FINEID - S1

Electronic ID Application

v1.12

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

Authors

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

Document history

Version	Date	Editor	Changes	Status
1.12	4.11.2002	AP	Paragraph 5.5.2: PSO: decipher command Lc parameter corrected (from 0x80 to 0x81).	Accepted
1.11	31.10.2002	AP	Minor editorial corrections and updates	Draft
1.1	24.10.1999	MKo	Specification used in production since 1.12.1999.	Accepted

Contents

1. Introduction 1
1.1. Normative references 1
1.2. Informative references 1
1.3. Related FINEID documentation 1
2. Abbreviations 2
3. File structure and contents 2
4. Command interface 2
4.1. SELECT FILE 3
4.2. GET RESPONSE 4
4.3. READ BINARY 4
4.4. VERIFY 5
4.5. MANAGE SECURITY ENVIRONMENT: RESTORE 5
4.6. MANAGE SECURITY ENVIRONMENT: SET 6
4.7. PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE 8
4.8. PERFORM SECURITY OPERATION: DECIPHER 8
4.9. CHANGE REFERENCE DATA 9
4.10. RESET RETRY COUNTER 9
4.11. UPDATE BINARY 10
4.12. ERASE BINARY 10
5. Implementation guidelines for software developers 12
5.1. Resource management 12
5.2. Resetting the card 13
5.3. File selection 13
5.4. Authentication objects 13
5.5. Private key operations (sign and decrypt) 15

1. Introduction

This document describes the command interface and the content of the Finnish Electronic Identification (FINEID) application.

The file structure is based on PKCS#15 v1.0. The command set supported by the card is based on ISO/IEC 7816-4 and ISO/IEC FDIS 7816-8.

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 FDIS 7816-8 (draft)
- PKCS#15 v1.0, Cryptographic Token Information Format Standard, April 23, 1999
- PKCS#15 v1.0 Amendment 1 Draft #1, October 20, 1999

The PKCS Standards are available from

http://www.rsasecurity.com/rsalabs/pkcs/index.html

1.2. Informative references

The following documents have also influenced this specification:

- PKCS#1 v2.0, RSA Cryptography Standard, October 1, 1998
- DIN NI-17.4 v1.0, DIN Specification of chipcard interface with digital signature application/function acc. to SigG and SigV, 15.12.1998
- WAP WIM proposed version 05-Jul-1999, Wireless Application Protocol Identity Module Specification, Part: Security

1.3. Related FINEID documentation

Related FINEID specifications are listed below:

- FINEID S2 – VRK (PRC) CA-model and certificate contents, v2.0
- FINEID S4-1 - Implementation Profile 1 for Finnish Electronic ID Card v1.31
- FINEID S4-2 - Implementation Profile 2 for Organizational Usage, v.1.31
- FINEID S5 – Directory Specification, v2.0

FINEID documentation is available from

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

2. Abbreviations

AID	Application Identifier
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
CRDO	Control Reference Data Object
DF	Dedicated File
EF	Elementary File
FCI	File Control Information
CLA	Class byte
MF	Master File
MSE	Manage Security Environment
PIN	Personal Identification Number
PSO	Perform Security Operation
RFU	Reserved for Future
SE	Security Environment
SW1-SW2	Status Words

3. File structure and contents

The file structure and contents shall be according to PKCS#15 v1.0 standard.

The card 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 PKCS#15 for additional information on the file structure and contents.

4. Command interface

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

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

Table 1. EID application related commands

Command	Standard	Functionality
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 FDIS 7816-8	Restore a predefined (or empty) security environment.
MANAGE SECURITY ENVIRONMENT: **SET**	ISO/IEC FDIS 7816-8	Set the security environment (algorithms, keys) that shall be used in the following PERFORM SECURITY OPERATION commands.
PERFORM SECURITY OPERATION: **COMPUTE DIGITAL SIGNATURE**	ISO/IEC FDIS 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 FDIS 7816-8	Decrypt data with a private key. The algorithm and key are specified with the MSE command.
CHANGE REFERENCE DATA	ISO/IEC FDIS 7816-8	Change the current reference data (e.g. PIN)
RESET RETRY COUNTER	ISO/IEC FDIS 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

4.1. 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).

