Aventra MyEID
PKI Smart Card

Reference manual
Ver. 3.0.6

Public

Contents

1 Version history

Date	Version	Description
28.1.2011	1.7.7	
17.12.2014	2.0.0	MyEID 4.x functionality added
29.1.2015	2.0.1	Corrections in references
5.2.2015	2.0.2	AES and DES, added response SW values
5.4.2015	2.0.3	Clarification on EC key lengths, added some error codes
9.4.2015	2.0.4	Added information about applet states
15.4.2015	2.0.5	Incorrect parameters in the data field / Wrong data (6A80)
4.5.2015	2.0.6	Init PIN flag default settings clarified. Moved General authenticate and clarified the command data structure.
15.5.2015	2.0.8	Specified how setting PIN and PUK retry counters work. Added AID in Activate Applet command's spec.
23.6.2015	2.0.9	Added PSO: ENCIPHER
10.3.2016	2.1.0	Added GET DATA: MyEID CARD CAPABILITIES
22.3.2016	2.1.1	Added symmetric keys to PUT DATA: LOAD KEY
12.4.2016	2.1.3	RESET RETRY COUNTER, specified C/R pins
28.4.2016	2.1.4	Corrections to EC related commands
7.9.2017	2.1.5	Corrected some references to e.g. tables
13.12.2017	2.1.6	Correction to P1 and P2 values in MSE:SET command when used before ECDH operation.
2.11.2018	2.2.0	Added key wrapping/unwrapping and session objects
25.3.2019	2.3.0	Added PUT DATA: Initialise PIV emulation
14.2.2020	2.4.0	Added description of 4096 bit RSA and APDU chaining. Added information about ATR.

7.2.2024	3.0.0	MyEID 5 documentation added
28.6.2024	3.0.1	Corrected GET DATA, P2=A8h or A9h specification. Improved description of user roles and recommendations. Small clarifications to command-specific instructions based on feedback from testing. Updated introduction and added a section about operating environment.
11.10.2024	3.0.2	- Updated definition of PIN reference number in VERIFY, CHANGE REF. DATA and RESET RETRY COUNTER commands to mention only range 01-0E of allowed values. Previously allowed values of MyEID v < 3 (only three pins) were mentioned first. Clarified how admin/global unblocker PINs function if used to unblock a PIN. - a correction to definition of P2 in GET CHALLENGE - added a recommendation for using a key integrity verification mechanism with key wrapping
14.10.2024	3.0.3	Removed PUT DATA: LOAD KEY PAIR from the command list, because actually there are no such command – all key components are loaded using PUT DATA: LOAD KEY Update the title of the document to be consistent with other CC EAL4+ documentation. Added information about potential compatibility issues.
16.12.2024	3.0.4	- Updated the description of PUT DATA: SET SECURE MESSAGING PARAMETERS command. Added a description of Exclusive PIN Mode flag. - Improved description of user role assignment - Improved and clarified guidance, requirements and recommendations for secure usage in section 17. - added descriptions of some GET DATA responses which were missing. - added information about security event. Internal review 16.12.2024 JS
6.2.2025	3.0.5	- added missing response APDU tables for some PUT DATA commands - minor clarifications to product identification - improved guidance on proper configuration of security attributes. - added Public marking Internal review 6.2.2025 JS
11.3.2025	3.0.6	- updated 17.3 regarding RSA with OAEP. - corrected wrong order of nonces in 16.3 step 7. - distinguished the integrity check on ACTIVATE APPLET from the self-test triggered by GET DATA. - added passing the self-test as criteria for product acceptance. Internal review 16.3.2025 JS

2 Introduction

This document describes Aventra MyEID PKI Smart Card, context and command interface.

MyEID PKI Smart Card is designed for user authentication, digital signature creation and protection of sensitive data. It stores private and secret keys securely on a microcontroller specially designed for this purpose and performs cryptographic computation on card. The smart card is based on JavaCard technology. The MyEID JavaCard applet provides the interface for software to interact with the security functions and manages the file system of the smart card.

The command set of the applet is based on ISO/IEC 7816-4 and ISO/IEC 7816-8 standards. The application related data is in files according to the ISO/IEC 7816-15 standard. The applet implements the FINEID S1 v1.12 specification and it can be configured to include all or any subset of the features specified in FINEID S4-1 or S4-2 specifications. The applet supports RSA key lengths from 2048 to 4096 bits and Elliptic Curve key lengths from 256 to 521 bits on NIST/SEC prime curves and Brainbool Standard curves.

The applet is fully compatible with JavaCard and GlobalPlatform specifications. The applet is set as the default applet on the card and it is automatically selected after the JavaCard has been powered up. MyEID 4.9.10 and 5.0.0 are delivered on JavaCard version 3.0.5 Classic and GlobalPlatform 2.3 platforms. MyEID supports file sizes up to 32767 bytes and up to 14 different PIN codes, which can be mapped into different security attributes and user roles. The number of private and secret keys is only limited by the available memory and maximum numbers of files (see *PUT DATA: INITIALISE APPLET*).

Documentation specific to MyEID 5 applies to MyEID 4.9.10 and newer versions. At the time of writing, MyEID 5.0.0 is being evaluated to Common Criteria EAL4+ level, and will be available when the evaluation and certification are ready. MyEID 4.9.10 has the same command interface and features.

2.1 References

The most relevant specifications and standards are:

- ISO/IEC 7816-4
- ISO/IEC 7816-8
- ISO/IEC 7816-9
- ISO/IEC 7816-15
- JavaCard 2.1.1, MyEID3: 2.2.1, MyEID5: 3.0.5

- GlobalPlatform 2.0.1 (Open Platform), MyEID3: GlobalPlatform 2.1.1, MyEID5: GlobalPlatform 2.3
- FINEID S1 and S4 documentation
- NIST Special Publication 800, NIST SP 800-73pt2-5: Interfaces for Personal Identity Verification, Part 2 - PIV Card Application Card Command Interface [PIV] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-73pt2-5.pdf
- Microsoft Smart Card Minidriver Specification v7.07
- Request for Comments 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard

2.2 Abbreviations

AC	Access Condition
AID	Application Identifier
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
CRDO	Control Reference Data Object
CIV	Commercial Identity Verification
CLA	Class Byte
C/R	Challenge/Response
DF	Dedicated File
EF	Elementary File
FCI	File Control Information
FID	File Identifier
LC	Length of APDU command data
MF	Master File
MSE	Manage Security Environment
P1/P2	APDU command parameter 1 and 2
PIN	Personal Identification Number
PIV	Personal Identity Verification
PSO	Perform Security Operation
RFU	Reserved for Future Use
SE	Security Environment
SM	Secure Messaging
SW	Status Word
SSCD	Secure Signature Creation Device

3 Usage environment

MyEID can be used with any device, which supports communication with smart cards according to ISO 7816 standard. A typical usage environment is a personal computer with Windows, Linux or MAC OS X operating system, equipped with an ISO 7816 and PC/SC compliant smart card reader.

