EstEID

ID – card specification

Document information	
Date of Creation	11.06.2012
Recipient	Police and Border Guard Board, Republic of Estonia
Publisher	AS Sertifitseerimiskeskus
Author	Trüb Baltic AS
Version	1.1

Version information		
Date	Version	Changes/notices
02.05.2011	0.1	- History of the changes devised - The addition of the clause „The use of data reading command SelectFile on the card“ and its subclause „Reading the certificate file and its size“.
03.05.2011		- A correction in the clause “Decrypting the session key” with reference to the modification
04.05.2011		- The addition of concepts Hash, Authorization, Verification and Authentication. - The addition of abbreviations ASN.1, ASCII, ECC, SHA-1, DES, 3DES, HEX, BCD and DEC. - The correction of references within the text.
15.07.2011		- The addition of abbreviations TLS, FCI, FCP, FMD. - The correction of references and the addition of links within the text.
31.10.2011		- Changed branding to Trüb Baltic AS - The addition of abbreviations MF, DF, EF, AID. - The addition of chapter “Recognising the application”
03.11.2011		- The correction of references and the addition of links within the text. - Correction of ATR bytes definition.
16.11.2011		- MICARDO related references integrated into this document as Appendix.

Version information		
Date	Version	Changes/notices
21.11.2011		- Fixed paragraph and clause numbering - Added captions to tables and these as well into table of contents.
30.11.2011		- Added descriptions for compatibility issues between different application versions.
19.04.2012	0.9	- Many wording improvements - Reference fixes - Improved figures - Some chapters have been moved as sub-chapters.
10.05.2012	1.0	- Some chapters have been moved as sub-chapters. - Improved command chaining description and added chapter describing separately the basics of command chaining. - Added chapter containing differences between different application versions.
11.06.2012		- Minor fixes: ▪ CMK keys naming in EstEID objects figure. ▪ Application version number corrections. ▪ Note for PIN usage.
30.11.2012		- Minor correction: Chapter 2.1, SHA-518 changed to SHA-512
03.05.2013	1.1	- Chapter 10.1, section 3) note added - Chapter 12.4, section 4) example part removed - Section 13, FID in table corrected: EF –> MF - Section 13.2, section 3) correction of P2 value - “SSL challenge” -> “SSL/TLS challenge” - Chapter 14, section 5) note added - Chapter 15.1, section 5) note added - Chapter 25.1.1 added new comment 6) - Chapter 25.1.2 added new comment 6) - Chapter 25.1.3 added new comment 3)
02.09.2013		- Chapter 3.2, note for v1.1 cards added - Chapter 11, comments added - Chapter 12.4, section 4) example added, note added - Chapter 13.4, note about root certificates added - Chapter 14, note at the end added
12.09.2013		- Table 2.1, version titles streamlined - Chapter 25.1.3, note point 1 amended - Chapert 25.1.3, note point 9 added

To the extent of 90%, the present document is based on the EstEID user manual from 2003. This is allowed by the fact that there are no significant changes made in the EstEID cards, instead only specific improvements and developments to the former card versions.

The aim of the present document is to specify the interface and data content of the security chip of the Estonian national public key infrastructure (EstEID).

It is meant for programmers designing applications communicating directly with EstEID cards or drivers communicating with the card via card reader drivers (for instance, PC/SC, CT-API etc). The document includes detailed descriptions of the use of the card commands.

The present document is not meant or necessary for programmers designing applications using EstEID cards via higher-level drivers (such as Cryptok or Microsoft CSP).

The document requires comprehension of the general principles of smart card communication.

The document employs the hexadecimal numeral system unless specifically marked with the decimal number system abbreviation (dec).

Abbreviations

Abbreviation	Definition
MF	Root directory in the security chip file system (Master File)
ASN.1	A standard used in telecommunication and computer networking to describe, encode, transmit and decode the message (application data unit) transmitted in the network.
APDU	The application protocol data unit of the chip (Application Protocol Data Unit)
ASCII	The standard 7-bit code table to present digitally the English alphabet and other keyboard symbols.
HEX	The symbol for the hexadecimal numeral system
BCD	The presentation of numbers in a way that the first 4 and last 4 bits of each byte could be viewed as separate digits ranging 0 through 9 in the hexadecimal numeral system.
DEC	The symbol for the decimal number system
Card Management Centre	The institution executing the EstEID card administration operations authorized by the chip administrator.
Card administrator	The institution responsible for the execution of the EstEID card administration procedures.
LSB	The least significant bit.
FID	The file identifier
FCI	File Control Information. Data FCP+FMD. (ISO/IEC 7816-4)
FCP	File Control Parameters. The given parameters include a list of logical, structural and security attributes. (ISO/IEC 7816-4)
FMD	File Management Data (ISO/IEC 7816-4)
DES	Data Encryption Standard

Abbreviation	Definition
3DES	Encryption Standard using DES application with three independent keys in EDE (Encipher-Decipher-Encipher) mode.
ICV	The initialization vector of 3DES algorithm
ECC	The alternative for RSA encryption system based on the calculation of the discrete logarithms of elliptic functions.
MAC	Message Authentication Code
SK1, SK2	Session Keys
SSC	Send Sequence Counter
SHA-1	Secure Hash Algorithm
TLS	Transport Layer Security is a cryptographic protocol responsible for communication security over the Internet. TLS encrypts the segments of network connections in the communication channel of the transport layer. TLS is described in IETF standard RFC 5246 based on the former SSL description by Netscape.
TLV	Tag Length Value
PIN	Personal Identification Number
RSA	(Rivest-Shamir-Adleman) algorithm for public key cryptography in data transmission.
DF	Dedicated File
EF	Elementary File
AID	Application Idenifier

Terms

Term	Definition
Authentication	Confirming the origin of somebody or something.
Authorization	The procedure during which it is established whether the given person has the rights for the particular action.
PIN – code	A code consisting of letters and/or digits used to authenticate the user identity (Personal Identification Number)
Passphrase	The objects used in EstEID cards for user authentication which could be employed similarly to PIN1 and PIN2.
Hash	A unique set of bits corresponding to a specific set of data.
Verification	The procedure for verifying the data validity

1. The objects and operations of EstEID security chip application

1.1. The objects of EstEID security chip

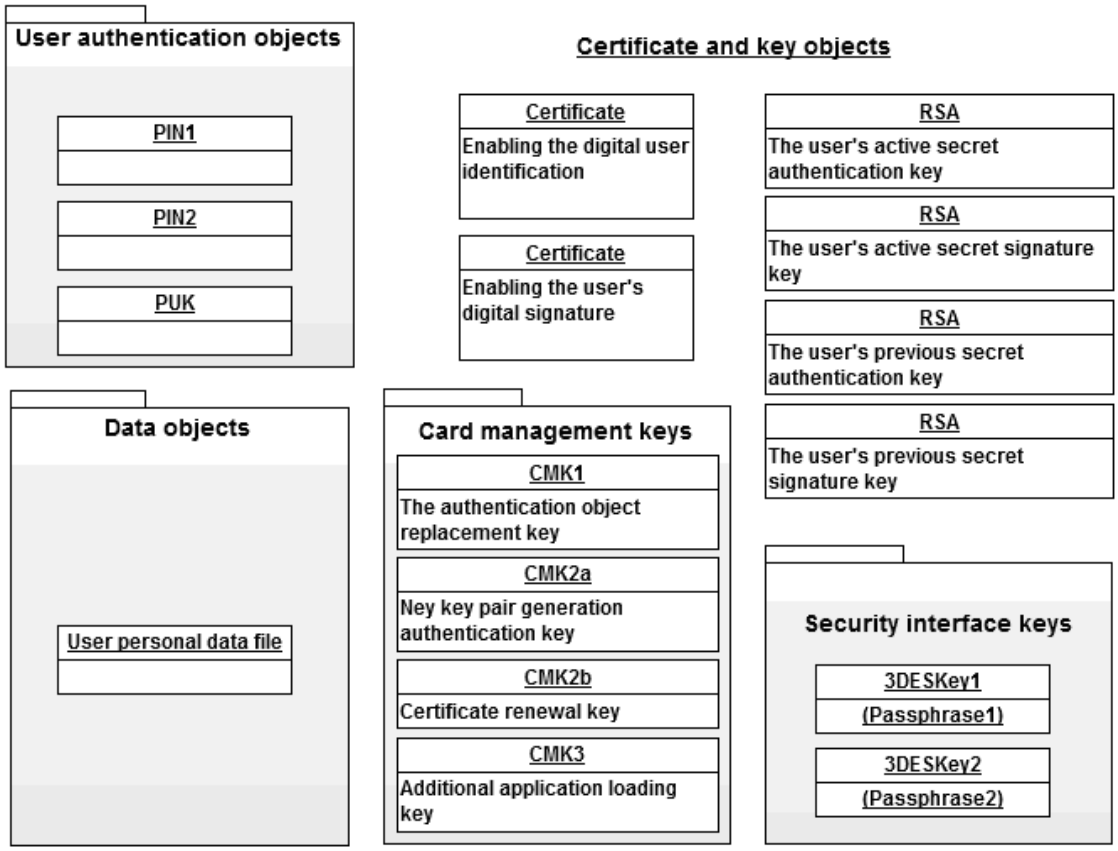

Figure 1: EstEID objects

1.2. The functions of EstEID security chip objects on the card

Table 1-1	The functions of EstEID security chip objects on the card
Object	Function on the card

Table 1-1 The functions of EstEID security chip objects on the card	
Object	Function on the card
PIN1	The authorisation of the card user: 1) for the use of the authentication key 2) for the execution of the following operations: a) the generation of new key pairs b) the loading of certificates c) the loading of additional applications (deletion)
PIN2	The authorisation of the card user: 1) for the use of the signature key 2) for setting the passphrases
PUK	The unblocking of PIN codes when they have been blocked after three consecutive incorrect entries.
3DESKey1	3DES key for secure data transmission in operations with the authentication key
3DESKey2	3DES key for secure data transmission in operations with the signature key
The certificate enabling user identification and secret authentication key.	The electronic card user identification
The user's previous authentication key	The card user's authentication key that was used prior to the generation of the new keys on the card
The certificate enabling the user's digital signature and the secret signature key.	Calculating and checking the card user's electronic signature.
The user's previous signature key	The card user's signature key that was used prior to the generation of the new keys on the card.
User personal data file	Includes the card user's personal data.
CMK1	3DES key which is used to secure the PIN code replacement procedure
CMK2a	3DES key which is used to authorise the new key pair generation.
CMK2b	3DES key which is used to form the secure command series to load the user certificates.
CMK3	3DES key which is used to the secure command series to load additional applications onto the card.

1.3. EstEID security chip operations

EstEID security chip enables the execution of the following operations:

1. The certificate and data reading operations
   a. Reading certificates; Certificate retrieval
   b. Reading the card user personal data file.
2. The administration of the card user authentication objects

a. Changing the values of PIN1, PIN2 and PUK;

b. Resetting the consecutive incorrect entries of PIN1 and PIN2;

c. Assigning values to 3DESKeys.

3. Card user authentication

   a. card user authentication with PIN1, PIN2 and PUK;

   b. card user authentication with 3DESKey1 and 3DESKey2.

4. Operations with secret keys

5. Card management operations

   a. Replacing authentication objects;

   b. Generating new key pairs;

   c. Loading certificates;

   d. Loading and deleting additional applications;

   e. Forming secure loading command series.

2. EstEID card principles

2.1. The used encryption algorithms and key lengths

EstEID card application version 3.0 supports operations with the following keys:

Table 2-1 EstEID card application versions' properties

	v1.0	v1.1 (DigiID)	v3.0	v3.4
Implementation platform	MICARDO	Multos	Java Card	
RSA module length	1024 bits		2048 bits	1024, 1280, 1536, 1984, 2048 bits
Hash algorithm support	SHA-1, SHA-224	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		

Table 2-1 EstEID card application versions' properties				
	v1.0	v1.1 (DigiID)	v3.0	v3.4
ECC module lengths	Not supported		The RSA key length of the same security level 160 bits ~1024 bits 192 bits ~1536 bits 224 bits ~2048 bits 256 bits ~3072 bits	
PKCS#1 padding support	v1.5			
Protocol support	T=0, T=1	T=0	T=0, T=1	T=0, T=1

Note: Module lengths of 1280, 1536 and 1984 are not actively used in case of EstEID v3.4 cards.

Note: ECC keys are not actively used in case of EstEID cards.

2.1.1. EstEID chip instruction set

The present EstEID version has been realized on Java platform

The card interface commands referred to in the specification may not be the only way to execute the given operation.

- EstEID card application version can be determined by comparing the ATR strings issued by the card.

2.1.2. Supported algorithms

- v1.0 and v1.1 support PKCS1 padding and SHA1 hashes with 1024bit RSA keys.
- v3.0 and further supports PKCS1 padding and SHA1, SHA-224 (not used as PKCS#11 does not support SHA-224 in v2.20) and SHA-256 hashes with 2048bit RSA keys

2.2. Card features

The EstEID card may simultaneously include up to:

- 8 authentication keys
  - 2 actively used and 1 active at a time in case of EstEID v1.0
  - 1 actively used and active at a time in case of other application versions
- 8 signature keys
  - 2 actively used and 1 active at a time in case of EstEID v1.0
  - 1 actively used and active at a time in case of other application versions
- 35 certificates or other data files
  - 2 actively used and active at a time (one for authentication and transport encryption and one for digital signing)

3. Card recognition

3.1. ATR strings and used protocols

The card recognition uses the ATR string issued by the card during the initial loading.

EstEID card has two ATR strings:

- So-called "cold ATR" that is issued by the card after the insertion into the reader
- So-called “warm ATR” that is issued by the card after the booting initiated by the card reader.

EstEID card application has been realized on various platforms. All application versions are compatible and with the same functionality, however, there are still some differences due to the use of various platforms that should be considered in interfacing EstEID cards.

All platforms have different ATR strings:

- The ATR strings of EstEID application realized on MICARDO cards are as follows:
  - EstEID v1.0 realised on Micardo Public 2.1

    Cold ATR: 3B FE 94 00 FF 80 B1 FA 45 1F 03 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30 43**

    Warm ATR: 3B 6E 00 FF **45 73 74 45 49 44 20 76 65 72 20 31 2E 30**
  - EstEID v1.0 realised on Micardo Public 3.0, issued since January 2006

    Cold ATR: 3B DE 18 FF C0 80 B1 FE 45 1F 03 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30 2B**

    Warm ATR: 3B 5E 11 FF **45 73 74 45 49 44 20 76 65 72 20 31 2E 30**
- The ATR strings of EstEID v1.1 for DigiID realised on MultOS (TM) by KeyCorp on IE4 cards are as follows:

  Cold ATR: 3B 6E 00 00 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30**

  Warm ATR: 3B FE 94 00 FF 80 B1 FA 45 1F 03 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30 43**

(same as "EstEID v1.0 cold" above)

Important TIP: DigiID doesn't support T=1 protocol. Always must be used protocol T=0.

- The ATR strings of EstEID application realized on Java card are as follows:
  - EstEID v3.0 (recalled 46 cards)

    Cold ATR: 3B FE 18 00 00 80 31 FE 45 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30 A8**

    Warm ATR: 3B FE 18 00 00 80 31 FE 45 **80 31 80 66 40 90 A4 16 2A 00 83 01 90 00 E1**

  - EstEID v3.0 (18.01.2011) and v3.4

    Cold ATR: 3B FE 18 00 00 80 31 FE 45 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30 A8**

    (same as the 46 recalled v3.0 cards above)

    Warm ATR: 3B FE 18 00 00 80 31 FE 45 **80 31 80 66 40 90 A4 16 2A 00 83 0F 90 00 EF**

When interfacing EstEID cards, it is recommended to prefer T=0 protocol for its less complicated structure.

The meanings of ATR string bytes have been defined in standard ISO 7816-3.

Bytes 0..8 in cold and warm ATR include the card communication protocol parameters. The parameters have no significance to the applications that communicate with the card via card reader drivers (for instance, PC/SC, CT-API) as this is handled by the driver.

Significant bytes are bytes counting from B in cold and warm ATR. These are so-called "historical bytes" that the application uses to identify the card type.

The historical bytes in cold ATR include the name of the card, in warm ATR information as defined in ISO/ICE 7816-4 standard.

On EstEID card, these bytes for cold ATR are as follows:

EstEID␣ver␣1.0

with ␣ marking the space character.

Note: *ATR historical bytes have not been standardised (ISO 7816-4 attempts to standardise the given bytes, however, it has not been used in practice). Thus, the fact that the smart card can represent itsself as EstEID card by return an ATR similar to the EstEID one. Furthermore, other smart cards may be programmed in a way that they operate similarly to EstEID card. The electronic attribute of the EstEID card is the secret keys on the card with its public side certified by the national certification centre and its visual attributes are the personalised security elements and the card holder's data. However, it may be stated that in case the card's ATR does not correspond to the EstEID card's ATR, it is not a EstEID card.*

The version number has not been changed in ATR as the application is compatible with the former version. However, the other parts of ATR have been changed and thus also the offset of the beginning of historical bytes.

In order to secure the smooth operation of EstEID card with as many PC/SC interface readers as possible, then in case of cards with application version v1.0 and v1.1, the value of TA3 parameter (the maximum length of data field) has been assigned as $250_{(dec)}$. In case of cards with application version v3.0 and 3.4 the value of TA3 parameter is $254_{(dec)}$.

3.2. Recognising the application

EstEID application is a default selected application on the card. For EstEID application recognition SELECT FILE command must be used (see chapter 9 „Using the data reading command SelectFile “).

- v3.0 and further cards must be sent SELECT FILE command:

CLA	INS	P1	P2	Lc	Application ID (AID)
00	A4	04	00	0E	F0 45 73 74 45 49 44 20 76 65 72 20 31 2E 30 (ASCII: "ðEstEID ver 1.0")

Card must respond with OK trailer: 0x9000. Otherwise the application selection failed with given AID.

Note: v1.1 and v3.0 cards answer with 0x9000 to any application selection command, regardless of whether there is an application with the given AID value in the card or not. See chapter 25.1"Differences between card applications" for more information.

- For cards below v3.0 must be used following SELECT FILE command:

CLA	INS	P1	P2	Lc	Application ID (AID)	Le
00	A4	04	00	10	D2 33 00 00 01 00 00 01 00 00 00 00 00 00 00 00	27

V 1.0 card must respond with 0x6127, which means that there are 0x27 bytes waiting to be read on from card. The reading can be done with GET RESPONSE command. If card responds something else then the application selection failed.

V1.1 card (Digi-ID) must respond with OK trailer 0x9000, otherwise the application selection fails.

4. The application data requests

4.1. The identification of the card application version

Card applications with v3.0 and v3.4 can be identified by the application version.

In order to identify the card application version, the GET DATA command must be sent to the card:

CLA	INS	P1	P2	Le
00	CA	01	00	02

The card with EstEID application version 1.1 issues:

Version number, most significant byte…least significant	OK trailer	
01 01	90	00

The card with EstEID application version 3.0 issues:

Version number, most significant byte...least significant	OK trailer	
03 00	90	00

The card with EstEID application version 3.4 issues:

Version number, most significant byte...least significant	OK trailer	
03 04	90	00

Note: *In case of EstEID v3.4 the card responds as well with data 0x0300.*

Note: *Only cards with version v1.1, v3.0 and v3.4 respond to the command with an application version number and OK trailer. Cards with application version v1.0 respond with 0x6D00 "Instruction code not supported". See chapter 25.1"Differences between card applications" for more information.*

4.2. The card's free memory query

For the given task the following command must be sent to the card:

CLA	INS	P1	P2	Le
00	CA	01	01	02

Only EstEID cards with application version 3.0 and 3.4 issue:

Free memory, most significant byte first	OK trailer	
NN NN	90	00

Important TIP: EstEID cards below v3.0 respond to given command with status 0x6D00.

Note: *In case the capacity is higher than 0x7FFF bytes, the card replies as well with 0x7FFF.*

Note: *The card's free memory query is not actively used in case of EstEID cards.*

5. T=0 or T=1

The standard ISO 7816-3 and its Appendix 1 include two protocols for the communication with smartcards:

1) T = 0
2) T = 1

The difference between the protocols must be considered not only inside the lower level driver but also in the application communicating, for instance, on the PC/SC level.

EstEID card supports both protocols. After the initial loading, the card activates T = 0 protocol. After the initial loading the application using the card may switch the card over to T = 1 protocol. In PC/SC there are no convenient opportunities for the execution of such an operation, however, the PC/SC drivers of some readers can handle it after they have been activated to communicate via T = 1 during initialisation.

The differences between T = 0 and T = 1 protocols must also be considered in case of commands where both the command to the card and the card's response include data.

Such commands are:

- file reading from the card
- calculating the response to SSL/TLS-challenge
- calculating the signature
- decrypting the session key.

Neglecting the consideration of the protocol feature leads to command error situations.

As in T = 0 protocol, all operations may be executed with the EstEID card, there is generally no need for T = 1, although all operations may also be executed under this protocol.

Important TIP: **EstEID cards with v1.1 support only T=0.**

However, in programming for T = 0 it is important to consider that the card may respond to the command with the code "61 L" as a positive response. L (one byte) here shows how many bytes of data are waiting for the release on the card. In order to read the given bytes, a new command must be given to the card:

CLA	INS	P1	P2	Le
00	C0	00	00	L

For example: The card replied to the command with the codes "61 0A" meaning that there are 10 (dec) bytes waiting on the card. In order to read the bytes, we send:

CLA	INS	P1	P2	Le
00	C0	00	00	0A

The card responds:

Released bytes	OK trailer
00 00 00 00 00 00 00 00 00 00	90 00

6. PIN1, PIN2, PUK code

PIN1, PIN2 and PUK consist of digits '0'..'9' (however, the card supports using all of the ASCII symbols in PIN and PUK codes with values ranging from 0x00 to 0xFF).

Note: *Although not restricted by the card application only the following subset of ASCII character set shall be used: 0, 1, 2, ...9. Other characters are not supported by the use cases.*

The following table includes the allowed length for PIN and PUK codes:

Table 6-1	Allowed lenghts for PIN1, PIN2 and PUK codes	
	Minimum length (digits)	Maximum length (digits)
PIN1	4	12
PIN2	5	12
PUK	8	12

EstEID card cannot handle the length for PIN or PUK which exceed the limit given in the table.

The length of PIN and PUK codes may vary, the applications communicating with the card must support PIN codes of varying length.

PIN codes are transferred onto the card in ASCII format and cannot be read from the card.

The card user authentication with PIN1, PIN2 or PUK is conducted by the command VERIFY. The given operation may be executed without restrictions.

Note: *v3.0 cards allow the same values for old and new PIN and PUK codes when changing a PIN or PUK code (with the command CHANGE REFERENCE DATA) or when unblocking and simultaneously changing a PIN code (with the command RESET RETRY COUNTER). See chapter 25.1.2 "Differences of application version v3.0."*

Note: *The following differences between card applications should be taken into account when using the command VERIFY:*

- *v3.0 cards respond with 0x630X trailer instead of 0x63CX when trying to verify PIN or PUK code with an incorrect value (with the command VERIFY when the PIN or PUK code's length is in its allowed value range).*
- *In case of an incorrect PIN or PUK input length used in the command (if the length is not in the allowed range), the card is expected to respond with 0x63CX error trailer and decrease the respective retry counter's value. However, version 3.0 cards respond with 0x6984 and version 1.0 cards respond with 0x6A80 ("Incorrect data in command field") and also decrease the retry counter's value.*

- *Version 1.1 and 3.0 cards respond to key operations (like INTERNAL AUTHENTICATE), if the required PIN has not been verified, with 0x6985 "Conditions of use not satisfied" instead of 0x6982 "Security status not satisfied".*

See chapter 25.1"Differences between card applications" for more information.

7. Changing PIN1, PIN2 and PUK code

The values of PIN1, PIN2 and PUK codes may always be changed with the command CHANGE REFERENCE DATA.

PIN1 and PIN2 may be given new values by using command RESET RETRY COUNTER[1]. The execution of the given operation requires the verification of PUK or the execution of the operation secured by 3DESKey for PIN1 and by 3DESKey2 for PIN2.

Note: v3.0 cards allow the same values for old and new PIN and PUK codes when changing a PIN or PUK code (with the command CHANGE REFERENCE DATA) or when unblocking and simultaneously changing a PIN code (with the command RESET RETRY COUNTER). See chapter 25.1.2 "Differences of application version v3.0" for more information.

The change of PIN1, PIN2 and PUK code is executed by the following command:

CLA	INS	P1	P2 (PIN-code number)	Lc	Previous PIN or PUK	New PIN or PUK
00	24	00	PUK = 00 PIN1 = 01 PIN2 = 02	The length of the previous PIN + length of the new PIN	As ASCII bytes	As ASCII bytes

For example: Let the previous PIN1 be "1234" and the new PIN1 "54321". We will change the PIN1-code with the following command:

CLA	INS	P1	P2	Lc	Previous PIN1	New PIN1
00	24	00	01	09	31 32 33 34	35 34 33 32 31

For example: Let the previous PUK be "12345678" and the new PUK "1234567890". We will change the PUK code with the following command:

CLA	INS	P1	P2	Lc	Previous PUK	New PUK
00	24	00	00	12	31 32 33 34 35 36 37 38	31 32 33 34 35 36 37 38 39 30

[1] RESET RETRY COUNTER command must be used for replacing the blocked PIN codes. Look chapter 8.2 "Unblocking the PIN code by replacing the blocked PIN code with a new one"

8. Unblocking PIN1 and PIN2 codes

In EstEID cards PIN1, PIN2 and PUK codes have retry counters with the initial value 3. In other words, after three consecutive incorrect entries the codes will be blocked.

Each code has a separate counter, the incorrect insertion of one code does not change the counters of other codes (i.e. inserting PIN1 incorrectly does not change the counter for PIN2 and PUK incorrect insertions and does not block them).

The blocked PIN1 and PIN2 may be unblocked with the PUK code. The retry counter may be reset with the command RESET RETRY COUNTER. The execution of the given operation requires the verification of PUK or the execution of the operation secured by 3DESKey for PIN1 and by 3DESKey2 for PIN2.

Both PIN codes may be unblocked by the PIN code replacement procedure. Given procedure can be found from Chapter 19.6.1 “PIN code replacement procedure”.