Table 2. SELECT FILE command APDU

Byte	Value
CLA	0Xh
INS	A4h
P1	00h - select EF, DF or MF by file identifier 04h - select DF by Application IDentifier (AID) 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 = 04h

	- AID value 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 3. SELECT FILE response APDU

Byte	Value
Data	File Control Information
SW1-SW2	Status bytes

4.2. 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 4. GET RESPONSE command APDU

Byte	Value
CLA	0Xh
INS	C0h
P1	00h
P2	00h
Lc	Empty
Data	Empty
Le	Maximum length of data expected in response

Table 5. GET RESPONSE response APDU

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

4.3. READ BINARY

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

Table 6. READ BINARY command APDU

Byte	Value
CLA	0Xh

INS	B0h
P1	XXh - MSB of the offset to the first byte to read (bit 8 = 0), or SFID (bit 8 = 1)
P2	YYh - LSB of the offset to the first byte to read
Lc	Empty
Data	Empty
Le	Number of bytes to read

Table 7. READ BINARY response APDU

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

4.4. 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.

Table 8. VERIFY command APDU

Byte	Value
CLA	0Xh
INS	20h
P1	00h
P2	XXh - PIN reference number (according to PKCS#15).
Lc	Empty or length of subsequent data field
Data	Empty or verification data (padded to the correct length). Padding is done according to PKCS#15.
Le	Empty

Table 9. VERIFY 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) or to check whether the verification is not required (SW1-SW2 = 9000h).

4.5. MANAGE SECURITY ENVIRONMENT: RESTORE

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

Table 10. MANAGE SECURITY ENVIRONMENT: RESTORE command APDU

Byte	Value
CLA	0Xh
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 11. MANAGE SECURITY ENVIRONMENT: RESTORE response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.6. MANAGE SECURITY ENVIRONMENT: SET

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

Table 12. MANAGE SECURITY ENVIRONMENT: SET command APDU

Byte	Value
CLA	0Xh
INS	22h
P1	XY000001b The bits XY specify the operation that relates to the CRDOs in the data field. - X is set = computation - Y is set = decryption
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 13. MANAGE SECURITY ENVIRONMENT:SET response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

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

Table 14. Control Reference Data Objects (CRDO)

Tag	Value	DST	CT
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 PKCS15Path.path	+	+

84h	Key reference (for referencing a private key in asymmetric cases) - used to identify the **key** to be used in cryptographic operations (e.g. if a key file contains multiple keys) - value (single byte) specified in PKCS#15 private key object's PKCS15CommonKeyAttributes.keyReference (optional)	+	+

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 15. 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): 1. 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. 2. 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 v2.0 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 v2.0 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

4.7. 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)

according to the selected algorithm reference value.

Table 16. PSO: COMPUTE DIGITAL SIGNATURE command APDU

Byte	Value
CLA	0Xh
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 or 22h - Hash code. Card encapsulates the hash into DigestInfo structure and pads it internally according to PKCS#1 v1.5 into full modulus length.
Le	Empty or maximum length of data expected in response

Table 17. PSO: COMPUTE DIGITAL SIGNATURE response APDU

Byte	Value
Data	Digital signature
SW1-SW2	Status bytes

4.8. 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.

Table 18. PSO: DECIPHER command APDU

Byte	Value
CLA	0Xh

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 19. PSO: DECIPHER response APDU

Byte	Value
Data	If 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

4.9. 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.

Table 20. CHANGE REFERENCE DATA command APDU

Byte	Value
CLA	0Xh
INS	24h
P1	00h - exchange reference data
P2	XXh - PIN reference number (according to PKCS#15).
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 PKCS#15.
Le	Empty

Table 21. CHANGE REFERENCE DATA response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.10. 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 22. RESET RETRY COUNTER command APDU

Byte	Value
CLA	0Xh
INS	2Ch
P1	00h - reset retry counter and set new verification data
P2	XXh - PIN reference number (according to PKCS#15).
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 PKCS#15.
Le	Empty

Table 23. 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).

4.11. UPDATE BINARY

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

Table 24. UPDATE BINARY command APDU

Byte	Value
CLA	0Xh
INS	D6h
P1	XXh - MSB of the offset to the first byte to update (bit 8 = 0), or SFID (bit 8 = 1)
P2	YYh - LSB of the offset to the first byte to update
Lc	Length of subsequent data field
Data	Data to be updated
Le	Empty

Table 25. UPDATE BINARY response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

4.12. 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.

Table 26. ERASE BINARY command APDU

Byte	Value
CLA	0Xh
INS	0Eh
P1	XXh - MSB of the offset to the first byte to erase (bit 8 = 0), or SFID (bit 8 = 1)
P2	YYh - LSB of the offset to the first byte to erase
Lc	Empty
Data	Empty
Le	Empty

Table 27. ERASE BINARY response APDU

Byte	Value
Data	Empty
SW1-SW2	Status bytes

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 (current file, security status etc.). 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. current file or security status) 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 PKCS#15 for additional information on global PINs.

