FINEID - S1
Electronic ID Application

v3.0

Population Register Centre (VRK)
Certification Authority Services
P.O. Box 123
FIN-00531 Helsinki
Finland
http://www.fineid.fi

Authors

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

Document history

Version	Date	Editor	Changes	Status
0.9	29.06.2016	MaSi		Initial version.
1.00	22.11.2016	AP	Document accepted.	Published version.
1.01	24.01.2018	MaSi	Following corrections: • The content of PIN attributes return by GET DATA command corrected (paragraph 3.14.3 Response). • Unblocking PIN method byte coding corrected (paragraph 3.14.3 Response).	
1.02	25.01.2018	AP	Document accepted.	Published version.

Table of contents

1. Introduction

This document describes the command interface of the Finnish Electronic Identification (FINEID) application version 3.0.

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: Inter-industry data elements, ISO/IEC 7816-6
  Part 8: Security related interindustry commands, ISO/IEC 7816-8
  Part 15: Cryptographic information application, ISO/IEC7816-15
- Global Platform, Card Specification, version 2.1.1.
- Technical Guideline TR-03110 (BSI) - Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control (EAC). V2.10 Part 1 and 3 — 20 March 2012.

1.2 Informative references

The following documents have also influenced this specification:

- PKCS#1 v2.2, RSA Cryptography Standard, October 27, 2012.
- FINEID – S1, Electronic ID application for Finnish Electronic ID card, v2.1.

1.3 Related FINEID documentation

FINEID documentation is available from:

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

2 Abbreviations

AC	Access Condition
AID	Application IDentifier
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
CIA	Cryptographic Information Application
CLA	Class byte
CT	Confidentiality Template
CRDO	Control Reference Data Object
CRT	Chinese Remainder Theorem
DF	Dedicated File
DO	Data Object
DST	Digital Signature Template
EF	Elementary File
ELC	ELliptic Curve
FCI	File Control Information
FCP	File Control Parameter
FINEID	Finnish Electronic Identification
Licc	Length of data available on the card
MF	Master File
MSE	Manage Security Environment
PACE	Password Authenticated Connection Establishment
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 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.

Table 1. EID application related commands

Command	Standard	Functionality
SELECT	Global Platform, Card Specification, version 2.1.1.	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
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: **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

3.1 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 0Ch - secure messaging
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.

3.2 SELECT FILE

The SELECT FILE command selects a DF or an EF.

If P1, P2 and Lc are all zero, the command selects the root but returns no data. To select the root and return the FCI information, you must select it by its file ID (3F00h).

Note: Some card implementations may process the command (that is, 0x A4 04 02 00) at the card operating system level before sending it to the FINEID application. Consequently non nominal cases can result in an unexpected status word. For all the application nominal cases, the behaviour is as described here.

3.2.1 Conditions of Use and Security

3.2.1.1 Usage

The MF and DFs can be selected by their file identifier, by their path or by their name, but not by partial name. EFs can be selected by their file identifier or by their path.

The command cannot be used to select a deactivated file.

3.2.1.2 Security

If the EF to be selected is protected by a security attribute, then this security attribute must be fulfilled in order to perform the SELECT FILE command.

The security attribute specifies if prior mutual authentication is necessary, and if so whether or not secure messaging is also necessary.

Note: For clarity, the command is described here with the data in plain text.

3.2.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	A4h
P1	00h - select EF, DF or MF by file identifier 02h - select EF by file identifier under current DF 04h - select DF or MF by name 08h - select file by absolute path from MF
P2	00h - FCI returned in response 04h - FCP returned in response 0Ch - no response
Lc	Empty or length of subsequent data field
Data	P1 = 00h - EF, DF or MF file identifier (or empty = MF) P1 = 02h - EF file identifier P1 = 04h - MF or DF name P1 = 08h - absolute path from MF without the identifier of MF (3F00h)
Le	Empty or maximum length of data expected in response

Note: When P1 is 00h, (file selection by file ID), but not under the current DF, all the files in the application domain will be searched in the order of creation. It selects the first file that matches the file ID specified in the data field, which might not be the file intended.

3.2.3 Response

Byte	Value
Data	File Control Information (FCI), File Control Parameters or empty
SW1-SW2	Status bytes

The card returns the data requested by the P2 reference control parameter, followed by the SW1, SW2 status codes. The response data is either in the FCI template, the FCP template or does not exist (P2 = 0Ch).

The FCP template contains the file control parameters that were specified when creating the file. The content of FCP is described in Annex A.

The FCI information returned by the command (P2 = 00h) is shown in Annex A.

Caution: When the command is sent with secure messaging, the tag (62h for FCP, 6Fh for FCI) and length of the FCP or FCI template are absent. Two trailer bytes with the value 00h are appended to the end of the FCP/FCI data. This is true for DF and EF selection.

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error. No FCI data returned
62h	83h	Deactivated file
64h	00h	Execution error, file descriptor checksum error.
67h	00h	Incorrect length, Lc.
69h	82h	Security status not satisfied, error during secure messaging
69h	99h	⁴ Select Application Failed ⁴ PACE authentication not open

6Ah	81h	Function not supported (cannot select MF by path)
6Ah	82h	File not found
6Ah	86h	Incorrect P1, P2 parameter

3.3 GET RESPONSE

The GET RESPONSE command returns response data from the card for case 4 APDU commands.

For example, this command is used in to get response data from commands

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

3.3.1 Conditions of Use and Security

3.3.1.1 Usage

The GET RESPONSE command must be used only after command case 4.

3.3.1.2 Security

The GET RESPONSE command can be accessed freely.

3.3.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	C0h
P1	00h
P2	00h
Lc	Empty
Data	Empty
Le	Maximum length of data expected in response

3.3.3 Response

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

Most status byte values for this command are managed by the card platform. The FINEID application can return only the following codes:

SW1	SW2	Description
68h	84h	The chaining mechanism is not available for the command.
90h	00h	Successful execution of the command (Le = Licc)

Once taken over, the card platform can also return the following codes:

SW1	SW2	Description
61h	XXh	Le < Licc, XXh is the remaining number of available data bytes in the card
69h	82h	Security status not satisfied
6Ch	Licc	if Le > Licc

3.4 READ BINARY

The READ BINARY command reads all or part of a transparent EF.

There are two ways of referencing the EF:

- Implicit selection. In this case, the EF is under the current DF, and is specified by its short file identifier (SFI). P1 specifies the SFI and P2 specifies the start address for the read (called the offset). The command cannot be used for files with an SFI of 1Fh.
- Direct selection. In this case the EF must have been previously selected using the SELECT FILE command. The data is read directly from the current EF. In this case the offset is specified over P1 and P2.

The command reads the number of bytes specified in Le from the offset, up to a maximum of 255 bytes (including secure messaging data if used).

3.4.1 Conditions of Use and Security

3.4.1.1 Usage

The command can be used on any activated EF.

In the case of implicit selection, the EF becomes the current EF after execution of the command.

3.4.1.2 Security

If the EF containing the data to be read is protected by a security attribute, then this security attribute must be fulfilled in order to perform the READ BINARY command.

The security attribute specifies if prior mutual authentication is necessary, and if so whether or not secure messaging is also necessary.

Note: For clarity, the command is described here with the data in plain text.

3.4.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	B0h
P1	See table below
P2	See table below
Lc	Empty
Data	Empty
Le	Number of bytes to read. If empty or zero, the command reads until the end of the file, up to a maximum of 255 bytes (including data for secure messaging if SM specified).

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	x	x	x	x	x	-	– short FID (value domain 1 – 30)

3.4.3 Response

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

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
62h	82h	End of file reached (offset + Le is higher than the end of the file)
64h	00h	Execution error, file descriptor checksum error.
69h	82h	Security status not satisfied, for example: - The security attributes are not satisfied - Error during secure messaging
69h	85h	Conditions of use not satisfied, for example: Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
69h	86h	Command not allowed (no current EF)
6Ah	82h	File not found (b8 of P1 = 1 indicating implicit selection, but no SFI given)
6Ah	86h	Incorrect P1, P2 (check that the offset is inside the EF and that the SFI is not 1Fh)
6Bh	00h	Incorrect P1, P2 (offset outside EF)

3.5 VERIFY

The VERIFY command authenticates a user by comparing the PIN entered in the command (the verification PIN) with the reference PIN.

If the VERIFY command is successful, the following actions take place:

- The reference PIN Try Counter is set to PIN Try Limit.
- The reference PIN validated flag is set to true.

- The reference PIN usage counter is decremented by 1 unless its value is FFh (no limit to the number of times the PIN can be used) or 00h (already been used maximum allowed number of times).
- The reference PINs Credentials Counter can be used.

If the Verify command fails, the following actions take place:

- The reference PIN Try Counter is decremented by 1. If the reference PIN Try Counter reaches zero as a result, the command returns 6983h.
- The reference PIN validated flag is set to false.
- The reference PIN usage counter is unchanged.

In "Unverify" Mode:

If the command is used to "unverify" a PIN, a successful execution of the command sets the reference PIN to "unverified".

3.5.1 Conditions of Use and Security

3.5.1.1 Usage

The length of the entered PIN must be equal to length of the PIN in the card, otherwise the VERIFY command fails.

Sending the command with Lc = 00h has a special meaning. It enables you to know if the PIN has already been successfully presented during the current session. If the PIN has been successfully presented already, then the VERIFY command returns a status code of 9000h. If the PIN has not been successfully presented already, the VERIFY command returns a status code of 63 Cxh, where x is the number of remaining presentation attempts allowed.

3.5.1.2 Security

The PIN Try Counter and the Usage Counter for the reference PIN must not be zero.

The use of secure messaging is determined by the security attribute of the PIN. If secure messaging is specified, the verification PIN provided by the terminal must be of the same length as the reference PIN and its value must match.

