FINEID - S1 Electronic ID Application

v 2.1

Population Register Centre (VRK)

Certification Authority Services P.O. Box 70 FIN-00581 Helsinki Finland http://www.fineid.fi



i

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
2.1	15.3.2004	AP	References to ISO/IEC FDIS 7816- 15 changed to ISO/IEC 7816-15	Accepted
2.0A	30.6.2003	AP	References to ISO FCD 7816-15 changed to ISO/IEC FDIS 7816-15	Accepted
2.0	10.3.2003	AP	Editorial changes, minor corrections	Accepted
1.0	7.11.2002	MaSi	First edition	Draft

Table of Contents

1	Intr	oduction	1
	1.1	Normative references	1
	1.2	Informative references	1
	1.3	Related FINEID documentation	1
2	Abb	previations	1
3	File	e structure and contents	2
4	Со	nmand interface	3
	4.1	SELECT	3
	4.2	SELECT FILE	4
	4.3	GET RESPONSE	5
	4.4	READ BINARY	5
	4.5	VERIFY	6
	4.6	MANAGE SECURITY ENVIRONMENT: RESTORE	7
	4.7	MANAGE SECURITY ENVIRONMENT: SET	7
	4.8	PERFORM SECURITY OPERATION: HASH	9
	4.9	PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE	10
	4.10	PERFORM SECURITY OPERATION: DECIPHER	11
	4.11	CHANGE REFERENCE DATA	11
	4.12	RESET RETRY COUNTER	12
	4.13	UPDATE BINARY	13
	4.14	ERASE BINARY	14
	4.15	GET DATA	14
5	Imp	elementation guidelines for software developers	16
	5.1		
	5.2	Resetting the card	16
	5.3	Application/File selection	17
		5.3.1 CIA application	17
		5.3.2 Path	17
	5.4	Authentication objects	17
		5.4.1 Accessing objects	18
		5.4.2 Auth required flag	19
	5.5	Private key operations (sign and decrypt)	19
		5.5.1 Signature operation	19
		5.5.2 Decryption operation	20
An	nex	A (Informative): Coding of the File Control Information template	21
		Access conditions	23
		Simple coding	24
		Adaptive coding	26
		SC byte 27	
An	nex	B (Informative): Status conditions	28

iii

Success conditions	
Warning conditions	28
wanning conditions	20
Error conditions	28

15.3.2004 1(29)

1 Introduction

This document describes the command interface and the content of the Finnish Electronic Identification (FINEID) application version 2.1. The FINEID application version 1.1 is described in the document "FINEID – S1, Electronic ID application, v1.1, Markku Kontio".

The file structure is based on ISO/IEC 7816-15. The command set supported by the card is based on ISO/IEC 7816-4 and ISO/IEC 7816-8.

All ASN.1 type, value and information object class definitions referred in this document are from module Cryptographic Information Framework defined in ISO/IEC 7816-15.

1.1 Normative references

The most important specifications are listed below:

- ISO, Information Technology Identification cards Integrated circuit(s) cards with contacts
 - Part 1: Physical Characteristics, ISO/IEC 7816-1
 - Part 2: Dimensions and location of the contacts, ISO/IEC 7816-2
 - Part 3: Electronic signals and transmission protocols, ISO/IEC 7816-3
 - Part 4: Interindustry commands for interchange, ISO/IEC 7816-4

Part 5: Numbering system and registration procedure for application identifiers, ISO/IEC 7816-5

Part 6: Interindustry data elements, ISO/IEC 7816-6

Part 8: Security related interindustry commands, ISO/IEC 7816-8

Part 15: Cryptographic information application, ISO/IEC 7816-15

- Open Platform, Card Specification, Version 2.0.1, Global Platform, 7 April 2000.

1.2 Informative references

The following documents have also influenced this specification:

- PKCS#1 v2.1, RSA Cryptography Standard, June 14, 2002.
- DIN NI-17.4 v1.0, DIN Specification of chipcard interface with digital signature application/function acc. to SigG and SigV, 15.12.1998.
- FINEID S1, Electronic ID application v1.12, 4.11.2002.

1.3 Related FINEID documentation

FINEID documentation is available at

- http://www.fineid.fi

2 Abbreviations

Access Condition
Application IDentifier
Application Protocol Data Unit
Abstract Syntax Notation One

CIA	Cryptographic Information Application
CLA	Class byte
СТ	Confidentiality Template
CRDO	Control Reference Data Object
CRT	Chinese Remainder Theorem
DF	Dedicated File
DO	Data Object
DST	Digital Signature Template
EF	Elementary File
FCI	File Control Information
FCP	File Control Parameter
FINEID	Finnish Electronic IDentification
MF	Master File
MSE	Manage Security Environment
PIN	Personal Identification Number
PKCS	Public-Key Cryptography Standards
PSO	Perform Security Operation
PUK	PIN Unblocking Key
RFU	Reserved for Future Use
RSA	Rivest, Shamir, Adleman
SC	Security Condition
SE	Security Environment
SFID	Short File IDentifier
S/MIME	Secure Multipurpose Internet Mail Extensions
SW1-SW2	Status bytes

3 File structure and contents

The file structure and contents shall be according to ISO/IEC 7816-15 standard.

The application should contain at least the following objects:

- private key(s),
- authentication object(s),
- card holder certificate(s) and
- trusted certificate(s).

The reader is advised to read ISO/IEC 7816-15 for additional information of the file structure and contents.

15.3.2004 3(29)

4 Command interface

This chapter describes the commands (and their parameters) that shall be supported by FINEID application. Additional commands may be supported by the application but they are not normally used by host applications utilizing the FINEID application.

The reader is advised to refer to ISO/IEC 7816-4 and ISO/IEC 7816-8 for more detailed information about the commands.