This document describes the command interface in level of APDU commands defined in ISO 7816 standard, which enables the reader to implement their own application or interface to communicate with the card. In typical usage scenarios, higher level communication interfaces are already available for applications. These include a Microsoft Windows minidriver, which is available from Aventra, and PKCS#11 driver, which is available in OpenSC smart card toolset/software library. Even when using a higher-level interface, this document can be useful for understanding how the smart card works and protects the sensitive data such as private keys and PIN codes.

For usage scenarios with high security requirements such as usage as a Secure Signature Creation Device as defined in Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408), a security specialist with appropriate knowledge of IT security and cryptography should verify that your operating environment complies with the requirements. The requirements are explained from MyEID's perspective in Annex I.

4 Version history

4.1 New in MyEID 5

Secure Messaging

MyEID 5 supports secure messaging according to NIST SP 800-73-4 specification (PIV Card Specification). Secure messaging helps to protect PIN codes, and private keys when imported to the card.

Extended length APDUs

MyEID 5 supports extended length APDUs as defined in ISO 7816-4:2005. For RSA operations with 2048 bit and longer keys, data no longer needs to be chained into multiple APDUs. The command data length is limited to 768 bytes and the response length to 32767 bytes. The command reference contains definition of Lc and Le fields according to short APDUs. Extended APDUs can be used for all commands as specified in the standard.

Security improvements

- To prevent possibility of accidentally issuing a card still in Creation State to a user, crypto-operations are now disabled in Creation State. They can be enabled until next reset to allow using them in card personalisation. With this mechanism, the card does not work normally if it is left in Creation State, making it easier to notice the situation.
- MyEID 5 includes a cryptographic mechanism to verify authenticity of the card.

AES C/R pins

Support for challenge / response PINs using AES algorithms has been added.

Brainpool ECC curves

MyEID 5 supports Brainpool elliptic curves between 256, 320, 384 and 512 bits as defined in RFC 5639.

Short RSA key lengths and DES deprecated

- RSA key lengths less than 2048 bits are no longer supported.
- DES keys are no longer supported, except for challenge-response PINs.

Stricter ISO 7816-3 compliance

Previous versions of MyEID behaved in not fully ISO 7816-3 compliant way in some situations. For example, MyEID treated the Le field as optional for commands which return data, and commands returned data even though the Le field was absent. MyEID 5 follows the standard strictly. A compatibility mode can be enabled to allow applications built for older versions to work without changes. Please contact Aventra for more information about the compatibility mode.

4.2 New in MyEID 4.5

MyEID version 4.5 introduces key wrapping and unwrapping, and session objects.

Key wrapping and unwrapping

This feature enables transferring keys from and to the card in encrypted form. In key wrapping, card encrypts key material from a key file using another key, and outputs the resulting cryptogram. In key unwrapping, a cryptogram is input into the card. The card decrypts the cryptogram using a selected key and places the resulting plaintext into a key file.

The following usage scenarios are supported:

- Unwrap symmetric keys using RSA keys
- Unwrap symmetric keys using symmetric keys
- Wrap symmetric keys using symmetric keys.

Session objects

Session objects are temporary objects. In terms of PKCS#11 specification, they are objects with CKA_TOKEN attribute set to CK_FALSE. They are cleared from the card automatically next time the card is reset. This mechanism ensures that if the software that is using the card has no possibility to clear the session objects before the card is ejected, they are cleared when the card is powered up next time.

4096 bit RSA keys

MyEID 4.5 adds support to RSA keys up to 4096 bit length. Crypto-operations with 2048 bit and larger keys require data to be split into multiple APDUs. This is handled with APDU chaining. In command APDUs, the CLA byte is set to 10h when there is more data to follow. In response APDU, SW1/SW2 value 61xx indicates that there is more response data waiting, and the next block can be requested using the GET RESPONSE command. SW2 indicates the number of bytes remaining, 00 indicates a full block. The old method of splitting a 2048 bit cryptogram in PSO: Decipher operations remains available for backwards compatibility.

4.3 New in MyEID 4

MyEID version 4.0 introduces several new features including Elliptic Curve Cryptography (ECC) support, challenge/response PINs and many new options for creating files and PINs. Some of the features are implemented specially to make the applet fulfil Microsoft's Smart Card Minidriver specification completely and make MyEID card a Windows Certified device.

ECC support

The applet supports ECDSA and ECDH algorithms with the following named prime curves, which are defined by NIST in document "RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE": P-192, P-224, P-256, P-384 and P-521. Some of the MyEID4 platforms do not support 384 and 521 bit EC keys.

Challenge/response PINs

When authenticating a challenge/response PIN the card generates a random challenge. The user encrypts the challenge with an AES or 3DES key that is also stored on the card, and sends the response to the card using VERIFY command APDU. The card decrypts the response and compares it with the original challenge.

Auto size files

Files created with the "auto size" flag set grow dynamically, when they are updated past their initial size.

Admin state

In previous applet versions each operation for each card object could be only accessed by one specified PIN. In some situations it is practical that an object can be accessed with different PINs. The admin state has been implemented to fulfil this requirement.