PIN codes of EstEID card may be unblocked using two methods by using RESET RETRY COUNTER command:

1) leaving the blocked PIN code the same

2) simultaneously replacing the blocked PIN code with a new one.

The PUK code is similarly blocked after three consecutive incorrect insertions. The card with the blocked PUK code could be reactivated by the PIN code replacement procedure. The blocked PUK code cannot be unblocked by the card user, the given procedure can be made only by the service provider.

8.1. Unblocking the PIN code by leaving the code unchanged

The PIN code may be unblocked by leaving the code unchanged with the following commands:

1) Verifying the PUK code:

CLA	INS	P1	P2	Lc	PUK
00	20	00	PUK code number = 00	PUK code length	As ASCII codes

2) Unblocking the PIN code:

CLA	INS	P1	P2
00	2C	03	PIN code number, PIN1 = 01 PIN2 = 02

For example: Let the PUK be "12345678" and the blocked PIN1. In order to unblock we send:

1) Verifying the PUK code:

CLA	INS	P1	P2	Lc	PUK
00	20	00	00	08	3132 33 34 35 36 37 38

2) Unblocking the PIN1 code:

CLA	INS	P1	P2
00	2C	03	01

8.2. Unblocking the PIN code by replacing the blocked PIN code with a new one

The PIN code may be unblocked by replacing the previous code with a new one with the RESET RETRY COUNTER command:

CLA	INS	P1	P2	Lc	PUK	New PIN
00	2C	00	PIN code number, PIN1 = 01 PIN2 = 02	PUK code length + new PIN length	As ASCII codes	As ASCII codes

For example: Let the PUK be "12345678" and blocked PIN1, which we want to replace with the value "1234" after unblocking.

In order to unblock we send:

CLA	INS	P1	P2	Lc	PUK	New PIN1
00	2C	00	01	0C	31 32 33 34 35 36 37 38	31 32 33 34

Note: In case the PIN code has not been blocked (i.e. it has not been inserted incorrectly on three consecutive occasions), the EstEID card responds to the unblock command with an error trailer. Cards with application version v1.0 and v3.4 respond with error trailer 0x6985; versions v1.1 and v3.0 respond with error trailer 0x6A86.

Note: v3.0 cards allow the same values for old and new PIN codes when unblocking and simultaneously changing a PIN code (with the command RESET RETRY COUNTER). See chapter 25.1.2 "Differences of application version v3.0."

9. Using the data reading command SelectFile

In order to read the data from EstEID card, the command SelectFile is used. The given command may be sent to the card in four ways:

CLA	INS	P1	P2	Command description
00	A4	0X	00	The card response includes FCI (FCP+FMD)
00	A4	0X	04	The card response includes FCP
00	A4	0X	08	The card response includes FMD
00	A4	0X	0C	There is no response to the command from the card (only OK trailer: 0x9000)

As the response to the SelectFile command does not include significant information, it is recommended to use the command format 00 A4 0X 0C.

Note: *v3.0 cards respond with a double 0x62 FCP tag to SELECT FILE commands which select an elementary file and require FCP value to be included in the response (i.e. the first parameter's value is set to P1=02 and the second parameter is set to P2=04). This should be taken into account when parsing the response. See chapter 25.1.2 "Differences of application version v3.0."*

In the given command table symbol "X" marks parameter, which defines the type of data structure to be selected. These data structures are:

Value of X	Description
0	Application Master File (MF) (no data in APDU data field)
1	Application Dedicated File (DF) (APDU data field contains DF identifier)
2	Application Elementary File (EF) (APDU data field contains EF identifier)
4	Card Application (APDU data field contains Application identifier (AID))

10. The card user personal data file

EstEID card includes a personal data file with the personal data of the card user and with FID (*File Identifier*) 'PD' or '50 44'. This is a variable-length record file including the electronic information (except photo and signature image) transferred onto the card during the personalisation process.

The file may be read by the personal data file reading procedure with the aim of transferring the card user data in electronic form:

- in case the card has not been loaded a certificate;
- in case the certificate is not processed by the host application;
- to form new certificate requests.

The card user personal data file is a variable-length formatted file. The maximum length of the file is 200_{hex} (512_{dec}) bytes, the maximum record length is 50_{dec} bytes and the number of records 16_{dec}. The file is completed during the personalisation process and the recorded data cannot be modified later.

There are technical possibilities to supplement the file during the personalisation process by adding records at the end of the file. Similarly the maximum length of the record may be changed.

The file includes the records given below. The symbols have been encoded according to ANSI code page no 1252 in keeping with ISO 8859-1.

Table 10 1	Personal Data File Contents	
Record number	content	Maximum length
1	Surname	28_d bytes
2	First name line 1	15_d bytes
3	First name line 2	15_d bytes
4	Sex	1_d bytes
5	Nationality (3 letters)	3_d bytes
6	Birth date (dd.mm.yyyy)	10_d bytes
7	Personal identification code	11_d bytes
8	Document number	8_d or 9_d bytes
9	Expiry date (dd.mm.yyyy)	10_d bytes
10	Place of birth	35_d bytes
11	Date of issuance (dd.mm.yyyy)	10_d bytes
12	Type of residence permit	50_d bytes
13	Notes line 1	50_d bytes
14	Notes line 2	50_d bytes
15	Notes line 3	50_d bytes
16	Notes line 4	50_d bytes

Note: *Application versions v3.0 and v3.4 have card numbers with the length of 9 bytes (instead of 8 bytes). This should be taken into account when reading the "Document number" record from the personal data file. See chapter 25.1.2 "Differences of application version v3.0" for more information.*

Note: *v3.0 cards have the personal data file (MF/EEEE/5004) records padded with spaces which should be taken into account when parsing the records. See chapter 25.1.2 "Differences of application version v3.0" for more information.*

10.1. Reading data from the personal data file

The personal data file content is always readable, authentication is not required.

In order to read the information, do the following:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select the catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Select the file 5044 with the command SELECT FILE. The command is given so that the EstEID card responds with the file header:

In case of T=0 we send:

CLA	INS	P1	P2	Lc	FID
00	A4	02	04	02	50 44

In case of T=1 we send:

CLA	INS	P1	P2	Lc	FID	Le
00	A4	02	04	02	50 44	00

The length of the file header is generally not known. Thus we give Le byte with the value 0.

The card responds:

Byte no	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	...
Value	62	17	82	05	04	41	00	32	10	83	02	50	44	85	02	01	00	

11	12	13	14	15	16	17	18
8A	01	05	A1	03	8B	01	01

The maximum length of the record is in byte no 7 and the number of records in byte no 8 (NB! hexadecimal numeric values).

Note: *EstEID card stores the information in file headers in the form of TLV (Tag Length Value) objects. The maximum length of the record is in byte 3 (counting starts from one) of the object 82 of the compound data object 62 and the number of entries in byte 4. The correct procedure would include parsing the file header content to find the maximum record length and the number of entries. However, as the file header in EstEID card (presumably) will not change, the reading of 7th and 8th byte would also (presumably) yield the correct result.*

Note: *v1.1, v3.0 and v3.4 cards respond with a different TLV value than v1.0 cards. V1.1 cards have the byte no 8 set to $0A_{hex}$; v3.0 and v3.4 cards have bytes no 7 and 8 set to $7F_{hex}$ and 14_{hex}. However, all card versions have the maximum record length 32_{hex} bytes and number of records 10_{hex} . See also chapter 25.1 Differences between card applications for more information.*

4) Read record one in the file with the command READ RECORD.

5) We read record no 1 from the file with the command READ RECORD:

CLA	INS	P1 (record number)	P2	Le
00	B2	01	04	00

In case protocol T=0 is used, Le may also be omitted. Le is 0 as the record length is not known.

In case protocol T=0 is used, the card responds:

The card has bytes waiting	The number of bytes waiting (length of surname)
61	...

Read the surname from the card:

CLA	INS	P1	P2	Le
00	C0	00	00	Length of surname

The card response (using T=1 protocol, we immediately get the following response):

Surname	OK Trailer
…	90 00

Other records are read in a similar manner.

11. Card user's secret keys

EstEID card has two secret keys that are respectively related to the certificate enabling digital identification of the card user and the certificate enabling the digital signature by the card user.

The secret keys are generated:

1. initially during the card personalisation (within the card)
2. during the new key pair generation procedure (also within the card)

The secret keys cannot be read from the card. The key length is 2048 bits, for v1 and v2 cards it is 1024 bits and EstEID saves the key in CRT format.[2]

The public exponent of the secret keys is generated as a random value, except for v3.4 cards where the exponent value could be also fixed value 65537.

The keys are related to the use counter with the initial value 0xFFFFFF which is decreased by one after each operation executed with the key.

As the given keys are not visible to external applications, the present document shall not discuss their location in the card.

12. Reading counters in the card

EstEID card has the following counters:

1) PIN1 retry counter
2) PIN2 retry counter
3) PUK retry counter
4) Passphrase 1 (3DESKey1)[3] retry counter
5) Passphrase 2 (3DESKey2) retry counter
6) Signature key version 1[4] use counter
7) Signature key version 2 use counter
8) Authentication key version 1 use counter
9) Authentication key version 2 use counter

PIN and PUK code retry counters are in the card's root directory Master File with the FID (File Identifier) 0016. The file record 1 corresponds to PIN1, record 2 to PIN2 and record 3 to PUK.

The passphrases are related to counters similar to PIN code retry counters (for further information on PIN code counters see Chapter 8 "Unblocking PIN1 and PIN2 codes"). Their values are located in records 4 (Passphrase 1 counter) and 5 (Passphrase 2 counter) in file MF/0016.

The passphrase retry counters operate on principles analogous to those of PIN code retry counters:

- The initial value of counters is 3;
- Each unsuccessful operation with a passphrase reduces the counter value by 1;
- A successful operation resets the initial value of the counter;
- A successful change of the passphrase resets the initial value of the counter.

[2] For different supported key lengths look chapter 2.1 „The used encryption algorithms and key lengths".

[3] Passphrases are the objects used in EstEID cards for the user authentication that could be employed similarly to PIN1 and PIN2. A more detailed use of the passphrases is given in the following part of the manual.

[4] EstEID card enables (after certain authentication operations) the generation of new key pairs. The previous key pairs are stored. The generation of new key pairs will be discussed below. After personalisation (and until the generation of new key pairs), version 1 of the keys will be used.

12.1. Reading the key types, module lengths and use counters on the card

File MF/EEEE/0013 includes altogether 16 records each of which related to one key:

Note: *In case of EstEID cards with version v1.0, there are 2 authentication keys and 2 signature keys actively used on the card. In case of other application versions, only 1 authentication key and 1 signature key is actively used.*

Table 12-1 File EEEE/0013 key records

The record number of file MF/EEEE/0013	Name of certificate	Key reference
1	Signature key 1	0x01
2	Signature key 2	0x02
3	Authentication key 1	0x11
4	Authentication key 2	0x12
5	Signature key 3	0x03
6	Signature key 4	0x04
7	Authentication key 3	0x13
8	Authentication key 4	0x14
9	Signature key 5	0x05
10	Signature key 6	0x06
11	Authentication key 5	0x15
12	Authentication key 6	0x16
13	Signature key 7	0x07
14	Signature key 8	0x08
15	Authentication key 7	0x17
16	Authentication key 8	0x18

The record of file /EEEE/0013 includes the following information on the key:

Table 12-2 EEEE/0013 key record descriptors		
Byte no (dec, counting from zero)	RSA key	ECC key
2	Key reference	
6	0xC0	0xE0
9	Module length*	(no significance)
12..14	Key use counter	
32	Key type and status**	
60	(no significance)	0x06
61	(no significance)	ECC group's OID length (bytes)
62..	(no significance)	ECC group's OID

* Byte value = (module length in bits – 1024) / 32.

** Byte values are as follows:

0xF6 – Signature key, not generated

0xE4 – Authentication key, not generated

0xB6 – Signature key, ready for use

0xA4 – Authentication key, ready for use

12.2. Finding the retry counter values

In order to find the retry counter values, do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select file 0016 with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID

00	A4	02	0C	02	00 16

3) Read record 1 of the file.

CLA	INS	P1 (record number)	P2	Le
00	B2	01	04	00

As file 0016 is a variable length record file, the amount of bytes to be read from the card cannot be predicted. Thus we give Le byte with the value 0. Using T = 0 protocol, Le may be omitted. In case T = 1, Le byte must be used.

The record is read from the card:

Byte no.	0	1	2	3	4	5	6	7	8	9
Value	80	01	03	90	01	03	83	02	00	00

PIN1 retry counter is in byte no 5.

NB! The initial value of retry counter is 3 and on the blocked card 0, thus the counter shows how many attempts to enter the PIN code are left.

Note: *EstEID card stores the information in file records in the form of TLV (Tag Length Value) objects. The PIN retry counter is in object tagged 90 and with length 01. The correct procedure would include parsing the file content to find the counter value. However, as the form of the record (presumably) will not change during the period of card usage, the reading of the 5th byte would also (presumably) yield the correct result.*

4) We use analogous command to read record 2, which includes the PIN2 retry counter:

CLA	INS	P1 (record number)	P2	Le
00	B2	02	04	00

5) In order to read the PUK retry counter, we read record 3:

CLA	INS	P1 (record number)	P2	Le
00	B2	03	04	00

The read record is as follows:

Byte no	0	1	2	3	4	5
Value	80	01	03	90	01	03

The retry counter is in byte no 5.

The passphrase retry counters are in Master File 0013 records 5 (passphrase 1) and 6 (passphrase 2).

12.3. Reading the passphrase retry counter

In order to read the passphrase 1 retry counter, do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Select file 0013 with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	02	0C	02	00 13

4) Read record 5 of the file:

CLA	INS	P1 (record number)	P2	Le
00	B2	05	04	00

File 0013 is also a variable length structured file. Using T = 0 protocol, Le may be omitted. In case T = 1, Le byte must be used.

The record is read from the card:

Byte no.	0	1	2	3	4	5	6	7	8	9	A	... total length of record is 28(hex) barite
Value	83	02	04	00	C1	02	81	10	90	01	FF	...(bytes)

The passphrase retry counter is in byte no A. Its initial value is FF and each incorrect passphrase insertion reduces the counter by one. When the counter reaches zero, the passphrase cannot be used any longer and the card must be authenticated with PIN codes or it must be exchanged. The passphrase retry counter does not reset with the correct passphrase insertion.

Passphrase 2 retry counter is analogously in record 6.

12.4. Reading secret key use counters

All secret keys on the card have their own use counter. This chapter describes how to read secret key use counter from the card.

Secret key use counters are in the following records of file 0013 of the catalogue EEEE:

Key version	Record number of file MF/EEEE/0013
Signature key version 1	1
Signature key version 2	2
Authentication key version 1	3
Authentication key version 2	4

In order to learn how many times the signature key version 1 has been used, do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select the catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Select file 0013 with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	02	0C	02	00 13

4) Read record 1 of the file:

CLA	INS	P1 (record number)	P2	Le
00	B2	01	04	00

File 0013 in the catalogue is a fixed-length structured file (record length 4F), however, the same rule applies: Using T = 0 protocol, Le may be omitted. In case T = 1, Le byte must be used.

The record is read from the card:

Byte no	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	...
Value	83	04	01	00	10	01	C0	02	81	80	91	03	FF	FF	FF	

The key use counter is in bytes no C..E as a 3-byte value with the older byte on the left. Thus, the counter is read as follows:

<Byte C Value >*10^4 + < Byte D Value >*10^3 + < Byte E Value >

The initial value of key use counter is FFFFFF and each key usage reduces it by one. In order to learn how many times the key has been used, the counter value read from the card must be subtracted from FFFFFF.

Note: *If the secret key's use counter reaches zero then the counter is reset to its initial value 0xFFFFFF.*

The number of usages of other keys is read from the card analogously.

12.4.1. Determining currently active keys version

On the EstEID card there can be many different secure keys. Still active at the same can be only one. Chapter 2.2 "Card features" describes the amount of keys on the card. Every key has its unique version number. Therefore key reference file which contains references to currently active key must have the key version number as well available.

Key reference file has FID 0033. As described in chapter 16.2 "Session key decryption in case a new key pair has been generated on the card after personalisation" in this file on place 9..A is a reference to currently active authentication key and on place 13..14 a reference to currently active signature key.

Key reference consists two bytes. First byte of the 2byte key reference holds the key identifier. Second byte holds the version of the active key.

To read the key version from the card it needs to be get MF/EEEE/0033 EF selected. File selection is done as follows:

1) Select the Master File (MF) with the command SELECT FILE:

CLA	INS	P1	P2
00	A4	00	0C

2) Select the catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Select file 0013 with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	02	0C	02	00 33

4) Read record 1 from EF 0033:

CLA	INS	P1 (record number)	P2	Le
00	B2	01	04	00

5) For example card responds to previous command with status 0x9000 and data:

Byte no	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
Value	00	A4	08	95	01	40	83	03	80	11	00	B6	08	95	01	40	83	03	80	01	00

In previous response byte on position A holds the key version for authentication key. Place 14 holds the key version for signature key.

13. Certificates

EstEID card has the following certificates:

- the certificate enabling digital identification of the card user
- the certificate enabling the digital signature by the card user

The certificates in the EstEID card are located in catalogue EEEE in the form of transparent files and they are always readable, authentication is not required. As certificate renewal is required during the validity period of the card, fixed size files are reserved during personalisation for the certificates to ensure the possibility for writing new (and possibly larger) certificates into the card.

The rest of the file is filled according to the following scheme:

Certificate	Byte 80	00 .. 00

i.e. byte 80 is appended at the end of the certificate and the rest of the file is filled with zero bytes.

Certificates are located in the following files:

Certificate	FID	Size (hex bytes)
Certificate enabling digital identification	MF/EEEE/AACE	600
Certificate enabling the digital signature	MF/EEEE/DDCE	600

File sizes may vary. The part below describes how to determine the actual size of the file on the card.

13.1. The certificate file reading and its content

If the selected file is a certificate file, the response includes the file size in bytes. In the application realized in Java card, the FCP and FMD content is set during personalisation. Therefore the file size cannot be found by checking the offset bytes of the response to the Select File command. The location of the bytes showing the file size may be changed by developing the personalisation process or the card application. The change may also be caused by the use of a new platform. In order to find the file size, the element with the Tag=0x85 must be found in the FCP TLV structure, for instance:

0x62 … … 0x85 0x02 0x06 0x00 …

The file size in the above example is 0x600 bytes.

The size of the certificate on the card may also be determined by reading the first block of the file (for example 250 bytes) and then looking for the certificate length ASN.1 syntax by parsing.

13.2. Reading the certificate on the card

In order to read the certificate enabling digital identification (similarly to other certificates), do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Select file AACE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	02	04	02	AACE

The card responds:

Byte no.	0	1	2	3	4	5	6	7	8	9	A	B	C	... total length of the response is 1A bytes
Value	62	18	82	01	01	83	02	AA	CE	85	02	06	00	...

File size is in the response bytes B and C, in this case $600_{(hex)}$.

4) Read the file content with the command READ BINARY:

CLA	INS	P1(offsetMSB)	P2(offsetLSB)	Le (number of bytes to read)
00	B0	00	00	64

The given command reads $64_{(hex)}$ bytes from the beginning of the file. Here Le is necessary in case of both T=0 and T=1, as it shows how many bytes must be read. P1 and P2 show the address in the file from which to read (for instance, in order to read from address $120_{(hex)}$ P1=01 and P2=20). The maximum value of Le is FE.

13.3. Certificate loading

The certificate loading is conducted by the sending of secured command series onto the EstEID card. The secured command series is formed by Card Centre using CMK2b key.[5]

Certificate loading may take place once PIN1 has been checked in order to guarantee that the card user is aware of the operation.

If necessary, the certificate loading command series modifies the secret key references in order to set a new key version as the active version.

13.4. Certificate renewal

In EstEID cards both user certificates (authentication certificate and signature certificate) and, if present, the root certificate may be renewed (note that EstEID card doesn't carry root certificates). Due to security considerations, the certificates cannot be overwritten 'directly' – overwriting has been protected by security mechanisms which also guarantee that the certificates may be loaded only into the EstEID card where they belong to. Any attempt to load the certificate into some other EstEID card results in an error situation.

The implemented security mechanism operates as follows: the Token Management Centre (TMC) converts the certificate code into commands that include the respective MAC code. Thus, the EstEID card where it belongs to shall also receive it accordingly.

The certificate loading procedure includes the following operations:

1. Select MF on the card.
2. Set the security environment no. 3 within the card.
3. Select catalogue EE EE on the card.
4. Set the security environment no 3 again in catalogue EE EE.
5. Verify PIN1 code. The aim of the verification of PIN1 code is to guarantee that the card user is aware of the operation.
6. Read the commands in the certificate loading code file given by TMC and send on to the card.

The commands in the certificate loading code files are in HEX (or BCD) format, one command on each line (lines are separated by LF - 0x0A).

On the command level, the certificate renewal process is as follows:

1) Select the Master File (MF) with the command SELECT FILE.

[5] Look chapter 19.5 "Generating certificate loading modules".

CLA	INS	P1	P2
00	A4	00	0C

2) Set the security environment no 3 within the card:

CLA	INS	P1	P2
00	22	F3	03

3) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

4) Set the security environment no 3 again in catalogue EE EE:

CLA	INS	P1	P2
00	22	F3	03

5) Verify PIN1 code:

CLA	INS	P1	P2	Lc	PIN1
00	20	00	PIN1 code number = 01	PIN1 code length	As ASCII codes

6) Load the new certificate onto the card according to the following algorithm:[6]

```
while (! EndOfFile)
{
    Line = ReadLineFromFile;
    Convert Line from BCD to Binary;
    If Protocol == T0 Remove Le byte;
    Transmit Line to EstEID card;
}
```

The certificate is renewed once all the commands in the certificate loading code file have been sent to the card.

Note: *Certificate renewal is actively used only in case of EstEID cards with application version v1.0.*

[6] Look chapter 19.5 „Generating certificate loading modules".

14. Calculating the response to SSL/TLS challenge

According to PKCS#1 ver. 1.5 block type 1, calculating the response to SSL/TLS challenge is the encrypting of the formatted challenge with a secret RSA key. In order to calculate the response in EstEID card, do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EE EE

3) Set the security environment no. 1 in the card:

CLA	INS	P1	P2
00	22	F3	01

4) Verification of PIN1:

CLA	INS	P1	P2	Lc	PIN1
00	20	00	PIN1 code number = 01	PIN1 code length	As ASCII codes

5) Send the command to the card to calculate the response to the challenge:

a) In case of T=0 send:

CLA	INS	P1	P2	Lc (challenge length)	Challenge
00	88	00	00	24	....

b) In case of T= 1 send

CLA	INS	P1	P2	Lc (challenge length)	Challenge	Le
00	88	00	00	24	....	80

EstEID card response is 80_{hex} bytes formatted according to PKCS#1 ver. 1.5 block type 1 followed by a challenge encrypted with the authentication key.

According to SSL standard, the length of the challenge is 24_{hex} bytes, however, EstEID card also enables the calculation of responses to challenges of other length.

Note: *In case of EstEID v1.0 cards, the maximum allowed length of SSL/TLS challenge is 30hex bytes. In case of v1.1 cards, the maximum length is 75hex bytes, in case of cards with version v3.0 and above, F5hex bytes. The response to the SSL/TLS challenge may also be calculated with ECC key. The following rules apply when calculating the challenge response with ECC key:*

1. The maximum length of the challenge is 32_{dec} bytes. Sending a longer data block results in an error.
2. In case the challenge is shorter than the module of the used key, a required amount of zeros is added to the block from the left.
3. In case the challenge is longer than the module of the used key, the data block is shortened from the right so that the length of the operation input would be equal to the length of the key module.

As EstEID card application version 3.0 uses keys of various lengths, also the responses to the challenges received from the card are of various lengths. Thus, the value of Le byte is not always 0x80 (as in case of RSA-1024). The value of Le byte may be 0.

Note: *In EstEID v3.0 and v3.4 the return length argument Le is not used. Response to SSL/TLS challenge is calculated and returned in spite of the provided Le value.*

Note: *During calculation of response to SSL/TLS challenge, if the security status has once been set and PIN1 code verified then the security status of the card remains unchanged after the SSL/TLS challenge calculation operation has been executed, i.e. it is possible to repeat the operation in the card without setting the security conditions again.*

15. Calculating the electronic signature

The calculation of electronic signature is the encryption of a hash object formatted according to PKCS#1 ver. 1.5 block type 1 with the secret RSA key.

EstEID enables the calculation of the electronic signature using the RSA key in two ways:

a) Giving the object including the hash of the text to be signed
b) Sending the text to be signed to the card in blocks and letting the card hash it.

The signature on a card with hash may be calculated when the operation takes place in a device with limited resources without the hashing function, for instance in a POS terminal.

Electronic signature may also be calculated with the ECC key.

As keys with various lengths may be used in calculating the signature, the Le byte may not always be 0x80. In the signature calculation command sent to the card, the value of Le may be 0. In case of RSA-2048 bit keys are used for signing then it is necessary to use APDU command chaining in the course of the operation.

Note: *In EstEID v3.0 and v3.4 the return length argument Le is not used. Signature is calculated and returned in spite of the provided Le value.*

15.1. Calculating the electronic signature when the hash is ready

Do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Set the security environment no 1 within the card:

CLA	INS	P1	P2
00	22	F3	01

4) Verify PIN2:

CLA	INS	P1	P2	Lc	PIN2
00	20	00	PIN2 code number = 02	PIN2 code length	As ASCIIcodes

5) Sending the command to the card to calculate the signature:

a) In case of T=0 send:

CLA	INS	P1	P2	Lc (hash object length)	Hash object
00	2A	9E	9A	23	....

b) In case of T = 1 send:

CLA	INS	P1	P2	Lc (hash object length)	Hash object	Le
00	2A	9E	9A	23	....	nn

Le value is related to RSA key length. For example RSA with length 1024 has value 80_{hex}. In case of RSA with length 2048 messages chaining must be used.

Note that the hash object has to be calculated by using the appropriate hash algorithm which is supported by the specific card application version. Supported algorithms are provided in chapter 2.1 The used encryption algorithms and key lengths.
The following list specifies hash algorithm identifiers:

- SHA-1: 3021300906052B0E03021A05000414hex
- SHA-224: 302D300D06096086480165030402040500041Chex
- SHA-256: 3031300D06096086480165030402010500420hex
- SHA-384: 3041300D06096086480165030402020500430hex
- SHA-512: 3051300D06096086480165030402030500440hex

EstEID card response is 80_{hex} bytes formatted according to PKCS#1 ver. 1.5 block type 1 followed by a hash object encrypted with the signature key.