Command	Standard	Functionality
SELECT	Open Platform, Card Specification, version 2.0.1	Select an application on the card.
SELECT FILE	ISO/IEC 7816-4	Select a file from the card's file system
GET RESPONSE	ISO/IEC 7816-4	Read response data from the card (in T=0 protocol)
READ BINARY	ISO/IEC 7816-4	Read binary data from a transparent (binary) file
VERIFY	ISO/IEC 7816-4	Verify reference data presented by user (e.g. PIN) with the reference data stored inside the card. The current verification status can be also queried with this command.
MANAGE SECURITY ENVIRONMENT: RESTORE	ISO/IEC 7816-8	Restore a predefined (or empty) security environment.
MANAGE SECURITY ENVIRONMENT: SET	ISO/IEC 7816-8	Set the security environment (algorithms, keys) that shall be used in the following PERFORM SECURITY OPERATION commands.
PERFORM SECURITY OPERATION: HASH	ISO/IEC 7816-8	Calculate a hash code. The algorithm is specified with the MSE command.
PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE	ISO/IEC 7816-8	Compute a digital signature with a private key. The algorithm and key are specified with the MSE command.
PERFORM SECURITY OPERATION: DECIPHER	ISO/IEC 7816-8	Decrypt data with a private key. The algorithm and key are specified with the MSE command.
CHANGE REFERENCE DATA	ISO/IEC 7816-8	Change the current reference data (e.g. PIN)
RESET RETRY COUNTER	ISO/IEC 7816-8	Unlock locked reference data (e.g. PIN)
UPDATE BINARY	ISO/IEC 7816-4	Update the contents of a transparent (binary) file
ERASE BINARY	ISO/IEC 7816-4	Erase the contents of a transparent (binary) file
GET DATA	ISO/IEC 7816-4	Retrieve the public part of a RSA key

Table 1. EID application related commands

4.1 SELECT

The Select command selects an application on the card. All successive commands are handled by the selected application until a new application selection is made.

Table 2. SELECT command APDU

Byte	Value
CLA	00h
INS	A4h
P1	04h – select by name (by Application IDentifier (AID))
P2	00h – select first or only occurrence
	02h – select next occurrence
Lc	length of subsequent data field
Data	AID
Le	00h

Table 3. SELECT response APDU

Byte	Value
Data	File Control Information (FCI)
SW1-SW2	Status bytes

The content of FCI is described in Annex A.

4.2 SELECT FILE

The SELECT FILE command selects a file from the card's file system according to file identifier, file path or application identifier (AID).

Byte	Value
CLA	00h
INS	A4h
P1	00h - select EF, DF or MF by file identifier 08h - select file by absolute path from MF 09h - select file by relative path from current DF
P2	00h - FCI returned in response
Lc	Empty or length of subsequent data field
Data	 P1 = 00h EF, DF or MF file identifier (or empty = MF) P1 = 08h absolute path from MF without the identifier of MF (3F00h) P1 = 09h relative path from the current DF without the identifier of the current DF
Le	Empty or maximum length of data expected in response

Table 5. SELECT FILE response APDU

Byte	Value
Data	File Control Information (FCI)
SW1-SW2	Status bytes

The content of FCI is described in Annex A.

4.3 GET RESPONSE

The GET RESPONSE command returns response data from the card in T=0 protocol.

This command is used in to get response data from commands

- SELECT FILE,
- PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE and
- PERFORM SECURITY OPERATION: DECIPHER.

Table 6. GET RESPONSE command APDU

Byte	Value
CLA	00h
INS	C0h
P1	00h
P2	00h
Lc	Empty
Data	Empty
Le	Maximum length of data expected in response

Table 7. GET RESPONSE response APDU

Byte	Value						
Data	Value of the response						
SW1-SW2 Status bytes							

4.4 READ BINARY

The READ BINARY command is used to read consecutive bytes from the current (transparent) elementary file.

Byte	Value						
CLA	00h						
INS	B0h						
P1	See table below						
P2	See table below						
Lc	Empty						
Data	Empty						
Le	Number of bytes to read						

Table 8. READ BINARY command APDU

Table 9. READ BINARY: coding of P1 and P2.

	Coding of P1 and P2								
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning
0	-	-	-	-	-	-	-	-	P1-P2 specifies a 15-bit offset of the data to be read
1	-	-	-	-	-	-	-	-	P1 specifies a short FID and P2 specifies an 8-bit offset of the data to be read
1	0	0	Х	х	x	х	x	-	– short FID (value domain 1 – 30)

Table 10. READ BINARY response APDU

Byte	Value					
Data	a Data read from the file					
SW1-SW2	Status bytes					

4.5 VERIFY

The VERIFY command is used to authenticate the user. Verification data (e.g. PIN) is compared with the reference data stored internally by the card.

Byte	Value					
CLA	00h					
INS	20h					
P1	00h					
P2 Qualifier of the PIN, see table below.						
Lc	Empty or length of subsequent data field					
Data	Empty or verification data (padded to the correct length).					
	Padding is done according to ISO/IEC 7816-15.					
Le	Empty					

Table 11. VERIFY command APDU

Table 12. Qualifier of the PIN

	Coding of the P2								
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning
0	-	-	-	-	-	-	-	-	Global reference data (card PIN). Not supported in FINEID context.
1	-	-	-	-	-	-	-	-	Specific reference data (DF specific PIN)
-	Х	Х	х	x	-	-	-	-	'0000' (Other values are RFU)
-	-	-	-	-	х	х	х	-	PIN number (according to ISO/IEC 7816-15)
-	-	-	-	-	0	0	0		RFU

Table 13. VERIFY response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

15.3.2004	
7(29)	

If $Lc = 00h$, the command can be used to retrieve the number X of further allowed
retries (SW1-SW2 = 63 CXh), or to check whether the verification is not required
(SW1-SW2 = 9000h).

4.6 MANAGE SECURITY ENVIRONMENT: RESTORE

The MANAGE SECURITY ENVIRONMENT: **RESTORE** command is used to restore a predefined (or empty) SECURITY ENVIRONMENT.

Table 14. MANAGE SECURITY ENVIRONMENT: RESTORE command APDU

Byte	Value				
CLA	00h				
INS 22h – MSE					
P1	11110011b = F3h - RESTORE				
P2	Number of the SE to be restored (00h is an empty SE)				
Lc	Empty				
Data	Empty				
Le	Empty				

Table 15. MANAGE SECURITY ENVIRONMENT: RESTORE response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.7 MANAGE SECURITY ENVIRONMENT: SET

The MANAGE SECURITY ENVIRONMENT: **SET** command is used to set attributes in the current SECURITY ENVIRONMENT.