Note: For clarity, the command is described with the data in plain text.

3.5.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	20h
P1	00h – Verify mode FFh – Unverify mode
P2	11h – global PIN 81h-8Fh - local PIN
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

3.5.3 Response

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

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
63h	Cxh	Reference PIN not verified.”x” attempts remaining
67h	00h	Incorrect length, Lc.
69h	82h	Security conditions not satisfied (error during secure messaging)
69h	83h	Authentication method blocked (reference PIN blocked)
69h	84h	PIN Try Counter or PIN Usage Counter for reference PIN has reached zero
69h	85h	Conditions of use not satisfied, for example: ④ Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded. (Possible in “Verify” mode but not “Unverify” mode).

69h	99h	PACE authentication not open
6Ah	86h	Incorrect P1, P2
6Ah	88h	Referenced data not found

3.6 MANAGE SECURITY ENVIRONMENT: SET

The MANAGE SECURITY ENVIRONMENT: SET command updates a CRT in the current SE. The value of the CRT in the original SE remains unchanged.

3.6.1 Conditions of Use and Security

3.6.1.1 Usage

The command applies to the current SE only.

3.6.1.2 Security

The MANAGE SECURITY ENVIRONMENT: SET command can be accessed freely.

Note: For clarity, the command is described here with the data in plain text.

3.6.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging 10h - transmit first portion of data in chaining mode with no secure messaging and wait for final portion
INS	22h
P1	01000001b = 41h – computation and decipherment
P2	P1 = SET - P2 = AAh hash template (HT) - 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. CRDOs are the items contained in the CRT. The data does not contain the tag and length of the CRT.
Le	Empty

3.6.3 Response

Byte	Value
Data	Empty
SW1-SW2	Status bytes

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
67h	00h	Incorrect length. Lc > 30
69h	82h	Security status not satisfied, error during secure messaging
69h	85h	Conditions of use not satisfied, for example wrong order of operations or context of use not respected.
6Ah	80h	Incorrect data, for example no algorithm ID referenced
6Ah	86h	Incorrect P1, P2

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

Table 4. Control Reference Data Objects (CRDO)

Tag	Value	DST	CT
80h	Algorithm reference	+	+
84h	Key reference	+	+

Table 5. MANAGE SECURITY ENVIRONMENT:SET supported P1-P2 combinations

Supported combinations of P1-P2				
P1	**P2**	**Meaning**	**CRDO in data field**	**Data field contents**
'41'	'AA'	SET SE for hashing	HT	' 80 01 xx '
'41'	'B6'	SET SE for digital signature	DST	' 80 01 xx 84 01 xx '
'41'	'B8'	SET SE for decipherment	CT	' 80 01 xx 84 01 xx ‘

The supported values for the CRDO algorithm reference (tag 80h) are specified in the tables below.

Table 6. Values for the algorithm reference used in Digital Signature Template

Algorithm reference	Details
0Xh	No hash indicated. Either the hash function is defined implicitly or is not applicable.
1Xh	SHA-1
3Xh	SHA-224
4Xh	SHA–256
5Xh	SHA-384
6Xh	SHA-512
X1h	RSA with padding according to ISO 9796–2.
X2h	**RSASSA-PKCS1-v1_5 signature scheme** (according to PKCS#1 v2.2 with RSA algorithm, compatible with PKCS#1 v1.5)
X3h	RSA with padding according to RFC 2409.
X4h	ECDSA.
X5h	RSA PSS.
XXh	All other values are RFU.

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 7. Values for the algorithm reference used in Confidentiality Template

Algorithm reference	Details
1Ah	RSASSA PKCS#1_v1.5
1Dh	RSAES OAEP SHA-1 (160 bits)
3Dh	RSAES OAEP SHA-224
4Dh	RSAES OAEP SHA–256
5Dh	RSAES OAEP SHA-384
6Dh	RSAES OAEP SHA-512

3.7 PERFORM SECURITY OPERATION: HASH

The PERFORM SECURITY OPERATION: HASH provides the hashed message to be used as input for the computation of a digital signature. There are three different cases:

- The hash is performed entirely by card
- The hash is performed externally but card performs the final round of hashing

- The hash is performed entirely outside the card and the PERFORM SECURITY OPERATION: HASH command is used to “set” the data in preparation for the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command

For information, the following table summarizes the various hash algorithm constants:

Table 8. Hash Algorithm Constants

Hash	Length of intermediate hash	Length of the counter indicating the number of bits already hashed (bytes) 1	Block length of hash algorithm	Length of message digest result
SHA-1	20 bytes	8 bytes	64 bytes	20 bytes
SHA-224	32 bytes	8 bytes	64 bytes	28 bytes
SHA-256	32 bytes	8 bytes	64 bytes	32 bytes
SHA-384	64 bytes	16 bytes	128 bytes	48 bytes
SHA-512	64 bytes	16 bytes	128 bytes	64 bytes

For the sake of simplicity, the counters indicating the number of bits already hashed can be coded on 8 or 16 bytes for all supported hash algorithms. However, when coded on 16 bytes, the 8 MSB (leftmost) bytes must be null. Thus a length coded on 8 or 16 bytes is equivalent.

3.7.1 Conditions of Use and Security

3.7.1.1 Usage

If the data is to be hashed entirely in the card and is more than 64 bytes, divide it into blocks of 64 bytes or less and perform the PERFORM SECURITY OPERATION: HASH command for each block. The final command has a different format (see “APDU Format to Hash the Final Block”). If the data is 64 bytes or less, use the format described for the final command. If a command other than PERFORM SECURITY OPERATION: HASH is sent, the hash session ends and previously hashed data is lost.

The result of the PERFORM SECURITY OPERATION: HASH is available until one of the following occurs:

- The FINEID application is deselected or reselected
- A further PERFORM SECURITY OPERATION: HASH command is issued
- A MANAGE SECURITY ENVIRONMENT: SET command is issued

3.7.1.2 Security

The current SE must contain a DST template or an HT template and the algorithm ID in this template must be consistent with the format of the command data.

If a MANAGE SECURITY ENVIRONMENT: SET command is sent during a hash sequence which references a different DST key, the hash sequence is aborted and the internal "hash integrity flag" is set to "unverified".

Note: For clarity, the command is described here with the data in plain text.

3.7.2 Format

This command format varies according to where the hash is performed. Each case is shown separately.

3.7.2.1 Hashing Performed Entirely by the Card

If the data is more than one block length, the command must be sent for each block. The block length is 64 bytes or 128 bytes according to the SHA used (see the "block length" column in Hash Algorithm Constants table).

The APDU format for all the blocks except the last one is as follows:

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ah
P1	90h
P2	80h
Lc	Length of subsequent data field
Data	Data to be hashed (no more than one block length as shown in "block length" column in Hash Algorithm Constants table
Le	Empty.

The APDU format for the last block or only block if the data is not more than one block length is as follows:

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ah
P1	90h
P2	A0h

Lc	Length of subsequent data field
Data	Data to be hashed (last block, in TLV format as follows (L is the length of the last data block): 80h || L || last data block).
Le	Length of the hash result. This is optional and is used if you want to return the final value of the hash in the command.

3.7.2.2 Hashing Performed Partially by the Card

In this case, all the data except the final block is hashed outside the card but the card hashes the final block. The APDU format is as follows:

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ah
P1	90h
P2	A0h
Lc	Length of subsequent data field
Data	Data is the concatenation of the immediate hash value and the final data block according to the hash algorithm used, as shown in Input Hash Lengths for Different Algorithms (SHAs) table.
Le	Empty.

Table 9. PSO–Hash Input Data (Hash Performed Partially by the Card)

Tag	Length	Value
90h	XXh	NN–byte intermediate hash value || 8–byte or 16–byte counter (number of bits already hashed). The length of the counter depends on the SHA algorithm as shown in in Input Hash Lengths for Different Algorithms (SHAs) table.
80h	L	Final data block Where: L is the length of the final data block, which must be between 0 and the block lengths as indicated in in Hash Algorithm Constants table.

Table 10. Input Hash Lengths for Different Algorithms (SHAs)

Hash	XX	Length of Intermediate Hash (NN bytes) and Counter
SHA-1	1Ch	20 + 8 bytes
SHA-224	28h	32 + 8 bytes
SHA-256	28h	32 + 8 bytes
SHA-384	48h or 50h	64 + 8 bytes or 64 + 16 bytes
SHA-512	48h or 50h	64 + 8 bytes or 64 + 16 bytes

3.7.2.3 Hashing Performed Entirely Externally

In this case, all the data is hashed outside the card. The APDU format is as follows:

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ah
P1	90h
P2	A0h
Lc	Length of subsequent data field
Data	Hash value, in TLV format as follows (L is the length of the hash value according to the algorithm ID): 90h || L || hash value). **Algorithm ID** **Length of hash value** 01h or 02h 1–36 bytes 03h 16–36 bytes 11h or 12h 20 bytes 32h 28 bytes 41h or 42h 32 bytes 52h 48 bytes 62h 64 bytes
Le	Empty.

3.7.3 Response

Byte	Value
Data	Empty
SW1-SW2	Status bytes

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
64h	00h	Incorrect checksum for hash value
68h	84h	The chaining mechanism is not available for the command.
69h	82h	Error during secure messaging
69h	85h	Conditions of use not satisfied, for example: ④ Algorithm ID must be one of those in Values for the algorithm reference used in Digital Signature Template table. ④ Length of input hash data not valid for algorithm ID ④ Security attribute for the current contact/contactless interface is absent,

		zero or not BER-TLV coded.
69h	86h	Command not allowed (use of SHA-1 has been prohibited by the cryptographic environment parameters data object)
6Ah	80h	Incorrect data. Input template in incorrect format

3.8 PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE

The PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command is used to compute a digital signature (from either a message or a hash computation).