For example: Let the SHA1 hash of the text to be signed be:
010203040506070809 0A0B0C0D0E0F1012131415 (hex),
PIN2 "12345" and the used protocol T=0. We calculate the electronic signature:

1) Select the root directory:

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Set the security environment no 1 within the card:

CLA	INS	P1	P2
00	22	F3	01

4) Verify PIN2:

CLA	INS	P1	P2	Lc	PIN2
00	20	00	02	05	31 32 33 34 35

5) Send the card the command to calculate the signature:

CLA	INS	P1	P2	Lc	Hash algorithm identifier || Hash of data (Hash object)
00	2A	9E	9A	23	30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 12 13 14 15

The card responds:

Bytes waiting in the card	The number of bytes waiting
61	80

6) Read the bytes waiting:

CLA	INS	P1	P2	Le
00	C0	00	00	80

The card responds:

Electronic signature: 80_{hex} bytes	OK trailer
—	90 00

15.1.1. Calculating the electronic signature with ECC key when the hash is ready

The following rules apply when calculating the signature with ECC key:

1. The maximum length of the data block given for signing is 32 bytes. Sending a longer data block results in error.
2. In case the given data block is shorter than the module of the used key, a required amount of zeros is added to the block from the left.
3. In case the given data block is longer than the module of the used key, the data block is shortened from the right so that the length of the operation input would be equal to the length of the key module.

Note: *Using ECC keys is not actively used in case of EstEID cards.*

15.2. Calculating the electronic signature in a card with hashing

In the EstEID application one command to calculate the hash may include one to three 0x40-byte data blocks. Thus there are three possible forms of command series (except the last) to send the data to be signed onto the card:

CLA	INS	P1	P2	Lc	Data: 80 40 <40(hex)-byte data block>
10	2A	90	A0	42	80 40 ….

CLA	INS	P1	P2	Lc	Data: 81 80 <80(hex)- byte data block >
10	2A	90	A0	83	80 81 80 ….

CLA	INS	P1	P2	Lc	Data: 81 C0 <C0(hex)- byte data block >
10	2A	90	A0	C3	80 81 C0 ….

Note: *Differently from the previous* version, *EstEID v3.0 and v3.4* applications *do not give a calculated hash and it cannot be read from the card in any other way.*

Note: *Calculating electronic signature with hashing in the card is not actively used in case of EstEID cards.*

In case RSA key is used, the hash algorithm is SHA-1.

In case ECC key is used, the card selects the hashing logarithm according to the key module length as follows:

ECC module length (bits)	The used SHA option
160	SHA-1
192	SHA-1
224	SHA-224
256	SHA-256

NB! In case the signature is calculated on a hashing card, the key reference must be set before hashing.

In calculating the electronic signature in a hashing card, the first four operations are not repeated.

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Set the security environment no 1 within the card:

CLA	INS	P1	P2
00	22	F3	01

4) Verify PIN2:

CLA	INS	P1	P2	Lc	PIN2
00	20	00	PIN2 code number = 02	PIN2 code length	As ASCII codes

5) Hash the text to be signed in the card.

The text must be sent to the card in 40hex-byte blocks for hashing, the last block may be shorter.

If the text to be sent to the card for hashing is longer than 80_{hex} bytes, then APDU command chaining must be used. In this case the text is divided into several parts in the way each one could fit in one of the command message described in the beginning of current chapter.

With the exception of the last one, blocks are in the following format:

CLA	INS	P1	P2	Lc	Data
10	2A	90	A0	42	80 40 <40(hex)- byte data block >
				83	80 81 80 <80(hex)- byte data block >
				C3	90 81 C0 <C0(hex)- byte data block >

The last block is in the form:

CLA	INS	P1	P2	Lc	Data: 80 <length of the last block><the last data block>

00	2A	90	A0	Length of the last block + 2	80L….

The length of the last block may also be 40_{hex} bytes. In case the protocol T=0 is used, the card responds:

Bytes waiting in the card	The number of bytes waiting - SHA-1 hash length
61	14

And the hash calculated on the card may be read with the command

CLA	INS	P1	P2	Le
00	C0	00	00	14

In case protocol T=1 is used, Le=14 may be added at the end of the command sending the last block to the card, then the card shall send back the calculated hash. In case Le has not been added at the end of the command, the card shall only respond 90 00 under the T=1 protocol.

6) Send the card the command to calculate the signature for the hash on the card. The card shall add the necessary bytes in front of the 20 (dec)-byte hash to form the ASN.1 SHA-1 hash object

a) In case of T=0 we send:

CLA	INS	P1	P2
00	2A	9E	9A

b) In case of T=1 we send:

CLA	INS	P1	P2	Le
00	2A	9E	9A	80

For example: The aim is to calculate the signature for the following text (text length is 8A bytes, ∪ marks the space character), let PIN2 be "12345" and the protocol used T=0.

```
Auto∪on∪sõitjate∪või∪veoste∪veoks∪või∪sõidukite∪haakes∪vedamiseks∪või
∪eritööde∪tegemiseks∪ettenähtud∪vähemalt∪neljarattaline∪mootorsõiduk.
```

1) Select the root directory:

CLA	INS	P1	P2

00	A4	00	0C

2) Select catalogue EEEE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Set the security environment no 1 within the card:

CLA	INS	P1	P2
00	22	F3	01

4) Verify PIN2:

CLA	INS	P1	P2	Lc	PIN2
00	20	00	02	05	31 32 33 34 35

5) As the text length is 8A bytes, the hash must be calculated by using APDU command chaining and sent to the card as blocks where the length of the first two is 40(hex) and the length of the last one $0A_{hex}$ bytes.

Block1:

CLA	INS	P1	P2	Lc	80 40 <first 40(hex)-byte data block>
10	2A	90	A0	42	80 40 Auto␣on␣sõitjate␣või␣veoste␣veoks␣või␣sõidukite␣ haakes␣vedamisek

Block 2:

CLA	INS	P1	P2	Lc	80 40 <second 40(hex)- byte data block >
10	2A	90	A0	42	80 40 s␣või␣eritööde␣tegemiseks␣ettenähtud␣vähemalt␣neljarattaline␣mootorsõiduk

Block 3:

CLA	INS	P1	P2	Lc	80 L <last 40(hex)- byte or shorter data block >
00	2A	90	A0	0C	80 0A torsõiduk.

The card responds:

Bytes waiting in the card	Number of bytes waiting
61	14

6) Read the calculated hash (this operation may be omitted):

CLA	INS	P1	P2	Le
00	C0	00	00	14

7) Calculate the electronic signature:

CLA	INS	P1	P2
00	2A	9E	9A

The card responds:

Bytes waiting in the card	Number of bytes waiting
61	80

8) Read the electronic signature from the card:

CLA	INS	P1	P2	Le
00	C0	00	00	80

The card responds:

80 (hex bytes)	OK trailer
……	90 00

16. Decrypting the session key

The decryption of the session key is possible only with the RSA keys. In case of long keys, the length of the command sent to the card may exceed the maximum length of APDU. In that case, the command chaining must be used. When securing the command chain with a passphrase, each command of the chain must be secured separately. The card user is authenticated once the last command of the chain is secured.

When decrypting the session key, the secret key decrypts the RSA crypto block received by encrypting the data block formatted according to PKCS#1 ver. 1.5 block type 2 with a respective public key. At the same time, EstEID card parses the data block received as a result of decryption and only sends out the data.

Thus NB! In case the data block given for decryption is not a data block encrypted according to PKCS#1 ver. 1.5 block type 2 with the respective public key, there will be an error situation in the EstEID card.

Decryption operation is possible with both the authentication and signature key.

In order to decrypt the session key, do as follows:

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Set the security environment no 6 within the card:

CLA	INS	P1	P2
00	22	F3	06

4) Modify the security environment by deleting the reference to the authentication key, i.e. tell the card that there is no need for authentication operations[7]:

CLA	INS	P1	P2	Lc	Key reference
00	22	41	A4	02	83 00

5) Modify the security environment by deleting the reference to the signature key, i.e. tell the card that there is no need for signing operations:

CLA	INS	P1	P2	Lc	Key reference
00	22	41	B6	02	83 00

6) Modify the security environment by setting a reference to the key that we want to use in the decryption operation.

In case we want to decrypt with the authentication key, we send the following command[8]:

[7] For more detailed information on the operations see Appendix Chapter „MANAGE SECURITY ENVIRONMENT".

CLA	INS	P1	P2	Lc	Key reference
00	22	41	B8	05	83 03 80 11 00

In case we want to decrypt with the signature key, we send the following command :

CLA	INS	P1	P2	Lc	Key reference
00	22	41	B8	05	83 03 80 01 00

7) In case we want to decrypt with the authentication key, we verify PIN1 code:

CLA	INS	P1	P2	Lc	PIN1
00	20	00	PIN1 code number = 01	PIN1 code length	As ASCII codes

In case we want to decrypt with the signature key, we verify PIN2 code:

CLA	INS	P1	P2	Lc	PIN2
00	20	00	PIN2 code number = 02	PIN2 code length	As ASCII codes

8) We send the card the command to decrypt the block. The length of the encrypted block is $80_{(hex)}$ bytes as the length of the keys used is 1024 bits.:

If the session key to be decrypted exceeds the maximum APDU length, encrypted session key must be sent successively as blocks. In the case of commands chaining all commands except the very last one have the following form with CLA set to 0x10:[9]

a) In case T=0 we send:

CLA	INS	P1	P2	Lc (1+80 bytes)	1 byte with value 00	Encrypted block
10	2A	80	86	81	00	

b) In case T=1 we have to use Le as well:

CLA	INS	P1	P2	Lc (1+80 bytes)	1 byte with value 00	Encrypted block	Le
10	2A	80	86	81	00	....	00

[8] This form of the command is used in case a new key pair has not been generated on the card after personalisation. In case the given premise cannot be used, some additional operations are necessary. More detailed description below.

[9] Similar chaining operation is performed in chapter 15.2 „Calculating the electronic signature in a card with hashing“ in clause 5).

The very last or only command has CLA value 0x00 and following form:

a) In case T=0 we send:

CLA	INS	P1	P2	Lc (1+80 bytes)	1 byte with value 00	Encrypted block
00	2A	80	86	81	00	

b) In case T=1 we have to use Le as well:

CLA	INS	P1	P2	Lc (1+80 bytes)	1 byte with value 00	Encrypted block	Le
00	2A	80	86	81	00	....	00

Le 00 is given, as the length of the decrypted data block in the crypto block is not known.

EstEID card responds with the data block in the crypto block.

For example: We need to decrypt the block with a signature key, let PIN2 be "12345" and the protocol used T=0.

1) Select the Master File (MF) with the command SELECT FILE. After the initial loading the given command may be omitted as the directory has been selected by default.

CLA	INS	P1	P2
00	A4	00	0C

2) Select catalogue EEEE with the command SELECT FILE:

CLA	INS	P1	P2	Lc	FID
00	A4	01	0C	02	EEEE

3) Set the security environment no 6 within the card:

CLA	INS	P1	P2
00	22	F3	06

4) Modify the security environment in the card by sending the following three commands:

CLA	INS	P1	P2	Lc	Key reference
00	22	41	A4	02	83 00

CLA	INS	P1	P2	Lc	Key reference
00	22	41	B6	02	83 00

We set the signature key for the decryption operation, i.e. we tell the card that the following decryption operations must be executed with the signature key. Therefore we send:

CLA	INS	P1	P2	Lc	Key reference
00	22	41	B8	05	83 03 80 01 00

5) Verifying PIN2 code:

CLA	INS	P1	P2	Lc	PIN2
00	20	00	02	05	3132 33 34 35

6) We send the card the command to decrypt the blocks:

CLA	INS	P1	P2	Lc (1+80 bytes)	1 byte with value 00[10]	Encrypted block
00	2A	80	86	81	00	

The card responds:

Bytes waiting in the card	Number of bytes waiting (the amount of data in the crypto block in bytes)
61	...

7) Read the decrypted data from the card:

CLA	INS	P1	P2	Le (number of bytes waiting)
00	C0	00	00	...

The card responds:

The data included in crypto block	OK Trailer
....	90 00

16.1. The derivation of the session keys with ECC keys

[10] Look Appendix subchapter „Decipher“ of chapter „PERFORM SECURITY OPERATION“ and inspect paragraph related to RSA for clearer understanding of prepadded 0x00 byte.

It is possible to use ECC authentication keys and EstEID card for the derivation of the TLS session premaster secret according to Diffie-Hellman key derivation algorithm.

The sequence of commands is as follows:

1) Set the reference key for the key derivation:

CLA	INS	P1	P2	Lc	Reference key
00	22	41	A6	05	83 03 80 11 00

2) Verify PIN1 code:

CLA	INS	P1	P2	Lc	PIN1
00	20	00	PIN1 code number = 01	PIN1 code length	As ASCII codes

3) The derivation of the premaster secret:

CLA	INS	P1	P2	Lc (length of the public key of the other participant)	The other participant's public key as octet strings
00	88	00	00	L	04 ….

The card response given is 20(dec) bytes SHA-1 hash of the calculated X coordinate point.

NB! Once the reference key for the key derivation is set, the card will calculate the premaster secret of the session key derivation process. If the key derivation reference has not been set, the challenge response shall be calculated.

4) Delete the reference key for the key derivation:

CLA	INS	P1	P2	Lc	Reference key deletion
00	22	41	A6	02	83 00

Note: *Using ECC keys are not actively used in case of EstEID cards.*

16.2. Session key decryption in case a new key pair has been generated on the card after personalisation

It is possible to generate a new RSA key pair on EstEID card for both the authentication key and the signature key. During the compilation of the present manual it was not known if and in what circumstances new key pairs shall be generated on the card in case such a technical option arises (key pair generation procedure is described in chapter 19.4 of the present manual).

The procedure of the calculation of the SSL/TLS challenge response and the electronic signature does not change in case a new key pair is generated on the card, as in case of key generation or the loading of new certificates also the key references on the card shall be changed. However, this is not the case with session key decryption operations. In case it is impossible to presume that new keys have not been generated on the card after personalisation, we must execute some additional operations before the decryption operations.

Note: Information regarding to Signature and Authentication keys references are located in chapter 19.4.1 "New key pair generation on EstEID card" clause 5).

In the card personalisation, Authentication key1 and Signature key1 are generated and Signature key2 and Authentication key2 are left blank (referring to them results in an error situation).

In case new key pairs are now generated, the new secret keys shall be placed respectively in the position of Signature key2 and Authentication key2 and the previous keys remain. In the next generation, the new secret keys shall be placed in the position of Authentication key1 and Signature key1 and the previous keys remain in the position of Authentication key2 and Signature key2. The keys that were formed during personalisation are removed as the result of the last key pair generation. There is no limit to the number of new key pair generations.

The present valid keys are written in file FID=0033 in catalogue EEEE. It is a structured file including one record which after personalisation is as follows:

Byte no	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
Value	00	A4	08	95	01	40	83	03	80	11	00	B6	08	95	01	40	83	03	80	01	00

The reference of the active authentication key is in bytes no 9..A and the signature key reference in bytes no 0x13..0x14.

In order to execute the decryption operation with the current key version, it must be first determined which of the keys is the current one by reading record no 1 in file MF/EEEE/0033. Then the security environment within the card must be modified by telling the card that the operation must be executed with the given key.

Thus, the procedure given in clause Modify the security environment by setting a reference to the key that we want to use in the decryption operation.4) of the Chapter 16 "Decrypting the session key" is as follows:

- We modify the security environment within the card by setting a reference to the key we want to use in decryption operation.

  Read record no 1 in file FID=0033.

  In case we want to decrypt with the authentication key, we send the following command:

CLA	INS	P1	P2	Lc	Key reference		
00	22	41	B8	05	83 03 80	Byte no 9 in record no 1 in file EEEE/0033	Byte no A in record no 1 in file EEEE/0033

  In case we want to decrypt with the signature key, we send the following command:

CLA	INS	P1	P2	Lc	Key reference		
00	22	41	B8	05	83 03 80	Byte no 0x13 in record no 1 in file EEEE/0033	Byte no 0x14 in record no 1 in file EEEE/0033

The same applies when authorising with the passphrase.

17. Setting and changing passphrases, executing operations by identifying the user with passphrases

17.1. The general principles of using passphrases

Passphrase is a series of tokens of unlimited length used to derive 3DES key. The same 3DES key has been saved on the EstEID card. The key may be used to establish a secure connection between EstEID card and the host application, but also to identify the user as similarly to PIN codes, the passphrase is known only to the lawful card user. Thus, also the user may enter the secret key (through the passphrase) to the host application to execute EstEID PKI operations. The card user is considered as authenticated when secure messaging takes place between the card and the host application with the given keys. There are no separate authentication procedures.

3DESKey1 and 3DESKey2 are derived from Passphrase 1 and Passphrase 2 according to the procedures of deriving 3DESKey from the passphrase.

Note: *Using passphrases is not actively used in case of EstEID cards.*

When using passphrases for identification, the sequence of operation procedure is as follows:

1. The user enters the passphrase;

2. The host application derives 3DES key from the passphrase;
3. During the mutual interaction, the card and the host application derive a unique session key;
4. The session key is used to encrypt the content of the commands sent to the card and to calculate the cryptographic check sums. The attempt to execute the operation with an incorrect key leads to an error situation. The responses from the EstEID card are also encrypted and include cryptographic check sums.

A detailed example of each given step is given below.

EstEID card has two keys derived from the passphrases:

1) 3DESKey1 – enables the execution of operations with an authentication key
2) 3DESKey2 – enables the execution of operations with signature key.

The attempt to execute the operation with another 3DES key (for instance: calculating the electronic signature with 3DESKey1) leads to an error situation.

In card personalisation, the value of the keys is set as 00..00 – the given key is not usable. In order to execute operations with the given keys, the card user must first set them on the card with 3DESKey setting procedure. The keys cannot be read from the card, but they can be replaced by the above-mentioned procedure.

The keys are related to retry counters with the initial value 0xFF. The counter value is reduced by one after each unsuccessful operation. The counter value is not reset – once the counter has reached zero, the key cannot be used any longer.

17.2. Deriving 3DES key from the passphrase

The procedure for deriving the key from the passphrase was a joint agreement, not a characteristic of EstEID card. The need for a uniform procedure is conditioned by the need to ensure the validity of the same passphrase in various applications.

The joint procedure to derive 3DesKey from the passphrase is as follows:

1. SHA-1 hash of the passphrase is calculated;
2. 16 bytes from the left of the has are taken;
3. the youngest bit of each byte are set so that the byte would be odd (EstEID card requirement: a byte with wrong parity in the EstEID card key leads to an error situation).

Note: *In production, the EstEID card's both 3DESKey values are 00..00 (all zero bytes). The given key cannot be used as zero bytes are even bytes.*

Let the passphrase1 (for the derivation of 3DESKey1) as follows:

KaksUkurjaUkukkeUkaklevadUkuudiUkatusel.

And passphrase2 (for the derivation of 3DESKey2) as follows (the given sentences shall also be used in the examples to follow):

See␣on␣minu␣paroollause2.

with ␣ marking the space character.

Calculate the SHA-1 hashes of the sentences:

The hash of passphrase1:

63	F0	EA	AD	E2	7E	5F	CA	D2	5A	09	CB	3F	E3	96	5E	43	7B	DA	F9

The hash of passphrase2:

30	A1	DB	94	DF	29	BC	65	A8	17	25	2B	A6	73	88	FE	9B	7B	D6	43

We take 16 bytes from the left of the hashes and set the youngest bit of each byte so that the byte would be odd. The result is:

3DESKey1:

62	F1	EA	AD	E3	7F	5E	CB	D3	5B	08	CB	3E	E3	97	5E

3DESKey2:

31	A1	DA	94	DF	29	BC	64	A8	16	25	2A	A7	73	89	FE

The given keys shall be used also in the examples to follow.

17.3. Setting passphrases onto the card

3DESKey1 is located on the Master File (MF) 3DESKeys file FID=0010 record no 1 and 3DESKey2 is located in record no.2 of the same file.

The keys are given values with the command UPDATE RECORD.

Setting 3DESKey includes writing the key value into the respective record with the following nuance:

Bytes '04 00' must be added in front of 3DESKey1 key value and bytes '05 00' added in front of 3DESKey2 values[11]. Thus the length of each key record is 0x12 bytes.

The key values are usually derived from passphrases according to 3DESKey value derivation procedure.

The condition for assigning 3DESKey values is that PIN2 be verified or the communication be secured with the same key (the previous version).

17.3.1. Setting the passphrase by authorising with PIN2 code

In order to set the passphrase by authorising with PIN2 code, do as follows:

1. Set the security environment no.1 within the card:

CLA	INS	P1	P2
00	22	F3	01

2. Verify PIN2 code:

CLA	INS	P1	P2	Lc	PIN2				
00	20	00	02	05	31	32	33	34	35

3. Select Master File and then file FID=0010

CLA	INS	P1	P2
00	A4	00	0C

CLA	INS	P1	P2	Lc	FID	
00	A4	02	0C	02	00	10

4. Overwrite record no 1 of the given file with the following command (P1 = record number):

CLA	INS	P1	P2	Lc	Key ID		Key value														
00	DC	01	04	12	04	00	62	F1	EA	AD	E3	7F	5E	CB	D3	5B	08	CB	3E	E3	97	5E

[11] These are the key (sequence) numbers for the inner structure of EstEID card. For additional info look Appendix.

In order to set Passphrase2, we write the key value in record no. 2 of the same file.

CLA	INS	P1	P2	Lc	Key ID		Key value														
00	DC	02	04	12	05	00	31	A1	DA	94	DF	29	BC	64	A8	16	25	2A	A7	73	89	FE

17.3.2. Setting the passphrase by authorising with the former passphrase

Passphrase may be set (i.e. write the new key value into the Master File FID=0010) also by authorising with the present key (i.e. the key in the Master File FID=0010).

It must be admitted that the given operation is more complicated than setting the passphrase with PIN2 authorisation, as it requires the establishment of EstEID secure messaging between the card and the host application.

NB! When authorising with PIN2 we may set both passphrase1 and passphrase2, however, when authorising with the passphrase, we must authorise the new passphrase1 with the present passphrase1 and respectively the new passphrase2 with present passphrase2.

Let the valid passphrases be the ones set in the previous chapter passphrase1 = “Kaks kurja kukke kaklevad kuudi katusel.” and passphrase2 = “See on minu paroollause2”

Let the new passphrase1 be as follows:

Kured␣läinud␣-␣kurjad␣ilmad.

with ␣ marking the space character.

Key 3DESKey1 derived from the passphrase is as follows:

85	92	9D	57	D9	40	76	7A	97	89	E6	32	DC	07	BA	70

In order to set the new value to 3DESKey1, do as follows:

1) Select Master File and then file FID=0010:

CLA	INS	P1	P2
00	A4	00	0C

CLA	INS	P1	P2	Lc	FID	
00	A4	02	0C	02	00	10

2) Set the security environment no.2 within the card:

CLA	INS	P1	P2
00	22	F3	02

3) Send the card the command GET CHALLENGE in order to receive the challenge consisting of eight randomly generated bytes:

CLA	INS	P1	P2	Le
00	84	00	00	08

Note: *From here onwards the given example cannot be exactly repeated, as the challenge by the card differs each time.*

In case of the given example, the card gave the following response:

challenge								OK trailer	
06	F3	22	BD	D4	84	88	A3	90	00

4) Next, we will generate 2 random values, the length of the first – RND_IFD – is 8 bytes and the second – K_IFD – 32_{dec} bytes. Let the given values in our example be:

RND_IFD = 01 02 03 04 05 06 07 08

and

K_IFD = 11 12 13 14 15 16 17 18
21 22 23 24 25 26 27 28
31 32 33 34 35 36 37 38
41 42 43 44 45 46 47 48

Note: *To ensure maximum security the applications used in practice must use better random value generation methods.*

5) Form the block RND_IFD || Challenge from the card || K_IFD, in our example:

RND_IFD, 8 bytes	Card challenge, 8 bytes	K_IFD, 32_{dec} bytes

01…08	06…A3	11..48

6) Encrypt the block with the 3DESKey1 valid at present (derived from the passphrase “Kaks kurja kukke kaklevad kuudi katusel.”) in 3DES CBC mode, whereas ICV=”00 00 00 00 00 00 00 00”. The result is as follows:

F2 89 C9 96 5D 10 DC DE

8E 88 58 10 FB D6 3D C5

9B E6 2E 20 D7 36 1E 8C

B5 C8 BB C7 1F E4 C9 D5

74 10 C1 7D 10 E9 F4 E8

F3 FF 7E D5 AE A8 90 17

7) We form the command MUTUAL AUTHENTICATE and send it onto the card. P2 = key number, which in case of 3DESKey1 is 04:

CLA	INS	P1	P2	Lc	Formed block	Le
00	82	00	04	30	F2…17	30

The card responds as follows:

Response block, 0x30 bytes	OK trailer	
….	90	00

In our example the response block is as follows:

2B 4A 0B E2 2B CF 2B FD

C1 ED 73 FA 5F D8 FE F2

74 D0 17 BA AB 48 DA 29

41 FF B8 33 47 2C 77 35

3B 56 C3 5A EA B6 31 70

52 D5 EA 6A 45 CA C4 5E

8) The response block is encrypted with 3DESKey1 in 3DES CBC mode with ICV=”00 00 00 00 00 00 00 00”. We decrypt the block, the result is as follows:

06	F3	22	BD	D4	84	88	A3
01	02	03	04	05	06	07	08
69	DA	9F	6F	F4	27	A7	8B
B6	8B	DB	B7	7A	C7	89	43
6F	33	DB	B5	9D	79	9F	F2
C1	AB	EA	1B	B8	44	81	48

9) We execute the following operation:

(K_IFD) XOR (decrypted response block [0x10..0x2F])

Thus, we XOR the K_IFD decrypted response block with bytes 16(dec)…47(dec). The result is as follows:

78	C8	8C	7B	E1	31	B0	93	97	A9	F8	93	5F	E1	AE	6B
5E	01	E8	81	A8	4F	A8	CA	80	E9	A9	5F	FD	02	C6	00

We have hereby reached our initial result – two EstEID secure messaging session keys and an 8-byte send sequence counter have been derived.