Byte	Value	
CLA	00h	
INS	22h	
P1	01000001b = 41h - computation and decipherment	
	-	
P2	P1 = SET	
	- $P2 = B6h$, value of DST in data field	
	- $P2 = B8h$, value of CT in data field	
Lc	Empty or length of subsequent data field	
Data	Concatenation of CRDOs	
Le	Empty	

Table 16. MANAGE SECURITY ENVIRONMENT: SET command APDU

Table 17. MANAGE SECURITY ENVIRONMENT:SET response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

FINEID SPECIFICATION	15.3.2004
FINEID - S1 / v2.1	8(29)

The table below describes the Control Reference Data Objects (CRDO) that are supported in Digital Signature Templates (DST) and Confidentiality Templates (CT).

Tag	Value	DST	СТ
80h	Algorithm reference	+	+
81h	 File reference (file identifier or a path) only file identifiers shall be used in FINEID context used to identify the key file to be used in cryptographic operations value specified in PKCS#15 private key object's Path.efidOrPath 	+	+

Table 19. MANAGE SECURITY ENVIRONMENT:SET supported P1-P2 combinations

	Supported combinations of P1-P2			
P1 P2 Meaning CRDO in data field Data field contents		Data field contents		
'41'	'B6'	SET SE for digital signature	DST	' 80 01 xx 81 02 xx xx '
'41'	'B8'	SET SE for decipherment	СТ	' 80 01 xx 81 02 xx xx '

The supported values for the CRDO algorithm reference (tag 80h) are specified in the table below. The coding is taken from DIN NI-17.4 version 1.0 specification (annex F table F.2) with some modifications. The high nibble of the algorithm reference specifies the hash algorithm used (if hashing is relevant for the algorithm). The low nibble specifies the rest of the details about the algorithm.

Table 20. Values for the algorithm reference

Algorithm reference	Details			
0Xh	No hash algorithm			
1Xh	SHA-1 hash algorithm (id-sha1)			
2Xh	RFU			
X0h	'Raw' RSA algorithm (card does not do any input or output formatting i.e. padding or hash encapsulation)			
	 Signature generation operation (PSO: COMPUTE DIGITAL SIGNATURE): Input data size must equal modulus length i.e. hash is NOT encapsulated or padded by the card. Modulus length shall be a multiple of eight for this algorithm. RSASP1 signature primitive is applied (RSA private key operation) 			
	 Decryption operation (PSO: DECIPHER): 1. RSADP decryption primitive is applied (RSA private key operation) 2. Padding is NOT removed by the card. 			
X1h	RFU			

X2h	RSASSA-PKCS1-v1_5 signature scheme (according to PKCS#1 v2.1 with RSA algorithm, compatible with PKCS#1 v1.5)			
	Signature generation operation (PSO: COMPUTE DIGITAL SIGNATURE):			
	1. The hash code is encapsulated into DigestInfo ASN.1 structure according to selected hash algorithm. If no hash algorithm is selected (02h), the hash encapsulation is not done by the card.			
	2. DigestInfo is padded to modulus length according to PKCS#1 v1.5 (block type 01h). The size of the DigestInfo shall not be more than 40% of modulus length.			
	3. RSASP1 signature primitive is applied (RSA private key operation)			
	RSAES-PKCS1-v1_5 encryption scheme (according to PKCS#1 v2.1 with RSA algorithm, compatible with PKCS#1 v1.5)			
	Decryption operation (PSO: DECIPHER):			
	1. RSADP decryption primitive is applied (RSA private key operation)			
	2. PKCS#1 v1.5 padding is removed			
X3h	RFU			
X4h	RFU			

4.8 PERFORM SECURITY OPERATION: HASH

PSO: HASH command calculates a hash sum over a large amount of data. The algorithm to be used must be specified using the MANAGE SECURITY ENVIRONMENT command (using DST CRDO in the data field). Currently only supported algorithm is SHA-1.

This command supports command chaining mechanism which utilizes the CLA value to indicate the end of the command chain. The command chain has CLA = 10h for all but the last command of the chain, which has CLA = 00h. In chained commands the commands with CLA = 10h shall carry only data quantities which are multiples of the block size of the hashing algorithm (64 bytes for SHA-1). The last command of the chain has no data length limitations. In order to be able to sign or verify the generated hash sum, the CLA must be 00h (end of chain) in the PSO: HASH command given immediately before the PSO: COMPUTE DIGITAL SIGNATURE command.

Byte	Value
CLA	00h/10h
INS	2Ah
P1	90h
P2	80h
Lc	Length of subsequent data field
Data	Data to be hashed
Le	Empty.

Table 21	PSO.	HASH	command APDU	
1 4010 21.	IDU.	117 1011		

The data field may contain zero or more (plain value) bytes to be integrated into the hash sum (if no bytes are provided, the initial hash state is generated). Length of the data field shall be multiple of the block size of the hashing algorithm (64 bytes for SHA-1) for all but the last command of the chain. The algorithm identifier specified in the previous MANAGE SECURITY ENVIRONMENT: SET command (using the DST CRDO) shall be 12h. Table 22. PSO: HASH response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

The calculated hash code is stored in the card and available for use in a subsequent command (PSO: COMPUTE DIGITAL SIGNATURE).

4.9 PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE

The PSO: COMPUTE DIGITAL SIGNATURE command calculates a digital signature. The private key and algorithm to be used must be specified using the MANAGE SECURITY ENVIRONMENT command.

The input to the command may be either

- a hash code (e.g. SHA-1 hash value 20 bytes),
- a DigestInfo ASN.1 structure encapsulating the hash code, or
- a full modulus size input buffer (padding done by host application), or
- empty (hash code is calculated by preceding PSO: HASH command(s))

according to the selected algorithm reference value.