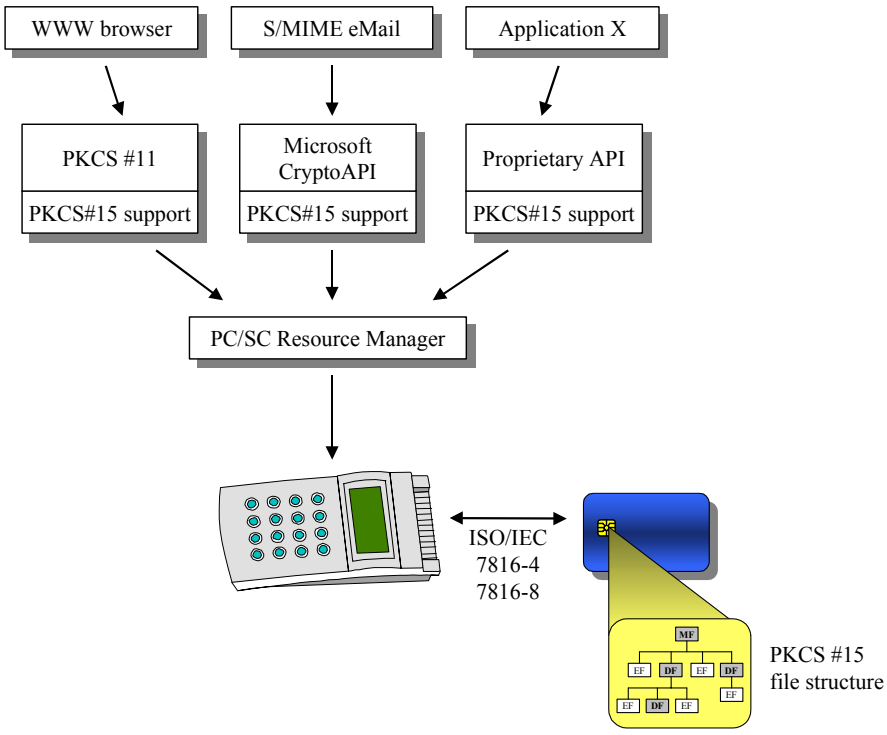

Figure 1. Example scenario of multiple host applications - single card

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.

5.3. File selection

5.3.1. PKCS#15 application

PKCS#15 application is selected using either

- PKCS#15 Application Identifier (AID) specified in PKCS#15 v1.0 or
- path.

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	-

Selection by path specified in EF(DIR) (example path OCTET STRING = 3F 00 11 22 AA BB):

Command	CLA	INS	P1	P2	Lc	Data	Le
SELECT	00	A4	08	00	04	11 22 AA BB (MF file identifier 3F 00 removed)	-

5.3.2. PKCS15Path

PKCS#15 uses PKCS15Path ASN.1 structure to reference various files. The PKCS15Path.path 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 current DF (which is not 3F 00)

Selection of a file using a relative path or a file identifier must be done after having selected the PKCS#15 application first to make sure that the scope for selection is correct. This should be ensured for each transaction. Absolute paths can be selected without first selecting the PKCS#15 application.

5.4. Authentication objects

In PKCS#15 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.

An object can not be protected with multiple authentication objects in PKCS#15. Furthermore, the specific access type (operation on the object) can not be specified. The following table lists the operations that can be protected with authentication objects in the PKCS#15 sense.

Table 28. 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 used in FINEID context**) - verify (PSO: VERIFY DIGITAL SIGNATURE) - encrypt (PSO: ENCIPHER)
Secret key	Secret key operations (**not used 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).

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.