A message is always hashed before signing. This hash can be performed by in one of three ways:

- entirely by card
- entirely externally
- partially externally and partially by card. In this case, the data is hashed externally but the last block is hashed by the card.

In all cases, card pads the hash to create Digital Signature Input (DSI) and signs this DSI by using the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command.

The current SE indicates the keys and algorithms to use for the signature and, if applicable, the hash algorithm.

3.8.1 Conditions of Use and Security

3.8.1.1 Usage

The command must be preceded by a PERFORM SECURITY OPERATION: HASH command which provides the hash or sets the hashed data if the hash is performed entirely externally.

None of the following actions must have taken place in between the execution of the PERFORM SECURITY OPERATION: HASH command and the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command otherwise the hash data is lost:

- Select FINEID application
- Another completed hash computation
- A MANAGE SECURITY ENVIRONMENT: SET command

The current SE must contain a DST which refers to a private key. This key must be a signature key and must have been initialized.

Caution: The key modulus value must be greater than the value of the padded message to be signed. Remember that messages padded with ISO 9796–2 padding begin with a padding byte of 6Ah, so any modulus value beginning with a value less than 6Ah is not allowed. In such a case, the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE does not return an error code in its status bytes, but generates a signature that has no meaning.

3.8.1.2 Security

In the application phase any security attributes set for the private key regarding the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE must be fulfilled.

Note: For clarity, the command is described here with the data in plain text.

If the signature key has the optional non-repudiation flag activated (value other than 00h or absent), the applet sets the PIN(s) protecting the key to “unverified” after the command (so the PIN needs to be presented again if the key is used again).

Note: If more than one PIN is protecting the signature key and the PINs are to be set to unverified afterwards, the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command unverifies ONE PIN only (the one that first granted access).

If the signature key has the optional non-repudiation flag deactivated (flag is present and value is 00h), the applet sets the PIN(s) protecting the key to “verified” after the command.

If the signature key is protected by more than one PIN defined in the security attributes as OR, then a correct presentation of any one of the PINs is valid for the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command. Thus, if a PIN had already been verified, the second PIN could be used to authorize another PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command, without the need to re-verify the first PIN. If the security attributes specified an AND

condition, then all the specified PINs need to be verified for ONE use of the PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE command.

If the signature was not completed because the APDU processing was interrupted due to the context not being properly established (that is, the applet returns status bytes of 69 82h) the PIN Presentation flag remains untouched.

If the signature key is protected by one or more PINs, and the "Change PIN before first use" option is active, then these PINs must have been changed at least once since personalization for the signature key to be available. In other words, the "changed" flag for the PIN must be true.

If the signature key has a signature counter associated with it, the counter is incremented when the command is performed successfully. If the incremented counter exceeds the maximum allowed value, the command fails.

3.8.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ah
P1	9Eh - digital signature data object is returned in response
P2	9Ah - data field contains data to be signed
Lc	Empty
Data	Empty
Le	Maximum length of the expected length of the response, that is, the length of the signature with no tag or length bytes.

Note: If the response is greater than 256 bytes including secure messaging (for example with key lengths of 2,048 bits), the card opens a retrieval sequence. The sequence differs according to the communication protocol in use (T=0 or T=1/T=CL). For details see Annex B.

3.8.3 Response

Byte	Value
Data	Digital signature

SW1-SW2	Status bytes

The card returns the value of the digital signature, followed by the SW1, SW2 status codes. RSA signatures are unformatted. ECDSA signatures are in the format r || s, where r and s are integers and are the co-ordinates of the point on the elliptic curve that correspond to the signature value.

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
69h	82h	Security conditions not satisfied, for example: - Security attribute not fulfilled for private key - Error during secure messaging - PIN protecting the signature key has not been presented (it is valid for one signature only) - Integrity of DTBS check failed
69h	85h	Conditions of use not satisfied, for example: - No hashed message available from PSO–Hash command. - No algorithm or private key in the current SE's DST. - Algorithm not recognized. - Private key in current SE's DST is not a digital signature key. - Private key in current SE's DST has not been initialized, that is, at least one of the elements has not been initialized using the Put Data command. - PIN protecting the signature key has not been changed since personalization, and the "Change PIN before first use" option is active. Table 29 on page 35. - Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
69h	86h	Command not allowed (use of SHA-1 has been prohibited by the cryptographic environment parameters data object)
6Ah	84h	Signature counter exceeds maximum allowed value.
6Ah	86h	Incorrect P1, P2. Must be 9Eh 9Ah

3.9 PERFORM SECURITY OPERATION: DECIPHER

The PERFORM SECURITY OPERATION: DECIPHER command decrypts a message using a decipher key stored in the card. It returns the message in plain data.

The PERFORM SECURITY OPERATION: DECIPHER command can be used in "chaining" mode. This means that if the data is too large to be sent in a single

PERFORM SECURITY OPERATION: DECIPHER command you can send it in two PERFORM SECURITY OPERATION: DECIPHER commands. The first command is sent with CLA = 1Xh. This tells the card to wait for the remaining data. Then issue another PERFORM SECURITY OPERATION: DECIPHER command with CLA= 0Xh. The card concatenates the data from the first command with that of the second and then performs the deciphering operation. You will need to use chaining mode to decipher data that was encrypted using a 2,048–bit key or longer.

3.9.1 Conditions of Use and Security

3.9.1.1 Usage

The private key used to decipher the data must be a decipher key and must be stored in the confidentiality template (CT - tag B8h) of the current SE. The encrypted message must be the same length as this key.

If you send the command with CLA = 1Xh to indicate chaining mode, the next command you issue must be another PERFORM SECURITY OPERATION: DECIPHER command with CLA = 0Xh, otherwise the chaining mechanism ends and the data is lost.

The total amount of data that can be sent in "chaining" mode is 742 bytes. If this figure is exceeded, the chaining session is broken and the card returns the error 6700h.

The message to be decrypted must have been encrypted using one of the following RSAES (RSA encryption schemes):

- RSA with either PKCS #1 v1.5 padding (block type 2 format)
- RSA with OAEP padding. The format of OAEP padding is outside the scope of this document. For more information please refer to the RSA PKCS#1 v2.1 standard.

Caution: OAEP has an optional label (to be associated with the message to encrypt), this label is hashed and concatenated with a padding and the message to encrypt. When deciphering the encoded data, the service uses the same label and verifies its hash (retrieved from the deciphered data). The hash of the label is used as an additional verification token to certify the encrypted data. However it is not available in the JavaCard 2.2.2 API (this is also true for JavaCard 3.0.1).

The PKCS #1 V2.1 RSAES OAEP contains inner parameters. The following parameters have been chosen:

- The inner hash is the same as the external hash
- The Label is an empty string
- The MGF function is the MGF1() defined in PKCS #1 V2.1

3.9.1.2 Security

If specified, the "Decipher" security attribute for the private key referenced in the CT of the current SE must be fulfilled.

Note: For clarity, the command is described here with the data in plain text.

The algorithm ID must be one of those listed in Values for the algorithm reference used in Confidentiality Template table.

3.9.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging 10h - transmit first portion of data in chaining mode with no secure messaging and wait for final portion 1Ch - transmit first portion of data in chaining mode with secure messaging and wait for final portion
INS	2Ah
P1	80h - decrypted value is returned in response
P2	86h - cryptogram in data field
Lc	Length of subsequent data field
Data	Cryptogram = PI || encrypted message. The encrypted message must be in the correct format (please refer to the PKCS#1 standard). PI=81h.
Le	Empty or maximum length of data expected in response

Note: The total length of the encrypted message (not including the 81h byte and secure messaging) must be the same length as the key used to decipher it.

3.9.3 Response

Byte	Value
Data	Decrypted cryptogram (padding is removed by the card and only the actual data is returned).
SW1-SW2	Status bytes

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
67h	00h	Incorrect length, Lc, for example: - Data is not the same length as the private key + 1 for padding indicator (PI). - Total data in chaining mode exceeds 742 bytes.
69h	82h	Security status not satisfied: - Security attribute not fulfilled for private key referenced in CT of current SE - Error during secure messaging
69h	84h	Referenced data (private key) invalidated or does not exist
69h	85h	Conditions of use not satisfied, for example: - No algorithm or private key in the current SE's CT - Algorithm ID must be 1A, 1D, 3D, 4D, 5D or 6Dh - Private key in current SE's CT is not a decipher key (that is, bit 2 of the private key usage byte is not set to 1) –see Table 83 on page 152 - Private key in current SE's CT has not been initialized, that is, not all the elements have been updated using the Put Data command - Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
69h	86h	Command not allowed (Algorithm ID indicates the use of RSAES OAEP algorithm but without a supported related hash method (i.e. algoID is 'xD' with x outside the values [1,3,4,5,6])
6Ah	80h	Incorrect data, for example: - PI must be 81h - The message before encryption and padding was longer than key length - 11 bytes - The message before encryption and padding was not in the format described by Table 121 on page 235 or the correct OAEP format defined in PKCS#1 V2.1 standard. - Incorrect tag or tag length (including TLVs used in secure messaging) - Mismatch between algoID used to cipher the message and the algoID relevant for Decipher
6Ah	86h	Incorrect P1, P2. Must be 80h 86h

3.10 CHANGE REFERENCE DATA

The CHANGE REFERENCE DATA command replaces the value of the reference PIN by a new value.

The current value of the reference PIN must be presented as part of the CHANGE REFERENCE DATA command.

If the CHANGE REFERENCE DATA command is successful, the following actions take place:

- The reference PIN Try Counter is set to PIN Try Limit.
- The reference PIN validated flag is set to false (because the new value has not yet been verified).
- The reference PIN value is updated with the value sent in CHANGE REFERENCE DATA command.
- The reference PIN's "PIN changed" flag is set to "true".
- The reference PIN usage counter is unchanged.