Byte	Value
CLA	00h
INS	2Ah
P1	9Eh - digital signature data object is returned in response
P2	9Ah - data field contains data to be signed
Lc	Length of subsequent data field
Data	If algorithm reference in $SE = 00h$
	- Data to be signed (e.g. encapsulated hash code). Padding is done to the full modulus length by the host application.
	If algorithm reference in $SE = 02h$:
	- Hash code encapsulated by the host application into DigestInfo structure. Padding is done internally by the card.
	If algorithm reference in $SE = 12h$
	- Hash code. Card encapsulates the hash into DigestInfo structure and pads it internally according to PKCS#1 v1.5 into full modulus length.
	- Empty. Hash code is calculated by preceding PSO: HASH command(s).
Le	Empty or maximum length of data expected in response

Table 23. PSO: COMPUTE DIGITAL SIGNATURE command APDU

Table 24. PSO: COMPUTE DIGITAL SIGNATURE response APDU

Byte	Value
Data	Digital signature
SW1-SW2	Status bytes

4.10 PERFORM SECURITY OPERATION: DECIPHER

The PSO: DECIPHER command decrypts an encrypted message (cryptogram). The private key and algorithm to be used must be specified using the MANAGE SECURITY ENVIRONMENT command.

Byte	Value
CLA	00h
INS	2Ah
P1	80h - decrypted value is returned in response
P2	86h - data field contains padding indicator byte (00h according to ISO/IEC 7816-4) followed by the cryptogram
Lc	Length of subsequent data field
Data	00h (padding indicator byte) cryptogram
Le	Empty or maximum length of data expected in response

Table 25. PSO: DECIPHER command APDU

Table 26. PSO: DECIPHER response APDU

Byte	Value						
Data	f algorithm reference in $SE = 00h$						
	- Decrypted cryptogram. Padding is not removed by the card.						
	If algorithm reference in $SE = 02h$:						
	- Decrypted cryptogram. PKCS#1 v1.5 padding is removed by the card and only the actual data is returned.						
SW1-SW2	Status bytes						

4.11 CHANGE REFERENCE DATA

The CHANGE REFERENCE DATA command is used to change the current internally stored reference data into a new value. Current reference data is first compared with verification data presented by the user.

Byte	Value
CLA	00h
INS	24h
P1	00h - exchange reference data
P2	Qualifier of the PIN, see table below.
Lc	Length of subsequent data field
Data	Existing reference data (padded to the correct length) followed by new reference data (padded to the correct length). Padding is done according to ISO/IEC 7816-15.
Le	Empty

Table 28. Qualifier of the PIN

	Coding of the P2										
b8	b7	b6	B5	b4	b3	b2	b1	Hex	Meaning		
0	-	-	-	-	-	-	-	-	Global reference data (card PIN). Not supported in FINEID context.		
1	-	-	-	-	-	-	-	-	Specific reference data (DF specific PIN)		
-	x	X	X	x	-	-	-	-	'0000' (Other values are RFU)		
-	-	-	-	-	X	x	x	-	PIN number (according to ISO/IEC 7816-15)		
-	-	-	-	-	0	0	0		RFU		

Table 29. CHANGE REFERENCE DATA response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.12 RESET RETRY COUNTER

The RESET RETRY COUNTER command is used when a PIN code has been locked due to too many consecutive unsuccessful verifications. Unlocking a PIN requires a resetting code (a.k.a. PIN Unlocking Key, PUK) to be presented to the card by the user.

Table 30. RESET RETRY COUNTER command APDU

Byte	Value
CLA	00h
INS	2Ch
P1	00h - reset retry counter and set new verification data
P2	Qualifier of the PIN, see table below.
Lc	Empty or length of subsequent data field
Data	Empty or resetting code (padded to the correct length) followed by new reference data (padded to the correct length). Padding is done according to ISO/IEC 7816-15.
Le	Empty

Table 31. Qualifier of the PIN

	Coding of the P2									
b8	b7	b6	B5	b4	b3	b2	b1	Hex	Meaning	
0	-	-	-	-	-	-	-	-	Global reference data (card PIN). Not supported in FINEID context.	
1	-	-	-	-	-	-	-	-	Specific reference data (DF specific PIN)	
-	X	X	X	х	-	-	-	-	'0000' (Other values are RFU)	
-	-	-	-	-	X	X	x	-	PIN number (according to ISO/IEC 7816-15)	
-	-	-	-	-	0	0	0		RFU	

Table 32. RESET RETRY COUNTER response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes
	If $Lc = 00h$, status bytes indicate the number X of further allowed retries (SW1-SW2 = 63CXh).

4.13 UPDATE BINARY

The UPDATE BINARY command is used update the contents of a transparent (binary) file.

Table 33. UPDATE BINARY command APDU

Byte	Value						
CLA	00h						
INS	D6h						
P1	See table below						
P2	e table below						
Lc	Length of subsequent data field						
Data	Data to be updated						
Le	Empty						

Table 34. UPDATE BINARY: coding of P1 and P2.

	Coding of P1 and P2										
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning		
0	-	-	-	-	-	-	-	-	P1-P2 specifies a 15-bit offset of the data to be read		
1	-	-	-	-	-	-	-	-	P1 specifies a short FID and P2 specifies an 8-bit offset of the data to be read		
1	0	0	х	х	х	х	х	-	– short FID (value domain 1 – 30)		

Table 35. UPDATE BINARY response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.14 ERASE BINARY

The ERASE BINARY command is used erase the contents of a transparent (binary) file. Erasing is done starting from the address specified in bytes P1 and P2 until the end of file.

Byte	Value
CLA	00h
INS	0Eh
P1	See table below
P2	See table below
Lc	Empty
Data	Empty
Le	Empty

Table 36. ERASE BINARY command APDU

Table 37. ERASE BINARY: coding of P1 and P2.

	Coding of P1 and P2									
b8	b8 b7 b6 b5 b4 b3 b2 B1 Hex Meaning									
0	-	-	-	-	-	-	-	-	P1-P2 specifies a 15-bit offset of the data to be read	
1	-	-	-	-	-	-	-	-	P1 specifies a short FID and P2 specifies an 8-bit offset of the data to be read	
1	0	0	х	х	x	х	x	-	- short FID (value domain $1 - 30$)	

Table 38. ERASE BINARY response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.15 GET DATA

The GET DATA command is used to retrieving a public key part of a RSA key pair. The file from which the key information is being retrieved must have been selected using the SELECT FILE command.