A PIN can be created with the administrator PIN flag set. When such PIN is verified, the admin state is enabled. Admin state gives special access to files and keys. If for example the normal user PIN is PIN #1 and PIN #3 is given admin flag, the administrator could be allowed to unblock PIN #1 or write to a file which has PIN#1 defined for writing in security attributes list. By default admin state does not allow special access to any file or PIN. To prevent unintentional access, the operations that are allowed in admin state must be defined separately for each card object.

PIV / CIV emulation

Since version 4, MyEID card can emulate a PIV/CIV card by mapping the ISO 7816-15 (PKCS#15) structure to the PIV/CIV command interface. The PIV/CIV emulation can be switched on and off with a single command. MyEID is not fully PIV/CIV compliant; the aim is to emulate the functionality to the extent of interoperability on command interface level, and to allow using MyEID cards in PIV compatible devices, in situations where native MyEID middleware cannot be installed.

All command parameters defined in the PIV specification are not supported. Returned data and error codes do not comply with the specification in all situations.

With the PIV/CIV emulation enabled the card will be identified as a PIV/CIV card and with the relevant AID.

Cards cannot be personalised using the PIV/CIV interface, it is read-only in this sense. The cards must be personalised using the MyEID command interface, and keys and certificates are mapped to the PIV/CIV objects using FIDs, by executing a PUT DATA: INITIALISE PIV EMULATION command.

4.4 Older versions

MyEID version 3 (MyEID3) uses the JavaCard 2.2.1 and GlobalPlatform 2.1.1 platform. It supports RSA keys from 512 up to 2048 bits in 64 bit increments. MyEID3 supports file sizes up to 32767 bytes and 14 different PIN-codes can be created and used.

4.5 Feature list table

In the following table is shown in which version of the applet specific features have been introduced.

Feature	MyEID 3	MyEID 4	MyEID 4.5	MyEID 5
Challenge/response PINs		*	*	*
Elliptic Curve Cryptography		*	*	*
Automatically growing files		*	*	*
Admin state		*	*	*
Global unblocker state		*	*	*
3DES encryption	(*) [1]	*	*	
AES encryption	(*) [1]	*	*	*
Key wrapping			*	*
Generic Secret key EF’s			*	*
Session objects			*	*
4096 bit RSA key length			*	*
1024 bit RSA key length	*	*	*	
Secure messaging				*
Card genuineness verification				*
AES challenge/response PINs				*
Brainpool Elliptic Curves				*
PIV interface		*	*	*
Extended APDUs				*

1) On MyEID 3 symmetric encryption is a special feature that must be ordered separately.

4.6 Compatibility issues

While in principle MyEID 4.9.10, 5.0.0 and newer are backward compatible with older versions, the following compatibility issues have been identified:

- Triple-DES Key EFs are not supported anymore.

- MyEID 5 enforces the integrity of the file system in a stricter way than previous versions, and does not allow activating the applet, in case the file structure contains references to uninitialized PINs. This has been identified to cause compatibility issues. For example, Active Process Manager or MyEID Minidriver Utility may use PIN #2 for DELETE access right of specific files, even if PIN#2 is not initialised.

- If the card is reset after applet initialisation and still in creation state, PERFORM SECURITY OPERATION commands are not allowed, unless explicitly enabled using PUT DATA: SET SESSION FLAGS command. This feature is to ensure that if a card is unintentionally left in the creation state, this will be noticed when trying to use the card.

5 Contents and File Structure

The applet follows the ISO 7816-15 standard (based on PKCS #15 specification) and implements all or any subset of files and contents of FINEID S4-1, S4-2, or ISO 7816-15 specifications.

The applet can contain the following objects:

- private RSA and ECC keys,
- public RSA and ECC keys,
- AES keys
- authentication objects,
- card holder certificates,
- trusted certificates, and
- any data objects

6 User roles

The identified user roles and their characteristics are described in the following table.

Table 1 – User roles

Role	**Description**
End User	A user who uses MyEID for digital signature authentication and encipherment/decipherment operations. End user can be a human person or a device, where a MyEID smart card is installed. End users have one or more personal user PINs to protect access to their private keys, but usually do not know their card's SO-PIN. End users may or may not possess the PUKs of their cards.
Card Issuance Operator	A person who operates software used to issue MyEID smart cards for end users. Card issuance is usually automated, so typically Card Issuance Operators do not need to understand details of MyEID's security architecture, or handle PIN or PUK codes of the cards manually. Authorized to initialize and configure the applet using automated tools.
Administrator	Users who perform maintenance tasks, such as unblocking or re-initialising end users' cards. These tasks involve operations which require authenticating to the card using SO-PIN. Administrator is called Security Officer in PKCS#11 and PKCS#15 terminology.
Application Developer	Develops applications which use MyEID's security functions. Developers typically use higher level interfaces such as CNG or PKCS#11 to access the security functionality, but often need to understand the card's internal structure and security architecture.
Security Specialist	Plans usage of MyEID cards in an organisation's environment. Needs to understand MyEID's security architecture, and makes decisions on how to configure MyEID's security attributes. Authorized to

	initialize and configure the applet and manage the configurations.

6.1 User role assignment

Administrator and End user roles are separated on the card by assigning a specific PIN for each role and by using File Security Attributes (13.2.1) which are configured before handing the card to the End user. On card level, all roles in the User roles table except End user are mapped to the Administrator role. In a typical scenario, PIN #1 is associated with the End user and PIN #3 is associated with the Administrator. For example, by setting Create DF and Create EF security attribute to 3 in MF and DF 5015, and Update / Erase attribute to 3 in the files within DF 5015, creation, deletion and modification of files is allowed only for Administrator. By setting Use security attribute of Key EFs to 1 and Put Data, Delete File and Generate attributes to 3, End user is allowed to use the Key EFs for cryptographic operations, but generating or importing a new key or deleting a key is allowed only for Administrator. If the Use security attribute of a Key EF is set to the User PIN (e.g. PIN #1), Administrator cannot by default use the key for cryptographic operations. Using a key can be enabled for both End user and Administrator by setting the "Admin use / wrap key" flag in the second status byte of the FCP structure of the key EF.

MyEID cards shall not be delivered to End users in Creation state.

The PIN associated with the Administator role must be set into the Recreate MF security attribute of the MF.