The top row of the above table "78…6B" is session key 1 – SK1 – and the row below "5E…00" is session key 2 – SK2.

We get the initial value of the send sequence counter as follows: send sequence counter bytes 0..3 are the decrypted response block bytes 0xC to 0xF[12] from the card (see "Reading counters in the card" Chapter 12) and send sequence counter bytes 4..7 are the bytes 4..7 of the same block. Thus in our example, send sequence counter is:

SSC = **05 06 07 08 D4 84 88 A3**

Next we may execute the operation under EstEID security communication.

10) The send sequence counter must be increased by one **before each** operation executed with session keys. **The youngest bite of the send sequence counter is on the right**. Thus, by increasing send sequence counter by one, we get

SSC = **05 06 07 08 D4 84 88 A4**

11) We form the overwriting command for record no.1 of file FID=0010 similarly to the authorisation with PIN2:

[12] Counting starts from zero.

CLA	INS	P1	P2	Lc	Key ID		Key value in new passphrase “Kured läinud – kurjad ilmad.”														
00	DC	01	04	12	04	00	85	92	9D	57	D9	40	76	7A	97	89	E6	32	DC	07	BA	70

However, the given command cannot be sent directly to the card. The data field (i.e. “Key ID” + “Key value”) must be processed with session keys and send sequence counter.

12) The data field of the given command must be encrypted with 3DES algorithm. Therefore, byte 0x80 must be added at the end of the block followed by as many zero bytes as needed to make the number of bytes in the block eightfold.

Note: *If the block length is eightfold already without adding supplementary bytes, we add 8 bytes: “80 00 00 00 00 00 00 00”.*

In the present case:

04	00	85	92	9D	57	D9	40	76	7A	97	89	E6	32	DC	07	BA	70	80	00	00	00	00	00

13) Next we encrypt the block with SK1 in 3DES CBC mode using send sequence counter as ICV. The result is:

08	BB	57	9A	AB	1B	A2	B5	D5	BB	1E	83	16	F0	AC	F8	94	DD	24	7F	CF	16	F9	FB

14) We reconstruct the command formed in clause 11 as follows:

CLA	INS	P1	P2	Lc	Fixed value	Length of the next block	Fixed value	Crypto block received in the previous clause
0C	DC	01	04	Open	87	19	01	08…FB

NB! *CLA byte = 0x0C, meaning that security communication is used in the command.*

15) We temporarily form the following block:

Command header, extended to the length 8 bytes || **Command data field, extended to the eightfold length**

In our case:

0C	DC	01	04	80	00	00	00
87	19	01	08	BB	57	9A	AB
1B	A2	B5	D5	BB	1E	83	16

F0	AC	F8	94	DD	24	7F	CF
16	F9	FB	80	00	00	00	00

16) We calculate the MAC code of the block in 3DES CFB mode using SK1 as the key and send sequence counter as ICV. The result is as follows:

MAC = 87 94 E4 7E E2 CB 00 1F

17) We form the final command adding MAC code to the command formed in clause 14 and establish Lc value. If necessary also Le is added (using T=1 protocol), which is always 0.

CLA	INS	P1	P2	Lc	Fixed value	Length of the next block	Fixed value	Crypto block from clause 13	Fixed value	Fixed value	Calculated MAC-code	Le
0C	DC	01	04	25	87	19	01	08…FB	8E	08	87..1F	0

The given command is sent to the card.

The card responds as follows:

Fixed values		OK Trailer'		Fixed values		Response MAC								OK Trailer	
99	02	90	00	8E	08	51	CB	48	70	4D	6C	DA	4C	90	00

OK Trailer "90 00" shows that the security communication operated correctly, OK Trailer "90 00" shows also that the card command transmitted under the security communication operated correctly.

Next we must check the authenticity of the card response with the MAC code of the response. Thus:

18) We increase the send sequence counter by one:

SSC = 05 06 07 08 D4 84 88 A5

19) We extend the data in the response to eightfold length:

99 02 90 00 80 00 00 00

20) We calculate the MAC code of the given block in 3DES CFB mode using SK1 as the key and send sequence counter as ICV (similarly to clause 16). The result is:

MAC = 51 CB 48 70 4D 6C DA 4C

Thus the calculated MAC code equals the MAC code from the card.

Thus, the process of replacing passphrase1 authorised with the former passphrase has been successfully completed. The new passphrase is "Kured läinud - kurjad ilmad".

Replacing passphrase2 by authorising it with the former passphrase includes the same procedures with the following variations:

1. The derivation of session keys and send sequence counter is conducted with 3DESKey2 which in EstEID card has the sequence number 5 (P2=5 in clause 7)
2. Record no.2 must be overwritten, i.e. P1 = 2 in the command formed in clause 11.

Note: *Error in deriving the session keys and send sequence counter (for example the use of the wrong key in the host application) results in 3DESKey retry counter being reduced by one. The initial value of the counter is 0xFF and it cannot be reset. Once the counter has reached zero, authorising with the given passphrase cannot be used any longer.*

For reading 3DESKey retry counter values from the card see chapter 12.3 "Reading the passphrase retry counter".

17.4. Executing operations with passphrase authorisation

The execution of the given operations with passphrase authorisation is analogous to "Setting the passphrase by authorising with the former passphrase" authorisation (see Chapter 17.3.2) including the following steps:

1. The session keys and the send sequence counter are derived from the respective 3DESKey.
2. The command is constructed analogously to the PIN code authorisation.
3. The command is secured with session keys and send sequence counter.
4. The command is sent to the card and the response read.
5. The authenticity of the response is checked.

Next we will consider each operation separately on the basis of an example.

The following table shows which passphrase must be used for various operations:

	Calculating response to SSL/TLS challenge	Calculating electronic signature	Decrypting the session key with authentication key	Decrypting the session key with signature key
Passphrase1	Yes		Yes	
Passphrase2		Yes		Yes

17.4.1. Calculating the response to SSL/TLS challenge

Let passphrase1 be “Kured läinud - kurjad ilmad” and SSL/TLS challenge “30 31 32 33 34 35 36 37 38 39” (we hereby use the shortened challenge).

As in the previous chapter, 3DESKey1 derived from the passphrase has been calculated as follows:

85	92	9D	57	D9	40	76	7A	97	89	E6	32	DC	07	BA	70

1) We set the security environment no.2 in Master File and EEEE catalogue. For that purpose:

We select the Master File:

CLA	INS	P1	P2
00	A4	00	0C

Set the security environment no.2 within the card:

CLA	INS	P1	P2
00	22	F3	02

Select catalogue EEEE:

CLA	INS	P1	P2	Lc	FID	
00	A4	01	0C	02	EE	EE

Set the security environment no.2 **again**:

CLA	INS	P1	P2
00	22	F3	02

2) Derive session keys SK1, SK2 and send sequence counter SSC as in chapter 17.3.2, see clauses 3)..9).

Note: *From here onwards the given example cannot be exactly repeated, as the session keys SK1, SK2 and send sequence counter SSC differ each time.*

Let in the present example

SK1 = BA DA 02 DE 36 FB A7 BE 17 1D 8E A8 77 22 0D 78

SK2 = 44 76 34 30 52 2B 5D A3 76 90 84 B2 DD F4 60 37

and SSC = 05 06 07 08 F9 2D E6 B3

3) We increase the send sequence counter by one:

SSC = 05 06 07 08 F9 2D E6 B4

4) We add the byte 0x80 to the right of the challenge and as many zero bytes as needed to make the block length eightfold.

Note: *If the block length is eightfold already without adding supplementary bytes, we nevertheless add 8 bytes: "80 00 00 00 00 00 00 00".*

In this case:

30	31	32	33	34	35	36	37	38	39	80	00	00	00	00	00

5) We encrypt the given block in 3DES CBC mode with the key SK1 and use send sequence counter as ICV (which was increased by one in clause 3). The result is:

29	5E	DE	E3	41	2A	82	30	0E	F8	74	C8	23	7E	6C	CD

6) We form the command block as follows:

Fixed value	Length of the following data	Fixed value	The block received from the previous step, length 0x10 bytes	Fixed value	Fixed value	Fixed value
87	11	01	29…CD	97	01	80

The bytes "97 01 80" added at the end mean that the length of the expected response is 0x80 bytes – 1024-bit RSA cryptoblock.

7) We form the header of the command to calculate the response to SSL/TLS challenge. This is analogous to the command header used in PIN1 authorisation, the only difference is CLA byte value 0x0C:

CLA	INS	P1	P2
0C	88	00	00

8) We temporarily form the following block:

Command header, extended to the length of 8 bytes || Data field formed in step 6, extended to eightfold length

In our example:

Header	Header extension to length 8	Block received in step 6	The extension of the received in step 6 block to eightfold length
0C 88 00 00	80 00 00 00	87…80	80 00

9) We calculate the MAC code of the block received in the previous step in 3DES CFB mode using SK2 as the key and send sequence counter as ICV. The result is:

MAC = 20 10 A7 C3 10 84 A0 59

10) We form the command to be sent to the card. In case the protocol is T=1, **Le=0** must be added at the end:

CLA	INS	P1	P2	Lc	Fixed value	Length of the next block	Fixed value	Crypto block received in clause 5	Fixed value	Fixed value	Calculated MAC-code	Le
0C	88	00	00	20	87	11	01	29…CD	8E	08	20..59	00

The card responds:

Fixed values				Encrypted challenge response, 0x88 bytes	Fixed values		MAC, 8 bytes	OK Trailer
87	81	89	01	….	8E	08	1E E7 D9 C2 42 41 D6 63	90 00

EstEID card adds a similar extension (to make the length eightfold) to the encrypted data objects as in case of objects sent to the card (see Chapter 14 "Calculating the response to SSL/TLS challenge" clause 4) upper part). As the length of the challenge response is 0x80 bytes which is eightfold, the card additionally appends 8 bytes at the end: "80 00 00 00 00 00 00 00".

Next we must check the authenticity of the response received from the card by calculating the MAC code and comparing it to the MAC code received from the card.

11) We increase the send sequence counter by one:

SSC = 05 06 07 08 F9 2D E6 B5

12) We temporarily form the next block from the response received from the card leaving out the MAC object at the end (tag 0x8E, length 0x08 and MAC value) and replace it with “80 00 00 00” to make the block length eightfold.

Fixed values				Encrypted challenge response, length 0x88 bytes	Fixed values
87	81	89	01	….	80 00 00 00

13) We calculate the MAC code of the block received in the previous step in 3DES CFB mode using SK2 as the key and send sequence counter as ICV. The result in our example is:

MAC = 1E E7 D9 C2 42 41 D6 63

This equals with the MAC code received from the card.

14) We decrypt the challenge response with key SK1 in 3DES CBC mode. The result is as follows:

Challenge response, length 0x80 bytes	Extension with the length 8 bytes
….	80 00 00 00 00 00 00 00

We leave out the extension at the end and get the calculated challenge response.

17.4.2. Calculating the electronic signature

Calculating the electronic signature with passphrase authorisation is similar to the calculation of SSL/TLS challenge response. The differences are as follows:

1) Passphrase2 must be used (3DESKey2); any attempt to authorise with passphrase1 results in an error situation.

2) Command header is as follows (see Chapter 15.2 “Calculating the electronic signature in a card with hashing” clause 6) *ülalpool*):

CLA	INS	P1	P2
0C	2A	9E	9A

and a hash object must be placed instead of the challenge in the (unsecured) data field (see Chapter 15.2 "Calculating the electronic signature in a card with hashing" clause 6)). More detailed information on the calculation of the electronic signature is given in chapter 15.1 "Calculating the electronic signature when the hash is ready " [13].

17.4.3. Decrypting session keys

Also the securing of session key decrypting commands with passphrase authorisation is analogous to securing the SSL/TLS challenge response and electronic signature calculation commands, however, with several significant nuances. Thus we will hereby consider the operation in detail.

Passphrase1 must be used for authorisation in decrypting the session key with authentication key, and passphrase2 used in authorising the decryption with the signature key. The reverse will lead to an error situation.

Let our passphrase2 be "See on minu paroollause2". And the crypto block including the session key with the length of 0x80 bytes as follows:

21 A6 … B9 E1

3DESKey2 derived from the passphrase has been calculated as in chapter 17.3.2:

30	A1	DB	94	DF	29	BC	65	A8	17	25	2B	A6	73	88	FE	9B	7B	D6	43

1) Set **security environment no.2 in Master File and security environment no 7 in EEEE** catalogue. Therefore:

Select Master File:

CLA	INS	P1	P2
00	A4	00	0C

Set the security environment **no.2** within the card:

CLA	INS	P1	P2
00	22	F3	02

Select catalogue EEEE:

[13] There is no data on the possibility of calculating the electronic signature with passphrase authorisation and hash use on the card. On the other hand, there is also no immediate need for the given operation as the passphrase is used in larger full keyboard devices which have the hash function included in the system software and there is no need for hashing on the card.

CLA	INS	P1	P2	Lc	FID	
00	A4	01	0C	02	EE	EE

Set the security environment **no7** within the card:

CLA	INS	P1	P2
00	22	F3	07

2) Next the security environment within the card must be modified. The given operations are analogous to the session key decryption with PIN code authorisation (see chapter 16 of the manual).

We modify the security environment within the card by deleting the reference to the authentication key, i.e. we tell the card that there is no need for authentication operations:

CLA	INS	P1	P2	Lc	Key reference
00	22	41	A4	02	83 00

We modify the security environment within the card by deleting the reference to the signature key, i.e. we tell the card that there is no need for signature operations:

CLA	INS	P1	P2	Lc	Key reference
00	22	41	B6	02	83 00

We modify the security environment within the card by setting a reference to the key that we want to use in the decryption operation. In case we want to decrypt with the signature key, we send the following command[14]:

CLA	INS	P1	P2	Lc	**Key reference**
00	22	41	B8	05	83 03 80 **01 00**

Note: *The key reference for decryption with authentication key would be "83 03 80 **11 00**".*

3) We derive session keys SK1, SK2 and send sequence counter SSC similarly as described in chapter "Setting the passphrase by authorising with the former passphrase" (see Chapter 17.3.2). 3DESKey2 must be used, i.e. the sequence number of the key in EstEID card is **5** (4 in case of 3DESKey1).

[14] The command is in the given format in case a new key pair has not been generated on the card after personalisation. In case such a premise cannot be used, see the next chapter.

In the example:

SK1 = EB 08 02 E5 4A 25 07 2B B6 98 33 6F 6C B3 E9 E5

SK2 = D9 B1 E7 B0 04 21 4F 42 E8 8D 9D CD 50 1D FD 82

and SSC = 05 06 07 08 B6 E4 C8 2C

4) We increase the send sequence counter by one:

SSC = 05 06 07 08 B6 E4 C8 2D

5) We add one zero byte in front of the crypto block (to the left) so that the block length would be 0x81bytes:

00 21 A6 … B9 E1

6) We add byte 0x80 to the challenge on the right and as many zero bytes as needed to make the block length eightfold. As the previous operation made the block length 0x81 bytes, after the given operation the block is as follows:

1 zero byte	Crypto block, length 0x80 bytes	Extension
00	21 A6 …B9 E1	80 00 00 00 00 00 00

7) We encrypt the given block in 3DES CBC mode with key SK1 and use send sequence counter as ICV (increased by one in clause 4). Thus as a result:

Encrypted block, length 0x88 bytes
36 0D … E2 53

8) We form the command block as follows:

Fixed values	The block received in the previous step, length 0x88 bytes	Fixed value	Fixed value	Fixed value (Le')
87 81 89 01	36 0D…E2 53	97	01	00

The bytes "97 01 00" added at the end mean that the expected response length is not known and thus the length value is given as 0. The crypto block usually includes the object encoded according to the SSL/TLS session data encryption secret key ASN.1, various crypto

algorithms of various key length may be used. Thus we cannot foresee the length of the key object in the crypto block and the data length expected from the card is set as Le'=0.

9) We form the command header to the calculation of SSL/TLS challenge response. This is analogous to the command header used in PIN1 authorisation, the only difference is in CLA byte value 0x0C:

CLA	INS	P1	P2
0C	2A	80	86

10) We temporarily form the following block:

Command header, extended to the length of 8 bytes || **Data field formed in step 8, extended to eightfold length**

In our case:

Header	Header extension to length 8	The block received in step 8	The extension of the block received in step 8 to eightfold length
0C 2A 80 86	80 00 00 00	87 81 89 01 36 … 53 97 01 00	80

11) We calculate the MAC code of the block received in the previous step in 3DES CFB mode using SK2 as the key and the send sequence counter as ICV. In our case the result is as follows:

MAC = 53 2B 42 7E 06 87 29 7A

12) We form the command to be sent to the card. In case protocol is T=1, **Le=0** must also be added at the end:

CLA	INS	P1	P2	Lc	Fixed values	Crypto block received in clause 7, length 0x88	Fixed values	Calculated MAC code	Le
0C	2A	80	86	99	87 81 89 01	36 0D … E2 53	8E 08	53..7A	0

The card responds:

Fixed value	The length of the next data block	Fixed value	Encrypted session key	Fixed value		MAC, 8 bytes	OK Trailer
87	11	01	F7 32 40 48 18 D5 67 96 26 1E 5B B8 6C BB 4D 14	8E	08	60 19 E8 61 7C 6D 88 25	90 00

Next the authenticity of the card response must be checked by calculating the MAC code and comparing it to the MAC code received from the card.

13) We increase the send sequence counter by one:

SSC = 05 06 07 08 B6 E4 C8 2E

14) We temporarily form the following block from the response received from the card: we leave out the MAC object at the end (tag 0x8E, length 0x08 and MAC value) and replace it with "80 00 00 00 00" so that the block length would be eightfold.

Fixed values			Encrypted session key	Fixed values
87	11	01	F7 32 40 48 18 D5 67 96 26 1E 5B B8 6C BB 4D 14	80 00 00 00 00

15) We calculate the MAC code of the block received in the previous step using SK2 as the key and send sequence counter as ICV. In our case the result is as follows:

MAC = 60 19 E8 61 7C 6D 88 25

16) We decrypt the data block received from the card using SK1 as 3DES key and send sequence counter as ICV. The result is:

30 31 32 33 34 35 36 37 38 39 80 00 00 00 00 00

17) We leave out the extension at the end giving us the session key object value "0123456789" in the form of ASCII.

18. Operations with the previous key versions

It is possible to execute operations on EstEID card with the previous key versions. The given procedures are similar to the operations executed with current keys, however, the current key version must be read from file EEEE/0033 and the security environment in the card modified by setting the references to the previous key version.

Note: *Operations with previous key versions are not actively used in case of EstEID cards.*

18.1. Calculating the SSL/TLS challenge response with the previous key version

When calculating the SSL/TLS challenge response with the previous authentication key version, do as follows:

1) Select the Master File.
2) Select catalogue EEEE.
3) Set the security environment no 1 within the card.
4) Read record no 1 in file FID=0033.
5) Set the key reference with the command

CLA	INS	P1	P2	Lc	Key reference	
00	22	41	A4	05	83 03 80	Key reference value

In case bytes no 9 and A from file FID=0033 are "1100", the key reference value is "1200" and vice versa in order to use the non-current key version in the operation execution.

6) Verify PIN1.
7) Send the card the command to calculate the challenge response:

CLA	INS	P1	P2	Lc (Challenge length)	Challenge	Le
00	88	00	00	24	....	80

8) The format of the card response does not differ from the response received in the operation executed with the current key.

In case passphrases are used for authorisation, the reading of file FID=EEEE/0033 and the setting of key references must be executed between the procedures of clause 1 and 2 (see "*Setting* the passphrase by authorising with the former passphrase" Chapter 17.3.2), i.e. after setting the security environment and prior to deriving the session keys.

18.2. Calculating the electronic signature with the previous key version

In order to calculate the electronic signature with the previous key version, do as follows (see also Chapter *15 "Calculating the electronic signature"*):

1) Select the Master File.
2) Select catalogue EEEE.
3) Set the security environment no 1 within the card.
4) Read record no 1 in file FID=0033.
5) Set the key reference with the command

CLA	INS	P1	P2	Lc	Key reference	
00	22	41	B6	05	83 03 80	Key reference value

In case bytes no 0x13 and 0x14 from file FID=0033 are “0100”, the key reference value is “0200” and vice versa in order to use the non-current key version in the operation execution.

6) Verify PIN2.

7) Send the card the command to calculate the electronic signature:

CLA	INS	P1	P2	Lc (hash object length)	Hash object	Le
00	2A	9E	9A	23	….	80

The format of the card response does not differ from the response received in the operation executed with the current key.

In case passphrases are used for authorisation, the reading of file FID=EEEE/0033 and the setting of key references must be executed after setting the security environments and prior to deriving the session keys.

18.3. Session key decryption with the previous key version

In order to execute the decryption operation with the previous key version, the operation must be executed similarly to the decryption with the current key, however, the key reference must be the opposite to the one written in file EEEE/FID=0033. If the authentication key value in the file is “1100”, we set it as “1200” in the card and vice versa. If the signature key value in the file is “0100”, we set it as “0200” in the card and vice versa.

19. Card management operations

19.1. Operations in general

Card management operations stand for operations executed with EstEID chip that are not thoroughly dependent on the card user but authorised by the card centre. The permission is given by cryptographic methods.

The card management system enables the execution of four operation types related to EstEID cards:

1. **Generating certificate loading codes** – in order to replace a certificate on a EstEID card, for instance after the expiry of the validity of the present certificate and the issuance of a new one, the certificate must be converted into a so-called special loading code. The loading codes can be generated only by the card centre with a respective secret key.

2. **Authorization of setting new PIN and PUK codes to the card if the PUK code's value is unknown –** in the given operation new PIN and PUK codes are set to the card without the need for previous codes. The card holder identity is identified by non-electronic means and in order to avoid the abuse of the operation, a respective system of organisational measures must be developed.

3. **Generating the modules for loading additional applications onto the card –** it is possible to load also other applications onto the EstEID card, however, this can only take place with the permission of the qualified institution. The additional application loading is technically supervised by the card centre. EstEID card accepts additional applications only when converted into a loading code by a specific procedure. The conversion operation is executed by the card centre with the respective secret key.

4. **Granting authorisation for the generation of new key pairs on EstEID card** – EstEID card enables the generation of new secret key pairs. The new key pair generation requires the authorisation by the card centre.

19.2. Card management keys: CMK1, CMK2a, CMK2b, CMK3

The security of EstEID card management system is based on four secret 3DES keys:

1. **Authentication object replacement key CMK1** – used to execute the authentication object (PIN code) replacement procedure;

2. **New key pair generation authentication key CMK2a** – used to authorise the new key pair generation (the card centre uses the given key to authorise the new key generation on the card);

3. **The key to form the certificate loading command sequence CMK2b** – used to form the certificate loading modules on the card and setting the key references after the key generation;

4. **The key to form the secure command sequence for application loading CMK3** – used to form the additional application loading modules.

The given four secret 3DES keys are generated by the card personalisation service provider in a secure environment. The keys are used to calculate the card-specific CMKs loaded onto the card during personalisation. The given keys cannot be read from the card or modified.

The example keys used in the present document are as follows:

1. CMK1 = A1A1A1A1A1A1A1A1A2A2A2A2A2A2A2A2
2. CMK2a = B0B0B0B0B0B0B0B0B3B3B3B3B3B3B3B3
3. CMK2b = C1C1C1C1C1C1C1C1C2C2C2C2C2C2C2C2
4. CMK3 = D0D0D0D0D0D0D0D0D3D3D3D3D3D3D3D3

19.3. Deriving the card-specific keys

3DES algorithm is a symmetrical crypto algorithm, the secret keys must be written also on each EstEID card. For security reasons, the main keys stored in the card centre are not written on the cards, but only the card-specific keys derived from the main key and the card holder's personal identification number. The card management code derivation procedure is as follows:

1. Calculating the SHA-1 hash of the card user's personal identification number.
2. Taking 16 bytes from the left of the given hash.
3. Encrypting the given bytes with the respective main key using 3DES algorithm, in CBC mode with ICV="00 00 00 00 00 00 00 00".
4. Setting the youngest bit of each byte so that the byte would be odd.

Example: Let the card user's personal identification number (ASCII) be "01234567890". We calculate the card-specific keys according to the sample main keys given in 19.2 ("Card management keys: CMK1, CMK2a, CMK2b, CMK3")

a. The hash of the given personal identification number is:

7E D1 0E 4A 58 9C 87 F9 E6 A8 5C 22 E4 B0 C3 8E CF 5F 50 59

b. We encrypt the 16 bytes on the left with the key CMK1 in 3DES CBC mode, ICV=0:

41 F9 AE 35 48 53 6F 19 B9 3F ED 4E F8 90 C9 3B

c. We set the youngest bits so that the bytes would be odd:

Card CMK1 = **40 F8** AE **34 49 52 6E** 19 B9 **3E EC 4F** F8 **91 C8** 3B

We also calculate:

Card CMK2a = 89 FB 5D 9B B0 83 D0 97 AB 13 5E BF 70 DF FD 86

Card CMK2b = 3B 8A BC 9B 98 1F 29 AB B3 0D 97 15 64 29 43 62

Card CMK3 = 6E DC 2A 25 D6 64 7C D0 C1 BF 01 16 08 51 F7 04

19.4. Generating new key pairs

The chapter only contains information about generating new key pairs in case of EstEID application version v1.0.

New key pairs may be generated on the card with the respective authorisation from the card centre using CMK2a key. The generated key becomes the new key version, at the same time the current key version remains. The new key version may be used with a special procedure.

The procedure related to certificate loading turns the current key version into a previous one and the new version the current one. The operations with the old key version may be executed with a special procedure.

19.4.1. New key pair generation on EstEID card

New keys are generated with module size 1024_{dec} bits.

The public key shall be read after the key pair generation from file FID=1000 similarly to the current card version. The public key length is dependent on the key module length.

The generation of a new key pair on EstEID card takes place as follows:

1) We read the card user's personal identification number written in the record no.7 of the personal data file FID=EEEE/5044. Let the personal identification number (ASCII) be 01234567890.[15]

2) We read record 1 from file FID=EEEE/0033 in order to get the current key references in the card. The current authentication key reference is read in record bytes 0x9..0xA and the signature key reference in bytes 0x13..0x14 (the byte counting begins with zero). The new key must be generated into the key that is not current at the moment. Thus the current key is stored in case it will be needed later. If the authentication key reference is "1100" in file FID=EEEE/0033, the reference of the generated key is "1200" and vice versa. If the signature key reference is "0100" in file FID=EEEE/0033, the reference of the generated key is "0200" and vice versa. There are no technical impediments to generating the current key reference over.