Table 39.	GET DATA	command APDU
-----------	----------	--------------

Byte	Value
CLA	00h
INS	CAh
P1	01h
P2	See table below
Lc	Empty
Data	Empty
Le	Number of bytes expected in response

	Coding of P2									
b8	b7	b6	b5	B 4	b3	b2	b1	Hex	Meaning	
0	0	0	0	0	0	0	1	'00'	Key info: algorithm identifier, length of modulus and length of public exponent	
0	0	0	0	0	0	1	0	'01'	Modulus	
0	0	0	0	0	0	1	1	'02'	Public exponent	
	Any other value -							-	RFU	

Table 41. GET DATA response APDU

Byte	Value				
Data	See table below				
SW1-SW2	V2 Status bytes				

Table 42. GET DATA: response APDU Data field

	GET DATA Response APDU Data field									
Value of P1	alue of P1 Value of P2 Data field coding									
01h	00h	Value	Length							
		algorithm identifier ('92 00' (RSA CRT) is the only currently supported value)	2							
		bit length of modulus	2							
		bit length of public exponent	2							
01h	01h	Value	Length							
		bit length of modulus	2							
		Modulus	var							
01h	02h	Value	Length							
		bit length of public exponent	2							
		Public exponent	var							

5 Implementation guidelines for software developers

5.1 Resource management

The FINEID card will be used by multiple host applications running simultaneously in the same PC. Because the FINEID card is internally a simple state machine, these host applications share the state of the FINEID card also. This sets some fundamental requirements for the host applications accessing the shared resource (i.e. the FINEID card and reader device):

- 1. Host applications must protect the command sequences they send to the FINEID card by locking the card exclusively to themselves (and blocking access from others) while doing these transactions.
- 2. The length of each transaction should be minimized.
- 3. Host applications should not assume that the state of the card (e.g. currently selected application) stays unmodified between transactions. The only exception to that rule is that the verification status of a successfully verified global PIN <u>should</u> be unaffected between transactions. Check ISO/IEC 7816-15 for additional information on global PINs.

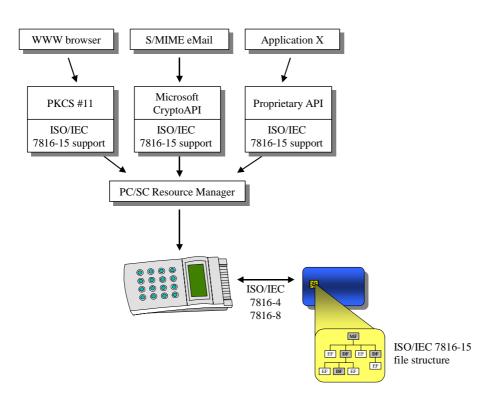


Figure 1. Example scenario of multiple host applications - single card

5.2 Resetting the card

Unnecessary resetting of the card should be avoided. When using PC/SC interface the card is resetted automatically by the Resource Manager so there is no need for the host application to explicitly reset the card before starting to use it.

15.3.2004 17(29)

5.3 Application/File selection

5.3.1 CIA application

CIA (Cryptographic Information Application) application is selected using following Application Identifier (AID):

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

Selection by Application identifier:

Command	CLA	INS	P1	P2	Lc	Data	Le
SELECT	00	A4	04	00	0C	A0 00 00 00 63 50 4B 43 53 2D 31 35	-

5.3.2 Path

CIA uses Path ASN.1 structure to reference various files. The Path.efidOrPath octet string contains:

- a file identifier if the length of the octet string is two bytes
- an absolute path if the octet string is longer that two bytes and starts with the file identifier of MF = 3F 00
- a relative path if the octet string is longer than two bytes and starts with the file identifier of the DF (which is not 3F 00)

5.4 Authentication objects

In CIA all objects (private keys, certificates etc.) can be protected with authentication objects (i.e. PINs). Each object may contain a pointer to an authentication object e.g. a private key object may contain a pointer to a PIN object. This means that the private key operation (decrypt or sign) can be done only after successful verification of the PIN code.

The following table lists the operations that can be protected with authentication objects in the CIA sense.

Object type	Operations protected with the authentication object						
Private key	Private key operations						
	- sign (PSO: COMPUTE DIGITAL SIGNATURE)						
	- decrypt (PSO: DECIPHER)						
Public key	Public key operations (not supported in FINEID context)						
	- verify (PSO: VERIFY DIGITAL SIGNATURE)						
	- encrypt (PSO: ENCIPHER)						
Secret key	Secret key operations (not supported in FINEID context)						
	- encrypt						
	- decrypt						
Certificate	Reading the contents of the certificate						
Data object	Reading the contents of data the object						
Authentication object	The authentication object can be used to unblock this authentication object (e.g. unblocking PIN is used).						

Table 43. Objects and protected operations

5.4.1 Accessing objects

The flowchart below describes one possible solution for accessing objects and fulfilling the authentication requirements (PIN verifications) of these objects.

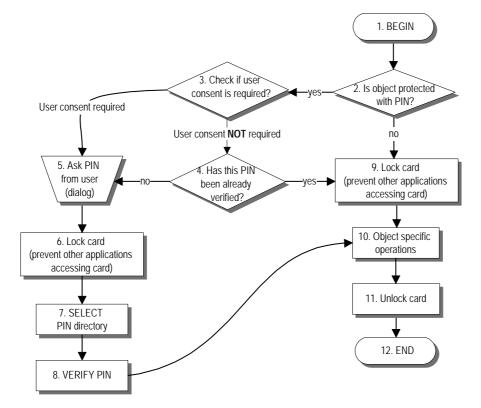


Figure 2. Example of PIN logic

The command sequence in PIN verification consists of two commands described below.

