
    },
    classAttributes
    {
      authId '01'H
    },
    typeAttributes
    {
      pwdFlags { case-sensitive, initialized, needs-padding,
exchangeRefData },
      pwdType ascii-numeric,
      minLength 4,
      storedLength 12,
      pwdReference 17,
      padChar '00'H,
    }
  }
value AuthenticationObjectChoice ::= pwd :
  {
    commonObjectAttributes
    {
      label "allekirjoitustunnusluku",
      flags { private, modifiable }
      authId '03'H

    },
    classAttributes
    {
      authId '02'H
    },
    typeAttributes
    {
```

```
      pwdFlags { case-sensitive, local, initialized, needs-padding,
exchangeRefData },
      pwdType ascii-numeric,
      minLength 6,
      storedLength 12,
      pwdReference 130,
      padChar '00'H,
    }
  }
value AuthenticationObjectChoice ::= pwd :
  {
    commonObjectAttributes
    {
      label "avaustunnusluku",
      flags { private, modifiable }
    },
    classAttributes
    {
      authId '03'H
    },
    typeAttributes
    {
      pwdFlags { case-sensitive, local, change-disabled, unblock-
disabled, initialized, needs-padding, unblockingPassword },
      pwdType ascii-numeric,
      minLength 8,
      storedLength 12,
      pwdReference 131,
      padChar '00'H,
    }
  }
```

ASN.1 Format

```
SEQUENCE {
  SEQUENCE {
    UTF8STRING { "perustunnusluku" }
    BIT STRING { 06 C0 }
    OCTET STRING { 03 }
  }
  SEQUENCE {
    OCTET STRING { 01 }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
        BIT STRING { 04 8C 10 }
        ENUMERATED { 01 }
        INTEGER { 4 }
        INTEGER { 12 }
        [CONTEXT-SPECIFIC 0] { 11 }
        OCTET STRING { 00 }
    }
  }
}
SEQUENCE {
  SEQUENCE {
    UTF8STRING { "allekirjoitustunnusluku" }
    BIT STRING { 06 C0 }
    OCTET STRING { 03 }
  }
  SEQUENCE {
    OCTET STRING { 02 }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
```

```
        BIT STRING { 04 CC 10 }
        ENUMERATED { 01 }
        INTEGER { 6 }
        INTEGER { 12 }
        [CONTEXT-SPECIFIC 0] { 00 82 }
        OCTET STRING { 00 }
    }
  }
}
SEQUENCE {
  SEQUENCE {
    UTF8STRING { "avaustunnusluku" }
    BIT STRING { 06 C0 }
  }
  SEQUENCE {
    OCTET STRING { 03 }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
        BIT STRING { 01 FE }
        ENUMERATED { 01 }
        INTEGER { 8 }
        INTEGER { 12 }
        [CONTEXT-SPECIFIC 0] { 00 83 }
        OCTET STRING { 00 }
    }
  }
}
```

8.1.6. 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.

Value notation example

ISO 7816-15 Interpretation

```
value PrivateKeyChoice ::= privateECKey :
  {
    commonObjectAttributes
    {
      label "todentamisavain",
      flags { private },
      authId '01'H,
      userConsent 1,
      accessControlRules
      {
        {
```

```
          accessMode { execute, pso_cds, pso_dec },
          securityCondition authId : '01'H
        }
      }
    },
    classAttributes
    {
      iD '45'H,
      usage { sign, derive },
      accessFlags { sensitive, alwaysSensitive, neverExtractable,
cardGenerated },
      keyReference 1
    },
    subClassAttributes
    {
      keyIdentifiers
      {
        {
          idType 4,
          idValue OCTET STRING :
'135651A53486D7B94C9EB76F49BAC5078EB2DE93'H
        }
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath ''H
          },
      keyInfo
      {
         namedCurve { "1.3.132.0.34" }
      }
    }
  }
value PrivateKeyChoice ::= privateECKey :
  {
    commonObjectAttributes
    {
      label "allekirjoitusavain ECC",
      flags { private },
      authId '02'H,
      userConsent 1,
      accessControlRules
      {
        {
          accessMode { execute, pso_cds },
          securityCondition authId : '02'H
        }
      }
    },
    classAttributes
    {
      iD '46'H,
      usage { nonRepudiation },
      accessFlags { sensitive, alwaysSensitive, neverExtractable,
cardGenerated },
      keyReference 2
    },
    subClassAttributes
    {
      keyIdentifiers
      {
        {
```