.

If the CHANGE REFERENCE DATA command fails, the following actions take place:

- The reference PIN Try Counter is decremented by 1.
- The reference PIN validated flag is set to false.
- The reference PIN usage counter is unchanged.

3.10.1 Conditions of Use and Security

3.10.1.1 Usage

3.10.1.2 Security

The PIN Try Counter and the Usage Counter for both the reference PIN and the change PIN must not be zero.

Note: For clarity, the command is described with the data in plain text.

3.10.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ch
P1	00h - reset retry counter and set new verification data
P2	Reference of the PIN:

	- 11h for global PIN - 81h–8Fh for local PIN
Lc	Length of subsequent data field
Data	Resetting code (padded to the correct length: 8–16 bytes (local PIN), 8-12 bytes (global PIN)) followed by new reference data (padded to the correct length 8–16 bytes (local PIN), 8-12 bytes (global PIN)). Padding is done according to ISO/IEC 7816-15.
Le	Empty

3.10.3 Response

Byte	Value
Data	Empty
SW1-SW2	Status bytes

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
63h	Cxh	Reference PIN not verified.”x” attempts remaining for change PIN
67h	00h	Incorrect length for example: ④ Lc incorrect ④ Length of new PIN value is not the same as length of existing value
69h	82h	Security conditions not satisfied (error during secure messaging or security attribute of reference PIN for Change Reference Data is never)
69h	83h	Authentication method blocked (Change PIN blocked)
69h	84h	Referenced data invalidated (PIN Try Counter or PIN Usage Counter for either the reference PIN or the change PIN has reached zero)
69h	85h	Conditions of use not satisfied, for example: ④ Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
6Ah	86h	Incorrect P1, P2
6Ah	88h	Referenced data not found

3.11 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.

If the RESET RETRY COUNTER command is successfully executed, the following actions take place:

- The reference PIN Try Counter is set to PIN Try Limit.
- The reference PIN validated flag is set to false (because the PIN is not yet verified)
- The reference PIN unblocking counter is decremented by 1, unless its value is A5h (no limit to the number of times reference PIN can be unblocked) or 00h (already been unblocked maximum allowed number of times).
- The reference PIN usage counter is unchanged.
- The PUK Try Counter is set to unblock PUK Try Limit
- The PUK validated flag is set to false
- The PUK usage counter is decremented by 1 unless its value is FFh (no limit to the number of times unblock PIN can be used) or 00h (already been used maximum allowed number of times).
- If a new value for the reference PIN is specified, the reference PIN value is updated with the value sent in the RESET RETRY COUNTER command
- If a new value for the reference PIN is specified, the reference PIN's "PIN changed" flag is set to "true"

If the RESET RETRY COUNTER command fails, the following actions take place:

- The PUK Try Counter is decremented by 1
- The PUK validated flag is set to false

3.11.1 Conditions of Use and Security

3.11.1.1 Usage

When used to unblock a PIN, a new value can be given to the reference PIN.

3.11.1.2 Security

None of the following counters must be zero:

- The Try Counter of the PUK
- The Unblocking Counter of the Reference PIN

④ The Usage Counters for both the reference PIN and the PUK.

Note: For clarity, the command is described with the data in plain text.

3.11.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	2Ch
P1	00h - reset retry counter and set new verification data 01h – only reset retry counter
P2	Reference of the PIN: - 11h for global PIN - 81h–8Fh for local PIN
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), or resetting code (padded to the correct length Padding is done according to ISO/IEC 7816-15.
Le	Empty

3.11.3 Response

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

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
63h	Cxh	PUK not verified.”x” attempts remaining for PUK

67h	00h	Incorrect length, Lc.
69h	82h	Security conditions not satisfied (error during secure messaging or security attribute for the reference PIN is never)
69h	83h	Authentication method blocked (the FP_PUK needed to unblock the reference PIN is blocked itself)
69h	84h	Referenced data invalidated (PIN Unblocking Counter for reference PIN or PIN Try Counter or PIN Usage Counter for FP_PUK has reached zero)
69h	85h	Conditions of use not satisfied, for example: ④ Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
6Ah	86h	Incorrect P1, P2
6Ah	88h	Referenced data not found

3.12 UPDATE BINARY

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

There are two ways of referencing the EF:

④ Implicit selection. In this case, the EF is under the current DF, and is specified by its short file identifier (SFI). P1 specifies the SFI and P2 specifies the start address for the update (called the offset). The command cannot be used for files with an SFI of 1Fh.

④ Direct selection. In this case the EF must have been previously selected using the SELECT FILE command. The data is updated directly in the current EF. In this case the offset is specified over P1 and P2.

3.12.1 Conditions of Use and Security

3.12.1.1 Usage

The command can be used on any activated EF.

In the case of implicit selection, the EF becomes the current EF after execution of the command.

3.12.1.2 Security

If the EF to be updated is protected by a security attribute, then this security attribute must be fulfilled in order to perform the UPDATE BINARY command.

Note: For clarity, the command is described here with the data in plain text.

3.12.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	D6h
P1	See table below
P2	See table below
Lc	Length of subsequent data field (up to a maximum of 248 bytes)
Data	Data to be updated
Le	Empty

Table 11. 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 updated
1	-	-	-	-	-	-	-	-	P1 specifies a short FID and P2 specifies an 8-bit offset of the data to be read
1	0	0	x	x	x	x	x	-	– short FID (value domain 1 – 30)

3.12.3 Response

Byte	Value
Data	Empty
SW1-SW2	Status bytes

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
62h	82h	End of file reached (offset + Lc is higher than the end of the file)
64h	00h	Execution error, file descriptor checksum error.
67h	00h	Incorrect length, Lc.
68h	84h	The chaining mechanism is not available for this command.
69h	82h	Security status not satisfied, for example: ④ The security attributes are not satisfied ④ Error during secure messaging
69h	85h	Conditions of use not satisfied, for example: ④ Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
69h	86h	Command not allowed (no current EF)
6Ah	82h	File not found (b8 of P1 = 1 indicating implicit selection, but no SFI given)
6Ah	86h	Incorrect P1, P2 (check that offset is inside the EF and that SFI is not 1Fh)
6Bh	00h	Incorrect P1, P2 (offset outside EF)

3.13 ERASE BINARY

The ERASE BINARY command is used erase the contents of a transparent (binary) file.

There are two ways of referencing the EF:

- ④ Implicit selection. In this case, the EF is under the current DF, and is specified by its short file identifier (SFI). P1 specifies the SFI and P2 specifies the start address for the erase (called the offset). The command cannot be used for files with an SFI of 1Fh.
- ④ Direct selection. In this case the EF must have been previously selected using the SELECT FILE command. The data is erased directly from the current EF. In this case the offset is specified over P1 and P2.

The command erases data from the offset until the end of the file is reached or until the end address in the data field, if one exists.

Note:

- The end address is in fact the address of the first byte NOT to be erased.
- After the ERASE BINARY command has been run, an implicitly selected EF has its file contents erased and becomes the current EF.

3.13.1 Conditions of Use and Security

3.13.1.1 Usage

The command can be used on any activated EF.

In the case of implicit selection, the EF becomes the current EF after execution of the command.

3.13.1.2 Security

If the EF containing the data to be erased is protected by a security attribute, then this security attribute must be fulfilled in order to perform the ERASE BINARY command.

Note: For clarity, the command is described here with the data in plain text.

3.13.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	0Eh
P1	See table below
P2	See table below
Lc	00h – no data field 02h – offset in data field
Data	If Empty - indicates the command erases data until the end of the file If present indicates the offset in bytes of the first byte NOT to be erased
Le	Empty

Table 12. ERASE 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	x	x	x	x	x	-	– short FID (value domain 1 – 30)

3.13.3 Response

Byte	Value
Data	Empty
SW1-SW2	Status bytes

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
62h	82h	End of file reached (address specified in data field is higher than the end of the file)
64h	00h	Execution error, file descriptor checksum error.
67h	00h	⑷ Incorrect length, Lc. ⑷ Offset given in data field is not higher than offset of start address in P1
69h	82h	Security status not satisfied, for example: ⑷ The security attributes are not satisfied ⑷ Error during secure messaging
69h	85h	Conditions of use not satisfied, for example: ⑷ Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
69h	86h	Command not allowed (no current EF)
6Ah	82h	File not found (b8 of P1 = 1 indicating implicit selection, but no SFI given)
6Ah	86h	Incorrect P1, P2 (does not match the functionality)
6Bh	00h	Incorrect P1, P2 (offset outside EF limits)

3.14 GET DATA

The GET DATA command is used to retrieve following BER-TLV objects:

- Public key elements
- PIN information

The data in the response is in TLV format.

3.14.1 Conditions of Use and Security

3.14.1.1 Usage

When retrieving ELC public key elements, the elements must be retrieved individually, that is, only one element per GET DATA command. For RSA public key elements, the elements can be retrieved individually or the command can retrieve ALL the elements in the one command.

3.14.1.2 Security

The use of the GET DATA command depends on the security attributes of the object whose value is to be retrieved.

Note: For clarity, the command is described here with the data in plain text.

3.14.2 Format

Byte	Value
CLA	00h - no secure messaging 0Ch - secure messaging
INS	CBh
P1	00h
P2	FFh
Lc	Length of subsequent data field
Data	See tables below.
Le	Number of bytes expected in response

Table 13. Get Data Coding Structure – Public Key Element (the modulus in this example)

Tag	Length	Value			Description	No. of Bytes
B6h	03h				Tag and length of DST Template	2
		83h	01h	KeyID	TLV of public key	3
7Fh 49h	02h				Tag and length of public key template	3
		Tag	Length			
		81h	00h		Tag and length of modulus	2
					TOTAL BYTES	**10**

Note: The length of 00h means that all the element is recovered.