Functions PUT DATA: INITIALISE APPLET, PUT DATA: INITIALISE PIN and PUT DATA: INITIALISE PIV EMULATION, PUT DATA: ACTIVATE OR DEACTIVATE PIV EMULATION and PUT DATA: SET SECURE MESSAGING PARAMETERS are available only for the Administrator role in Operational state. These functions require authenticating the PIN associated with the Recreate MF security attribute.

Security attributes, and requirements and recommendations for configuring them are further explained in 13.2.1, 17 and ANNEX I – Common Criteria EAL4+ Compliance.

7 Applet states

To simplify card personalisation, MyEID applet includes a concept of **Creation State**. An empty MyEID card is in Creation State, where access conditions are not effective. This way the whole card personalisation can be completed without verifying PINs required to access the files included in the process. After personalisation ACTIVATE APPLET command must be issued, which switches the applet to **Operational State**. The only way to switch the applet back to Creation State is to reinitialise the applet, which empties the applet's contents (files, keys and PINs) completely. Since MyEID 5.0.0, cryptographic operations are disabled in Creation State when the card is powered on. They can be enabled until next reset using **PUT DATA: Set Sessions Flags** command.

Admin State and **Global Unblocker** states (from MyEID4 onwards) allow changing and unblocking other PINs. These features are optional properties of PINs and they are explained in PUT DATA: INITIALISE PIN command's description.

8 Application Identifier (AID)

AID of the MyEID applet is A000000063504B43532D3135.

9 Answer to Reset

MyEID cards up to version 3.3.3 return the following ATR:

```
3B F5 18 00 00 81 31 FE 45 4D 79 45 49 44 9A
```

Since version 4.0.0, most cards have been shipped with this ATR:

```
3B F5 96 00 00 81 31 FE 45 4D 79 45 49 44 14
```

In the new ATR, TA1 byte has been changed to enable faster communication with smart card readers. To avoid compatibility issues, it is recommended to identify MyEID cards with only the historical characters instead of the whole ATR string. Cards with the old ATR are still available upon request.

Different ATRs may be used on customer's request or because of technical reasons.

10 Preparative procedures

10.1 Product identification

The Common Criteria EAL4+ certified version is identified with the following parameters, which can be obtained from the card using GET DATA commands as described under the next heading.

Name	MyEID
Major version	5
Minor version	0
Version revision	0
Security certifications	Bit 0 set (value: 01h)

10.2 Acceptance

The following steps shall be taken to ensure, that the delivered product is a genuine MyEID PKI Smart Card produced by Aventra, and the version and configuration are as ordered:

- Check that the cards are packed in packaging marked with Aventra logo, and that the boxes or envelopes are not physically damaged in delivery.

- Perform the Card genuineness verification as described in 16. Aventra recommends that at least 1% of the total quantity of ordered cards are checked. Check cards from different boxes. Aventra provides a software tool to perform the verification. Please see the downloads section at Aventra web site or contact Aventra for more information.

- Check that the ATR matches the ATR of the order form.

- Execute **GET DATA: Applet information** command to check the product name and version. See Table 14 for details.

- Execute **GET DATA: MyEID card capabilities** to check the security certification status. See Table 15 for details.

- Execute **GET DATA: Self-test** command (see 13.1)**.** The self-test verifies internal integrity of the MyEID applet and correct operation of the cryptographic functions. Verify that the card returns status code 9000h. In case the self-test returns an error, please contact Aventra.

The commands needed for the acceptance procedure can be executed using software tools provided by Aventra, using other smart card tools such as OpenSC, or using custom-made software.

10.3 Preparation for use

MyEID smart card can be used in many different scenarios, and preparative procedures vary. The following is only an overview of the preparative procedures. A security specialist with basic knowledge of cryptography and smart cards should familiarize oneself with this document and verify that the configuration is appropriate for a specific usage scenario.

The following steps are needed to prepare the card for usage:

- Execute PUT DATA: INITIALISE APPLET command. This is not mandatory, because the card is delivered as initialised, in Creation State. It is necessary, if other than default settings are needed.

- Create a file structure. Typically, the file structure defined in ISO 7816-15 standard is used, but custom file structures can be used for special scenarios. See 17.1 for recommended security attributes.

- Initialise PIN codes. See 17.5 for security considerations.

- Generate or import key pairs and load certificates as needed. This can be also done after activation.

- Use ACTIVATE APPLET command to transfer the applet to Operational State. At this point, the applet checks integrity of the file system, by checking that all PINs referenced in file security attributes have been initialized. If the integrity check passes, the access conditions become effective and crypto-operations are enabled.

- Generate or import key pairs and load certificates post-activation as needed.

Document **"Getting Started with MyEID Cards and Tokens"** is available at Aventra's web site for guidance on how to prepare MyEID ready for use in a typical usage scenario.

11 Security events

The following security events can occur on the smart card:

- PIN blocking: A PIN is blocked, if the number of allowed attempts is exceeded, i.e. the retry counter reaches zero. A blocked PIN cannot be authenticated.

- PUK blocking: A PUK is blocked, if the number of allowed attempts is exceeded. A blocked PUK cannot be used to unblock PINs. If the card is in Operational State and the PIN and PUK associated with the Administrator Role (Recreate MF security attribute) are blocked, there is no way to access PUT DATA: INITIALISE APPLET to reinitialise the card and restore the file system to the initial state.

- Platform security events: The smart card platform and microcontroller MyEID is built on includes advanced intrusion detection mechanisms, which protect the card from attempts to extract secret information such as private keys electronically or physically, and attempts to intercept the cryptographic operations and gain knowledge of the platform's internal processing. If the platform detects more than allowed number of events it considers attack-attempts, it may self-destruct the card and prevent any further access and usage. For security reasons, details of the platform's security mechanisms are not available in public.

12 Command Interface

This chapter describes the commands of the applet with their parameters.

The commands are split into two categories:

- regular commands
- personalisation and management commands

The regular commands are compatible with the FINEID S1 v1.12 specifications that comply with the ISO/IEC 7816-4 and 7816-8 standards.

The personalisation and management commands comply with the ISO/IEC 7816-4 and 7816-9 standards.