```
          idType 4,
          idValue OCTET STRING :
'BC6DD617A35E49B63A5F774CC1E5A93D06F398FB'H
        }
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath ''H
          },
      keyInfo
      {
         namedCurve { "1.3.132.0.34" }
      }
    }
  }
value PrivateKeyChoice ::= privateRSAKey  :
  {
    commonObjectAttributes
    {
      label "allekirjoitusavain RSA",
      flags { private },
      authId '02'H,
      userConsent 1,
      accessControlRules
      {
        {
          accessMode { execute, pso_cds },
          securityCondition authId : '02'H
        }
      }
    },
    classAttributes
    {
      iD '47'H,
      usage { nonRepudiation },
      accessFlags { sensitive, alwaysSensitive, neverExtractable,
cardGenerated },
      keyReference 3
    },
    subClassAttributes
    {
      keyIdentifiers
      {
        {
          idType 4,
          idValue OCTET STRING :
'AA6DD617A35E49B63A5F774CC1E5A93D06F398BB'H
        }
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath ''H
          },
      modulusLength 3072
    }
  }
```

ASN.1 Format

```
[CONTEXT-SPECIFIC 0] {
  SEQUENCE {
       UTF8STRING { "todentamisavain" }
       BIT STRING { 07 80 }
       OCTET STRING { 01 }
       INTEGER { 1 }
       SEQUENCE {
          SEQUENCE {
             BIT STRING { 00 25 }
             OCTET STRING { 01 }
          }
       }
  }
  SEQUENCE {
       OCTET STRING { 45 }
       BIT STRING { 07 20 80 }
       BIT STRING { 03 B8 }
       INTEGER { 1 }
  }
  [CONTEXT-SPECIFIC 0] {
       SEQUENCE {
          [CONTEXT-SPECIFIC 0] {
             SEQUENCE {
                  INTEGER { 4 }
                  OCTET STRING { 13 56 51 A5 34 86 D7 B9 4C 9E B7 6F 49
                  BA C5 07 8E B2 DE 93 }
          }
       }
  }
  }
  [CONTEXT-SPECIFIC 1] {
       SEQUENCE {
          SEQUENCE {
             OCTET STRING { }
          }
          SEQUENCE {
           OBJECT_IDENTIFIER { "1.3.132.0.34" }
          }
       }
  }
}
[CONTEXT-SPECIFIC 0] {
  SEQUENCE {
       UTF8STRING { "allekirjoitusavain ECC" }
       BIT STRING { 07 80 }
       OCTET STRING { 02 }
       INTEGER { 1 }
       SEQUENCE {
          SEQUENCE {
             BIT STRING { 02 24 }
             OCTET STRING { 02 }
          }
       }
  }
  SEQUENCE {
       OCTET STRING { 46 }
       BIT STRING { 06 00 40 }
       BIT STRING { 03 B8 }
       INTEGER { 2 }
  }
  [CONTEXT-SPECIFIC 0] {
       SEQUENCE {
          [CONTEXT-SPECIFIC 0] {
             SEQUENCE {
                  INTEGER { 4 }
```