Table 14. Tags for ELC public key elements

Tag	Description
81h	p: prime modulus according to curve type
82h	a: 1st coefficient of curve
83h	b: 2nd coefficient of curve
84h	G: coordinates X and Y in F; defining a curve point G of order n. DO is formatted as follows (uncompressed format): 04h || XG || YG
85h	n: order of the base point (positive prime integer)
86h	Q - coordinates X and Y in F defining the public key point Q in E. DO is formatted as follows (uncompressed format): 04h || XQ || YQ
87h	h: cofactor

Table 15. Get Data Coding Structure – entire RSA public key (modulus and exponent elements)

Tag	Length	Value			Description	No. of Bytes
B6h	03h				Tag and length of DST Template	2
		83h	01h	KeyID	TLV of public key	3
7Fh 49h					Tag of public key template	2
		Tag				
		80h			Tag of entire public key	1
					TOTAL BYTES	**8**

Caution: Remember that this would not be possible for ELC public keys.

Table 16. Get Data Coding Structure – PIN Container

Tag	Length	Value			Description	No. of Bytes
A0h	03h	Tag	Length	Value	Tag and length of FP_PIN Template	2
		83h	01h	xxh	xxh is FP_PIN Reference as follows: 81h-8Fh for local PIN 11h for global PIN	3

The “PIN changed” flag is returned only if the “Return PIN changed flag” parameter is set.

Caution: This is not ISO-compliant since the tag A0h means constructed tag, which this isn’t. However, if you require your application to be ISO-compliant, you can safely add additional tags to the A0h template as this will not affect the response data (all the data for the PIN is returned).

3.14.3 Response

Byte	Value
Data	Value of retrieved Data Object. See examples below.
SW1-SW2	Status bytes

Table 17. Get Data Response Structure – Public Key Element (the modulus in this example)

Tag	Length	Value			Description	No. of Bytes
B6h	03h				Tag and length of DST Template	2
		83h	01h	KeyID	TLV of public key	3
7Fh 49h	8183h				Tag and length of public key template	4
		Tag	Length	Value		
		81h	8180h	ValMod	TLV of modulus (value coded on 128 bytes)	131
					TOTAL BYTES	**140**

Table 18. Get Data Response Structure – entire RSA public key (modulus and exponent elements)

Tag	Length	Value			Description	No. of Bytes
B6h	03h				Tag and length of DST Template	2
		83h	01h	KeyID	TLV of public key	3
7Fh 49h	81 8Dh				Tag and length of public key template	4
		Tag	Length	Value		
		81h	81h 80h	ValMod	TLV of modulus (value coded on 128 bytes)	131
		82h	LEXP	ValEXP	TLV of exponent (value coded on 8 bytes)	10
					TOTAL BYTES	**150**

Table 19. Get Data Response Structure – PIN Information

Tag	Length	Value			Description	No. of Bytes
A0h	LA0	Tag	Length	Value	Tag and length of PIN Template	2
		83h	01h	xxh	xxh is PIN Reference byte (81h-8Fh for local PIN, 11h for global PIN)	3
		8Ch	L8C		Security Attributes - contact interface	Var
		9Ch	L8C		Security Attributes - contactless interface	Var
		DFh 21h	04h		PIN attributes	7
		DFh 27h	02h		PIN credentials counter	5
		DFh 28h	01h		PIN length	4
		DFh 2Fh	01h		PIN changed flag (1 means PIN has been changed)	3
					TOTAL BYTES	**Var**

If the PIN being returned is the global PIN, and the global PIN has been initialized outside FINEID application, the PIN length value is returned as 00h. If however, the global PIN has been initialized and is managed by the FINEID application, then the PIN length will be accurate.

The content of PIN attributes are

Field	Length	Description
#1	1 byte	PIN Try Counter. Stores the number of remaining attempts that can be made to present the correct PIN before it is blocked.
#2	1 byte	Usage counter (range 01h – FFh; FFh means no limit).
#3	1 byte	Unblocking counter (range 00h – 0Fh or A5h for no limit)
#4	1 byte	Unblocking PIN method. See table below.

Table 20. Unblocking PIN method byte coding

Meaning	b8	b7	b6	b5	b4	b3	b2	b1
Unblock PIN not allowed (neither by unblock PIN nor by MS 3DES3 External Authentication	0	0	0	0	0	0	0	0
MS 3DES3 Ext Auth grants unblock PIN		1						
MS 3DES3 Ext Auth does not grant unblock PIN		0						
FP_PUK Ref to be used	X		X	X	X	X	X	X
Not checked	1	1	1	1	1	1	1	1

The range of values for the FP_PUK reference are 01h – 0Fh for a local PIN, 11h for the global PIN.

The unblocking PIN method byte is coded as an OR condition between the MS 3DES3 External Authentication and the FP_PUK reference. Here are some example values:

- 84h: Unblocking possible with PIN#4 (PIN ref is 84h)

- 40h: Unblocking possible only if MS 3DES3 Ex Auth has taken place.
- C4h: Unblocking possible with PIN#4 or if MS 3DES3 Ex Auth has taken place.
- 00h: NEVER

Note: b7 is taken into account EXCEPT when the byte is FFh.

Credentials Counter

Certain objects may be protected by security attributes that specify that user authentication (PIN) is necessary before access is granted to a particular command. The security attribute therefore checks the PIN status when the command is issued.

The aim of the credentials counter is to limit the number of times that such checks are made, either for the current session or for the life of the applet. The counter comes into operation after a successful PIN verification.

The counter can exist for both local PINs and the global PIN.

Caution: In the case of global PINs, the counter is internal to the FINEID application. If used by the FINEID application, the counter is correctly reset in RAM after a successful global PIN presentation. However it is NOT reset if the global PIN was successfully verified by another applet. Consequently, the mechanism is not triggered in such a case and so access requiring the global ¨PIN is not granted, even though the global PIN has been successfully verified.

The first byte of the counter is the counter type and can be “RAM” (00h) or “EEPROM” (80h). The two types of counter work differently.

RAM Type

The counter limits the number of times that the PIN status is checked for the current session. A session starts for each successful Verify on a given PIN, and ends when the credential counter reaches 0. It works as follows:

1. When the PIN is successfully verified, the “current credential counter value” (stored in RAM) is reset to the credentials counter value.

2 Each time a security attribute or any action checks the PIN status (such as an access condition or a PIN used as an FP_PUK), the "current credential counter" decrements by 1. The credentials counter value in the PIN is NOT decremented.

3 If the current credential counter reaches zero, operations that test if this PIN access is granted will fail.

EEPROM Type

The counter limits the number of times that the PIN status is checked for the life of the applet. It works as follows:

1 When the PIN is successfully verified, the counter comes into operation.

2 Each time a security attribute or any action checks the PIN status (such as an access condition or a PIN used as an FP_PUK), the credentials counter value in the PIN decrements by 1.

3 If the credentials counter value reaches zero, it remains as zero, meaning that the PIN can no longer be used for access rights. However, the PIN can still be used with the CHANGE REFERENCE DATA and RESET RETRY COUNTER commands.

The Credentials Counter is a two-byte value coded as follows:

Table 21. Credentials Counter Coding

Counter Type	00h	RAM
	80h	EEPROM
Counter Value	00h to FEh	④ If RAM type, the current credentials counter in RAM is set to this value when the PIN is verified. It is the RAM value that is decremented when the PIN status is checked – NOT the Counter Value in the PIN. ④ If EEPROM type, this value is decremented each time the PIN status is checked
	FFh	There is no limit to the number of PIN status checks. This is true for both types of Credentials Counter.

The "PIN changed" flag is returned only if the "Return PIN changed flag" parameter was set to true when the applet was personalized.

Possible values for the SW1, SW2 status codes are as follows:

SW1	SW2	Description
90h	00h	Command processed without error
69h	82h	Security status not satisfied, error during secure messaging
69h	85h	Conditions of use not satisfied, for example: Security attribute for the current contact/contactless interface is absent, zero or not BER-TLV coded.
6Ah	80h	Incorrect FP_PIN reference or incorrect data field when retrieving PIN information
6Ah	86h	Incorrect P1, P2.

4 Implementation guidelines for software developer

4.1 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 should be unaffected between transactions. Check ISO/IEC 7816-15 for additional information on global PINs.

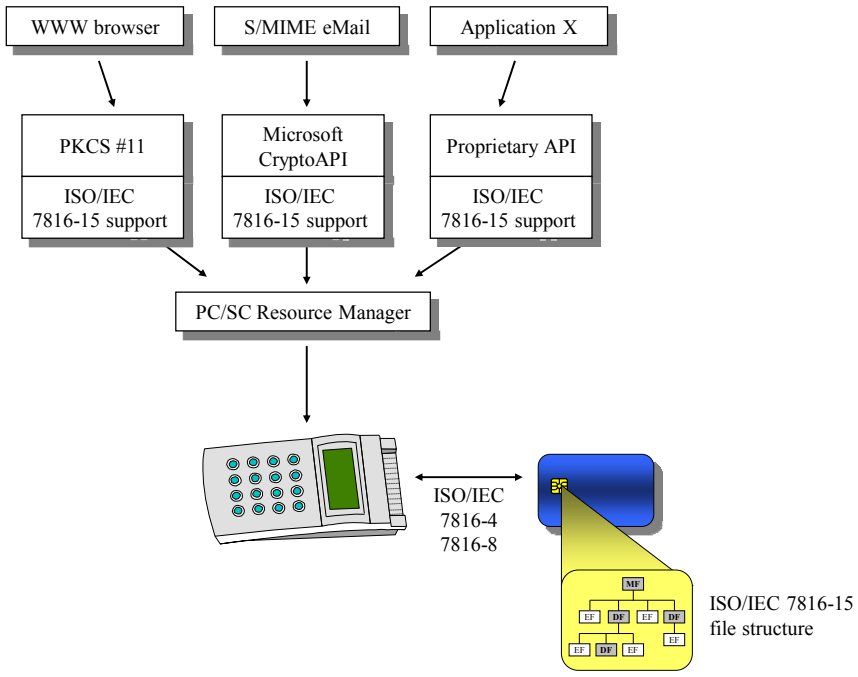


Figure 1. Example scenario of multiple host applications - single card

4.2 Resetting the card

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

4.3 Application/File selection

4.3.1 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	-

4.4 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)

4.5 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.