12.1 General guidance

12.1.1 About the command definitions

The contents of the LC field, the data field and the LE field are defined according to short APDU format without secure messaging. When using CLA byte value XCh for secure messaging, please refer to NIST SP 800-73pt2-5 section 4 "Secure Messaging" on how to encapsulate the commands into secure messaging data objects and how to decode the response.

Please refer to the ISO 7816-3 standard on how to format and interpret LC and LE fields with extended lengths APDU commands. Extended length APDUs can be used together with secure messaging.

12.1.2 Integer format

Commands for loading key components, which take integer data as parameters, use big endian big integer format. All RSA and ECC key components are positive integers. For flexibility and compatibility with different standards, MyEID accepts both unsigned and signed representations for key components, expect for RSA public exponent. In signed representation, integers with most significant byte value bigger than 7Fh have a leading zero to mark the value positive. If a leading zero is not included, MyEID interprets the value as unsigned, positive value, and the result is the same.

12.1.3 Secure parameter values

In general, MyEID's command interface is designed to allow only parameter values which are considered secure, and to return an error code if an insecure parameter value is provided. For example, MyEID does not allow setting zero (Allowed at all times) for the USE security attribute of Key EFs, minimum PIN length less than 4 or RSA key length less than 2048. However, security aspects should be considered especially when configuring security attributes of the file system, and awareness of security aspects is needed for using the available combinations of cryptographic algorithms and padding modes in a secure way. Please see 17 and Annex I for guidance for configuring and using MyEID in a secure way and for ensuring compliance with Common Criteria EAL4+ requirements when using MyEID as an SSCD.

12.2 Command lists

12.2.1 Regular commands

Table 2 - Regular usage related commands of the MyEID applet.

Command	Standard	Functionality
GET DATA (INS = CAh)	ISO/IEC 7816-4	Retrieves applet version and other information. See Table 4 for GET DATA with INS = CBh.
SELECT FILE	ISO/IEC 7816-4	Selects the applet or a file within the applet.
GET RESPONSE	ISO/IEC 7816-4	Reads the remaining or next block of the card's response, when there is more response data available than expected by the command APDU.
READ BINARY	ISO/IEC 7816-4	Reads contents from a transparent (binary) file.
UPDATE BINARY	ISO/IEC 7816-4	Updates contents in a transparent (binary) file.
ERASE BINARY	ISO/IEC 7816-4	Erases contents in a transparent (binary) file.
VERIFY	ISO/IEC 7816-4	Verifies reference data presented by the user (PIN) with the reference data stored in the applet. The current verification status can be queried with this command.
CHANGE REFERENCE DATA	ISO/IEC 7816-8	Changes the current reference data (PIN).
RESET RETRY COUNTER	ISO/IEC 7816-8	Unblocks a blocked reference data (PIN).
MANAGE SECURITY ENVIRONMENT: **RESTORE**	ISO/IEC 7816-8	Restores a predefined or empty security environment.
MANAGE SECURITY ENVIRONMENT: **SET**	ISO/IEC 7816-8	Sets security environment parameters that will be used with subsequent PSO or GENERAL AUTHENTICATE command(s).
PERFORM SECURITY OPERATION: **COMPUTE DIGITAL SIGNATURE**	ISO/IEC 7816-8	Computes a digital signature using a private key. The algorithm and the key are selected with the preceding MSE command.
PERFORM SECURITY OPERATION: **DECIPHER**	ISO/IEC 7816-8	Decrypts data with a private or secret key. The algorithm and the key are selected with the preceding MSE command.

PERFORM SECURITY OPERATION: **ENCIPHER**	ISO/IEC 7816-8	Encrypts data with a secret key. The algorithm and the key are selected with the preceding MSE command.
GET CHALLENGE	ISO/IEC 7816-4	Retrieves a challenge for authenticating a challenge/response PIN. Can be used to request random bytes from the card's RNG also for other purposes.
GENERAL AUTHENTICATE	ISO/IEC 7816-4	INS = 86h: Calculates and returns a shared secret using the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol. INS = 87h: Establishes a secure messaging session or computes a digital signature in the PIV interface mode. (see Table 4)
INTERNAL AUTHENTICATE	ISO/IEC 7816-4	This command is used to authenticate the card, i.e. verify that it is a genuine MyEID card manufactured by Aventra. The card signs random data provided by both the card and the user, combined with version information. The signature can be verified with a signature chain which leads to a public key published by Aventra.
DEAUTHENTICATE	-	Deauthenticates selected access conditions without resetting the card.

12.2.2 Personalisation and management commands

Table 3 – MyEID applet personalisation and management related commands.

Command	**Standard**	**Functionality**
PUT DATA: **INITIALISE APPLET**	ISO/IEC 7816-4	Initialises the file system and pre-creates the permanent files (MF and DF 5015 (PKCS#15).
PUT DATA: **INITIALISE PIN**	ISO/IEC 7816-4	Initialises the requested PIN reference data and its unblocking reference data (PUK).
ACTIVATE APPLET	ISO/IEC 7816-9	Activates the applet. Sets files to Activated State.
CREATE FILE	ISO/IEC 7816-9	Creates a file in the applet's file system under the current DF.
DELETE FILE	ISO/IEC 7816-9	Deletes the current file.
GENERATE KEY PAIR	ISO/IEC 7816-8	Generates an RSA or ECC key pair in the current key EF. *
PUT DATA: **LOAD KEY**	ISO/IEC 7816-4	Loads a component of an externally generated RSA* or ECC key pair, or a DES or AES key to a key EF
PUT DATA: **SET SESSION FLAGS**	ISO/IEC 7816-4	This command sets parameters that remain effective until next reset of the card.
PUT DATA: **INITIALISE PIV EMULATION**	ISO/IEC 7816-4	Initialise PIV emulation and map key and certificate EFs to objects defined in PIV specification.
PUT DATA: **ACTIVATE AND DEACTIVATE PIV EMULATION**	ISO/IEC 7816-4	Switch between MyEID native interface and PIV interface.
PUT DATA: **SET SECURE MESSAGING PARAMETERS**	ISO/IEC 7816-4	Set parameters for Secure Messaging.