```
                    OCTET STRING { BC 6D D6 17 A3 5E 49 B6 3A 5F 77 4C C1
                    E5 A9 3D 06 F3 98 FB }
                }
            }
        }
    }
    [CONTEXT-SPECIFIC 1] {
        SEQUENCE {
            SEQUENCE {
               OCTET STRING { }
            }
            SEQUENCE {
             OBJECT_IDENTIFIER { "1.3.132.0.34" }
            }
        }
    }
}
SEQUENCE {
  SEQUENCE {
        UTF8STRING { "allekirjoitusavain RSA" }
        BIT STRING { 07 80 }
        OCTET STRING { 02 }
        INTEGER { 1 }
        SEQUENCE {
            SEQUENCE {
               BIT STRING { 02 24 }
               OCTET STRING { 02 }
            }
        }
    }
    SEQUENCE {
        OCTET STRING { 47 }
        BIT STRING { 06 00 40 }
        BIT STRING { 03 B8 }
        INTEGER { 3 }
    }
    [CONTEXT-SPECIFIC 0] {
        SEQUENCE {
            [CONTEXT-SPECIFIC 0] {
               SEQUENCE {
                    INTEGER { 4 }
                    OCTET STRING { AA 6D D6 17 A3 5E 49 B6 3A 5F 77 4C C1
                    E5 A9 3D 06 F3 98 BB }
               }
            }
        }
    }
    [CONTEXT-SPECIFIC 1] {
        SEQUENCE {
            SEQUENCE {
               OCTET STRING { }
            }
            INTEGER { 3072 }
        }
    }
}
```

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

8.1.7. Private ELC Key #1

Description

This object contains the private ELC **'authentication key'**.

PIN 1 must be verified before ELC computation can be performed. PIN 1 verification status is dropped to state 'not verified' automatically by the card after each Compute Signature operation 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 Compute Signature operation.

Access conditions

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

8.1.8. Private ELC key #2

Description

This object contains the private ELC **'signature key ECC'**.

PIN 2 must be verified every time before ELC computation can be performed. PIN 2 verification status is dropped to state 'not verified' automatically by the card after each Compute Signature operation 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 Compute Signature operation.

Access conditions

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

8.1.9. Private RSA key #3

Description

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

PIN 2 must be verified every time before RSA computation can be performed. PIN 2 verification status is dropped to state 'not verified' automatically by the card after each Compute Signature operation 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 Compute Signature operation.

Access conditions

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

8.1.10. EF.CD #1

Description

This transparent elementary file contains attributes and pointers to card holder certificates. This certificates is 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 4.4. Card holder certificates.

Value notation example

ISO 7816-15 Interpretation

```
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "todentamisvarmenne",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '45'H,
      identifier
      {
        idType 3,
        idValue OCTET STRING :
'5340DFE896EFB3F4AE1748DF839AD2833873624B'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F004331'H
          }
```

```
     }
   }
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "allekirjoitusvarmenne ECC",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '46'H,
      identifier
      {
        idType 3,
        idValue OCTET STRING :
'2A211FAA8024BDECF2C119E10F45EC281348A88C'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F0050164332'H
          }
    }
  }
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "allekirjoitusvarmenne RSA",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '47'H,
      identifier
      {
        idType 3,
        idValue OCTET STRING :
'3B211FAA8024BDECF2C119E10F45EC281348A89D'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F0050164333'H
          }
    }
```

```
  }
```

ASN.1 Format

```
 SEQUENCE {
  SEQUENCE {
    UTF8STRING { "todentamisvarmenne" }
    BIT STRING { 00 }
    SEQUENCE {
         SEQUENCE {
            BIT STRING { 07 80 }
            NULL { }
         }
    }
  }
  SEQUENCE {
    OCTET STRING { 45 }
    SEQUENCE {
         INTEGER { 3 }
         OCTET STRING { 53 40 DF E8 96 EF B3 F4 AE 17 48 DF 83 9A D2 83 38
         73 62 4B }
    }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
         SEQUENCE {
            OCTET STRING { 3F 00 43 31 }
         }
    }
  }
 }
SEQUENCE {
  SEQUENCE {
  UTF8STRING { "allekirjoitusvarmenne ECC" }
  BIT STRING { 00 }
    SEQUENCE {
         SEQUENCE {
            BIT STRING { 07 80 }
            NULL { }
         }
    }
  }
  SEQUENCE {
    OCTET STRING { 46 }
    SEQUENCE {
         INTEGER { 3 }
         OCTET STRING { 2A 21 1F AA 80 24 BD EC F2 C1 19 E1 0F 45 EC 28 13
         48 A8 8C }
    }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
         SEQUENCE {
            OCTET STRING { 3F 00 50 16 43 32 }
         }
    }
  }
}
SEQUENCE {
  SEQUENCE {
  UTF8STRING { "allekirjoitusvarmenne RSA" }
  BIT STRING { 00 }
    SEQUENCE {
         SEQUENCE {
            BIT STRING { 07 80 }
```