Table 22. Objects and protected operations

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).

4.6 Accessing objects

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

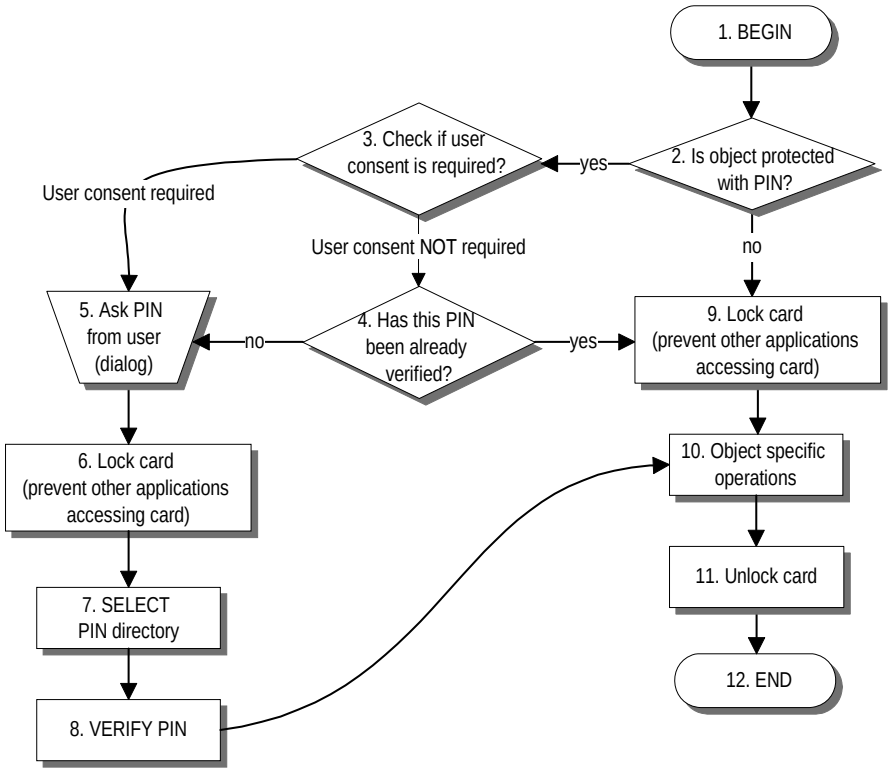


Figure 2. Example of PIN logic

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	0C	31 32 33 34 35 36 00 00 00 00 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 (Compute Digital Signature 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 Compute Digital Signature operation. Requiring user interaction for all Compute Digital Signature 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 22.

4.7 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.

4.7.1 Signature operation

It is assumed that corresponding PIN verification has already been done.

Set the following properties into the SE Digital Signature Template:

- algorithm reference (= 42 i.e. RSASSA-PKCS1-v1_5 signature with SHA-256, card does padding and DigestInfo encapsulating of the hash)
- key reference (= 01 derived from PrivateKeyAttributes.keyReference)

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE: SET	00	22	41	B6 DST in data field	06	80 01 42 (algorithm reference = 42) 84 01 01 (private key reference)	-

The hash is performed entirely by the card, only one block to hash:

Command	CLA	INS	P1	P2	Lc	Data	Le
PSO: HASH	00	2A	90	A0	16	80 14 4B 52 16 5B 4A B6 54 C3 E5 4F 64 B5 F1 EE A6 45 D4 6B 65 C8	-

Sign the hash calculated by the card:

Command	CLA	INS	P1	P2	Lc	Data	Le
PSO: COMPUTE DIGITAL SIGNATURE	00	2A	9E	9A	-	-	00

4.7.2 Decryption operation

It is assumed that corresponding PIN verification has already been done.

Set the following properties into the SE Confidentiality Template:

- algorithm reference (= 1A i.e. RSASSA PKCS#1_v1.5 decryption, card removes padding)
- key reference (= 01 derived from PrivateKeyAttributes.keyReference)

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE: SET	00	22	41	B8 CT in data field	06	80 01 1A (algorithm reference = 1A) 84 01 01 (private key reference)	-

Decrypt the modulus size (example 2048 bits), divided to two APDUs, the first contains the 255 bytes and the second rest 2 bytes (notice CLA = 10 in the first APDU):

Command	CLA	INS	P1	P2	Lc	Data	Le
PSO: DECIPHER	10	2A	80	86	FF	81 (padding indicator byte) 4B 52 16 … 54 C3 E5 (first 254 bytes of cryptogram)	-

Command	CLA	INS	P1	P2	Lc	Data	Le
PSO: DECIPHER	00	2A	80	86	02	52 16 (last 2 bytes of cryptogram)	XX

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

Get the response in T=0 protocol:

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

5 PACE

5.1 What is PACE Authentication?

Password authenticated connection establishment (PACE) is a service that can be used by the FINEID application. The PACE Protocol is a password authenticated Diffie-Hellman key agreement protocol that provides secure communication (secure messaging) and password-based authentication of the smart card and the inspection system (that is, the smart card and inspection system share the same password).

PACE is independent of the FINEID application. It interacts between the software application that wants to communicate with the card and the FINEID application. It is used with the FINEID application instance as a whole, not with its internal data objects. You cannot for example set the access condition of an FINEID application data object to require PACE authentication.

PACE is active once the PACE module is fully personalized. This means:

- A CardAccess EF has been created and updated directly below the MF. This file contains all the information needed by the PACE application.
- The PACE passwords has been configured.
- The restriction lists are updated to set up the rights to access FINEID application. These lists are described in “How Does PACE Authentication Work?” In FINEID context the contactless communication is protected by PACE. The only exception is Card Manager application which can be accessed without PACE protection, although contactless communication is used.

If present, PACE can specify two levels of security:

- PACE authentication only.
- PACE authentication + secure messaging (always ENC + MAC).

In FINEID context PACE authentication + secure messaging (always ENC + MAC is used. These two levels are described in “PACE Authentication Levels”.

5.2 How Does PACE Authentication Work?

If the CardAccess EF is present, PACE is used to determine if the external system is authorized to access the FINEID application. In FINEID context the contactless communication is protected by PACE. The only exception is Card Manager

application,which can be accessed without PACE protection, although contactless communication is used.

These are the steps.

1. Card power-up or re-set
2. Determine if mode is contact or contactless (based on the ATR/ATS historical bytes)
3. Read CardAccess EF to determine how card has been personalized with PACE.
4. Do PACE authentication, then select FINEID application using PACE secure messaging and continue communication using PACE secure messaging.

5.3 PACE Authentication Levels

This section provides a brief description of the different authentication levels.

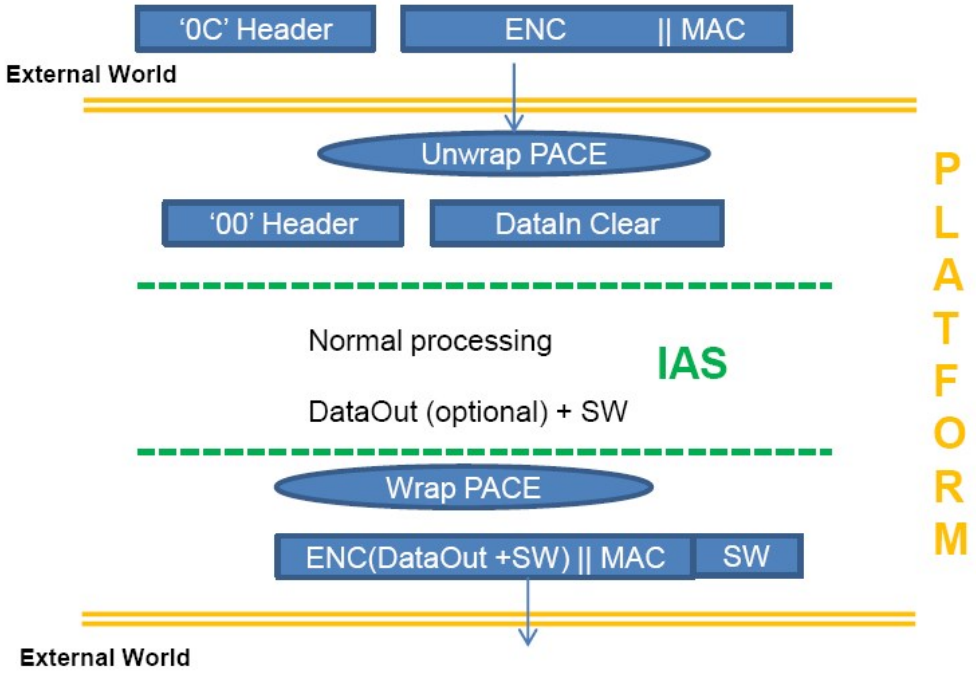
5.3.1 PACE Authentication Only

PACE authentication means that a secret shared by the card and terminal must be successfully presented in order to access the FINEID application. Typically this is the global PIN, but it could equally be the card access number (CAN) or the MRZ data. In FINEID context all these passwords are used.