* Applet version 2.2.0 onwards also supports 2048 bit keys. Earlier versions support only 1024 bit keys. MyEID3 supports key sizes from 512 to 2048 bit in 64 bit increments and MyEID 4.5 and newer supports RSA keys up to 4096 bit key length.

12.2.3 PIV interface commands

The commands listed in the following table are available in the PIV / CIV emulation mode. Documentation of the commands is available in [PIV].

Table 4 – Commands available in the PIV emulation mode

Command	**Standard**	**Functionality**
SELECT	ISO/IEC 7816-4, NIST.SP.800-73-5	Selects the PIV Card Application.

VERIFY	ISO/IEC 7816-4, NIST.SP.800-73-5	Verifies reference data presented by the user (PIN) with the reference data stored in the applet. *
CHANGE REFERENCE DATA	ISO/IEC 7816-8, NIST.SP.800-73-5	Changes the current reference data (PIN).
RESET RETRY COUNTER	ISO/IEC 7816-8, NIST.SP.800-73-5	Unblocks a blocked reference data (PIN).
GET DATA (INS=CBh)	ISO/IEC 7816-4, NIST.SP.800-73-5	Retrieves the data content of the single data object whose tag is given in the data field.
GENERAL AUTHENTICATE (INS = 87h)	ISO/IEC 7816-4, NIST.SP.800-73-5	Compute a digital signature using the PIV authentication key or establish a secure messaging session.

* MyEID emulation maps PIV Card Application PIN to PIN#1. Other pins defined in [PIV] are not supported.

12.3 Command response status bytes

Table 5 – General status bytes

SW1 / SW2	Functionality
9000h	Command successful
61xxh	Bytes remaining, xx = number of bytes
63Cxh	PIN verification trials left, x = trials left
6700h	Incorrect data length or wrong length
6881h	Logical channel not supported
6882h	Secure messaging not supported
6884h	Command chaining not supported
6981h	Incorrect file type
6982h	Security status not satisfied
6983h	PIN blocked or PIN not initialised or File invalid
6984h	Invalid data
6985h	Conditions not satisfied
6986h	Command not allowed
6987h	Expected secure messaging data objects missing
6988h	Incorrect secure messaging data objects
6A80h	Incorrect parameters in the data field / Wrong data
6A81h	Function not supported
6A82h	File or tag not found
6A83h	PIN not initialised
6A84h	Not enough memory space in the file
6A86h	Incorrect P1 and/or P2 parameters
6A88h	Referenced data not found
6A89h	File already exists
6A8Ah	DF name already exists
6B00h	Offset exceeds file length
6Cxxh	Correct length, xx bytes
6D00h	Unsupported command instruction
6E00h	Class not supported
6F00h	No precise diagnosis

12.4 Mapping between commands, user roles and states

The following tables contain mapping of commands to user roles and applet states, aiming to show in which states the commands are available, is authentication required, and in case authentication is required, which user role must be authenticated to execute the commands. The commands which are available in Creation State do not require authentication in Creation State but may require authentication in the Operational State. Commands available in Creation State are considered to be available only for the Admin role, because even though they do not require authentication, the TOE is not delivered to End Users in Creation State.

Admin role is mapped to the PIN which is assigned to the Recreate MF security attribute. End User role is mapped to any other PIN. Operations, which are marked "by sec. attrs." can be configured by setting the security attributes of files to be available to Admin, End user or freely without authentication with the exception of crypto-operations, which cannot be assigned the "Allowed at all times" security attribute. If an operation is allowed for User for a specific file, it is not allowed for Admin and vice-versa. However, when access is allowed for the user role, access can be allowed also for the admin role using the admin state. To enable admin access for a file/operation mapped to the user role, the admin PIN must have the admin-flag and the file must have admin access bit flag for the operation set, e.g. Admin use key.

See 13.2.1 and Table 24 - EF File Control Information data for details about security attributes and admin access flags.

Table 6 – Commands to states and user roles mapping – regular commands

Command	Creation State	Operational State	Without authentication	End User	Admin
GET DATA (INS = CAh)	X	X	X	X	X
SELECT FILE	X	X	X	X	X
GET RESPONSE	X	X	X	X	X
READ BINARY	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
UPDATE BINARY	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
ERASE BINARY	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
VERIFY	X	X	X	X	X
CHANGE REFERENCE DATA	X	X		X	X
RESET RETRY COUNTER	X	X		X	X
MANAGE SECURITY ENVIRONMENT: **RESTORE**	X	X	X	X	X
MANAGE SECURITY ENVIRONMENT: **SET**	X	X	X	X	X
PERFORM SECURITY OPERATION: **COMPUTE DIGITAL SIGNATURE**	If enabled *	X		by sec. attrs.	by sec. attrs.
PERFORM SECURITY OPERATION: **DECIPHER**	If enabled *	X		by sec. attrs.	by sec. attrs.
PERFORM SECURITY OPERATION: **ENCIPHER**	If enabled *	X		by sec. attrs.	by sec. attrs.
GET CHALLENGE	X	X	X	X	X
GENERAL AUTHENTICATE	If enabled *	X	**	X	X
INTERNAL AUTHENTICATE	X	X	X	X	X
DEAUTHENTICATE	X	X	X	X	X

* Cryptographic operations are enabled in Creation State, when the Key EF is in Created State or when enabled using PUT DATA: Set Session Flags command. If enabled, crypto-operations remain enabled until the next reset.

** GENERAL AUTHENTICATE requires authentication for ECDH and when used with PIV authetication key. When used for establishin a secure messaging session, authentication is not required.

Table 7 – Commands to states and user roles mapping – Personalisation and management commands

Command	**Creation State**	**Operational State**	**Without authentication**	**End User**	**Admin**
PUT DATA: **INITIALISE APPLET**	X	X			X
PUT DATA: **INITIALISE PIN**	X	X			X
ACTIVATE APPLET	X		X		X
CREATE FILE	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
DELETE FILE	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
GENERATE KEY PAIR	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
PUT DATA: **LOAD KEY**	X	X	by sec. attrs.	by sec. attrs.	by sec. attrs.
PUT DATA: **SET SESSION FLAGS**	X	X	X	X	X
PUT DATA: **INITIALISE PIV EMULATION**	X	X			X
PUT DATA: **ACTIVATE AND DEACTIVATE PIV EMULATION**	X	X			X
PUT DATA: **SET SECURE MESSAGING PARAMETERS**	X	X			X

13 Regular Commands

13.1 GET DATA

The GET DATA command retrieves information about the applet and its current state. The type of the returned information depends on parameter P2.