```
            NULL { }
          }
      }
    }
    SEQUENCE {
      OCTET STRING { 47 }
      SEQUENCE {
          INTEGER { 3 }
          OCTET STRING { 3B 21 1F AA 80 24 BD EC F2 C1 19 E1 0F 45 EC 28 13
          48 A8 9D }
      }
    }
    [CONTEXT-SPECIFIC 1] {
      SEQUENCE {
          SEQUENCE {
            OCTET STRING { 3F 00 50 16 43 33 }
          }
      }
    }
}
```

Files 3F00/4331, 3F00/5016/4332 and 3F00/5016/4333 should contain DER encoded certificate structure in accordance with ISO/IEC 9594-8.

8.1.11. Certificate #1

Description

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

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.1.12. 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)

8.1.13. 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:

- ‘DVV Gov. Root CA - G3 ECC’ (self signed)
- ‘DVV Gov. Root CA - G3 RSA’ (self signed)
- **‘**DVV Citizen Certificates - G4E**’** (signed by ‘DVV Gov. Root CA - G3 ECC’)
- **‘**DVV Citizen Certificates - G4R**’** (signed by ‘DVV Gov. Root CA - G3 RSA’)

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

Value notation example

ISO 7816-15 Interpretation

```
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "DVV Gov. Root CA - G3 ECC",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '50'H,
      authority TRUE,
      identifier
      {
        idType 2,
        idValue OCTET STRING :
'B97680C383F8CF22DA85EBCE144E55AC5E5A0A90'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F004334'H
          }
    }
```

```
  }
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "DVV Gov. Root CA - G3 RSA",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '51'H,
      authority TRUE,
      identifier
      {
        idType 2,idValue OCTET STRING :
'5B01E0CF5ED0C480B5508A8138878392BF150C8C'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F004335'H
          }
    }
  }
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "DVV Citizen Certificates - G4E",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '52'H,
      authority TRUE,
      identifier
      {
        idType 2,idValue OCTET STRING :
'DF9CBCBC82E8A1A8532697F974A3ECEA1DC997CE'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F004336'H
          }
    }
  }
```

```
value CertificateChoice ::= x509Certificate :
  {
    commonObjectAttributes
    {
      label "DVV Citizen Certificates - G4R",
      flags {  },
      accessControlRules
      {
        {
          accessMode { read },
          securityCondition always : NULL
        }
      }
    },
    classAttributes
    {
      iD '53'H,
      authority TRUE,
      identifier
      {
        idType 2,idValue OCTET STRING :
'08D14EB9F21359FFBBF51C2ED823DCD6C1FE3FC4'H
      }
    },
    typeAttributes
    {
      value indirect : path :
          {
            efidOrPath '3F004337'H
          }
    }
  }
```

ASN.1 Format

```
SEQUENCE {
  SEQUENCE {
    UTF8STRING { "DVV Gov. Root CA - G3 ECC" }
    BIT STRING { 00 }
    SEQUENCE {
        SEQUENCE {
           BIT STRING { 07 80 }
           NULL { }
        }
    }
  }
  SEQUENCE {
    OCTET STRING { 50 }
    BOOLEAN { TRUE }
    SEQUENCE {
        INTEGER { 2 }
        OCTET STRING { B9 76 80 C3 83 F8 CF 22 DA 85 EB CE 14 4E 55 AC 5E
        5A 0A 90 }
    }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
        SEQUENCE {
           OCTET STRING { 3F 00 43 34 }
        }
    }
  }
}
SEQUENCE {
  SEQUENCE {
```