5.3.2 PACE Authentication + Secure Messaging

PACE secure messaging is applied to an APDU at the platform level. The PACE SM is then unwrapped and the APDU data is available in clear text for the FINEID application (with no possibility of the external world seeing this data in clear text, but note the Caution in “PACE and FINEID application Secure Messaging”). After being processed by the FINEID application, the APDU is then wrapped in PACE SM again before being returned to the external world.

The following diagram illustrates this.



Although the PACE SM is "unwrapped" before the APDU is sent to FINEID application, the use of secure messaging does have an impact on the length of APDU commands and responses at the FINEID application level. Please refer to "Effect on APDU Command and Response Length".

5.3.3 PACE and FINEID application Secure Messaging

The important point to note is that PACE SM and FINEID application SM are never in use at the same time.

Once FINEID application SM is established, the PACE SM "sleeps" until the FINEID application completes the APDU processing and returns its SW (and maybe DataOut).

Note: PACE SM sleeps only on the logical channel on which FINEID application is selected and is still active on other logical channels.

PACE SM wakes under one of the following circumstances:

- FINEID application secure messaging error
- FINEID application is reselected.

Caution: When a FINEID application AES SM session is in progress, sending an APDU in clear text is considered as an SM error and therefore wakes PACE SM. In 3DES SM, it is not considered as an error, therefore the 3DES SM session is not broken and PACE SM

does not take over. Therefore the only way to guarantee protection against eavesdropping with the PACE SM is to make sure that the FINEID application SM is AES.

5.3.3.1 Effect of Breaking PACE SM

If PACE SM is broken (for example, due to an APDU sent with incorrect SM or sent in clear when SM was expected) then all applications using PACE SM on other logical channels are deselected on their respective channels.

5.3.4 Impacts of PACE on FINEID application

5.3.4.1.1 Effect on APDU Command and Response Length

When using PACE SM, the APDU commands are unwrapped before being sent to the FINEID application. However, because some bytes are taken up with the SM construction, the unwrapped APDU command is smaller. The effect of this is that there is a stricter limitation on the number of bytes available for APDU commands at the FINEID application level.

The same is true when the FINEID application returns data to the software application. It has to allow for the response to be sent in PACE SM and therefore for some bytes to be used for the SM. Therefore fewer bytes are available for the response at the FINEID application level.

The number of bytes taken up by PACE SM construction differs according to whether the PACE SM is DES or AES.

The maximum number of bytes available for APDU commands and responses are:

- For DES PACE SM: 239 bytes
- For AES PACE SM: 231 bytes

Annex A (Informative): Coding of the File Control Parameters and File Control Information templates

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

FCI for DFs

Offset	Data	Description
0	6Fh	Tag of FCI template
1	L	Length of FCI data
2	83h	Tag of File ID
3	02h	Length of File ID
4–5	File ID	Value of File ID
6	8Ch	Tag of security attributes
7	L	Length of security attributes
8	AMB	Access mode byte
9–(8+X)	SCB	Security condition bytes (X)
9+X	84h	Tag of DF Name
10+X	L	Length of DF Name
11+X –	DF name	Value of DF name (up to 16 bytes)

Note: In the example above, the contact interface security attribute is given (tag 8Ch). This could alternatively be replaced by the contactless interface security attribute, (tag 9Ch) or both interfaces could be present.

FCI for EFs

Offset	Data	Description
0	6Fh	Tag of FCI template
1	L	Length of FCI data
2	81h	Tag of File Size
3	02h	Length of File Size
4–5	File Size	Value of File Size.
6	82h	Tag of FDB
7	01h	Length of FDB
8	FDB	Value of FDB
9	83h	Tag of File ID
10	02h	Length of File ID
11–12	File ID	Value of File ID
13	8Ah	Tag of Life Cycle Status byte for file
14	01h	Length of Life Cycle Status byte for file
15	Var.	Value of Life Cycle Status byte for file
16	8Ch	Tag of security attributes
17	L	Length of security attributes
18	AMB	Access mode byte
19–(18+X)	SCBs	Security condition bytes (X)

Note: In the example above, the contact interface security attribute is given (tag 8Ch). This could alternatively be replaced by the contactless interface security attribute, (tag 9Ch) or both interfaces could be present.

FCP for DFs

Tag	Length	Tag	Length	Value	Description
62h	0Bh–25h				Tag and length of FCP template
		82h	01h	38h	TLV of FDB
		83h	02h	Var.	TLV of File ID
		8Ch	02h–04h	Var.	TLV of Security attributes (access mode and security condition bytes) - contact interface
		9Ch	02h–04h	Var.	TLV of Security attributes (access mode and security condition bytes) - contactless interface
		84h	01-10h	Var.	TLV of DF name

The DF name is optional and if used must be the final parameter. The FDB and file ID are mandatory.

Only one security attribute is mandatory (it is possible to have 8Ch or 9Ch or both).

The structure of the FCP of a dedicated file is as follows:

Field	Length	Description
FDB	1 byte	The file descriptor byte (FDB) is set to 38h when a DF is created.
File ID	2 bytes	This is allocated when the file is created.
Security Attribute	2–4 bytes	The security attributes are made up of one Access Mode byte which indicates the commands to be "controlled", followed by 1–3 security condition bytes which indicate the conditions for each command indicated in the Access Mode byte. The security attributes must be present for at least one of the interfaces (contact or contactless). Both interfaces can be present if required. The coding is identical for the contact and contactless interfaces.
DF Name	1–16 bytes	This contains the name of the DF up to a maximum of 16 bytes. It can be used to reference a file in a command

FCP for EFs

Tag	Length	Tag	Length	Value	Description
62h	12h–1Eh				Tag and length of FCP template
		81h	02h	Var.	TLV of File size (in bytes)
		82h	01h	01h	TLV of FDB
		83h	02h	Var.	TLV of File ID
		8Ah	01h	Var.	TLV of Life cycle status
		8Ch	02h–06h	Var.	TLV of Security attributes (access mode and security condition bytes) - contact interface
		9Ch	02h–06h	Var.	TLV of Security attributes - contactless interface

All these elements are mandatory. However only one security attribute is mandatory (it is possible to have 8Ch or 9Ch or both).

The structure of the FCP of an elementary file is as follows:

Field	Length	Description
File Size	2 bytes	This specifies the size of the file.
FDB	1 byte	The file descriptor byte (FDB) is set to 01h, meaning transparent file, when an EF is created.
Identifier (FID)	2 bytes	This is allocated when the file is created. The short file identifier corresponds to the 5 least significant bits of the file identifier. It is used to reference a file in a command.
Life Cycle Status	1 byte	The life cycle status shows the status of the EF. The LCS byte is coded as described in the table below.
Security Attribute	2-6 bytes	The security attributes are made up of one Access Mode byte which indicates the commands to be “controlled”, followed by 1–5 security condition bytes which indicate the conditions for each command indicated in the Access Mode byte. The security attributes must be present for at least one of the interfaces (contact or contactless). Both interfaces can be present if required. The coding is identical for the contact and contactless interfaces.

Table 23. Life Cycle Status byte coding

b8…b5	b4	b3	b2	b1	State
0..0	0	0	0	1	CREATED
0..0	0	0	1	1	INITIALIZED
0..0	0	1	–	1	OPERATIONAL (ACTIVATED)
0..0	0	1	–	0	OPERATIONAL (DEACTIVATED)

EF Types

All EFs managed by FINEID application are transparent EFs. These have an FDB value of 01h. A transparent file consists of an unstructured sequence of bytes that can be accessed by specifying an offset relative to the start of the EF. The offset size is given in bytes. The first byte of a transparent EF has the relative address 00h.

Security Attributes

Files and data objects are protected by security attributes, that must be fulfilled before access is granted to the file or data object. These security attributes are defined at the time of file creation.

Each security attribute is made up of:

- One access mode byte (AMB), that defines the command(s) to be protected against
- One security condition byte (SCB) for each bit set to 1 in the AMB

Each SCB defines the conditions that must be fulfilled in order to allow the corresponding command to be performed.

Note:

1. Security attributes are checked by commands only after personalization.
2. Security Environments do not have security attributes because they cannot be modified.

The security attributes are coded in TLV format.

Contact Vs. Contactless interface

Each file or data object can have one security attribute for the contact interface (tag 8Ch) and/or one security attribute for the contactless interface (tag 9Ch). At least one of these must be present.

Reminder: The security attributes coding (AMB and SCBs) applies only to the application phase.

Access Mode Byte

The access mode byte determines the commands that are to be controlled. Its coding has a different meaning for files and data objects since the commands that are used with them are different.

Caution: In both cases, if a bit in the AMB is set to zero, that is there is no corresponding SCB, then the security attribute for the corresponding command is NEVER.

The AMB is coded as follows:

Table 24. Access Mode Byte Coding – For DFs

Meaning	b8	b7	b6	b5	b4	b3	b2	b1
	0		0	0	0			0
Delete DF (self deletion)		1						
Create DF (applies only to the root)						1		
Create EF							1	

As DFs can be created under the root only, the Create DF condition is used only by the root.

Table 25. Access Mode Byte Coding – For EFs

Meaning	b8	b7	b6	b5	b4	b3	b2	b1
	0		0			0		
Delete EF (current EF)		1						
Activate File				1				

Deactivate File					1			
Update Binary, Erase Binary							1	
Read Binary								1

Bit 8 must be zero for EFs. Bits b7 to b1 are independent of each other, that is, more than one bit can be set to 1.

Table 26. Access Mode Byte Coding – For Data Objects

Meaning	**Type of Data Object**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**
	—	1					0		
Reset Retry Counter	PIN / FP		1						
Change Reference Data	PIN			1					
Verify or **PSO-Hash** and **PSO Compute Digital Signature (DTBS)**	PIN / FP or Private Key				1				
PSO Compute Digital Signature, **PSO: Decipher**	Private Key					1			
Put Data (Update) **Generate PK Pair**	Secret/Private/ Public Key Private Key DH Key Parameters Private Key							1	
Get Data	DH Key Parameters Public Key								1

Bit 8 must be 1 for data objects, which are for secret keys, private and public keys and PINs. Bits b4 and b2 are independent of each other, that is, both can be set to 1.