Select PIN directory specified in PassWordAttributes.path (example path OCTET STRING = 3F 00 50 16):

Command	CLA	INS	P1	P2	Lc	Data	Le
SELECT FILE	00	A4	08	00	02	50 16	-
						(MF file identifier 3F 00 removed)	

Verify PIN. Padding is done according to PassWordAttributes (storedLength, padChar). The P2 value is taken from PassWordAttributes.pwdReference (example value 82):

Command	CLA	INS	P1	P2	Lc	Data	Le
VERIFY	00	20	00	82	08	31 32 33 34 35 36 00 00 (PIN = 123456 in ASCII with 00 padding)	-

The verification status of a PIN may be dropped automatically to state 'not verified' by the card operating system after performing e.g. a private key operation. This is indicated by the **userConsent** element of the private key object. E.g. **userConsent** value set to one for a private key object indicates that the card holder must manually enter the PIN for each private key operation. Requiring user interaction for all operations done with a specific private key is a trade-off between usability and security. It is anticipated that this feature will be used for performing legally binding non-repudiable digital signatures only.

The object specific operations in step 10 include the ones listed in the Table 43.

FINEID SPECIFICATION	15.3.2004
FINEID - S1 / v2.1	19(29)

5.4.2 Auth required flag

In addition to the 'on demand' access control of objects in CIA it is also possible to protect some of the object directory files. The EF.CIAInfo contains a CardFlags.authRequired flag indicating that the first authentication object in the AOD is used to protect other object directory files than OD and AOD (ISO/IEC 7816-15 chapter 8.10 and annex B).

5.5 Private key operations (sign and decrypt)

There may be multiple private keys in the same CIA application. The host application must first determine which one of these private keys to use. This can be done e.g. based on the information inside card holder certificates according to application specific criteria (e.g. key usage bits and CA policy OIDs). Each certificate contains a pointer to the corresponding private key object.

Private keys are accessed like any other objects according to Figure 2. The command sequence of step 10 of that flowchart is described below.

5.5.1 Signature operation

It is assumed that PIN verification is already done and current DF is the DF, where the RSA key file containing the RSA key to be used is located.

Restore the empty SE:

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE:	00	22	F3	00	-	-	-
RESTORE							

Set the following properties into the SE Digital Signature Template:

- algorithm reference (= 12 i.e. RSASSA-PKCS1-v1_5 signature with SHA-1, card does padding and DigestInfo encapsulating of the hash)
- key file path (= 4B 02 derived from PrivateRSAKeyAttributes.value path)

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE: SET	00	22	41 computation	B6 DST in data	07	80 01 12 (algorithm reference = 12)	-
			Ĩ	field		81 02 4b 02 (private key file identifier)	

Sign the hash calculated by the host application:

Command	CLA	INS	P1	P2	Lc	Data	Le
PSO:	00	2A	9E	9A	14	4B 52 16 5B 4A B6 54 C3 E5 4F	XX
COMPUTE						64 B5 F1 EE A6 45 D4 6B 65 C8	
DIGITAL							
SIGNATURE							

XX is the maximum length of the digital signature returned in response.

Get the response in T=0 protocol:

15.3.2004 20(29)

Command	CLA	INS	P1	P2	Lc	Data	Le
GET RESPONSE	00	C0	00	00	-	-	XX

5.5.2 Decryption operation

It is assumed that PIN verification is already done and current DF is the DF, where the RSA key file containing the RSA key to be used is located.

Restore the empty SE:

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE:	00	22	F3	00	-	-	-
RESTORE							

Set the following properties into the SE Confidentiality Template:

- algorithm reference (= 02 i.e. RSAES-PKCS1-v1_5 decryption, card removes padding)

- key file path (= 4B 01 derived from PrivateRSAKeyAttributes.value path)

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE: SET	00	22	41 decryption	B8 CT in data field	07	80 01 02 (algorithm reference = 02) 81 02 4B 01 (private key file identifier)	-

Decrypt the modulus size (example 1024 bits) cryptogram:

Command	CLA	INS	P1	P2	Lc	Data	Le
PSO: DECIPHER	00	2A	80	86	81	00 (padding indicator byte)	XX
DECIFIER						4B 52 16 54 C3 E5	
						(cryptogram)	

XX is the maximum length of the decrypted cryptogram. PKCS#1 v1.5 padding is removed by the card when using the algorithm 02.

Get the response in T=0 protocol:

Command	CLA	INS	P1	P2	Lc	Data	Le
GET	00	C0	00	00	-	-	XX
RESPONSE							

Annex A (Informative): Coding of the File Control Information template

The FCI template returned by SELECT and SELECT FILE commands is a TLV (Tag-Length-Value) coded data structure.

Coding of the FCI template:

Tag	Length	Value
6Fh	XXh	File Control Parameter Data Objects (FCP DOs)

Coding of the FCP DOs:

	File Control Parameter DOs									
Tag	L	Value	Applies to							
81h	2	Number of data bytes in the file, including structural information if any	Any file							
82h	1	File descriptor byte	Any file							
83h	2	File identifier (FID)	Any file							
84h	1 – 16	DF name	DFs							
86h	var	Security attributes, proprietary coding	Any file							
8Ah	1	Life Cycle Status Integer (LCSI)	Any file							
A5h	var	Proprietary information (constructed)	DFs							

File size:

The file size (tag 81h) codes the number of bytes to be reserved for the data:

transparent EF:	amount of data
PIN file:	Not applicable, use value '0000'. PIN files reserves always the fixed size of memory.
RSA key file:	No applicable, use value '0000'. RSA key file reserves always the fixed size of memory.
DF:	Number of sub-files.

File descriptor byte:

The available file descriptor (tag 82h) values are:

	Available file descriptor values												
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning				
0	0	0	0	0	0	0	1	01h	Transparent (binary) EF				
0	0	0	0	1	0	1	0	0Ah	PIN file				
0	0	0	1	0	0	0	1	11h	RSA key file				
0	0	1	1	1	0	0	0	38h	DF				

Table 44. File descriptor values

File identifier:

The file identifier (tag 83h) codes the identity of EFs and DFs within the current DF.

DF name:

The DF name (tag 84h) codes the Application Identifier (AID) for DFs.