```
      UTF8STRING { "DVV Gov. Root CA - G3 RSA" }
      BIT STRING { 00 }
      SEQUENCE {
           SEQUENCE {
              BIT STRING { 07 80 }
              NULL { }
           }
      }
    }
    SEQUENCE {
      OCTET STRING { 51 }
      BOOLEAN { TRUE }
      SEQUENCE {
           INTEGER { 2 }
           OCTET STRING { 5B 01 E0 CF 5E D0 C4 80 B5 50 8A 81 38 87 83 92 BF
           15 0C 8C }
      }
    }
    [CONTEXT-SPECIFIC 1] {
      SEQUENCE {
           SEQUENCE {
              OCTET STRING { 3F 00 43 35 }
           }
      }
    }
  }
  SEQUENCE {
    SEQUENCE {
      UTF8STRING { "DVV Citizen Certificates - G4E" }
      BIT STRING { 00 }
      SEQUENCE {
           SEQUENCE {
              BIT STRING { 07 80 }
              NULL { }
           }
      }
    }
    SEQUENCE {
      OCTET STRING { 52 }
      BOOLEAN { TRUE }
      SEQUENCE {
           INTEGER { 2 }
           OCTET STRING { DF 9C BC BC 82 E8 A1 A8 53 26 97 F9 74 A3 EC EA 1D
           C9 97 CE }
      }
    }
    [CONTEXT-SPECIFIC 1] {
      SEQUENCE {
           SEQUENCE {
              OCTET STRING { 3F 00 43 36 }
           }
      }
    }
  }
  SEQUENCE {
    SEQUENCE {
      UTF8STRING { "DVV Citizen Certificates - G4R" }
      BIT STRING { 00 }
      SEQUENCE {
           SEQUENCE {
              BIT STRING { 07 80 }
              NULL { }
           }
      }
    }
```

```
  SEQUENCE {
    OCTET STRING { 53 }
    BOOLEAN { TRUE }
    SEQUENCE {
         INTEGER { 2 }
         OCTET STRING { 08 D1 4E B9 F2 13 59 FF BB F5 1C 2E D8 23 DC D6 C1
         FE 3F C4 }
    }
  }
  [CONTEXT-SPECIFIC 1] {
    SEQUENCE {
         SEQUENCE {
            OCTET STRING { 3F 00 43 37 }
         }
    }
  }
}
```

Files 3F00/4334, 3F00/4335, 3F00/4336 and 3F00/4337 should contain DER-encoded CA certificates in accordance with ISO/IEC 9594-8.

8.1.14. CA Certificate #1

Description

This file contains the trusted root CA certificate “DVV Gov. Root CA - G3 ECC“ (keyLength 384 bits, self-signed). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.1.15. CA Certificate #2

Description

This file contains the trusted root CA certificate “DVV Gov. Root CA - G3 RSA“ (keyLength 4096 bits, self-signed). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.1.16. CA Certificate #3

Description

This file contains the trusted intermediate CA certificate “DVV Citizen Certificates - G4E” (keyLength 384 bits, signed by “DVV Gov. Root CA - G3 ECC“). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.1.17. CA Certificate #4

Description

This file contains the trusted intermediate CA certificate “DVV Citizen Certificates - G4R” (keyLength 4096 bits, signed by “DVV Gov. Root CA - G3 RSA“). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.1.18. 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)

8.1.19. 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)

8.1.20. 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

ISO 7816-15 Interpretation

```
value UnusedSpace ::=
{
  path
  {
    path '3F00433F'H,
    index 0,
    length 8192
  },
  authId '01'H,
  accessControlRules
  {
    {
      accessMode { read },
      securityCondition always : NULL
    },
    {
      accessMode { update },
      securityCondition authId : '01'H
    }
  }
}
value UnusedSpace ::=
{
  path
  {
    path '3F00433E'H,
    index 0,
    length 4096
  },
```

```
   authId '01'H,
   accessControlRules
   {
     {
       accessMode { read, update },
       securityCondition authId : '01'H
     }
   }
 }
```

ASN.1 Format