Table 8 - GET DATA command APDU

Bytes	Value
CLA	00h or 0Ch for secure messaging
INS	CAh: GET DATA CBh: GET DATA, PIV interface. See [PIV] for details.
P1	01h: Get MyEID proprietary information 7Fh: Get Windows Smart Card Framework Card Identifier
P2	**P1 = 01h:** Get Key File information (a key file must be selected): 00h: algorithm identifier, see Table 10. 01h or 81h: modulus (RSA keys only), see Table 11. 02h or 82h: public exponent (RSA keys only), see Table 12. 50h: PIV state of "Conditions not Satisfied" in case the PIV emulation feature is not activated. See Table 75 for details. 55h: Secure Messaging parameters. See Table 81 for details. 5Fh: List of session objects in the selected DF. See Table 16 for details. ECC key information (for currently selected ECC key EF): 81h-85h: Elliptic curve parameters, see Table 13. 86h: ECC key's public point in uncompressed format, DER encoded within tag 86h. e.g. [CONTEXT_SPECIFIC 6] IMPLICIT 04 ?? ?? ?? ?? ?? … 87h: ECC key's public point in uncompressed format, without DER tagging. 88h: ECC curve OID. Returns the object identifier of the named curve of the selected ECC key EF. See the Encoded OID column in Table 93. Get other information: A0h: MyEID Applet information, see Table 14. A1h: current DF file list, see Table 16. A2h: same as A1h, but only includes EF's A3h: same as A1h, but only includes DF's A4h: same as A1h, but only includes RSA key EF's A5h: same as A1h, but only includes ECC key EF's A6h: same as A1h, but only includes symmetric key EF's A7h: same as A1h, but only includes generic secret key EF's.

	A8h: Get full path of currently selected EF, see Table 17. A9h: Get full path of currently selected DF, see Table 17 AAh: MyEID card capabilities, see Table 15. * ABh: Trigger a self-test. Returns 9000h on success. *** ACh: Access conditions states, see Table 19 AEh: Active features. One byte with combination of bits indicating which optional features of MyEID are currently active. See Table 20. ADh: CPLC data. Information about the IC and the operating system. ** Get PIN information: BXh: X defines the PIN reference number. See Table 18. Get session flags: E5h: Get current session flags. See data field in 14.11. Card genuineness verification: E6h: Get card authentication public key. Returns the public key of the card's authentication key pair. (EC public point in uncompressed format). E7h: Get card authentication signature chain. See 16.2 for details. Free space: F5h: Get available free space on the card. Returns a big endian four-byte integer. **P1 = 7Fh:** 68h: Get Windows Smart Card Framework Card Identifier. See Microsoft Smart Card Minidriver Specification v7.07, Appendix D, 4.6 for details.
LC	Empty
Data	Empty
LE	Zero or length of requested data

* P2 value AAh: MyEID card capabilities is available beginning from MyEID 4.0.0
** Format of the returned data is out of scope of this document
*** If the self-test fails, remove the card from use and contact Aventra.

Table 9 - GET DATA response APDU

Bytes	**Value**
Data	Requested data, see details in tables below
SW1 – SW2	Status Bytes

The GET DATA response command APDU data contains the information requested by with the specific combination of parameters P1 and P2.

Table 10 – Algorithm Identifier

Bytes	Length	Value
Algorithm Identifier	2	9200h: RSA
Modulus size	2	XXXXh: length of modulus in bits
Public exponent size	2	XXXXh: length of public exponent in bits

Table 11 – Modulus

Bytes	Length	Value
Modulus	N	RSA key modulus, N bytes.

Table 12 – Public Exponent

Bytes	Length	Value
Public Exponent	N	RSA key public exponent, N bytes.

Table 13 – Elliptic Curve Parameters

P2	Value
81h	Prime (a number denoted as p coded on z bytes)
82h	First coefficient (a number denoted as a coded on z bytes)
83h	Second coefficient (a number denoted as b coded on z bytes)
84h	Generator (a point denoted as PB on the curve, coded on 2z bytes with leading uncompressed point indicator 04h)
85h	Order (a prime number denoted as q, order of the generator PB, coded on z bytes)

Table 14 – Applet Information

Bytes	Length	Value
Name	5	“MyEID”
Major version	1	XXh: major version number
Minor version	1	XXh: minor version number
Version revision	1	XXh: revision number
Unique identifier	10	Unique identifier, different from card to card
Change counter	2	XXXXh: Incremented after every successful command that changes the contents of the applet.

Table 15 – MyEID card capabilities

Bytes	Length	Value
Structure version	1	1: MyEID card capabilities – version 1 or 2. Version number can be increased in future to indicate that new information is added to the end of the structure
MyEID card capabilities	2	Specifies capabilities supported in this version of MyEID. bit flags, bytes in big endian order: bit 0 (0001h): RSA algorithm bit 1 (0002h): DES/3DES algorithm bit 2 (0004h): AES algorithm bit 3 (0008h): ECDSA and ECDH algorithms bit 4 (0010h): GridPIN bit 5 (0020h): PIV emulation bit 6 (0040h): Supports Windows Smart Card Framework ID. bits 6-15: RFU
Maximum RSA key length	2	Maximum key size in bits
Maximum DES/3DES key length	2	Maximum key size in bits
Maximum AES key length	2	Maximum key size in bits
Maximum ECC key length	2	Maximum key size in bits
Security certifications	v < 5: 0 v >= 5: 1	Structure version 1: not included Structure version 2 (since MyEID 5): Bit flags: Bit 0: If set, the card is of Common Criteria certified version.

Table 16 – Current DF file list

Bytes	Length	Value
First FID	2	XXXXh: FID of the first file in current DF
Second FID	2	XXXXh: FID of the second file in current DF
…	…	…
Nth FID	2	XXXXh: FID of the Nth file in current DF

Table 17 – Current DF or EF path

Response data
The path is returned in reverse order, current file ID first. For example: 4B0150153F00 If no file is selected, empty data is returned with 9000h status code.