3) We set the security environment no.3 both in Master File and EEEE catalogue:

Select the Master File:

CLA	INS	P1	P2
00	A4	00	0C

Set the security environment no.3 within the card:

CLA	INS	P1	P2
00	22	F3	03

Select catalogue EEEE:

[15] Look chapter 25.1.2 „Differences of application version v3.0".

CLA	INS	P1	P2	Lc	FID	
00	A4	01	0C	02	EE	EE

Set the security environment no.3 **again**:

CLA	INS	P1	P2
00	22	F3	03

4) Verify PIN1 code. The aim of the verification of PIN1 code is to guarantee that the card user is aware of the operation.

5) In catalogue EEEE of the EstEID card is file FID=EEEE/0013 including the service information regarding the secret keys. It is a fixed-length structured file (record length 0x4F) including four records – each record corresponds to one secret key:

Key	Key ID	Record no in file FID=EEEE/0013
Signature key1	0100	1
Signature key2	0200	2
Authentication key1	1100	3
Authentication key2	1200	4

The new key generation is lead by record byte no 0x32 (counting begins from zero). Its values in authentication and signature keys are different:

Key	Byte no 0x32 value in the respective record of FID=EEEE/0013	Key status
Signature key1 and Signature key2	F6	No key – must be generated.
	B6	Key has been generated – ready to use.
Authentication key1 and Authentication key2	E4	No key – must be generated.
	A4	Key has been generated – ready to use.

In order to generate the new key pair, the byte value must be set so that the key generation would be allowed. This may be done, for instance, by reading the given record with the command READ BINARY (records are readable), changes the value of byte no. 0x32 so that the key generation would be allowed and then write the record back into the file.

However, overwriting the record in file FID=EEEE\0013 requires the card centre authorisation. Next we do as follows:

6) Form the command to overwrite the record in the file FID=EEEE\0013. The following command is to overwrite record no 4 (concerning Authentication key2):

Header	Lc	Data
00 DC 04 04	4F	830412001012C00281809103FFFFFF7B 18800100A10A8B080030010302040305 E40795014089022113 7B11800106A103 8B010BB807950140890211307B118001 07A1038B010CB80795014089021130

7) We change the command header as follows:

Header
0C DC 04 04

CLA = 0C means that the command is forwarded to EstEID card under secure messaging. However, in the given operation no session key derivation or encrypting is used. Only MAC code is calculated.

8) We form the following block:

Fixed value	Data block length = Lc	Data block
81	4F	83 04…11 30

9) We temporarily form the following block:

Command header	Command header extension to length 8	The block formed in the previous step	The extension of the block formed in the previous step to eightfold length
0C DC 04 04	80 00 00 00	81 4F 83 04…11 30	80 00 00 00 00 00 00

10) We calculate the MAC code of the block in 3DES CBC mode with the card-specific key CMK2b with ICV="00 00 00 00 00 00 00 00". In our case the result is as follows:

9F 7F CD 8B 02 F8 56 C4

11) We form the following command:

Header	Lc	The block formed in step 8	Fixed values	MAC calculated in step 10	Le
0C DC 04 04	5B	81 4F 83 04…11 30	8E 08	9F 7F CD 8B 02 F8 56 C4	00

12) We send the command to the card. In case protocol T=1 is used, there must be Le=0 at the end. The card responds:

Fixed values	OK Trailer'	Fixed values	MAC	OK Trailer
99 02	90 00	8E 08	F9 5D 3F 23 71 D5 B7 11	90 00

If both OK Trailer and OK Trailer' are "90 00", we may proceed.

Note: *After the given operation the processed secret key is not usable any longer. If the operation is discontinued for some reason, it must be started again from the beginning.*

Note: *The command formed in step 11 may be reused as it does not contain session-specific elements.*

13) Next we will derive the session key similarly to the procedures in chapter *17.3.2 clauses* 3)..9). We use the card **CMK2a** as 3DES key with the sequence number **2**.

In our case let it be:

SK1 = 95 CC 39 2F 24 38 AA 24 1F B9 DC F4 77 77 30 FD

SK2 = 30 FD B4 B6 7B A8 74 6E 0E 67 43 A3 72 90 E9 5D

SSC = 05 06 07 08 4E 56 55 CB

14) Key generation requires specific changes in the security environment of the card. The given changes are different for generating the authentication key and the signature key.

In generating the authentication key we set the security environment as follows:

a) We delete the reference to the signature key:

CLA	INS	P1	P2	Lc	Data
00	22	41	B6	02	83 00

b) We set a reference to the authentication key to be generated:

CLA	INS	P1	P2	Lc	Data	
					Fixed values	Key reference
00	22	41	A4	05	83 00 80	12 00

In generating the signature key we set the security environment as follows:

a) We set a reference to the signature key to be generated:

CLA	INS	P1	P2	Lc	Data	
					Fixed values	Key reference
00	22	41	B6	05	83 00 80	02 00

b) We set a reference to the generated authentication key:

CLA	INS	P1	P2	Lc	Data	
					Fixed values	Key reference
00	22	41	A4	05	83 00 80	12 00

Setting the authentication key reference in the generation of the signature key is needed as the EstEID card calculates a signature for the generated public key using the authentication key for both the authentication key and the signature key (thus, in case of the authentication key itself). Therefore, when renewing both keys, we first generate the authentication key followed by the signature key.

Note: *In generating the signature key, a reference must be set to the authentication key generated earlier. Setting a reference to an authentication key that has not been generated results in an error situation.*

15) We form the command to generate the key:

CLA	INS	P1	P2	Lc	Fixed values	Public key file FID (fixed value)
00	46	00	00	04	8302	1000

The public key formed in the key pair generation is located in the file FID=1000.

Next, the command must be forwarded onto the card under EstEID security communication using the session keys and send sequence counter formed in step 13. In the secure messaging only the protection with MAC codes is used, encrypting is not used.

16) We increase the send sequence counter by one:

SSC = 05 06 07 08 4E 56 55 CC

17) We change the command header as follows:

CLA	INS	P1	P2
0C	46	00	00

18) We temporarily form the following block:

Header	Header extension to length 8	Data	Data block extension to length 8
0C 46 00 00	80 00 00 00	81 04 83 02 10 00	80 00

19) We calculate the MAC code of the formed block with the card key CMK2b in 3DES CFB mode with ICV="00 00 00 00 00 00 00 00". The result is as follows:

MAC = 2B 08 D5 EF B2 53 FC C6

20) We form the command to be sent to the card:

Header	Lc	Fixed values	Fixed values	Fixed values	MAC calculated in the previous step	Le
0C 46 00 00	10	81 04	83 02 10 00	8E 08	2B 08 D5 EF B2 53 FC C6	00

The card responds:

Fixed values	OK Trailer'	Fixed values	MAC	OK Trailer
99 02	90 00	8E 08	69 CA B6 19 A6 EC 60 C9	90 00

Next we check the authenticity of the received response:

21) We increase the send sequence counter by one:

SSC = 05 06 07 08 4E 56 55 CD

22) We temporarily form the block:

Card response	Card response extension to length 8
99 02 90 00	80 00 00 00

23) We calculate the MAC code of the formed block with card key CMK2b in 3DES CFB mode with ICV="00 00 00 00 00 00 00 00". The result is as follows:

MAC = 69 CA B6 19 A6 EC 60 C9

As the result equals the MAC code received from the card, it may be said that the key has been successfully generated.

Next, the public key must be read from the card in order to use it, for instance, to form the certificate request.

Note: *The public key must be read from the card before the next key generation operation at the latest, as EstEID card has only one public key file (FID=EEEE\1000) and the content of the file is overwritten during the next key generation operation.*

24) We read the public key from the card. The public key is in the file FID=EEEE\1000. It is a sequence file with the length of 0x12C bytes. The file structure is as follows:

Bytes	Content
0..0x19	Fixed values
0x20..0x9F	Module
0xA0..0xA1	Fixed values
0xA1	The length of public exponent, 0x3 or 0x4 bytes
0xA2..0xA4 või 0xA2..0xA5	Public exponent
0xA6..0xAB	Fixed values
0xAC..0x12B	Key signature calculated with the authentication key (RSA, SHA-1, PKCS#1)

In order to set the given key as the current key, record no 1 of the file FID=EEEE\0033 (the only record in the file) must be modified. This may take place, for example, after loading the certificate about the generated key onto the card.

The modification of record no 1 of the file FID=EEEE\0033 may take place so that the record is read from the file, the key reference generated into the record is replaced and then the content of the record is written back into the file. The record is readily readable in the file, the card centre authorisation is needed only for overwriting.

In order to receive the authorisation, we do as follows:

Let the new content of the record be as follows:

Fixed values	Authentication key reference	Fixed values	Signature key reference
00 A4 08 95 01 40 83 03 80	12 00	B6 08 95 01 40 83 03 80	02 00

Such a record content is formed, for instance, when a new authentication key and a new signature key have been generated after the card personalisation.

25) Set the security environment no 3 in both the Master File and EEEE catalogue:

Select the Master File:

CLA	INS	P1	P2
00	A4	00	0C

Set the security environment no 3 within the card:

CLA	INS	P1	P2
00	22	F3	03

Select catalogue EEEE:

CLA	INS	P1	P2	Lc	FID	
00	A4	01	0C	02	EE	EE

Set the security environment no 3 **once again**:

CLA	INS	P1	P2
00	22	F3	03

26) Select file FID=0033.

27) Verify PIN1.

28) Form the following command:

Header	Lc	Record content	Le
00 DC 01 04	15	00 A4 08 95 01 40 83 03 80 12 00 B6 08 95 01 40 83 03 80 02 00	00

29) Modify the record header as follows:

Header
0C DC 01 04

30) We form the following block:

Fixed values	Record content
81 15	00 A4 08 95 01 40 83 03 80 12 00 B6 08 95 01 40 83 03 80 02 00

31) We temporarily form the following block:

Header	Header extension to length 8	The data block formed in the previous step	The extension of the data block formed in the previous step to eightfold length
0C DC 01 04	80 00 00 00	81 15 …02 00	80

32) We calculate the MAC code of the given block in 3DES CBC mode with the **card-specific key CMK2b** with ICV="00 00 00 00 00 00 00 00". In our case the result is:

MAC = **4C C0 4E A0 22 E6 2D 9F**

33) We form the following command to be sent to the card:

Header	Lc	The block formed in step 30	Fixed values	MAC calculated in step 32	Le
0C DC 01 04	21	81 15 … 02 00	8E 08	4C C0 4E A0 22 E6 2D 9F	00

Note: *Similarly to the command formed in step 11 also the present command may be reused as it does not contain session-specific elements.*

The card responds:

Fixed values	OK Trailer'	Fixed values	MAC	OK Trailer
99 02	90 00	8E 08	F9 5D 3F 23 71 D5 B7 11	90 00

If both OK Trailer and OK Trailer' are "90 00", then the content of record no.1 of file FID=EEEE\0033 has been successfully overwritten and the keys with the shown references set as the current keys.

19.5. Generating certificate loading modules

Generating certificate loading modules operation is a part of certificate renewal process and is actively used only in case of cards with EstEID application v1.0.

Certificate loading module includes APDU command sequences which once sent to EstEID card will load a new certificate on to it. The file structure is as follows:

Row 1: command SELECT FILE selecting the correct file for overwriting.

Rows 2..N: commands UPDATE BINARY writing a new certificate into the selected file.

The commands are encoded in the form of BCD HEX.

UPDATE BINARY commands in rows 2..N must be secured by using the card-specific key CMK2b. The command SELECT FILE in row 1 does not have to be secured.

The certificate file size on EstEID card is 0x600 bytes (which may be increased if necessary on the order of a competent institution). The space left empty in the file after writing the certificate must be filled as follows:

0x600 bytes		
Certificate	Byte 0x80	Zero bytes to the end of the certificate 00…00

Le=0 must be added at the end of secured UPDATE BINARY commands. The software loading the certificate onto the card using protocol T=0 must ignore the byte.

Let us consider the generation of the certificate loading module on the basis of the following example. Let us have a new authentication certificate (for the sake of clarity the certificate has been shortened at the beginning of the example):

Bytes no	Values
0..F	30 82 04 A5 30 82 03 8D A0 03 02 01 02 02 04 3C
10..1F	02 31 6F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 05
20..2F	05 00 30 68 31 0B 30 09 06 03 55 04 06 13 02 45
30..3F	45 31 22 30 20 06 03 55 04 0A 13 19 41 53 20 53
40..4F	65 72 74 69 66 69 74 73 65 65 72 69 6D 69 73 6B
50..5F	65 73 6B 75 73 31 10 30 0E 06 03 55 04 0B 13 07
60..6F	54 45 53 54 2D 53 4B 31 0A 30 08 06 03 55 04 04
70..7F	13 01 31 31 17 30 15 06 03 55 04 03 13 0E 54 45
80..8F	53 54 2D 45 53 54 45 49 44 2D 53 4B 30 1E 17 0D
90..9F	30 31 31 31 32 36 31 32 31 31 32 37 5A 17 0D 30
…..	….
400..40F	C0 88 E9 94 DB 6C DE FD 2A C8 5E 45 24 05 06 31
410..41F	04 25 69 D3 BF 74 38 7B 03 1C 12 D0 21 B6 A2 13
420..42F	58 5A 37 FE 1C B1 D9 3F 6E F7 E6 27 A1 EC B2 3C
430..43F	02 59 85 F3 36 7B 7B 14 A2 14 80 33 67 51 24 43
440..44F	B3 A2 55 8E F2 37 9C 6B 38 D0 EC 24 9D 37 5F 73
450..45F	99 E2 CE E0 5B 82 00 1B 69 4B AE 04 53 CA 39 EC
460..46F	42 1F 77 0E F1 44 10 C0 33 61 8D C3 B5 43 90 F7
470..47F	1E 6E EC EE 3F B3 8F C7 A9 CC 1F 07 B5 15 DD C8
480..48F	6F 4C C4 40 2A C0 A4 FF B7 2E 6D 98 FE 5A 06 D2
490..49F	DD 52 48 B9 F6 2A DE 9C DE 0C 8B 1F 84 44 5A D3
4A0..4A8	08 A8 AB 02 53 53 6D 80 5D

1) We form a data block with the length of 0x600 bytes to be written into the file:

Bytes no	Values
0..F	30 82 04 A5 30 82 03 8D A0 03 02 01 02 02 04 3C
10..1F	02 31 6F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 05
20..2F	05 00 30 68 31 0B 30 09 06 03 55 04 06 13 02 45
30..3F	45 31 22 30 20 06 03 55 04 0A 13 19 41 53 20 53
40..4F	65 72 74 69 66 69 74 73 65 65 72 69 6D 69 73 6B
50..5F	65 73 6B 75 73 31 10 30 0E 06 03 55 04 0B 13 07
60..6F	54 45 53 54 2D 53 4B 31 0A 30 08 06 03 55 04 04
70..7F	13 01 31 31 17 30 15 06 03 55 04 03 13 0E 54 45
80..8F	53 54 2D 45 53 54 45 49 44 2D 53 4B 30 1E 17 0D
90..9F	30 31 31 31 32 36 31 32 31 31 32 37 5A 17 0D 30
…..	….
400..40F	C0 88 E9 94 DB 6C DE FD 2A C8 5E 45 24 05 06 31
410..41F	04 25 69 D3 BF 74 38 7B 03 1C 12 D0 21 B6 A2 13
420..42F	58 5A 37 FE 1C B1 D9 3F 6E F7 E6 27 A1 EC B2 3C
430..43F	02 59 85 F3 36 7B 7B 14 A2 14 80 33 67 51 24 43
440..44F	B3 A2 55 8E F2 37 9C 6B 38 D0 EC 24 9D 37 5F 73
450..45F	99 E2 CE E0 5B 82 00 1B 69 4B AE 04 53 CA 39 EC
460..46F	42 1F 77 0E F1 44 10 C0 33 61 8D C3 B5 43 90 F7
470..47F	1E 6E EC EE 3F B3 8F C7 A9 CC 1F 07 B5 15 DD C8
480..48F	6F 4C C4 40 2A C0 A4 FF B7 2E 6D 98 FE 5A 06 D2
490..49F	DD 52 48 B9 F6 2A DE 9C DE 0C 8B 1F 84 44 5A D3
4A0..4A8	08 A8 AB 02 53 53 6D 80 5D **80** 00 00 00 00 00 00
4B0..4BF	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
…..	…..
590..59F	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

2) We form the first row – command SELECT FILE:

Header	Lc	Certificate file FID
00 A4 02 0C	02	AA CE

The FID of the signature certificate is **DDCE**.

We form secured UPDATE BINARY commands to write the certificate on to the card. It is recommended to send the amount of 0x40 or 0x40 bytes (i.e. 0x80 or 0xC0) to the card with one command, as these values are optimal for the EstEID chip hardware.

Next we will consider the formation of secure UPDATE BINARY command from certificate bytes 0…0x40.

3) We form the command header:

CLA	INS	P1 Offset older byte	P2 Offset younger byte
0C	D6	00	00

4) We form the next block

Fixed value	The length of the next data block	Certificate bytes 0..0x40
81	40	30 82 04 A5 .. 41 53 20 53

Note: When sending 0x80 or 0xC0 with one command, the given data block must be formed as follows (in case of 0x80 bytes):

Fixed value	Fixed value	The length of the next data block	Certificate bytes 0..0x80
81	81	80	30 82 04 A5 .. 03 55 04 04

5) We temporarily form the following block:

Header	Header extension to length 8	The block formed in the previous step	The extension of the block formed in the previous step to eightfold length
0C D6 00 00	80 00 00 00	81 40 30 82 04 A5 … 41 53 20 53	80 00 00 00 00 00

6) We calculate the MAC code of the block in 3DES CBC mode with card-specific key **CMK2b** with ICV="00 00 00 00 00 00 00 00". The result is as follows:

MAC = 91 18 2F 52 15 BA 44 74

7) We form the command as follows:

Header	Lc	The block formed in step 4	Fixed values	MAC calculated in the previous step	Le
0C D6 00 00	4C	81 40 30 82 04 A5 … 41 53 20 53	8E 08	91 18 2F 52 15 BA 44 74	00

NB! *Byte Le=0 must be added at the end of the command. The certificate loading software using protocol T=0 must eliminate the byte before sending the command to the card.*

Thus, the second row (the first is the command SELECT FILE) of the loading code file is ready.

We also calculate the following rows. The result is:

Row no	Header	Lc	Data	Le
1	00A4020C	02	AACE	
2	0CD60000	4C	8140308204A53082038DA00302010202043C02316F300D06092A864886F70D0101050500306831 0B30090603550406130245453122302006035504A1319415320538E0891182F5215BA4474	00
3	0CD60040	4C	81406572746966697473656572696D69736B65736B75733110300E060355040B1307544553542D5 34B310A300806035504041301313117301506035504031130E54458E08A0CA8956185B8EFC	00
4	0CD60080	4C	814053542D4553544549442D534B301E170D3031313132363132313132375A170D303230343134 3132353434355A308191310B3009060355040613026565310F300D8E08716286E53AF42D1C	00
5	0CD600C	04	C8140060355040A13064553544549443117301506035504B130E61757468656E7469636174696F 6E3121301F060355040313184549 4B454547492C454556492C30308E0809AA438B73B3F7BC	00
6	0CD60100	4C	81403030303030303030303110300E0603550404130745494B45454749310D300B060355042A130 44545564931143012060355040513 0B30303030303030303030308E08318588C51D157F1F	00
7	0CD60140	4C	81403081A0300D06092A864886F70D010101050003818E0030818A02818100BF588CF11BC7CA7 D97A3D2F60CA8AF34212A76A6BEDB0A2FB6BA4D6620347A605A0F7B8E0851A8E38B4BA89D6 F	00
8	0CD60180	4C	81409761BB66846A6B82016F6558E580404D4DF4ECE49035D9207ADE4C7A2497507D1AEF587C AE644705A7F6C145BF61C48D7CA0E4943200EFFBB5096A8A86D71E7A8E084DDE851D9D4FEEC D	00
9	0CD601C0	4C	814052F615B815AE33E2E81D4C10CCF11A4ADD86867FA87AB5B5E41604397F02040231763FA3 8201AE308201AA300E0603551D0F0101FF0404030204B0301D0603558E08ABDCB9BD89895124	00
0xA	0CD60200	4C	81401D250416301406082B0601050507030206082B06010505070304303806035 51D1F0431302F3 02DA02BA0298627687474703A2F2F7777772E736B2E65652F74658E08ADF4F99806354632	00
0xB	0CD60240	4C	8140737463726C2F6573746569642F63726C2E63726C300F0603551D110408300681046E6F6E653 082012C0603551D20048201233082011F3082011B06092B0604018E080A4D5D4A7ECB547E	00
0xC	0CD60280	4C	8140CE1F0201023082010C3081E206082B0601050507020230 81D51E81D20053006500650020 00 730065007200740069006600690 06B00610061007400200 06F006E8E089CB18AF02C3142E2	00
	…	…	…	…
0x14	0CD60480	4C	81406F4CC4402AC0A4FFB72E6D98FE5A06D2DD5248B9F62ADE9CDE0C8B1F84445AD308A8AB 0253536D805D800000000000000000000000000000000000000000000000008E08A68444C76607248D	00
0x15	0CD604C0	4C	81400000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000008E08B31CB25697B76AC8	00
	…	…	…	…
0x19	0CD605C0	4C	81400000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000008E081105CEDAD9016710	00

19.6. Replacing PIN codes

The replacement of PIN codes consists of the following parts:

1) Authorisation from the card centre is received with CMK1
2) The card is set new PIN1, PIN2 and PUK (may be entered by the card user in the same terminal)

The further changing of PIN1, PIN2 and PUK is up to the card user.

The procedure takes place as follows (the procedure must take place in an authorised office):

1. After the visual identification of the card user the terminal will read the challenge from the card.
2. The terminal sends a secure challenge, i.e. signed by the service staff to the card management centre.
3. The card management centre checks if the service staff is authorised to execute the procedure and grants the authorisation using CMK1.
4. After receiving the authorisation from the card management centre, the card allows the use of the command RESET RETRY COUNTER without further authorization and replace the PIN and PUK codes without knowing their previous values.

The replacement of PIN codes requires the following two operations:

1. Overwriting the file records storing PIN code values.
2. Initialising the retry counters.

19.6.1. PIN code replacement procedure

We will consider the PIN code replacement procedure on the basis of the following example:

Let the card user's personal identification number be "01234567890", new PIN1="1234", new PIN2="12345" and new PUK="12345678".

1) The retry counter may be initialised only if its value is 0 (the card is blocked). Thus we need to read the retry counter values from the card (see Chapter 12), and in case these are not zeros we execute the PIN control operation with the incorrect PIN code until the counter values are zeros.

2) We select the Master file:

CLA	INS	P1	P2
00	A4	00	0C

3) We select FID=0012 (the given file stores the PIN code values)

CLA	INS	P1	P2	FID
00	A4	02	0C	00 12

4) We set the security environment no 4 in the Master file:

CLA	INS	P1	P2
00	22	F3	04

Next we derive the session key similarly to the procedures in Chapter 17.3.2. The card's **CMK1** is used as 3DES key with the sequence number 1.

In our case

SK1 = 19 EB A2 29 ED EC 4F E9 E9 27 B4 FC 35 9E 8F DF

SK2 = 36 C3 42 23 9C B6 7F 0B EC 3B 04 8D C5 66 F2 30

SSC = 05 06 07 08 D2 86 CA 6F

5) PIN codes are not stored unrestrained on EstEID card (despite the fact that reading their file FID=0012 is forbidden). The record of FID=0012 file is 9 bytes long with the following structure:

Byte no.	0	1	2	3	4	5	6	7	8
Content	PIN code length	The block is sent as follows: 1. PIN code is in BCD HEX format and values F are added to the right with the length of 7 bytes. 2. Byte with value 0x20 + PIN code length are added to the left 3. The block is encrypted with DES algorithm using itself as the key.							

In the given example we use PIN1 code as follows:

PIN block in the form of BCD HEX:

Byte no.	0	1	2	3	4	5	6	7
Value	24	12	34	FF	FF	FF	FF	FF

The block DES encrypted with itself: **7E B3 98 5F 23 22 9E 91**

Thus the record of PIN1 = "1234" is as follows:

Byte no..	0	1	2	3	4	5	6	7	8
Value	04	7E	B3	98	5F	23	22	9E	91

PIN2 = “12345” block is: **05 1F 8A 25 12 DC EF 34 6C**.

PUK = “12345678” block is: **08 E6 76 6E 41 75 43 43 D5**.

Next we shall consider the operation on the basis of PIN1. The operation is similar with PIN2 and PUK codes, the differences have always been marked separately.

Next we will execute the operation UPDATE RECORD.

6) We form the command header:

CLA	INS	P1 – PIN record number in the file	P2 – fixed value
0C	DC	PIN1: 01 PIN2: 02 PUK: 03	04

In case of PIN1:

CLA	INS	P1	P2
0C	DC	01	04

7) We increase the send sequence counter by one:

SSC = 05 06 07 08 D2 86 CA **70**

8) We form the next block:

PIN file record	The extension of PIN file record to eightfold length
04 7E B3 98 5F 23 22 9E 91	80 00 00 00 00 00 00

9) We encrypt the block with key SK1 in 3DES CBC mode with ICV="00 00 00 00 00 00 00 00". The result is:

05 F0 E5 98 6D AD 9A 7E 92 02 27 EA B6 5A 75 5E

10) We form the next block

Fixed values	The crypto block received in the previous step
87 11 01	05 F0 E5 98 6D AD 9A 7E 92 02 27 EA B6 5A 75 5E

11) We temporarily form the next block:

Header	Header extension to length 8	The block formed in the previous step	The extension of the block formed in the previous step to eightfold length
0C DC 01 04	80 00 00 00	87 11 01 … 5A 75 5E	80 00 00 00 00

12) We calculate the MAC code of the block in 3DES CFB mode with key **SK2** and the send sequence counter as ICV. The result is:

MAC = **AA 51 BE C6 12 AE 73 32**

13) We form the command to be sent to the card (in case protocol T=0, there is no need to send Le):

Header	Lc	The block formed in step 11	Fixed values	MAC received in the previous step	Le
0C DC 01 04	1D	87 11 01 … 5A 75 5E	8E 08	AA 51 BE C6 12 AE 73 32	00

The card responds:

Fixed values	OK Trailer'	Fixed values	MAC	OK Trailer
99 02	90 00	8E 08	EC 51 2D D9 B1 23 65 16	90 00

Next we check the response from the card.