```
SEQUENCE {
  SEQUENCE {
    OCTET STRING { 3F 00 43 3F }
    INTEGER { 0 }
    [CONTEXT-SPECIFIC 0] { 20 00 }
  }
  OCTET STRING { 01 }
  SEQUENCE {
    SEQUENCE {
         BIT STRING { 07 80 }
         NULL { }
    }
    SEQUENCE {
         BIT STRING { 06 40 }
         OCTET STRING { 01 }
    }
  }
}
SEQUENCE {
  SEQUENCE {
    OCTET STRING { 3F 00 43 3E }
    INTEGER { 0 }
    [CONTEXT-SPECIFIC 0] { 10 00 }
  }
  OCTET STRING { 01 }
  SEQUENCE {
    SEQUENCE {
       BIT STRING { 06 C0 }
       OCTET STRING { 01 }
    }
  }
}
```

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).

8.1.21. 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.

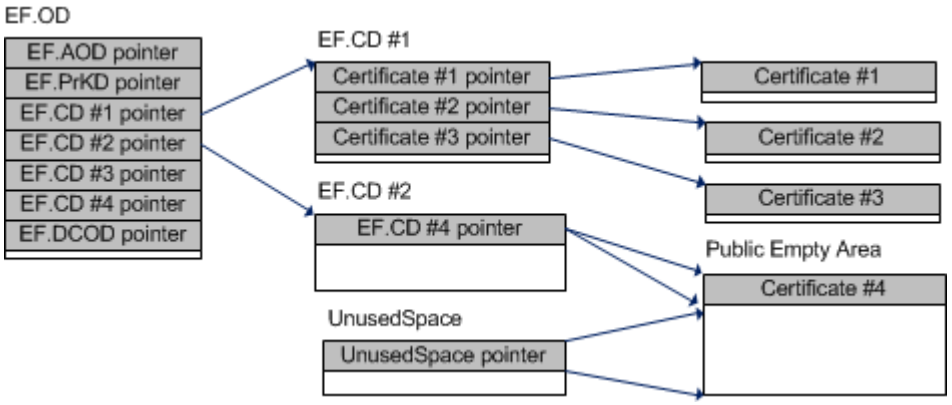
This file is optional.

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 #4 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.



8.1.22. 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.

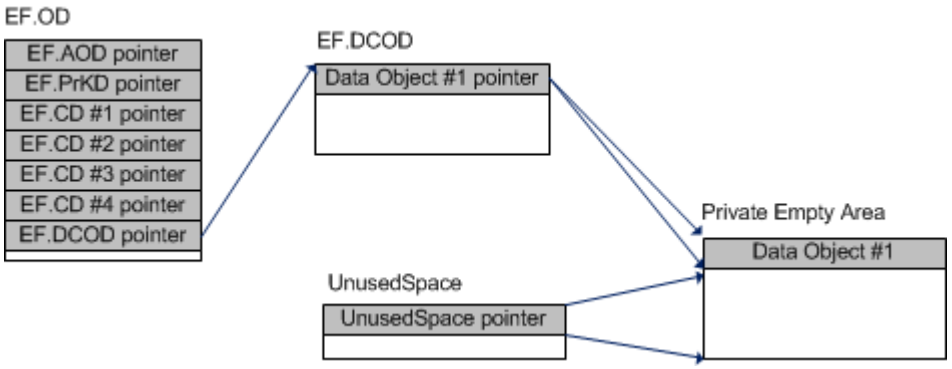
This file is optional.

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.



8.2. DF.ESIGN

Description

The DF.ESIGN represents the subdirectory of the FINEID application, where Certificate #2 and Certificate #3 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

8.2.1. Certificate #2

Description

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

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

8.2.2. Certificate #3

Description

This file contains the card holder's **'signature certificate RSA'** containing the public key corresponding to the private **'signature key RSA'** (Private RSA Key #3). The certificate in this file is DER encoded.

Access conditions

Access method	Access condition
Read	ALW
Update	NEV

9. Certificates

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