Security attributes:

The security attributes (tag 86h) codes the AC information for the file or directory. Interpretation of the AC information is file type dependent. The structure and coding of the AC are defined in the end of this annex.

Life cycle status integer:

The life cycle status integer (LCSI) (tag 8Ah) codes the status of the file.

Coding of LCSI is as follows:

	Coding of the Life Cycle Status Integer (LCSI)													
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning					
х	х	х	х	-	-	-	-	-	RFU (currently no effect)					
-	-	-	-	х	х	х	х	-	life cycle status information					
-	-	-	-	0	0	0	1	'X1'	creation state					
-	-	-	-	0	1	1	1	'X7'	operational state – activated					
-	-	-	-	other values			es	-	RFU					

Table 45. LCSI coding

PIN definitions:

The proprietary constructed information field (tag A5h) codes the PIN referencing. This gives the possibility to locally refer to PINs actually stored in the parent level. This field is optional if PINs are referenced locally.

It contain one DO, PIN definition (tag C1h). The coding is described below.

Tag = C1h

Length = XXh, where XX is between 00 and 02

Value = 'ZZh ZZh ... ZZh', where each ZZ is coded as follows

	Coding of PIN definition byte												
b8	b7	b6	b5	b4	b3	b2	B1	Hex	Meaning				
0	-	-	-	-	-	-	-	-	 Reference to parent DF PIN definition 				
1	-	-	-	-	-	-	-	-	- Reference to this local DF PIN definition				
-	-	х	х	х	х	х	х	-	RFU				

Table 46. PIN definition byte coding

The first value field is for PIN 1, second for PIN 2.

Access conditions

The main principles of the AC coding are the following:

- There is a proprietary TLV-like coding inside the value field of DO 86h in file header, expressing a list of AC definitions.
- There is a distinction between simple coding and adaptive coding. In simple coding most of the commands are covered and can be coded efficiently. In adaptive coding it is possible to express more complex specifications for commands and also to express AC for additional commands. The adaptive coding is used to differentiate the access condition of different modes of PERFORM SECURITY OPERATION (PSO) command
- Non-existent AC is interpreted as denied access. If conditions for a command is not specified in the ACI, the command is forbidden (unless the command is of type which does not require AC check, e.g. PIN verification or file selection). Therefore NEV conditions need not be explicitly expressed.
- A bitmap expresses a set of commands to which an accompanied condition applies. Hence several commands that have equal conditions can be expressed in one AC. This is called the simple coding.
- A byte string (called definition list) expresses the set of commands or commandparameter combinations to which an accompanied condition applies. Hence AC for any additional commands, possibly complex parameter-dependent variants also, can be expressed. This is called the adaptive coding.

The access control interpretation in simple coding format is different for DFs and EFs. There are also differences between types of EF. This means that the bitmap is used to refer to different set of commands depending on the type of file.

The first byte in an AC description (one DO of the proprietary TLV-like coded list) codes what fields are included, the operational mode and the length of the DO value. In context of TLV-coding it thus codes the (proprietary) T-L part and is hence called the PTL byte.

				C	Codi	ing	of tł	ne prop	rietary tag-length (PTL) byte of AC DO
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning
х	-	-	-	-	-	-	-	-	Coding switch
0	-	-	-	-	-	-	-	-	- simple coding
1	-	-	-	-	-	-	-	-	- adaptive coding
-	х	-	-	-	-	-	-	-	Life cycle context indicator
-	0	-	-	-	-	-	-	-	- the AC DO applies only in operational state
-	1	-	-	-	-	-	-	-	– RFU
-	-	х	-	-	-	-	-	-	RFU
-	-	-	х	-	-	-	-	-	RFU
-	-	-	-	х	х	х	х	-	Length of the entry

Table 47. PTL byte coding

After the PTL byte there comes one or more AM bytes, coding the access mode, i.e. the commands which the conditions are valid.

Simple coding

If the simple coding is used, there is always one AM byte. The coding of this byte is defined in tables from Table 48 to Table 51.

	Coding of the AM command bitmap for DF (simple coding)												
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning				
х	-	-	-	-	-	-	-	-	RFU				
-	х	-	-	-	-	-	-	-	RFU				
-	-	х	-	-	-	-	-	-	RFU				
-	-	-	х	-	-	-	-	-	RFU				
-	-	-	-	х	-	-	-	-	RFU				
-	-	-	-	-	1	-	-	-	CREATE FILE (DF creation)				
-	-	-	-	-	-	1	-	-	CREATE FILE (EF creation)				
-	-	-	-	-	-	-	1	-	DELETE FILE (child)				

Table 48. AM command bitmap for DF (simple coding)

			С	odi	ng c	of th	ne A	M com	nand bitmap for binary EF (simple coding)
b8	b7	B 6	b5	b4	b3	b2	b1	Hex	Meaning
х	-	-	-	-	-	-	-	-	RFU
-	1	-	-	-	-	-	-	-	DELETE FILE
-	-	х	-	-	-	-	-	-	RFU
-	-	-	х	-	-	-	-	-	RFU
-	-	-	-	х	-	-	-	-	RFU
-	-	-	-	-	х	-	-	-	RFU
-	-	-	-	-	-	1	-	-	UPDATE BINARY, ERASE BINARY
-	-	-	-	-	-	-	1	-	READ BINARY

Table 49. AM command bitmap for binary EF (simple coding)

				Coc	ding	of	the	AM con	nmand bitmap for PIN EF (simple coding)
b8	b7	B 6	B5	b4	b3	b2	b1	Hex	Meaning
x	-	-	-	-	-	-	-	-	RFU
-	1	-	-	-	-	-	-	-	DELETE FILE
-	-	Х	-	-	-	-	-	-	RFU
-	-	-	х	-	-	-	-	-	RFU
-	-	-	-	х	-	-	-	-	RFU
-	-	-	-	-	х	-	-	-	RFU
-	-	-	-	-	-	1	-	-	PUT DATA
-	-	-	-	-	-	-	х	-	RFU

Table 50. AM command bitmap for PIN EF (simple coding)