14) We increase the send sequence counter by one:

SSC = 05 06 07 08 D2 86 CA **71**

15) We temporarily form the following block:

Response from the card	The extension of the card response up to length 8
99 02 90 00	80 00 00 00

16) We calculate the MAC code of the block in 3DES CFB mode with key SK2 and the send sequence counter as ICV. The result is as follows:

MAC = EC 51 2D D9 B1 23 65 16

As the MAC code received from the card is equals with the calculated MAC code, we may consider the file FID=0012 (PIN code file) as successfully overwritten.

Next we must restore the value of PIN code retry counter. We do it with the command RESET RETRY COUNTER

17) We form the header of the command:

CLA	INS	P1 – fixed value	P2 – PIN sequence number in the card
0C	2C	03	PIN1: 01 PIN2: 02 PUK: 00

In case of PIN1:

CLA	INS	P1	P2
0C	2C	03	01

18) We increase the send sequence counter by one:

SSC = 05 06 07 08 D2 86 CA 72

19) We temporarily form the following block:

Header	Header extension to length 8
0C 2C 03 01	80 00 00 00

20) We calculate the MAC code of the block in 3DES CFB mode with key SK2 and the send sequence counter as ICV. In our case the result is as follows:

MAC = 45 F6 E8 BB 84 DE C8 35

21) We form the command that we send to the card (in case protocol T=0, there is no need to send Le):

Header	Lc	Fixed values	MAC received from the previous step	Le
0C 2C 03 01	0A	8E 08	45 F6 E8 BB 84 DE C8 35	00

The card responds:

Fixed values	OK Trailer'	Fixed values	MAC	OK Trailer
99 02	90 00	8E 08	4A EC FC D6 EB 6D BF 0A	90 00

Next we check the response from the card.

22) We increase the send sequence counter by one:

SSC = 05 06 07 08 D2 86 CA **73**

23) We temporarily form the following block:

Card response	The extension of the card response to length 8
99 02 90 00	80 00 00 00

24) We calculate the MAC code of the block in 3DES CFB mode with key **SK2** and the send sequence counter as ICV. The result is as follows:

MAC = **4A EC FC D6 EB 6D BF 0A**

As the MAC code received from the card equals with the calculated MAC code, the initial value of the retry counter has been successfully restored.

In case we wish to replace also PIN2 and PUK codes, the steps 7..24 must be repeated with the respective values.

19.7. Loading and deleting additional applications

In order to protect the EstEID issuer rights, the loading and deletion of applications on to the card must be authorised by the issuer. Similarly the card issuer has the control over the structure and principles of the loaded applications.

EstEID card accepts commands forming new catalogues and data files only in command sequences secured by CMK3.

Secure loading command sequences are formed with the key responding to APDU commands by calculating MAC codes.

The loading command sequences are formed as card-specific by the card management centre and these are generally available.

The formation procedure of the additional application loading command sequence is as follows:

- The code forming the application is given to the card administrator.
- The card administrator inspects the code evaluating the memory capacity used by the application etc, and in case of confirmation begins to form secure loading code sequences.

The same procedure applies also to the application deletion command sequences.

Application loading may take place once PIN1 is checked in order to guarantee that the card user is aware of the operation.

Note: *Loading and deleting additional applications is not actively used in case of EstEID cards.*

19.8. Application loading module generation

Additional applications may be loaded on to the EstEID card, however, the card accepts them only when secured with the **card-specific CMK3** key.

We will consider the application loading module generation on the basis of an example. The following APDU command loads the application on to the card with one file with the length of 0x100 byte. The file is always readable, however, PIN1 must be checked in order to write into the file[16].

Header	Lc	Data
00E03800	77	62218201388302FFDD8410D233000001000001000000000000000018A0105A1038B0101640062178205044100110 2830200308502001 68A0105A1038B01016400731E8302003085058001019000851180010190008001 02A40795010 8830200016213820101830200 0A850201008A0105A1038B01026400

In order to secure the command we do as follows:

1) We form the following block:

Fixed value	Data block length	Data block of APDU command
81	77	62 21 82 … 02 64 00

2) We modify the command header as follows:

CLA	INS	P1	P2
0C	E0	38	00

3) We temporarily form the following block:

[16] Application programming is not considered in the present manual.

Header	Header extension to length 8	The block formed in step 1	The extension of the block formed in step 1 to eightfold length
0C E0 38 00	80 00 00 00	81 77 62 … 02 64 00	80 00 00 00 00 00 00

4) We calculate the MAC code of the block in 3DES CBC mode with **card-specific key CMK3** with ICV="00 00 00 00 00 00 00 00". In our example the result is:

MAC = **4B 5A F7 09 E8 53 10 1C**

5) We form the command as follows:

Header	Lc	The block formed in step 3	Fixed values	MAC calculated in the previous step	Le
0C E0 38 00	83	81 77 62 … 02 64 00	8E 08	4B 5A F7 09 E8 53 10 1C	00

NB! Byte Le=0 must be added at the end of the command. The application loading software using protocol T=0 must eliminate the byte before sending the command to the card.

Application loading is similar to certificate loading (see Chapter 13.3), however, catalogue EEEE does not need to be selected and the security environment no 5 must be set.

19.9. Creating and deleting files

It is possible to create new sequence files and variable-length files on EstEID cards. There could be altogether 35 files. The created files may be later deleted. In that case also the memory space used by the file and the reference in the file table will be vacated.

File creation and deletion require that PIN1 be checked and the command secured either with CMK2a or CMK2b.

Storing data in the created files similarly requires that PIN1 be checked and the command secured either with CMK2a or CMK2b.

There are no restrictions to reading from the files.

Note: Creating and deleting files is not actively used in case of EstEID cards.

19.9.1. File creation

Sequence files and variable-length record files may be created on the card.

The CREATE FILE command to create a sequence file is as follows:

Position		Value (HEX)	Comment
CLA		0x0N	N – value according to the security communication type used
INS		**0xE0**	
P1		**0x01**	
P2		0x00	
Lc		MM+4	Data field length
Data field	0..6	62**MM**8201018302	Fixed values. **MM – the number of following bytes, except 0x6400 footer. Thus MM=Lc-4.**
	7..8	FID	
	9..10	8502	Fixed values
	11..12	File size	Older byte first.
	13..15	8A0105	Fixed values
	16..17+NN	A1NN <Data>	In the current EstEID version: file access conditions. Not used in EstEID JavaCard version and the element may be left out of the command.
	18+NN..19+NN	6400	Fixed values.

The CREATE FILE command to create a variable-length sequence file is as follows:

Position		Value (HEX)	Comment
CLA		0x0N	N – value according to the security communication type used
INS		**0xE0**	
P1		**0x04**	
P2		0x00	
Lc		MM+4	Data field length
Data field	0..6	62**MM**8205044100	Fixed values. **MM – the number of following bytes, except 0x6400 footer. Thus MM=Lc-4.**
	7	Maximum record length	Maximum value: 0xFE.
	8	Maximum number of records	Maximum value: 0xFE.
	9..10	8302	Fixed values
	11..12	FID	

	13..14	8502	Fixed values
	15..16	Maximum file capacity	Older byte first.
	17..19	8A0105	Fixed values
	20..21+NN	A1NN <Data>	In the current EstEID version: file access conditions. Not used in EstEID JavaCard version and the element may be left out of the command.
	22+23..19+NN	6400	Fixed values.

19.9.2. File deletion

The files created during card personalisation or later may be deleted.

The file deletion DELETE FILE command is as follows:

CLA	INS	P1	P2	Lc	File FID
00	E4	00	02	02	NN NN

20. EstEID card's symmetric crypto operations

20.1. 3DES encrypting in CBC mode

3DES encrypting in CBC mode takes place as follows (here ICV is the initialisation vector and 3DESKey the key):

```
for ( i = 0 ;  i < NumBlocks;  i++ )
{
    if ( i == 0 ) XOR(Block[0], ICV);
    else XOR( Block[i], Block[i-1]);
    3DESEncrypt ( Block[i], 3DESKey);
}
```

20.2. In 3DES MAC CBC mode

Calculating 3DES MAC code in CBC mode takes place as follows (here 3DESKey is the key and CalculatedMAC receives the final result):

```
CalculatedMAC = Block[0];

for ( i = 0; i < NumBlocks; i++ )
{
    if ( i == (NumBlocks - 1) ) // Last block
    {
        // 3DES only with the last block
        3DESEncrypt (CalculatedMAC, 3DESKey);
    }
    else
    {
        // with all other blocks single DES
        // with the left part of 3DES key
        DESEncrypt (3DESKey[K1], CalculatedMAC);
        XOR ( CalculatedMAC, Block[i+1] );
    }
}
```

20.3. 3DES MAC in CFB mode

Calculating 3DES MAC code in CFB mode takes place as follows (here 3DESKey is the key, MACICV is the initialisation vector and CalculatedMAC receives the final result):

```
CalculatedMAC = MACICV;
DESEncrypt (CalculatedMAC, 3DESKey[K1]); // Single DES
                                         // with 3DES key
                                         // leftmost part
XOR(CalculatedMAC, Block[0]);

for ( i = 0; i < NumBlocks; i++ )
{
    if ( i == (NumBlocks - 1) ) // Last block
    {
        // 3DES only with the last block
        3DESEncrypt (CalculatedMAC, 3DESKey);
    }
    else
    {
        // with all other blocks single DES
        // with the left part of 3DES key
        DESEncrypt (3DESKey[K1], CalculatedMAC);
        XOR ( CalculatedMAC, Block[i+1] );
    }
}
```

21. EstEID file system

EstEID file system has been described in the figure below. It does not show files that external applications have no access to.

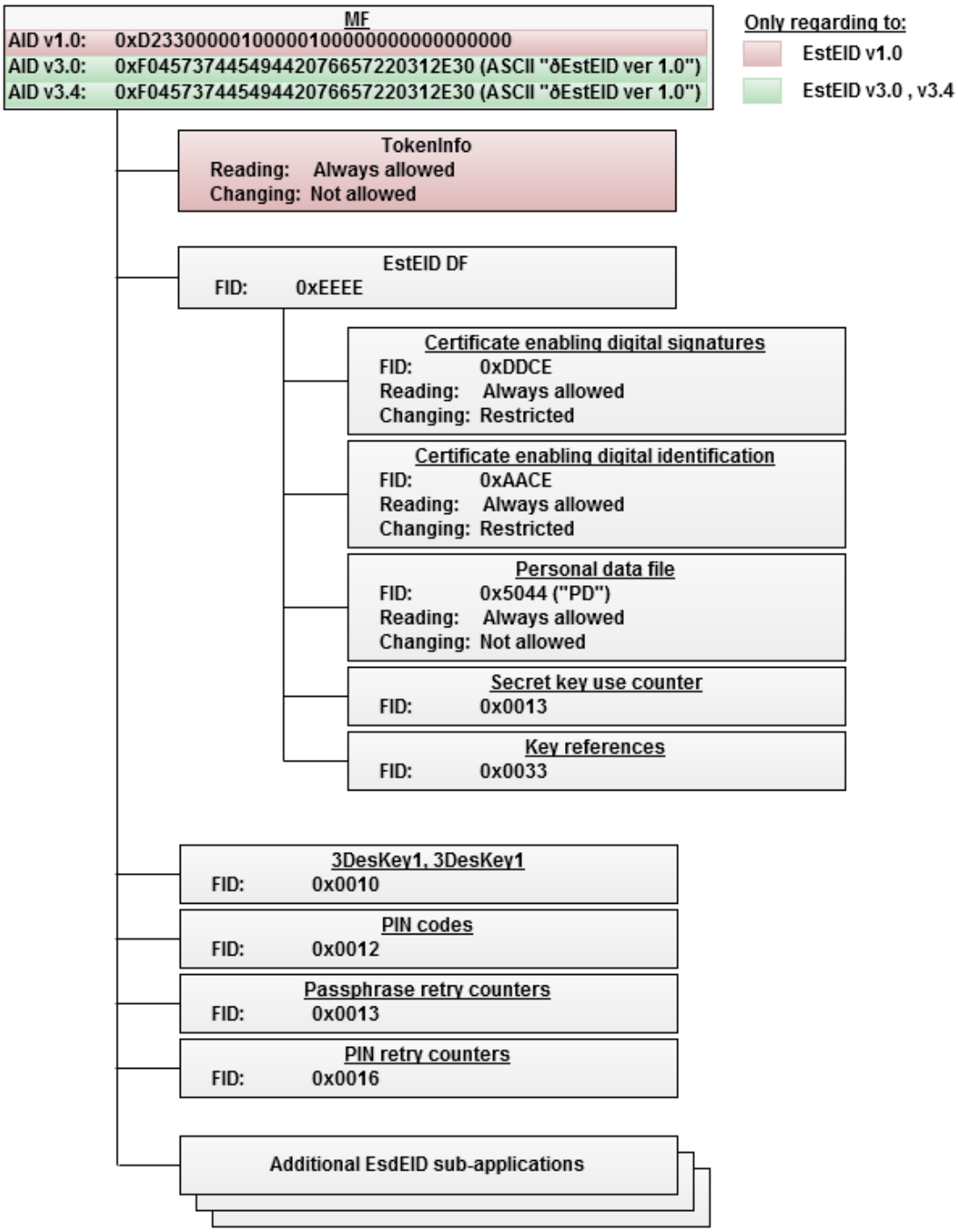

Figure 2: EstEID file system

TIP: Same AIDs as in figure are used also in Chapter 3.2 „Recognising the application".

22. Objects on EstEID card at issuance

When issued to the user, the EstEID card includes the following data:

Object	Content
Personal data file	Completed during personalisation
Authentication key	Generated on the card during personalisation
The former authentication key	Unused
Certificate enabling identification	Loaded on the card during personalisation
Signature key	Generated on the card during personalisation
The former signature key	Unused
Certificate enabling signing	Loaded on the card during personalisation
PIN1, PIN2, PUK	Set during personalisation
Card management keys CMK1, CMK2a, CMK2b, CMK3	Set during personalisation
3DESKey1 and 3DESKey2	Completed during personalisation with values 00...00

The personalisation service provider prints the PIN1, PIN2 and PUK codes in the security envelope to be delivered to the card holder.

23. EstEID card security structure

Each logical operation group on EstEID card has been located in a security environment restricting the selection of available operations so that the access is granted only to operations necessary within the given group.

For instance, operations with secret keys are not accessible during the certificate loading as the latter may take place in an insecure environment or an environment uncontrolled by the card user.

23.1. The cross-table of security environments and operations

The following table shows the security environments used in EstEID cards and the availability of the operations:

	Reading personal data file	Reading user certificates	Using the authentication key	Using the signature key	PIN1 unblocking (also with PIN1 replacement)	PIN2 unblocking (also with PIN2 replacement)	Changing PIN1, PIN2 and PUK	Assigning values to PIN1	Assigning values to PIN2	Assigning values to 3DESKey1	Assigning values to 3DESKey2	PUK unblocking and assigning new value	Generating new key pairs	Loading certificates	Creating new files in MF catalogue
PKI environment (SE#01)	ALW	ALW	PIN1	PIN2 For each op	PUK	PUK	ALW	NEV	NEV	PIN2	PIN2	NEV	NEV	NEV	NEV
PKI environment, secured communication (SE#02)	ALW	ALW	3DK1	3DK2	3DK1 and PUK	3DK2 and PUK	ALW	NEV	NEV	3DK1	3DK2	NEV	NEV	NEV	NEV
Certificate loading (SE#03)	ALW	ALW	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	PIN1 and A2	PIN1 and SC2	NEV
Replacing authentication objects (SE#04)	ALW	ALW	NEV	NEV	A1	A1	NEV	A1	A1	NEV	NEV	A1	NEV	NEV	NEV
Loading applications (SE#05)	ALW	ALW	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	NEV	PIN1 and SC3
Decrypting with PIN authentication (SE#6)	ALW	ALW	PIN1	PIN2 For each op	PUK	PUK	ALW	NEV	NEV	PUK	PUK	NEV	NEV	NEV	NEV
Decrypting with passphrase authentication (SE#7)	ALW	ALW	3DK1	3DK2	NEV	NEV	ALW	NEV	NEV	3DK1	3DK2	NEV	NEV	NEV	NEV

Definitions:

Marking	Explanation
ALW	Operation is always available
NEW	Operation is not available in the given security environment
PIN1	Operation is available, if PIN1 is checked
PIN2	Operation is available, if PIN2 is checked
PIN2 for each op	Operation is available, if PIN2 is checked and the checking is valid for only one operation
PUK	Operation is available, if PUK is checked
3DK1	Operation is available, if the communication with the card is secured with 3DESKey1

Marking	Explanation
3DK2	Operation is available, if the communication with the card is secured with 3DESKey2
A1	Operation is available, authorisation from the card management centre using CMK1
A2	Operation is available, authorisation from the card management centre using CMK2a
SC2	Operation is available, if the command sequence is secured with CMK2b
SC3	Operation is available, if the command sequence is secured with CMK3.

23.2. Comments on the security environments

1. PKI environment (SE#01)

This is the security environment for PKI operations where the user authentication is executed with PIN codes. The connection between the card and the card application is not separately secured. The security environment is suitable for secure environments with limited keyboard, such as ATM machines. Environments with full keyboard may use the security environment SE#2 where the communication between the host application and the card is secured.

2. PKI environment, secured communication (SE#02)

This is the security environment for PKI operations where the user authentication is executed with 3DESKey1 and 3DESKey2. The security environment may be used in full keyboard devices as 3DESKey1 and 3DESKey2 are derived from passphrases. As the exchange of data between the host application and the card is secured, the pirate codes have less opportunities.

3. Certificate loading environment (SE#03)

The operations used in the given security environment include the new key pair generation and certificate loading. The operations can take place once PIN1 is checked in order to guarantee that the card user is aware of the operation.

4. Authentication object replacement environment (SE#04)

Only the authentication object replacement operation may be used in the given security environment.

5. Application loading environment (SE#05)

Only the additional application loading operation may be used in the given security environment. The operation requirement is PIN1 check in order to guarantee that the card user is aware of the operation.

6. Decrypting operation environments (SE#6, SE#7)

In the given security environment, the decryption of the cryptograms formed with the card user's public keys is executed with the card user's secret keys.

24. EstEID card error messages

Error code	Mnemonics	Explanation
62 81	CorruptDataWarning	Check sum is incorrect – usually refers to a damaged card
62 82	EndOfFileWarning EndOfRecordWarning	Le is too large. Refers to an attempt to read too many bytes from the file at once (the maximum is FE).
62 83	FileInvalidWarning	The selected file is deactivated (Not in case of EstEID)
64 00	CorruptDataError ExecutionError InconsistentDataError FileInvalidError UndefinedTechnicalProblemError	Refers to serious data inconsistency either in the card application or in the command sent to the card. The error message may also result from an attempt to execute an inapplicable operation, for instance to read a secret key from the card. The inconsistency may also be caused by a damaged card.
65 81	MemoryFailureError	EstEID card EEPROM writing failed, the card is damaged.
67 00	TinyLeError	Le is too small. The given error occurs also when Le has been omitted in using T=1 protocol.
69 00	NoChvReferenceError NoKeyreferenceError	The command sent to the card refers to a PIN code or key not on the card.
69 81	IncompatibleFileStructureError	The file structure on the card does not enable the fulfilment of the command. For instance, it is attempted to add a record into a file that already consists of the maximum number of records.

Error code	Mnemonics	Explanation
69 82	SecurityStatusNotSatisfiedError	The command sent to the card requires that the card user be authorised beforehand. For instance, PIN control has not been executed. The given error occurs also in case the PIN code was checked but it was incorrect.
69 83	AuthenticationBlockedError	The number of allowed incorrect PIN insertions has been exceeded. PIN is blocked.
69 84	ReferencedDataInvalidatedError	PIN code or key cannot be used any longer.
69 85	CommandExecutionOrderError, ConditionOfUseNotSatisfiedError	The command sent to the card requires the use of such an object (usually a key) that is not allowed to be used in the given circumstances. This usually occurs when the wrong security environment has been set.
69 86	CommandNotAllowedError	The command sent to the card is not allowed in the given circumstances.
69 87	SmDataObjectsMissingError	The command sent to the card does not include certain data necessary for secure messaging.
69 88	SmDataObjectsIncorrectError	The data included in the secure command sent to the card are incorrect. The given error may occur in connection with certificate loading when the command is either incorrectly formed or it is attempted to load the certificate into the wrong card.
69 89	SmWithoutSessionkeysError	There are no session keys needed for secure messaging.
6A80	IncorrectParametersDatafieldError	Erroneous data in the command
6A82	FileNotFoundError	File has not been found
6A83	RecordNotFoundError	Record has not been found.

Error code	Mnemonics	Explanation
6A84	NotEnoughMemorySpaceError	There is not enough card memory space (should not occur in connection with the operations included in the present manual)
6A86	IncorrectParameterError	Invalid parameters in the command.
6A87	LcInconsistentWithP1P2Error	Such a Lc cannot be used in the given command.
6A88	ReferencedDataNotFoundError	No referenced data.
6A89	FileExistError	The file already exists.
6A8A	DfNameExistError	The catalogue already exists..
6B00	WrongParametersError	Wrong parameters in the command.
6D00	InstructionCodeNotSupportedError	The card does not support the instruction byte (INS) given in the sent command.
6E00	ClassNotSupportedError	The card does not support the class byte (INS) given in the sent command.
6F00	NoPreciseDiagnosisError	Refers to the fact that the card is not compatible with the given reader or interface.

25. Instructions for the writers of the applications using the card

The specification of the EstEID interface and principles given in the present document do not impose mandatory precepts for the programming of applications using EstEID card.

However, some instructions may be useful in programming the given applications.

25.1. Differences between card applications

The following subchapters describe differences between card application versions in comparison with the version v1.0.

25.1.1. Differences of application version v1.1 (DigiID)

1) v1.1 cards support only protocol T=0.

2) Information about v1.1 cards' protocol support is different in cold and warm ATR strings that are issued by the card. Cold ATR does not claim T=1 support, warm ATR mistakenly does.

3) v1.1 cards answer with OK trailer 0x9000 to all application selection commands (application recognition with SELECT FILE), regardless of whether there is an application with the given AID present in the card or not. After the application selection has been performed with a

non-present AID value then the card responds with error trailer 0x6A83 "Record not found" to all further commands, including the application selection command that tries to select the correct application. The card has to be reset in order to recover. Due to responding with 0x9000 to all application selection commands, v1.1 cards get auto-detected as "NIST SP 800-73 ![PIV]" cards in Windows 7.

4) v1.1 cards respond to application version identification (GET DATA command), with application version number 0x0101 and OK trailer 0x9000. Cards with version v1.0 respond with 0x6D00 “Instruction code not supported” instead. For more detailed description of the command, see chapter 4.1 “The identification of the card application version”.

5) v1.1 cards have only “Document number” record filled in personal data file. "Personal identification code" record is filled with 11 space characters, each of the remaining records contain one zero byte.

6) v1.1 cards respond to personal data file’s (MF/EEEE/5044) SELECT FILE command (00:A4:02:04:02:50:44) with different file header TLV value than v1.0 cards. V1.1 cards have the byte no 8 (indicating the number of records in the file) set to 0Ahex, v1.0 cards have byte no 8 set to 10hex. By default, the number of records in personal data file is 10hex as in case of v1.0 cards.

7) v1.1 cards respond with 0x63CX and decrease the PIN or PUK retry counter respectively when trying to verify PIN or PUK code by using a code value that has invalid length (the allowed range is 4..12 digits in case of PIN1, 5..12 digits in case of PIN2 and 8..12 digits in case of PUK). v1.0 cards respond with trailer 0x6A80 instead.

8) When changing the PIN or PUK code (command CHANGE REFERENCE DATA) by using a new code with an unallowed length then v1.1 cards respond with 0x6400, retry counter value is not decreased. v1.0 cards respond with trailer 0x6A80 instead.

9) When trying to unblock PIN code with the command RESET RETRY COUNTER if the code has not been blocked previously (i.e. the retry counter value is not 0), then v1.1 (and v3.0) cards respond with error trailer 0x6A86. v1.0 cards respond with error trailer 0x6985 instead.

10) v1.1 cards respond to key operations (like INTERNAL AUTHENTICATE) if the required PIN has not been verified with 0x6985 "Conditions of use not satisfied" instead of 0x6982 "Security status not satisfied".

11) Certificate loading to cards during personalization is actively used only in case of v1.1 cards.

25.1.2. Differences of application version v3.0

1) v3.0 cards answer with OK trailer 0x9000 to all application selection commands (application recognition with SELECT FILE), regardless of whether there is an application with the given AID present in the card or not.

   a. Due to responding with 0x9000 to all application selection commands, v3.0 cards get auto-detected as "NIST SP 800-73 ![PIV]" cards in Windows 7.

   b. Additionally, in some cases, after responding with 0x9000 to selecting an application that is not present in the card, version v3.0 cards erroneously respond to an undocumented APDU 80:20:00:00 (ISO VERIFY with a proprietary CLA byte) with 0x6984 “Incorrect reference data” and decrease the PUK retry counter value. If the application selection is called out repeatedly then eventually the PUK code’s retry counter expires and the code gets blocked.

2) v3.0 cards respond to application version identification (GET DATA command), with application version number 0x0300 and OK trailer 0x9000. Cards with version v1.0 respond with 0x6D00 “Instruction code not supported” instead.

    a. For more detailed description of the command, see chapter 4.1 “The identification of the card application version”.

3) v3.0 cards respond with a double 0x62 FCP tag to SELECT FILE commands which select an elementary file and require FCP value to be included in the response (i.e. the first parameter's value is set to P1=02 and the second parameter is set to P2=00). This should be taken into account when parsing the response.

4) v3.0 cards have the personal data file (MF/EEEE/5004) records padded with spaces to the maximum extent of the respective record's length. v1.0 cards have the records with the exact length.