When bit 5 is used for private keys, it is intended to protect the DSI before performing a signature.

Note: The **Put Data (Update)** and **Generate PK Pair** commands share the same bit, b2. It is possible upon SDO creation to define which of these commands (or both), the access mode bit refers to.

Security Condition Byte

There is one security condition byte (SCB) for each bit set to one in the AMB. Each SCB is coded as follows:

Meaning	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**
No condition (ALWAYS)	0	0	0	0	0	0	0	0
Never	1	1	1	1	1	1	1	1
Conditions	x	x	x	x				
At least one condition (OR) (b7 to b5)	0							
All conditions (AND) (b7 to b5)	1							
Secure Messaging (command & response)		1						
Mutual Authentication			1					
User Authentication (PIN, FP or MS 3DES3 Auth)				1				
Security Environment Reference					x	x	x	x
No SE referenced					0	0	0	0
SE# 0001–1110					—	—	—	—
RFU					1	1	1	1

Bits 8–5 indicate the conditions that must be fulfilled to access the file or data object.

Bit 7 set to 1 indicates that the command and the response must be sent with secure messaging.

Security Attributes Example

The AMB and SCBs are encapsulated together in TLV format under the tag 8Ch for the contact interface security attribute and the tag 9Ch for the contactless interface security attribute. The length depends on the number of SM bytes specified in the AMB. The following table shows an example of how a security attribute could be coded:

Table 27. Security Attributes Coding Example for an EF with Both Interfaces

Tag	Length (bytes)	Value	Meaning
8Ch	06h		Tag and length of the security attribute for contact interface (AMB + 4 SCBs)
		5Bh	AMBs. All five of the SCB for files are present.
		12h	SCB for **Delete File** command. PIN protection defined in SE#2.
		14h	SCB for **Activate File** command. PIN protection defined in SE#4.
		35h	SCB for **Deactivate File** command. **Mutual Authentication** OR PIN protection (using the PIN referenced in SE#5).
		B3h	SCB for **Update Binary** and **Erase Binary** commands. **Mutual Authentication** AND PIN protection (using the PIN referenced in SE#3).
		00h	SCB for **Read Binary** command. No condition.
9Ch	04h		Tag and length of the security attribute for contactless interface (AMB + 3 SCBs)
		19h	AMB. Three SCBs for files are present.
		14h	SCB for **Activate File** command. PIN protection defined in SE#4.
		35h	SCB for **Deactivate File** command. **Mutual Authentication** OR PIN protection (using the PIN referenced in SE#5).
		00h	SCB for **Read Binary** command. No condition.

Caution: Care should be taken when coding security attributes, particularly when they involve security environments and linking

command usage to life cycle status. To illustrate the potential problems, here is an example.

A security attribute is coded so that in SE#4, the ACTIVATE FILE command can be performed only when the life cycle status of the file is INITIALIZED. If the file is later deactivated by a DEACTIVATE FILE command, you will never be able to reactivate it in SE#4 because the status cannot revert to INITIALIZED.

Annex B (informative) Command–Response Pairs

As the command APDU and the response APDU may or may not contain data, there are four cases of command–response pairs as described in ISO 7816–4. These pairs can be summarized as follows:

Case 1: No Input/No Output

APDU Command

CLA	INS	P1	P2	---	-----	---

APDU Response

-----	SW1	SW2

Case 2s: No Input/Output of Expected Length

APDU Command

CLA	INS	P1	P2	---	-----	Le

APDU Response:

Data Field	SW1	SW2

Case 3s: Input/No Output

APDU Command

CLA	INS	P1	P2	Lc	Data Field	---

APDU Response

-----	SW1	SW2

Case 4s: Input/Output of Expected Length

APDU Command

CLA	INS	P1	P2	Lc	Data Field	Le

APDU Response

Data Field	SW1	SW2

For Input/Output of expected length, when a command with response parameters/data is used, the terminal must send a GET RESPONSE command to retrieve the response message.

T=0 Protocol

When using the T=0 protocol, FINEID application returns status bytes of SW1 = 61h, SW2 = Licc (the total number of bytes in the response) and the terminal must send a GET RESPONSE command to retrieve the response message.

Retrieval Sequence for Large Outgoing Data

If Licc >= 256 bytes, FINEID application returns 61h 00h and opens a retrieval sequence as follows:

1. The initial command (status 61h 00h - no data is returned).
2. One or more GET RESPONSE commands.

The following table illustrates the possible responses to the first (and possibly subsequent) Get Response commands where there is more than 256 bytes in the card's buffer):

Table 28. Get Response when Licc > 256 Bytes

Le: Licc	Data Returned	Status Bytes Returned	Sequence Open/Closed
Le = 00h (256 bytes)	256 bytes	61h Licc–256	Open
Le < 256	"Le" bytes	61h Licc–Le	Open

The following table illustrates the possible responses to the final GET RESPONSE command (where there is 256 bytes or less in the card's buffer):

Table 29. Get Response when Licc <= 256 Bytes

Le: Licc	Data Returned	Status Bytes Returned	Sequence Open/Closed
Le > Licc	None	6Ch Licc	Open
Le = Licc	"Licc" bytes	90h 00h	Closed
Le < Licc	"Le" bytes	61h Licc–Le	Open

T=1 and T=CL Protocol

When using the T=1 or T=CL protocol, when Licc is 256 bytes or less, FINEID application returns one of the following to the initial case 4 command:

Table 30. Response to Case 4 Command Licc <= 256 bytes

Licc	Data Returned	Status Bytes Returned	Sequence Open/Closed
Le > Licc	Licc bytes	90h 00h	Closed
Le = Licc	Licc bytes	90h 00h	Closed
Le < Licc	Le	61h XXh (XXh=Licc–Le)	Open

Retrieval Sequence for Large Outgoing Data

If Licc > 256 bytes, then one or more GET RESPONSE commands must be sent. With these protocols the GET RESPONSE command is not sent automatically and must be explicitly sent by the external device (for example a terminal).

A retrieval sequence opened as follows:

1. The initial command (status 61h 00h - 256 bytes of data are returned).
2. One or more GET RESPONSE commands.

If there are still more than 256 bytes in the card's memory buffer, the possible responses to a GET RESPONSE command are:

Table 31. Get Response when Licc > 256 Bytes

Le = XXh	Data Returned	Status Bytes Returned	Sequence Open/Closed
Le = 00h (256 bytes)	256 bytes	61h XXh (XXh=Licc–256	Open
Le < Licc	Le bytes	61h XXh (XXh=Licc–Le) XXh=00h if XX > 256 bytes	Open

If there are 256 bytes or less in the card's memory buffer, the possible responses to a GET RESPONSE command are:

Table 32. Get Response when Licc <= 256 Bytes

Le = XXh	Data Returned	Status Bytes Returned	Sequence Open/Closed
Le > Licc	Licc bytes	90 00h	Closed

Le = Licc	Licc bytes	90 00h	Closed
Le < Licc	Le bytes	61h XXh (XXh=Licc–Le)	Open

Comparison Example for T=0 and T=1

In this example, a PERFORM SECURITY OPERATION–COMPUTE DIGITAL SIGNATURE command (PSO-CDS) with a 4096-bit RSA key, using secure messaging returns 535 bytes of data.

Table 33. T=0 Example Retrieval Sequence when Licc > 256 Bytes

Command	**Data Returned**	**Status Bytes Returned**	**Sequence Open/ Closed**
PSO-CDS (Le=00h)	None	61h 00h (535 bytes remaining)	Open
Get Response (Le=00h)	256 bytes	61h 00h (279 bytes remaining)	Open
Get Response (Le=00h)	256 bytes	61h 17h (23 bytes remaining)	Open
Get Response (Le=17h)	23 bytes	90h 00h	Closed

Table 34. T=1 and T=CL Example Retrieval Sequence when Licc > 256 Bytes

Command	**Data Returned**	**Status Bytes Returned**	**Sequence Open/ Closed**
PSO-CDS (Le=00h)	256 bytes	61h 00h (279 bytes remaining)	Open
Get Response (Le=00h)	256 bytes	61h 17h (23 bytes remaining)	Open
Get Response (Le=17h)	23 bytes	90h 00h	Closed

FINEID application does not support extended length for data retrieval, so the retrieval sequence in T=1 and T=CL protocols is similar to T=0 protocol, except that 256 bytes of data is returned in the initial command.

All Protocols

After a command returns a status of 61xxh, the next command sent to the FINEID application must be a GET RESPONSE command, otherwise the retrieval sequence is

broken. This means the remaining data is lost. The FINEID application processes the "non GET RESPONSE command" as normal.

When the FINEID application receives a subsequent GET RESPONSE command with a length "Le" = Licc, the card must return Licc bytes followed by the SW_OK (0x9000h) status. The retrieval sequence is closed.

When the retrieval sequence is broken or closed, if a GET RESPONSE is sent to the card, the card returns SW_INS_NOT_SUPPORTED (0x6D00h).

When using secure messaging on outgoing data, the MAC is computed on the entire data (concatenation of all the consecutive data fields sent by all the subsequent GET RESPONSE commands). It is sent just before the status SW_OK.

The GET RESPONSE command used to retrieve the data does not use secure messaging, regardless of the security attributes.

Caution: The management of errors during a "retrieval sequence" may differ from one platform to another. The nominal "retrieval sequence" returns all the data prepared by the application. If an error happens in the "retrieval sequence" the terminal may not be able to recover the remaining bytes.