			(Cod	ing	of t	he /	AM com	mand bitmap for RSA EF (simple coding)
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning
х	-	-	-	-	-	-	-	-	RFU
-	1	-	-	-	-	-	-	-	DELETE FILE
-	-	1	-	-	-	-	-	-	PUT DATA (RSA key)
-	-	-	х	-	-	-	-	-	RFU
-	-	-	-	х	-	-	-	-	RFU
-	-	-	-	-	1	-	-	-	GENERATE PUBLIC KEY PAIR
-	-	-	-	-	-	1	-	-	ERASE BINARY
-	-	-	-	-	-	-	1	-	GET DATA (public part of the key)

Table 51. AM command bitmap for RSA EF (simple coding)

Adaptive coding

If the adaptive coding is used, there are always more than one AM byte. The coding of the first byte – defining what elements are present in the AM list – is defined in Table 52. Bits b7-b4 define the amount of bytes in one command description, while bits b2-b1 define the number of command descriptions in the definition list. The number of AM bytes following the first AM byte is then the product of these values, i.e. (length of command description)*(number of command descriptions).

It should be noted that in adaptive coding the INS byte is always present in the command description of the AM entry and is therefore not identified in the AM byte.

			Co	ding	of	the	firs	t byte o	f Access Mode (AM) of AC (adaptive coding)
b8	b7	b6	b5	b4	B3	b2	b1	Hex	Meaning
х	-	-	-	-	-	-	-	-	RFU
-	х	х	х	х	-	-	-	-	Definition list bitmap for adaptive coding
-	х	-	-	-	-	-	-	-	RFU present in command descriptions of definition list
-	-	1	-	-	-	-	-	-	P1 present in command descriptions of definition list
-	-	-	1	-	-	-	-	-	P2 present in command descriptions of definition list
-	-	-	-	х	-	-	-	-	RFU
-	-	-	-	-	Х	-	-	-	RFU
-	-	-	-	-	-	х	х	-	Number of command descriptions in the definition list $(1-2)^{1}$

Table 52. First AM byte coding (adaptive coding)

¹ Value 00 is not practical as it includes no command descriptions and such condition can never allow any command(s) to be executed.

FINEID SPECIFICATION	15.3.2004
FINEID - S1 / v2.1	27(29)

SC byte

The AM byte(s) is/are followed by max one SC byte. There can be no SC bytes (ALWAYS condition) or one SC byte.

	Coding of the PIN related Security Condition (SC) of AC DO												
b8	b7	b6	b5	b4	B3	b2	b1	Hex	Meaning				
x	-	-	-	-	-	-	-	-	Reference context				
0	-	-	-	-	-	-	-	-	– RFU				
1	-	-	-	-	-	-	-	-	- local PIN reference				
-	х	Х	х	х	-	-	-	-	RFU				
-	-	-	-	-	х	х	х	-	PIN number (1-2) ²				

Table 53. PIN related SC coding

 $^{^{2}}$ Value 000 is not practical as it refers to non-existing PIN and such condition can never be fulfilled.

Annex B (Informative): Status conditions

Success conditions

Success conditions			
'61xx'	RES_MORE	OK, 'xx' data bytes available for GET RESPONSE	
'9000'	RES_OK	ОК	

Warning conditions

Warning conditions			
'6200'	RES_AUTH_WARN	Authentication failed	
'6281'	RES_CORRUPT_WARN	Part of the returned data may be corrupted	
'6282'	RES_EOF_WARN	End of file reached before reading L_{e} bytes	
'6283'	RES_INVALID_WARN	Selected file invalidated	
'6284'	RES_FCI_WARN	FCI not formatted according to ISO/IEC 7816-4, subclause 5.1.5	
'6300'	RES_GEN_WARN	No information given (verification failed)	
'63Cx'	RES_VERIF_WARN	Counter (verification failed; x indicates the number of further allowed retries)	

Error conditions

Error conditions			
'6400'	RES_EXEC_ERR	Execution error	
'6581'	RES_MEM_ERR	Memory failure (unsuccessful writing)	
'6600'	RES_SE_ERR	The security environment cannot be set or modified, no further information	
ʻ6700'	RES_LEN_ERR	Wrong length	
'6882'	RES_CMD_CLA_ERR	SM not supported for the command	
'6981'	RES_FILE_ERR	Command incompatible with file structure	
'6982'	RES_AC_ERR	Security status not satisfied	
'6983'	RES_BLOCKED_ERR	Authentication method blocked	
ʻ6984'	RES_REF_INVALID_ERR	Referenced data invalidated	
'6985'	RES_COND_ERR	Conditions of use not satisfied	

RES_NOT_EF_ERR	Command not allowed (no current EF)
RES_NO_SM_ERR	Expected SM data objects missing
RES_SM_ERR	SM data objects incorrect
RES_FUNC_ERR	Function not supported
RES_NO_FILE_ERR	File not found
RES_NO_RECORD_ERR	Record not found
RES_FULL_ERR	Not enough space in the file
RES_PAR_ERR	Incorrect parameters P1-P2
RES_PAR_INCONS_ERR	L _c inconsistent with P1-P2
RES_NO_DATA_ERR	Referenced data not found
RES_EF_EXISTS_ERR	File already exists
RES_DF_EXISTS_ERR	DF name already exists
RES_PAR_ADDR_ERR	Wrong parameters (offset outside the EF)
RES_REDO_ERR	Wrong length (wrong L_e field; 'xx' indicates the exact length)
RES_INS_ERR	Incorrect INS byte
RES_CLA_ERR	Incorrect command class (CLA)
RES_GEN_ERR	No precise diagnosis is given
	RES_NO_SM_ERR RES_SM_ERR RES_FUNC_ERR RES_FUNC_ERR RES_NO_FILE_ERR RES_NO_RECORD_ERR RES_FULL_ERR RES_PAR_ERR RES_PAR_ERR RES_PAR_INCONS_ERR RES_DF_EXISTS_ERR RES_DF_EXISTS_ERR RES_DF_EXISTS_ERR RES_PAR_ADDR_ERR RES_REDO_ERR RES_INS_ERR RES_INS_ERR

[©] Väestörekisterikeskus 2005