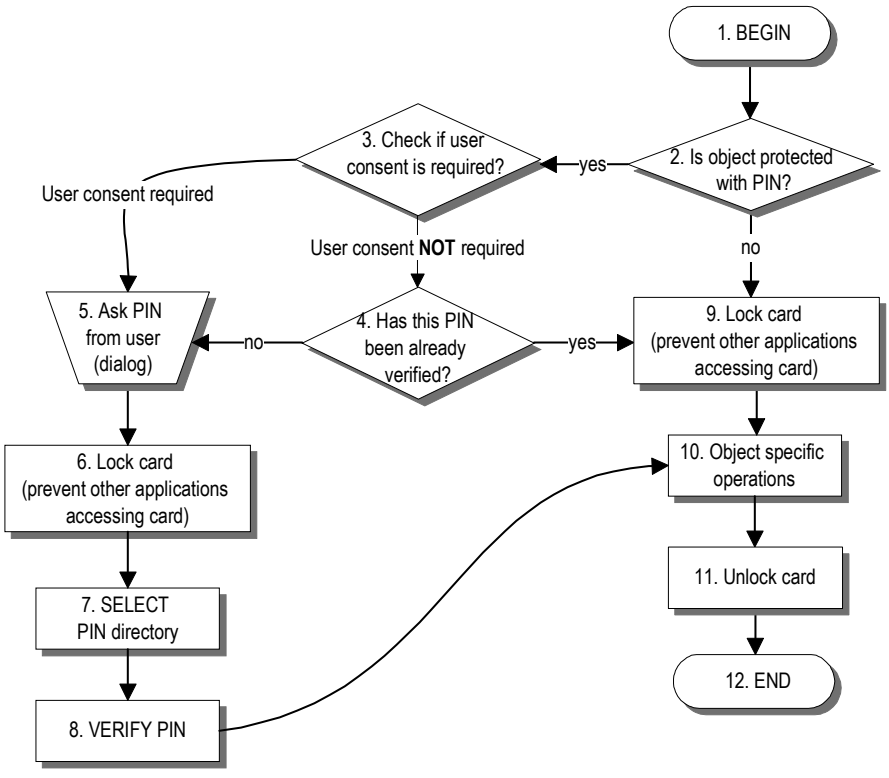

Figure 2. Example of PIN logic

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

Select PIN directory specified in PKCS15PinAttributes.path (example path OCTET STRING = 3F 00 11 22):

Command	CLA	INS	P1	P2	Lc	Data	Le
SELECT	00	A4	08	00	02	11 22 (MF file identifier 3F 00 removed)	-

Verify PIN. Padding is done according to PKCS15PinAttributes (storedLength, padChar). The P2 value is taken from PKCS15PinAttributes.pinReference (example value 01)

Command	CLA	INS	P1	P2	Lc	Data	Le
VERIFY	00	20	00	01	08	31 32 33 34 00 00 00 00 (PIN = 1234 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 (this feature was introduced in PKCS#15 v1.0 Amendment 1 Draft #1). 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 in the Table 28. Objects and protected operations.

5.4.2. Login required flag

In addition to the ‘on demand’ access control of objects in PKCS#15 it is also possible to protect some of the object directory files. The EF(TokenInfo) contains a PKCS15TokenFlags.loginRequired flag indicating that the first authentication object in the AODF is used to protect other object directory files than ODF and AODF (PKCS#15 chapter 7.9 and annex B).

5.5. Private key operations (sign and decrypt)

There may be multiple private keys in the same PKCS#15 card. 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 PKCS#15 DF.

Restore the empty SE:

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

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 (= 1122 from PKCS15PrivateRSAKeyAttributes.value path)
- key reference (= 00 from PKCS15CommonKeyAttributes.keyReference)

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE: SET	00	22	81 computation	B6 DST in data field	0A	80 01 12 (algorithm reference = 12) 81 02 11 22 (private key file identifier) 84 01 00 (key reference)	-

Sign the hash calculated by the host application:

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

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

Get the response in T=0 protocol:

Command	CLA	INS	P1	P2	Lc	Data	Le
GET RESPONSE	00	C0	00	00	-	53 B7 FF 19 A4 … A3 90 8E 4A (e.g. 128 bytes of digital signature if 1024 bit modulus)	XX

5.5.2. Decrypt operation

It is assumed that PIN verification is already done and current DF is the PKCS#15 DF.

Restore the empty SE:

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

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 (= 33 44 from PKCS15PrivateRSAKeyAttributes.value path)
- key reference (= 00 from PKCS15CommonKeyAttributes.keyReference)

Command	CLA	INS	P1	P2	Lc	Data	Le
MSE: SET	00	22	41 decryption	B8 CT in data field	0A	80 01 02 (algorithm reference = 02) 81 02 33 44 (private key file identifier) 84 01 00 (key reference)	-

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) 4B 52 16 … 54 C3 E5 (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 02.

Get the response in T=0 protocol:

Command	CLA	INS	P1	P2	Lc	Data	Le
GET RESPONSE	00	C0	00	00	-	11 22 33 44 55 66 77 88 99 00 11 22 (decrypted payload)	XX