5) Versions v3.0 and 3.4 have card numbers with the length of 9 bytes (instead of 8 bytes). This should be taken into account when reading the “Document number” record from the personal data file.

6) v3.0 cards respond to personal data file’s (MF/EEEE/5044) SELECT FILE command (00:A4:02:04:02:50:44) with different file header TLV value than v1.0 cards. v3.0 cards have the byte no 7 (indicating the maximum length of a record) and byte no 8 (indicating the number of records in the file) set to 7Fhex and 14hex respectively. v1.0 cards have byte no 7 set to 32hex and byte no 8 set to 10hex. By default, the maximum length of a record in personal data file is 32hex and the number of records is 10hex as in case of v1.0 cards.

7) v3.0 cards respond with 0x630X trailer instead of 0x63CX when trying to verify PIN or PUK code with an incorrect value (with the command VERIFY when the PIN or PUK code’s length is in its allowed value range).

8) v3.0 cards respond with 0x6984 and decrease the PIN or PUK retry counter respectively when trying to verify PIN or PUK code by using a code value that has invalid length (the allowed range is 4..12 digits in case of PIN1, 5..12 digits in case of PIN2 and 8..12 digits in case of PUK). v1.0 cards respond with trailer 0x6A80 instead.

9) v3.0 cards allow the same values for old and new PIN and PUK codes when changing a PIN or PUK code (with the command CHANGE REFERENCE DATA) or when unblocking and simultaneously changing a PIN code (with the command RESET RETRY COUNTER).

10) When changing the PIN or PUK code (command CHANGE REFERENCE DATA) by using a new code with an unallowed length then v3.0 and v3.4 cards respond with 0x6984, retry counter value is not decreased. v1.0 cards respond with trailer 0x6A80 instead.

11) When trying to unblock PIN code with the command RESET RETRY COUNTER if the code has not been blocked previously (i.e. the retry counter value is not 0), then v3.0 (and v1.1) cards respond with error trailer 0x6A86. v1.0 cards respond with error trailer 0x6985 instead.

12) v3.0 cards respond to key operations (like INTERNAL AUTHENTICATE) if the required PIN has not been verified with 0x6985 "Conditions of use not satisfied" instead of 0x6982 "Security status not satisfied".

13) Versions 3.0 and 3.4 support ECC keys.

14) Renewing certificates and generating new key pairs in the card is not actively used in case of card application versions v3.0 and 3.4.

15) v3.0 cards’ public key does not comply with ASN.1 standard. Due to that, some information systems may not accept the certificates on v3.0 cards.

25.1.3. Differences of application version v3.4

1) v3.4 cards respond to application version identification (GET DATA command) with two different application version number. Some cards respond with application version number 0x0300 and OK trailer 0x9000 and some respond with application version number 0x0304 and OK trailer 0x9000 . Cards with version v1.0 respond with 0x6D00 “Instruction code not supported” instead.

    a. For more detailed description of the command, see chapter 4.1 “The identification of the card application version”.

2) Versions v3.0 and 3.4 have card numbers with the length of 9 bytes (instead of 8 bytes). This should be taken into account when reading the “Document number” record from the personal data file.

3) v3.0 cards respond to personal data file’s (MF/EEEE/5044) SELECT FILE command (00:A4:02:04:02:50:44) with different file header TLV value than v1.0 cards. v3.0 cards have the byte no 7 (indicating the maximum length of a record) and byte no 8 (indicating the number of records in the file) set to 7Fhex and 14hex respectively. v1.0 cards have byte no 7 set to 32hex and byte no 8 set to 10hex. By default, the maximum length of a record in personal data file is 32hex and the number of records is 10hex as in case of v1.0 cards

4) v3.4 cards respond with 0x63CX and decrease the PIN or PUK retry counter respectively when trying to verify PIN or PUK code by using a code value that has invalid length (the allowed range is 4..12 digits in case of PIN1, 5..12 digits in case of PIN2 and 8..12 digits in case of PUK). v1.0 cards respond with trailer 0x6A80 instead.

5) When changing the PIN or PUK code (command CHANGE REFERENCE DATA) by using a new code with a not allowed length then the v3.4 (and v3.0) cards respond with 0x6984, retry counter value is not decreased. v1.0 cards respond with trailer 0x6A80 instead.

6) When calculating an electronic signature with hashing in the card then v3.4 cards do not return the calculated hash value as a response and it cannot be read from the card in any other way.

7) Versions 3.0 and 3.4 support ECC keys.

8) Renewing certificates and generating new key pairs in the card is not actively used in case of card application versions v3.0 and 3.4.

9) The public exponent of the secret keys is generated as a random value, except for v3.4 cards where the exponent value could be also fixed value 65537.

25.2. Determining the card application version

EstEID card application version can be determined by comparing the ATR strings issued by the card with the ATR values that have been defined for the specific EstEID card versions (see chapter "3.1. ATR strings and used protocols"). However, due to the similarities of ATR strings of some card versions, additional comparison should be conducted in the following cases:

1) If the card responds with ATR value that corresponds to the cold ATR of application v1.0:

    3B FE 94 00 FF 80 B1 FA 45 1F 03 **45 73 74 45 49 44 20 76 65 72 20 31 2E 30 43**

    In this case, in order to distinguish between application versions v1.0 and v1.1 (DigiID), do as follows:

    - Read a record from the personal data file in the card (i.e. the record of first name, last name or personal identification code).

- If the card responds with error trailer 0x6A83 “Record not found” then the card’s version is v1.1 (DigiID) as v1.1 cards contain data only in “Document number” field. Otherwise, the card's version is v1.0.

2) If the card responds with either cold or warm ATR value of any of v3.0 cards, then in order to distinguish between card versions 3.0 and 3.4, do as follows:

- Send an undocumented APDU command to the card in the form of “80 20 00 00” (VERIFY command of PUK code).
- Card v3.0 responds with trailer 0x6984 to the command. Card v3.4 responds with trailer 0x63CX.

Note: *Sending the abovementioned VERIFY command to the card causes the PUK retry counter value to be decreased by one.*

25.3. Card APDU message chaining

This chapter explains the basis of APDu message chaining in case of larger data to be transmitted.

JavaCard framework 2.2.2 and above support extended length APDU messages. Unfortunately EstEID v3.0 and v3.4 mimicry EstEID v1.0 and do not have implemented extended length APDU processing. Thus the data to be transmitted between the host and the card which doesn’t fit into the usual length of APDU messages must be transmitted as chained.

EstEID applications do not respond with data longer than maximum of standard APDU response messages. Thus there is no need for message chaining in given case.

25.3.1. APDU command chaining

EstEID cards use APDU command chaining to receive data from host for its internal processing. Data is devided into several blocks before sent to the card. These blocks are sent to the card successively one by one. By sending chained commands, inside the card the data is concatenated for processing.

To inform the card that APDU chaining is being used the CLA byte in command APDU must be set to 0x10. Only the very last APDU command in chaining sequence must have CLA with usual value 0x00. This informs the card that no further data is being sent.

EstEID applications use APDU command chaining only for signing and decrypting operations.

For example we have 3 blocks of data we have to send to the card. In this case first two APDU commands look something similar to following with CLA byte set to 0x10:

1) First APDU command:

CLA	INS	P1	P2	Lc	Block nr1	Le
10	XX	XX	XX	40	01…40	00

2) Second APDU command:

CLA	INS	P1	P2	Lc	Block nr1	Le
10	XX	XX	XX	40	01…40	00

Last APDU command has CLA with value 0x00 informing the card that no more data needs to be waited from host:

3) Third and last APDU command:

CLA	INS	P1	P2	Lc	Block nr1	Le
00	XX	XX	XX	32	01…32	00

Inside the card these blocks will be concatenated and ready for performing some operation with them.

25.4. Variable-length PIN codes

The PIN and PUK codes of EstEID chip are variable-length codes and the application cannot foresee the code length. Thus the application must allow the entry of variable-length codes.

25.5. Minimising transaction time

As EstEID card is used in environments where various programs are used simultaneously and the user's control over the environment is generally low, it is recommended to programme the applications so that the card operations would be executed in a short period of time (one after another) and the card be switched off after the completion of operations (receiving the results) but still in the application's use. Firstly, the given action reduces the possibility that the user removes the card from the reader prematurely and secondly, avoids the pirate code access to the card.

25.6. Signing application codes

Application codes must be signed so that the user could check the authenticity of the software processing the EstEID card.

Appendix

1. MICARDO related APDU commands and responses

Following appendix consists APDU command descriptions for v1.0 EstEID cards. Same card behaviour is implemented also in following EstEID versions so it could mimicry the first version.

EstEID v1.0 card can respond to commands that are not described here. Appendix consists of descriptions for commands that are used in current document.

Appendix Table 1-1 APDU commands

Command	INS	Brief description	See page
CHANGE REFERENCE DATA	'24'	Changes the password for cardholder authentication	117
CREATE FILE	'E0'	Creates a directory or data field	120
EXTERNAL AUTHENTICATE /MUTUAL AUTHENTICATE	'82'	Authenticates the external world / external world and chip card	121
GENERATE PUBLIC KEY PAIR	'46'	Generates the public and the private part of a RSA key pair.	124
GET CHALLENGE	'84'	Generates and outputs a random number	125
GET RESPONSE	'C0'	Reads out the response data (T=0)	126
INTERNAL AUTHENTICATE	'88'	Authenticates the chip card or application	117
MANAGE SECURITY ENVIRONMENT	'22'	Passes on key references, random numbers, and data to be used for key derivation	129
PERFORM SECURITY OPERATION	'2A'	Various functions using symmetrical and asymmetrical keys: - COMPUTE DIGITAL SIGNATURE - DECIPHER - HASH	131 132 133 135
READ BINARY	'B0'	Reads from a transparent data field	137
READ RECORD	'B2'	Reads from a formatted data field	138
RESET RETRY COUNTER	'2C'	Resets the counter of failed attempts	139
SELECT FILE	'A4'	Selects a directory or data field	141
UPDATE BINARY	'D6'	Modifies a transparent data field	143
UPDATE RECORD	'DC'	Modifies a formatted data field	144
VERIFY	'20'	Authenticates the cardholder	146

1.1. Command Class

The following table lists the possible values for the class byte (CLA) in the command header.

Appendix Table 1-2					APDU Command Classes
b8...b5	b4	b3	b2	b1	Description
x					Type of command :
´00 00´	…	…	…	…	Command and response APDU as specified by ISO; in chaining mode, this is the only command or the last one
´00 01´	…	…	…	…	Indicates chaining mode as specified by ISO
…	x	x	…	…	Secure messaging format (SM):
…	0	0	…	…	No SM, or SM is not shown
…	0	1	…	…	RFU, treated as an error
…	1	0	…	…	SM, file header is not MAC-secured
…	1	1	…	…	SM, file header is MAC-secured
…	…	…	x	x	Logical channel number

1.2. CHANGE REFERENCE DATA

The command CHANGE REFERENCE DATA launches a comparison between the data that accompanied the command and a reference value stored in the chip card. The parameter P2 specifies which reference value is to be used for the comparison.

If the comparison is acceptable, the security status of the chip card is changed and the old password replaced by the new password contained in the command data.

In the command data, the old password is immediately followed by the new one, without any kind of separator. The additional information relating to the password, which is referenced by the parameter P2, tells the chip card the transfer format and the length of the old password. This information is sufficient to enable it to explicitly extract the old password and the new one from the command data.

TIP: Suppose that a user wishes to replace his old password ABC by the new one 123. If he inadvertently enters ABCD for the old password, then the data field for the command will contain ABCD123. Since the beginning of the data field does correctly correspond to the old password, the command CHANGE REFERENCE DATA will be executed and will mistakenly set up the new password as D123. Since no separator is used, the chip card has no way of recognizing this situation as an error unless both the old password and the new one are of maximum length.

Appendix Table 1-3		Change Reference Data command APDU	
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´24´	Command code
P1	1	´00´	Fixed value
P2	1	´XX´	Check byte (+)
Lc	1	´01´...´FF´	Length of the command data

Appendix Table 1-3	Change Reference Data command APDU		
	Length	Contents	Description
Data	Lc	´XX...YY´	Old password/new password, both in the same transfer format
Le	0	-	Empty

Appendix Table 1-4	Change Reference Data response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	No errors and no warnings
		´63 CX´	As for ´90 00´, but problems with writing

Appendix Table 1-5	Change Reference Data response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Inconsistent or incorrect data
		´65 81´	Write error
		´66 12´	Parity error in the cryptographic algorithm
		´67 00´	Response data has incorrect length, Lc is missing or incorrect, Le is superfluous
		´69 82´	Security conditions not fulfilled
		´69 83´	Retry counter expired
		´69 85´	Referenced password cannot be used
		´69 88´	Incorrect data objects for secure messaging
		´6A 80´	Password has incorrect transport format or old and new passwords are identical
		´6A 83´	Record not found
		´6A 86´	P1 or P2 possesses an illegal value
		´6A 88´	Referenced password not found
		´6E 00´	CLA and INS are inconsistent
		*	Trailers from the VERIFY command are also possible

Appendix Table 1-6	Coding of P2 for Change Reference Data								
b8	b7	b6	b5	b4	b3	b2	b1	Description
x	…	…	…	…	…	…	…	Password:

1	…	…	…	…	…	…	…	Global, all information in the MF
0	…	…	…	…	…	…	…	Local, all information under the MF
…	0	0	0	0	0	…	…	Other values are RFU
…	…	…	…	…	…	x	x	Password number (PwdID)

1.3. CREATE FILE

The command CREATE FILE enables you to create a lower level file within the current directory. All the necessary information regarding the directory or data field that is to be created is contained in the command data.

In general it will not be possible to accommodate all the information within one command message, so for creating new directories you can execute the command in chaining mode.

If a successful subcommand is followed by a reset or some other command, then the command CREATE FILE will be aborted and the reset or the other interrupting command will be executed. Any changes already carried out by the command CREATE FILE will be retained.

The command data may contain a variety of data objects:

1. File header for a DF that is to be created
2. File header for an EF that is to be created
3. Record data, tag ´73´ with the following data elements:

Tag	Length	Value
´83´	´01´	FID
´85´	´XX...YY´	Contents of the records
´85´	´XX...YY´	Continuation of record contents, if necessary
´85´	´XX.....	....

The above-mentioned data objects can be used as follows:

For a DF:	Data object 1 at the beginning, followed by <n> occurrences of data objects 2 and 3 in an arbitrary order
For an EF:	One data object 2

Appendix Table 1-7		Create File command APDU	
	Length	Contents	Description
CLA	1		Command class
		´0X´	Final subcommand
		´1X´	Subcommand, not the final command
INS	1	´E0´	Command code

Appendix Table 1-7	Create File command APDU		
	Length	Contents	Description
P1	1	´XX´	First byte of the file descriptor
P2	1	´00´	Fixed value
Lc	1	´01´...´FF´	Length of the command data
Data	Lc	´XX...YY´	Data objects 1, 2 and/or 3 (see above)
Le	0	-	Empty

Appendix Table 1-8	Create File response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	Command was performed correctly
		´63 CX´	As for ´90 00´, but problems with writing

Appendix Table 1-9	Create File response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Current DF is deactivated, inconsistent or incorrect data
		´65 81´	Write error
		´67 00´	Response data has incorrect length, Lc is missing or incorrect, Le is superfluous
		´69 82´	Security conditions not fulfilled
		´69 88´	Incorrect data objects for secure messaging
		´6A 80´	Error in the content of the command data
		´6A 84´	Not enough memory available
		´6A 86´	P1 or P2 possesses an illegal value or P1 varies within a chain
		´6A 89´	FID already present in the current DF
		´6A 8A´	Name of the DF is already present in the chip card
		´6E 00´	CLA and INS are inconsistent or the subcommands have different low half-bytes

1.4. EXTERNAL AUTHENTICATE / MUTUAL AUTHENTICATE

The command EXTERNAL AUTHENTICATE modifies the security status of the chip card, provided the encrypted data contained in the command matches the token that was previously requested from the external world. The key that is used for this purpose is either specified by the parameter P2 or referenced via the active security environment of the current directory. Which cryptographic algorithm is to be used can be derived from the additional information for the referenced key.

It is absolutely essential, before issuing the command EXTERNAL AUTHENTICATE, to use the command GET CHALLENGE to pass a random number to the external world.

In the variant MUTUAL AUTHENTICATE, the card token that has been processed by the external world is received together with a random number that is then processed by card and sent back to the external world. This makes it possible to perform mutual authentication with only one command.

Appendix Table 1-10	External/Mutual Authentication command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´82´	Command code
P1	1	´00´	Fixed value
P2	1	´00´	Fixed value
Lc	1	´01´...´FF´	Length of the command data
Data	Lc	´XX...YY´	Command data (+)
Le	0 1	- ´00´...´FF´	Empty for one-sided authentication Length of the response data

Appendix Table 1-11	External/Mutual Authentication response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0 var.	- ´XX...YY´	Empty for one-sided authentication Mutual authentication
Trailer	2	´90 00´	Authentication was successful
		´63 CX´	As for ´90 00´, but problems with writing, X indicates the number of attempts
		´63 CX´	Authentication not successful, X indicates the retry counter for the key

Appendix Table 1-12	External/Mutual Authentication response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´65 81´	Write error
		´67 00´	Response data has incorrect length, Lc is missing or incorrect, Le is superfluous
		´69 00´	Missing key reference
		´69 82´	Security condition for K0.IFD not fulfilled
		´69 83´	Retry counter has expired

		´69 84´	Usage counter has expired
		´69 85´	Incorrect algorithm ID or second key reference is missing, incorrect command sequence, referenced key cannot be used
		´69 98´	Error encountered while deriving the session key
		´6A 86´	P1 or P2 possesses an illegal value
		´6A 88´	Referenced key not found
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-13			Coding of P2 for External/Mutual Authentication					
b8	b7	b6	b5	b4	b3	b2	b1	Description
1	…	…	…	…	…	…	…	Global key, all information in the MF
0	…	…	…	…	…	…	…	Local key, all information under the MF
…	0	0	…	…	…	…	…	Other values are RFU
…	…	…	x	x	x	x	x	Key reference: KID || KV = ´FF´

The following table shows how the command data is to be laid out depending on the algorithm ID, i.e. depending on the security feature:

Procedure/Algorithm ID	Command data
Authentication, one-sided external, DES { 2 1 2 1 }	eK(RND.ICC) Length: 8 bytes
Authentication, mutual external first, DES { 2 2 1 2}	eK(RND.ICC) || RND.ICC Lengths: 8 bytes || 8 bytes
Key management, key transport, negotiation DES3, 1 SK, without SSC { 3 2 1 1 1 0 } DES3, 1 SK, with SSC { 3 2 1 1 3 0 }	eK(RND.IFD || RND.ICC || K.IFD) Lengths: 8 bytes || 8 bytes || 16 bytes
Key management, key transport, negotiation DES3, 2 SK, without SSC { 3 2 1 1 2 0 } DES3, 2 SK, with SSC { 3 2 1 1 4 0 }	eK(RND.IFD || RND.ICC || K.IFD) Lengths: 8 bytes || 8 bytes || 32 bytes

1.5. GENERATE PUBLIC KEY PAIR

The command GENERATE PUBLIC KEY PAIR generates the private and the public part of a RSA key pair.

First, all required EEEE/0010 data fields have to be created. The TLV structure in the data fields has to be defined, too.

The private part of the RSA key is stored in the EF_Key_RSA_Private data field. The command expects that the FID is referenced in the EEEE/0013 data field.

The public part of the RSA key is stored in the EF_Key_RSA_Public data field. The command expects that the necessary information for key generation is available.

TIP: Chapter 19.4 “Generating new key pairs” contains a detailed description of an RSA key generation.

Appendix Table 1-14	Get Public Key Pair command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´46´	Command code
P1	1	´00´	Fixed value
P2	1	´00´	Fixed value
Lc	1	´04´	Length of the command data
Data	Lc	´83 02 XX...YY´	FID of EF_Key_RSA_Public
Le	0	-	Empty

Appendix Table 1-15	Generate Public Key Pair response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	Retry counter is set to its initial value (from the command)

Appendix Table 1-16	Generate Public Key Pair response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´65 81´	EF_Key_RSA_Public is incorrect
		´67 00´	Lc is missing or incorrect, Le exists
		´69 00´	Missing key reference

		´69 82´	Security condition for key not fulfilled
		´69 85´	Key reference is missing or incorrect
		´6A 80´	Command data are wrong
		´6A 82´	EF_Key_RSA_Public is missing
		´6A 86´	P1 or P2 possesses an illegal value
		´6E 00´	CLA and INS are inconsistent

The following table describes the command data:

Appendix Table 1-17	Generate Public Key Data command data description
Byte number	Command data
1	Value: '83'
2	Value: '02'
3-4	FID of the EF_Key_RSA_Public

1.6. GET CHALLENGE

The command GET CHALLENGE returns a certain number of bytes. Since the contents of these bytes are determined by an internal hardware random number generator, the byte sequence can be considered as a random number.

The number of bytes required is specified in the command.

This command is generally used to prepare chip card input data that is to be encrypted, or transferred with the protection of a cryptogram. The random number supplied by the chip card is then used to prepare the input data as IcvSecureMessaging.

Appendix Table 1-18	Get Challenge command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´84´	Command code
P1	1	´00´	Fixed value
P2	1	´00´	Fixed value
Lc	0	-	Empty
Data	0	-	Empty
Le	1	´00´ '01...´FF´	Length of the response data: Lr = 8 bytes Lr = Le

Appendix Table 1-19	Get Challenge response APDU if the command was completed successfully		
	Length	Contents	Description
Data	Var.	´XX...YY´	Random number
Trailer	2	´90 00´	No errors and no warnings

Appendix Table 1-20	Get Challenge response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´67 00´	Lc is superfluous, Le is missing
		´69 85´	Could not create random number
		´69 86´	P1 or P2 possesses an illegal value
		´6E 00´	CLA and INS are inconsistent

1.7. GET RESPONSE

The command GET RESPONSE is used to send response data from the chip card to the external world.

TIP: The command is relevant only for the chip card protocol T=0.

Appendix Table 1-21	Get Response command APDU		
	Length	Contents	Description
CLA	1	´00´	Command class
INS	1	´C0´	Command code
P1	1	´00´	Fixed value
P2	1	´00´	Fixed value
Lc	0	-	Empty
Data	0	-	Empty
Le	1	´00´...´FF´	Response data expected

Appendix Table 1-22	Get Response response APDU if the command was completed successfully		
	Length	Contents	Description
Data	var.	´XX...YY´	Data requested from user data
Trailer	2	´90 00´	No errors and no warnings

Appendix Table 1-22	Get Response response APDU if the command was completed successfully		
	Length	Contents	Description
		´62 82´	Length of the response data is less than Le
		´62 81´	User data and checksum are inconsistent

Appendix Table 1-23	Get Response response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´67 00´	No errors and no warnings
		´69 85´	Length of the response data is less than Le
		´6A 86´	P1 or P2 possesses an illegal value
		´6E 00´	CLA and INS are inconsistent
		´6C XY´	Send same command with Le = 'XY'

1.8. INTERNAL AUTHENTICATE

The command INTERNAL AUTHENTICATE has the effect of generating authentication data from a token passed to the chip card. The key that is used for this purpose is either specified by the parameter P2 or referenced via the active *security environment* of the current directory.

The additional information regarding the referenced key indicates how the response data is to be derived from the token.

Appendix Table 1-24	Internal Authenticate command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´88´	Command code
P1	1	´00´	Fixed value
P2	1	´00´ ´XY.	Fixed value Key reference (+)
Lc	0 1	- ´01´...´FF´	No command data Length of the command data
Data	0 Lc	- ´XX...YY´	Token is already present in the chip card Token that is to be encrypted or signed
Le	1	´00´...´FF´	Length of the response data

Appendix Table 1-25	Internal Authenticate response APDU if the command was completed successfully		
	Length	Contents	Description

Appendix Table 1-25	Internal Authenticate response APDU if the command was completed successfully		
	Length	Contents	Description
Data	var.	´XX...YY´	Encrypted or signed token
Trailer	2	´90 00´	No errors and no warnings
		´63 CX´	Repeated write attempts were necessary

Appendix Table 1-26	Internal Authenticate response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´65 81´	Write error
		´67 00´	Lc is missing or incorrect, Le is missing, Le is smaller than Lr
		´69 00´	Missing key reference
		´69 82´	Security condition for K0.ICC not fulfilled
		´69 83´	Retry counter has expired
		´69 84´	Usage counter has expired
		´69 85´	Incorrect algorithm ID or second key reference is missing, incorrect key length, referenced key cannot be used
		´69 98´	Error encountered while deriving the session key
		´6A 86´	P1 or P2 possesses an illegal value
		´6A 88´	Referenced key not found
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-27	Coding of P2 for Internal Authenticate							
b8	b7	b6	b5	b4	b3	b2	b1	Description
0	…	…	…	…	…	…	…	Global key, all information in the MF
1	…	…	…	…	…	…	…	Local key, all information under the MF
…	0	0	…	…	…	…	…	Other values are RFU
…	…	…	x	x	x	x	x	Key reference: KID || KV = ´FF´

The following table shows how the command data is to be laid out depending

on the algorithm ID, i.e. depending on the security feature:

Procedure/Algorithm ID	Command data

Procedure/Algorithm ID	Command data
Authentication, one-sided internal, DES { 2 1 1 1 } internal, DES3 { 2 1 1 2 }	RND.IFD Length: 8 bytes
Authentication, one-sided internal, RSA { 2 1 1 4 }	RND.IFD Length: 1 byte <= length RND.IFD <= 40 % modulus length

1.9. MANAGE SECURITY ENVIRONMENT

There are two variants of the command MANAGE SECURITY ENVIRONMENT:

1. The SET variant, in which P1=´X1´

   This variant modifies components of the active security environment in the current directory. The parameters P1 and P2 specify which CRT components are affected. New values for the affected components are passed on in the command.

   This variant can also be used to pass data for key derivation and initiate such a key derivation process.

2. The RESTORE variant, in which P1=´F3´

   This variant replaces the active security environment in the current directory by one which is permanently stored in the chip card. The parameter

   P2 specifies the number of this security environment, which comes into effect straightaway.

Appendix Table 1-28	Manage Security Environment command APDU		
	Length	Contents	Description
CLA	1	´00´	Command class
INS	1	´22´	Command code
P1	1	´X1´ ´F3´	SET Manipulates the active SE in the current DF (+) RESTORE Replaces the active SE in the current DF (+)
P2	1	´´XY.	P1=´X1´: tag of a CRT component P1=´F3´: SE#
Lc	1 0	´XY´ -	P1=´X1´: length of the command data P1=´F3´: empty

Appendix Table 1-28	Manage Security Environment command APDU		
	Length	Contents	Description
Data	Lc 0	´XX...YY´ -	P1=´X1´: command data P1=´F3´: empty
Le	0	-	Empty

Appendix Table 1-29	Manage Security Environment response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	No errors and no warnings

Appendix Table 1-30	Manage Security Environment response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´67 00´	Lc is not equal to the length of the command data, Lc is missing or is superfluous, Le is missing or is superfluous
		´69 85´	Chip card contains no data from which to derive a key, referenced key cannot be used
		´6A 80´	Incorrect command data
		´6A 86´	P1 or P2 possesses an illegal value
		´6A 88´	Referenced key not found
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-31	Manage Security Environment command parameter P1 coding							
b8	b7	b6	b5	b4	b3	b2	b1	Description
x	x	x	x	0	0	0	1	SET variant:
1	…	…	…	0	0	0	1	- External and mutual authentication (AT)
…	1	…	…	0	0	0	1	- Data authentication (DST), data confidentiality (CT), internal and mutual authentication (AT)
…	…	1	…	0	0	0	1	- Secure messaging for response to message (CCT, CT)
…	…	…	1	0	0	0	1	- Secure messaging for command message (CCT, CT)
1	1	1	1	0	0	1	1	RESTORE
All other values RFU								Not supported

Appendix Table 1-32	Manage Security Environment command parameter P2 coding
P2	Significance
	RESTORE variant:
´XY´	Number of the new SE, ´XY´=´01´...´FE´ except ´EF´
	SET:
´A4´	AT components in the data portion of the command
´B4´	CCT components in the data portion of the command
´B6´	DST components in the data portion of the command
´B8´	CT components in the data portion of the command
All other values RFU	

1.10. PERFORM SECURITY OPERATION

The command PERFORM SECURITY OPERATION (PSO) can be used to call a variety of cryptographic algorithms. The following algorithms are offered:

1. Computation of a digital signature
2. Hashing of data
3. Decryption of data

These functions are specified as PSO commands. They access various cryptographic algorithms and keys, so you may need to adjust the security environment before you execute a PSO command. This will usually be done by calling the command MANAGE SECURITY ENVIRONMENT.

The desired PSO command is specified using parameters P1 and P2. The contained data may vary depending on which PSO command is used.

Appendix Table 1-33	Perform Security Operation command APDU		
	Length	Contents	Description
CLA	1	 ´0X´ ´1X´	Command class Final subcommand Subcommand, not the final command
INS	1	´22´	Command code for PSO command
P1	1	´XX...YY´	Tag of the data object that is used in the response data (+)
P2	1	´XX...YY´	Tag of the data object compiled from Lc and Data (+)
Lc	0 1	- ´XX...YY´	Empty Length of the command data (Length)
Data	0 Lc	- ´XX...YY´	Empty Command data (Value)
Le	0	-	Empty

Appendix Table 1-33	Perform Security Operation command APDU		
	Length	Contents	Description
	1	´00´...´FF´	Length of the response data

Appendix Table 1-34	Perform Security Operation response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´6A 86´	Illegal combination of P1 and P2 for PSO commands

Appendix Table 1-35	Perform Security Operation command permitted P1 and P2 combinations
P1 P2	PSO Function
´9E 9A´	COMPUTE DIGITAL SIGNATURE
´80 86´	DECIPHER
´90 80´	HASH, command data is not TLV coded
´90 A0´	HASH, command data is TLV coded

1.10.1. COMPUTE DIGITAL SIGNATURE

The PSO command COMPUTE DIGITAL SIGNATURE calculates a digital signature using an asymmetrical cryptographic algorithm and a referenced private key. The calculation is performed according to the algorithm ID specified by the key reference.

The desired key for the calculation must previously have been set up using the command MANAGE SECURITY ENVIRONMENT, unless it has already been defined. The referenced key is located in the relevant data field EF containing key.

You can select the cryptographic algorithm:

- Digital signature using RSA, DSI according to PKCS#1

EstEID card formats the transferred data according to the format that is currently set up and returns the calculated digital signature in its response.

The program supports only algorithm IDs that are defined for the calculation of digital signatures.

TIP: This command can use a command specific access condition. In this case, the command expects in the SC data object only one further data object with the tag ' A4'. It could be of the type AUT or of the type PWD. This indicates that prior to each signature creation a key or password based authentication must be performed. The security state is reset during command execution.

Appendix Table 1-36	Compute Digital Signature command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´2A´	Command code for PSO command

Appendix Table 1-36	Compute Digital Signature command APDU		
	Length	Contents	Description
P1	1	´9E´	Plaintext data object, not TLV coded
P2	1	´9A´	Padding indicator followed by a cryptogram
Lc	0 1	- ´XX´	Length of the command data in bytes
Data	0 ´XX´	- ´XX...YY´	Command data, not TLV coded
Le	1	´00´...´FF´	Length of the response data

Appendix Table 1-37	Compute Digital Signature response APDU if the command was completed successfully		
	Length	Contents	Description
Data	8	´XX...YY´	Plaintext dataDigital signature
Trailer	2	´90 00´	Command was performed correctly

Appendix Table 1-38	Compute Digital Signature response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Hardware fault during the operation, inconsistencies in EEEE/0013
		´64 00´	Inconsistent or incorrect data
		´65 81´	Error encountered while writing the retry counter
		´67 00´	Response data has incorrect length, Lc is not equal to the length of the command data, Le is smaller than Lr
		´69 82´	Access conditions for the key not fulfilled
		´69 83´	Retry counter is zero
		´69 84´	Usage counter is zero
		´69 85´	Usage conditions for the key not fulfilled, referenced key cannot be used
		´69 88´	DSI has incorrect length, incorrect dataobjects for secure messaging
		´69 88´	Key or hash function not found, referenced key not found

1.10.2. DECIPHER

The PSO command DECIPHER decrypts a cryptogram that is passed together with the command. The following cryptographic algorithms can be used:

- DES and DES3 decryption in CBC mode

- RSA decryption using the private key

The command MANAGE SECURITY ENVIRONMENT is used to specify the key that is to be used and hence the cryptographic algorithm. The length of the response data is always a multiple of the length of the key. The first byte of the command data is the padding indicator which shows whether the chip card processes the plaintext data at the end of the operation, and if so, how. If a defined padding indicator is present, card will remove the padding bytes and output only the relevant plaintext data.

DES/DES3:

In the case of DES/DES3, the input has to be a multiple of 8 bytes plus padding indicator. The only defined padding indicator is ´01´ (ISO padding).

If the key requires an ICV, you must set up one using the command MANAGE SECURITY ENVIRONMENT. After the command call the current block is output decrypted. The decryption uses a key from the relevant data field EF containing key. This key must be authorised for the decryption.

RSA:

In the case of RSA the chip card decrypts the transferred cryptogram (length = modulus) using a private key from the relevant data field EF containing key. This key must be authorised for the decryption. Card assumes that the plaintext data of the cryptogram is formatted according to PKCS#1 (block type 01).

The byte ´00´ is placed before the code in the command call as the padding indicator (no further indication, PKCS#1 is assumed). Card interprets the plaintext data sufficiently that only data which is identified as plaintext data is output. When the decryption has been carried out, card also checks whether the data has the correct PKCS#1 format. If this is not the case, then no response data will be output and the retry counter for the private key will be decreased.

The program supports only algorithm IDs that are defined for decryption.

Appendix Table 1-39	Decipher command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´2A´	Command code for PSO command
P1	1	´80´	Plaintext data object, not TLV coded
P2	1	´86´	Padding indicator followed by a cryptogram
Lc	1	´XX´	Length of the command data in bytes
Data	Lc	´XX...YY´	Command data, not TLV coded
Le	1	´00´...´FF´	Length of the response data

Appendix Table 1-40	Decipher response APDU if the command was completed successfully		
	Length	Contents	Description
Data	8	´XX...YY´	Plaintext data
Trailer	2	´90 00´	Command was performed correctly

Appendix Table 1-41	Decipher response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty

Appendix Table 1-41		Decipher response APDU in the event of an error	
	Length	Contents	Description
Trailer	2	´64 00´	Hardware fault during the operation, inconsistencies in file with FID 0013, inconsistent or incorrect data
		´65 81´	Error encountered while writing retry counter or usage counter
		´67 00´	Response data has incorrect length, Lc is not equal to the length of the command data, Le is smaller than Lr
		´69 82´	Access conditions for the key not fulfilled
		´69 83´	Retry counter is zero
		´69 84´	Usage counter is zero
		´69 85´	Usage conditions for the key not fulfilled
		´69 85´	Referenced key cannot be used
		´69 88´	Contents of the cryptogram are not correct
		´69 88´	Contents of the cryptogram are not correct
		'6A 88´	Referenced key not found

1.10.3. HASH

The PSO command HASH calculates a 160-bit hash value from the data transferred in the command. The calculated hash value is output in the response to the PSO command, in chaining mode it is output in the response to the last block.

The result is also stored temporarily in RAM, to be used in the event that the next command is COMPUTE DIGITAL SIGNATURE or INTERNAL AUTHENTICATE.

We distinguish two versions of this PSO command:

Hash performed entirely within the chip card:

The plaintext data passed in the command is formatted by the chip card as specified by the SHA conventions (padding). The command can be used in chaining mode; in this case a plaintext block must always have a length of 64 bytes. The final block may be smaller, in which case it will be padded by the card.

Hash of previous round(s):

If the task of hashing is shared between the card and the external world, the data field is laid out as follows:

1. Without chaining mode or one single block
   (data object with tag ´90´, followed by one data object with tag ´80´):
   - data object ´90´ with 20 bytes of a hash value block, followed by 8 bytes of counter (number of bits already hashed)
   - data object ´80´ containing the unpadded text remaining to be hashed (of length up to 64 bytes). The padding will be carried out by the card, if necessary.
2. Using chaining mode:
   - First command: data object ´90´ (28 bytes, laid out as for 1)
   - Second to (n-1)th command: data object ´80´ and 64 bytes of text

- nth command: data object ´80´ containing the final unpadded text block (of length up to 64 bytes), the padding will be carried out by the chip card, if necessary.

Appendix Table 1-42	Hash command APDU		
	Length	Contents	Description
CLA	1	´00´ ´10´	Final command in chaining mode Chaining mode, additional command calls will follow
INS	1	´2A´	Command code for PSO command
P1	1	´90´	Create hash value
P2	1	´80´ ´A0´	Plaintext data object, not TLV coded TLV coded data objects in the data field
Lc	1	´XX´	Length of the command data in bytes
Data	Lc	´XX...YY´ ´XX...YY´ ´XX...YY´ ´XX...YY´ ´XX...YY´	Message data, not TLV coded, or ´90 14´ - ´ XX...YY´ hash value ´90 1C´ - hash block + 8-bytes counter || ´80´ - L - text to be hashed ´80 40´ - text to be hashed in chaining mode ´80´ - L - final portion of text to be hashed (nth command)
Le	1	´00´...´FF´	Length of the response data

Appendix Table 1-43	Hash response APDU if the command was completed successfully, CLA = '00'		
	Length	Contents	Description
Data	20	´XX...YY´	Hash value
Trailer	2	´90 00´	Command was performed correctly

Appendix Table 1-44	Hash response APDU if the command was completed successfully, CLA = '10'		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	Command was performed correctly

Appendix Table 1-45	Hash response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´67 00´	Lc is not equal to the length of the command data, Le is not equal to

			20 bytes, or Le is superfluous
		´69 87´	Counter or hash value missing
		´69 88´	Length of data object is not 28 or 64 bytes

1.11. READ BINARY

The command READ BINARY outputs all or part of the contents of a transparent data field.

The data field can either be selected previously, using the command SELECT FILE, or you can use the short file identifier (SFI) given in parameter P1 to make this data field the current data field.

In the command, the parameters P1 and P2 specify the first byte to be read relative to the beginning of the data field and Le supplies the number of bytes that are to be read.

Appendix Table 1-46	Read Binary command APDU		
	Length	Contents	Description
CLA	1	´00´	Command class
INS	1	´B0´	Command code
P1	1	´XX´	SFI or MSB of the offset (+)
P2	1	´00´...´FF´	LSB of the offset
Lc	0	-	Empty
Data	0	-	Empty
Le	1	´00´...´FF´	Length of the response data

Appendix Table 1-47	Read Binary response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0 var.	- ´XX...YY´	Empty, no data available Response APDU or part of it
Trailer	2	´90 00´	No errors and no warnings
		´61 XX´	Additional response data in the chip card

Appendix Table 1-48	Read Binary response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´67 00´	Lc is superfluous, Le is missing

		´69 85´	There is no data to be fetched
		´6A 86´	P1 or P2 possesses an illegal value
		´6E 00´	CLA and INS are inconsistent
		´6C XY´	Send same command with Le = 'XY'
		´69 88´	Incorrect data objects for secure messaging
		´6A 82´	File not found via SFI
		´6A 86´	Parameters P1 - P2 incorrect
		´6B 00´	Offset is outside the EF
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-49 Coding of P1 for Read Binary								
b8	b7	b6	b5	b4	b3	b2	b1	Description
0	x	x	x	x	x	x	x	Offset (MSB)
1	0	0	x	x	x	x	x	5-bit SFI

1.12. READ RECORD

The command READ RECORD outputs the entire contents of a record within a linear data field.

The data field can either be selected previously, using the command SELECT FILE, or you can use the short file identifier (SFI) given in parameter P2 to make this data field the current data field.

In the command, the parameters P1 and P2 specify which record is to be read and Le supplies the length of the response data.

Appendix Table 1-50 Read Record command APDU			
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´B2´	Command code
P1	1	´01´...´FE´	Number of the record that is to be read
P2	1	´XX´	Type of addressing (+)
Lc	0	-	Empty
Data	0	-	Empty
Le	1	´00´...´FF´	Length of the response data

Appendix Table 1-51 Read Record response APDU if the command was completed successfully			
	Length	Contents	Description

Appendix Table 1-51	Read Record response APDU if the command was completed successfully		
	Length	Contents	Description
Data	var.	´XX...YY´	Contents of a record
Trailer	2	´90 00´	No errors and no warnings
		´62 82´	Length of the response data is smaller than Le
		´62 81´	Record and its checksum are inconsistent

Appendix Table 1-52	Read Record response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	EF to be read out is deactivated, inconsistent or incorrect data
		´67 00´	Response data has incorrect length, Lc is superfluous, Le is missing, Le is smaller than Lr
		´69 81´	EF to be read out is not formatted
		´69 82´	Security conditions not fulfilled
		´69 86´	Command not allowed, no EF is selected
		´69 88´	Incorrect data objects for secure messaging
		´6A 82´	File not found via SFI
		´6A 83´	Referenced record does not exist
		´6A 86´	Parameters P1 - P2 incorrect
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-53	Coding of P2 for Read Record							
b8	b7	b6	b5	b4	b3	b2	b1	Description
								SFI usage:
0	0	0	0	0	…	…	…	References the current EF, no SFI
x	x	x	x	x	…	…	…	5-bit SFI
…	…	…	…	…	x	x	x	Type of addressing:
…	…	…	…	…	1	0	0	Absolute; record number is given in P1

1.13. RESET RETRY COUNTER

The command RESET RETRY COUNTER has the effect of resetting the retry counter for a password to its initial value. Which password the operation refers to is specified by the parameter P2.

It is also possible to replace the old password by a new one (P1=´00´).The replacement of the old password is authorized by a resetting code (password) which is integrated in the command.

NOTICE: An unauthorized reset of the retry counter should be prevented for reasons of security. As the execution of the command is determined by access rules, e. g. a preceding successful authentication procedure (which may be password or key based) can be demanded. Alternatively the reset can depend on a resetting code.

Appendix Table 1-54	Reset Retry Counter command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´2C´	Command code
P1	1	´00´ ´F3´	Command with replace Command without replace
P2	1	´XX´	Check byte (+)
Lc	1 0	´01´...´FF´ -	P1=´00´: Length of the command data P1=´03´: Empty
Data	Lc 0	´XX...YY´ -	P1=´00´: Resetting code || new password P1=´03´: Empty
Le	0	-	Empty

Appendix Table 1-55	Reset Retry Counter response APDU if the command was completed successfully		
	Length	Contents	Description
Data	var.	-	Empty
Trailer	2	´90 00´	Retry counter is now set to its initial value
		´63 CX´	As for ´90 00´, but problems with writing

Appendix Table 1-56	Reset Retry Counter response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Inconsistent or incorrect data
		´65 81´	Write error
		´66 12´	Parity error in the cryptographic algorithm
		´67 00´	Response data has incorrect length, command data is superfluous, Le is superfluous
		´69 82´	Security conditions not fulfilled
		´69 85´	Retry counter is not zero, referenced password cannot be used

		´69 88´	Incorrect data objects for secure messaging
		´6A 80´	Incorrect transport format for the password
		´6A 83´	Record not found
		´6A 86´	P1 or P2 possesses an illegal value, no reference to the resetting code
		´6A 88´	Referenced password not found
		´6E 00´	CLA and INS are inconsistent
		*	Trailers from the VERIFY and CHANGE REFERENCE DATA commands are also possible

Appendix Table 1-57				Coding of P2 for Reset Retry Counter				
b8	b7	b6	b5	b4	b3	b2	b1	Description
x	…	…	…	…	…	…	…	Password:
0	…	…	…	…	…	…	…	Global, all information in the MF
1	…	…	…	…	…	…	…	Local, all information under the MF
…	0	0	0	0	0	…	…	Other values are RFU
…	…	…	…	…	…	x	x	Password number (PwdID)

1.14. SELECT FILE

You can use the command SELECT FILE to select a file in the file system and make it the current file. Depending on the parameters, this command selects either a directory (MF or DF) or a data field (EF).

When this command has been successfully executed the situation is as follows:

- If a directory was selected, the directory pointer points to this selected directory and the field pointer is undefined.
- If a data field was selected, the directory pointer remains unchanged and the field pointer points to this selected data field.

If the command cannot be completed successfully, both the directory pointer and the field pointer remain unchanged.

The security status values, the active security environments (including volatile ones) and all derived and negotiated keys will also remain unchanged if the command is unsuccessful.

Appendix Table 1-58		Select File command APDU	
	Length	Contents	Description
CLA	1	´00´	Command class
INS	1	´A4´	Command code
P1	1	´0X´	Type of navigation (+)
P2	1	´0X´	Quantity of information in the response data (+)

Appendix Table 1-58	Select File command APDU		
	Length	Contents	Description
Lc	1 0	- ´01...FF´	Empty Length of the command data
Data	0 Lc	- ´XX...YY´	Empty Optional command data containing the FID or the name of the DF
Le	0 1	- ´00...FF´	No response data Length of the response data

Appendix Table 1-59	Select File response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0 var.	- ´XX...YY´	Empty Information about the file (+)
Trailer	2	´90 00´	No errors and no warnings
		´62 83´	File has been deactivated

Appendix Table 1-60	Select File response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	File header and checksum are inconsistent,
		´67 00´	Lc is not equal to the length of the command data, Le is missing or is superfluous, Le is smaller than Lr
		´6A 80´	P1=´00´ yet FID is not ´3F00´
		´6A 82´	File not found
		´6A 86´	P1 or P2 possesses an illegal value
		´6A 87´	Lc is missing or is superfluous
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-61	Coding of P1 for Select File								
b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	0	0	Select MF
…	…	…	…	…	x	x	x	Selection of DF or EF
…	…	…	…	…	…	…	0	Selection of MF
…	…	…	…	…	0	0	1	Select DF in the current DF
…	…	…	…	…	0	1	0	Select EF in the current DF

Appendix Table 1-61				Coding of P1 for Select File				
b8	b7	b6	b5	b4	b3	b2	b1	Description
…	…	…	…	…	0	1	1	Select the DF at next higher level above the current one
…	…	…	…	…	1	0	0	Make selection using the name of the DF
All other values RFU								Are not supported

Appendix Table 1-62				Coding of P2 for Select				
b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	x	x	0	0	Contents of the response:
0	0	0	0	0	0	0	0	Return data object consists of FCP and FMD
0	0	0	0	0	1	0	0	Return FCP
0	0	0	0	1	0	0	0	Return FMD
0	0	0	0	1	1	0	0	No response data
All other values RFU								Are not supported

1.15. UPDATE BINARY

The command UPDATE BINARY overwrites all or part of the contents of a transparent data field.

The data field can either be selected previously, using the command SELECT FILE, or you can use the short file identifier (SFI) given in parameter P1 to make this data field the current data field.

In the command, the parameters P1 and P2 specify the first byte to be overwritten, relative to the beginning of the data field, and pass the new data.

Appendix Table 1-63		Update Binary command APDU	
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´D6´	Command code
P1	1	´XX´	SFI or MSB of the offset (+)
P2	1	´00´...´FF´	LSB of the offset
Lc	1	´01´...´FF´	Length of the command data
Data	Lc	´XX...YY´	New data
Le	0	-	Empty

Appendix Table 1-64		Update Binary response APDU if the command was completed successfully	
	Length	Contents	Description
Data	0 var.	- ´XX...YY´	Empty Information about the file (+)
Trailer	2	´90 00´	No errors and no warnings
		´63 CX´	Repeated write attempts were necessary

Appendix Table 1-65		Update Binary response APDU in the event of an error	
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Referenced EF is deactivated, user data and checksum are inconsistent, inconsistent or incorrect data
		´65 81´	Write error
		´67 00´	Response data has incorrect length, command data is missing or Lc is not equal to the length of the command data, Le is superfluous
		´69 81´	Referenced EF is not transparent
		´69 82´	Security conditions are not fulfilled
		´69 86´	Command not allowed, no EF is selected
		´69 88´	Incorrect data objects for secure messaging
		´6A 82´	File not found via SFI
		´6A 84´	Length of user data is less than offset + data length
		´6A 86´	Parameters P1 - P2 incorrect
		´6B 00´	Offset lies outside the EF
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-66				Coding of P1 for Update Binary				
b8	b7	b6	b5	b4	b3	b2	b1	Description
0	x	x	x	x	x	x	X	Offset (MSB)
1	0	0	x	x	x	x	x	5-bit SFI

1.16. UPDATE RECORD

The command UPDATE RECORD overwrites the entire contents of a record.

The data field can either be selected previously, using the command SELECT FILE, or you can use the short file identifier (SFI) given in parameter P2 to make this data field the current data field.

In the command, parameters P1 and P2 specify which record is to be overwritten.

Appendix Table 1-67	Update Record command APDU		
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´DC´	Command code
P1	1	´01´...´FE´	Record number
P2	1	´XX´	Type of addressing (+)
Lc	1	´01´...´FF´	Length of the command data
Data	Lc	´XX...YY´	New contents of the record that is to be overwritten
Le	0	-	Empty

Appendix Table 1-68	Update Record response APDU if the command was completed successfully		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	No errors and no warnings
		´63 CX´	Repeated write attempts were necessary

Appendix Table 1-69	Update Record response APDU in the event of an error		
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Referenced EF is deactivated, inconsistent or incorrect data
		´65 81´	Write error
		´67 00´	Response data has incorrect length, command data is missing, or Lc is not equal to the length of the command data, or the data length is not equal to the fixed record length, Le is superfluous
		´69 81´	Referenced EF is not formatted
		´69 82´	Security conditions are not fulfilled
		´69 86´	Command not allowed, no EF selected
		´69 88´	Incorrect data objects for secure messaging
		´6A 82´	File not found via SFI
		´6A 83´	Referenced record does not exist
		´6A 84´	Too much data for the EF
		´6A 86´	Parameters P1 - P2 incorrect
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-70 Coding of P2 for Update Record								
b8	b7	b6	b5	b4	b3	b2	b1	Description
								SFI usage:
0	0	0	0	0	…	…	…	References current EF, no SFI
x	x	x	x	x	…	…	…	5-bit SFI
…	…	…	…	…	x	x	X	Type of addressing:
…	…	…	…	…	1	0	0	Absolute; record number is given in P1

1.17. VERIFY

The command VERIFY launches a comparison between data that is sent with the command and a reference value stored in the chip card.

The parameter P2 specifies which reference value is to be used for the comparison. If the comparison is acceptable, the security status of the chip card is changed.

Appendix Table 1-71 Verify command APDU			
	Length	Contents	Description
CLA	1	´0X´	Command class
INS	1	´20´	Command code
P1	1	´00´	Fixed value
P2	1	´XX´	Check byte (+)
Lc	1	´01´...´FF´	Length of the command data
Data	Lc	´XX...YY´	Password in transfer format
Le	0	-	Empty

Appendix Table 1-72 Verify response APDU if the command was completed successfully			
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´90 00´	Correct password
		´63 CX´	As for ´90 00´, but problems with writing
		´63 CX´	Incorrect password

Appendix Table 1-73 Response APDU for Verify command in the event of an error			
	Length	Contents	Description
Data	0	-	Empty
Trailer	2	´64 00´	Lc is not equal to the length of the command data, Le is not equal to

Appendix Table 1-73		Response APDU for Verify command in the event of an error	
	Length	Contents	Description
			20 bytes, or Le is superfluous
		´65 81´	Counter or hash value missing
		´66 12´	Length of data object is not 28 or 64 bytes
		´67 00´	Response data has incorrect length, Lc is missing or incorrect, Le is superfluous
		´69 82´	Security conditions are not fulfilled
		´69 83´	Retry counter has expired
		´69 84´	Usage counter has expired
		´69 85´	Referenced password cannot be used
		´69 88´	Incorrect data objects for secure messaging
		´6A 80´	Password has incorrect transport format
		´6A 83´	Record not found
		´6A 86´	P1 or P2 has an illegal value
		´6A 88´	Referenced password not found
		´6E 00´	CLA and INS are inconsistent

Appendix Table 1-74			Verify command P2 coding					
b8	b7	b6	b5	b4	b3	b2	b1	Description
x	…	…	…	…	…	…	…	Password:
0	…	…	…	…	…	…	…	Global, all information in the MF
1	…	…	…	…	…	…	…	Local, all information under the MF
…	0	0	0	0	0	…	…	Other values are RFU
…	…	…	…	…	…	x	x	Password number (PwdID)