Table 18 – PIN information

Bytes	Length	Value
PIN tries remaining	1	Number of verification tries remaining
PUK tries remaining	1	Number of unblocking tries remaining
PIN tries	1	Number of verification tries initially or after successful VERIFY or RESET RETRY COUNTER command
PUK tries	1	Number of verification tries initially or after successful RESET RETRY COUNTER command.
Status byte	1	Status byte containing PIN flags, see Initialise PIN command. Bit 6 (40h) is set if the PIN is in authenticated state.
PIN type	1	00h = Normal PIN, 01h = Grid PIN, 02h = Challenge / response PIN with 3DES algorithm, 03h = Challenge / response PIN with AES algorithm.
Grid size	1	Grid size for Grid PINs or 00h
Minimum length of PIN	1	Minimum length of PIN reference data
Minimum length of PUK	1	Minimum length of PUK reference data

Table 19 – Access condition states – two bytes of bit flags

Value	Meaning
0x8000	Admin state active
0x4000	Global unblocker state active
0x2000	PIN 14 verified
...	PINs 2-13
0x0001	PIN 1 verified

Table 20 – Active features

Value	Meaning
02h (bit 1)*	Windows Smart Card Framework Identifier (WSCF ID)
04h (bit 2)	PIV emulation*
08h (bit 3)	Creation state compatibility mode
10h (bit 4)	Generic compatibility mode

* index of the lowest bit is zero.

** This value indicates that PIV emulation is available. To use the PIV emulation feature, it must be initialised using PUT DATA: INITIALISE PIV EMULATION command.

13.2 SELECT FILE

The SELECT FILE command selects the applet itself with the Application Identifier (AID) or a file within the applet. A file can be selected by a single File Identifier (FID) or by relative or absolute path. A path consists of a sequence of FID's.

Table 21 - SELECT FILE command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	A4h
P1	00h: select EF, DF or MF by single FID 04h: select applet by AID or a DF by its name 08h: select file by absolute path from MF 09h: select file by relative path current DF
P2	00h: FCI returned in response
LC	absent or length of data

Data	P1 = 00h:
	EF, DF or MF FID (or empty for MF)
	P1 = 04h:
	AID value (for applet or DF name)
	P1 = 08h:
	absolute path from MF without the FID for MF
	P1 = 09h:
	relative path from the current DF without the FID of current DF

LE	00h or length of the expected data.

Table 22 - SELECT FILE response APDU

Byte	Value
Data	File Control Information (FCI), empty if protocol is T=0, when LE is absent or on error
SW1 – SW2	Status Bytes 0x6A82 – File not found

The SELECT FILE response command APDU data holds specific information about the selected file.

Table 23 - MF, DF, and Applet File Control Information data

Byte	Length	Value
FCI tag	1	6Fh
Length	1	Length of the DER encoded structure
File Size tag	1	81h
Length	1	02h
File Size	1-2	XXXXh: maximum size of a new file *
File Description tag	1	82h
Length	1	01h
File Descriptor	1	38h: DF or MF
File Identifier tag	1	83h
Length	1	02h
File Identifier	2	XXXXh: FID
Security Attributes tag	1	86h
Length	1	03h
Security Attributes	3	XXXXXXh: See Table 26 for details of MF See Table 27 for details of DF
Proprietary Information tag	1	85h: File status byte
Length	1	02h
Proprietary Information	2	First byte: RFU Second byte: bit flags bit 1 (02h): permanent file; DF cannot be deleted with DELETE FILE command bit 5 (20h): Admin state allows to initialise applet (MF), Admin state allows to delete this file (DF) bit 6 (40h): Admin state allows to create EFs bit 7 (80h): Admin state allows to create DFs other bits RFU.
Life Cycle Status tag	1	8Ah
Length	1	01h
Life Cycle Status	1	X1h: creation state X7h: operational state – activated The high nibble is RFU

DF Name tag	1	84h: Optional tag
Length	1	01h – 10h
DF Name	1-16	

* The maximum value returned withing the file size tag is 7FFFh (32767). Please use GET DATA with P1=01h and P2=F5h to determine total available free space on the card. Please note that a new file requires slightly more memory than the value returned in FCI of MF, because of the overhead needed for the internal Java object which represents the file.

Table 24 - EF File Control Information data

Byte	Length	Value
FCI tag	1	6Fh
Length	1	Length of the DER encoded structure
File Size tag	1	80h
Length	1	01h or 02h
File Size	1-2	Transparent EF’s: XXXXh: File size in bytes RSA key EF’s: 0400h to 1000h* for a private RSA key EF (=key size in bits, or modulus, of the key EF) MyEID3 accepts values from 0200h up to 0800h in 64 bit increments, MyEID 4.5 accepts values up to 1000h, MyEID 5 accepts values between 0800h to 1000h. ECC key EF’s: XXXXh: key length in bits (field size): 256, 320**, 384**, 512** or 521** DES key EF’s: **** 0040h: 64 bit DES key 0080h: 128 bit 3DES key (two key) 00C0h: 192 bit 3DES key (three key) AES key EF’s: 0080h: 128 bit key 0100h: 256 bit key
File Description tag	1	82h
Length	1	01h

File Descriptor	1	01h: transparent EF 11h: private RSA key EF 19h: DES key EF **** 22h: EC key EF 29h: AES key EF 41h: Generic secret key EF ***
File Identifier tag	1	83h
Length	1	02h
File Identifier	2	XXXXh: FID
Security Attributes tag	1	86h
Length	1	03h
Security Attributes	3	XXXXXXh: See Table 28 for details of transparent EF See Table 29 for details of private key EF
Proprietary Information tag	1	85h: File status byte
Length	1	02h

Proprietary Information	2	First status byte: Transparent EF's 00h: RFU Key EF's X0h: key not valid X1h: key valid X3h: key valid, key generated on card X = AC to be cleared after a cryptographic operation. The remaining bits are RFU for EF Key, and for the other file types the whole byte is RFU. Second status byte: Bit flags: bit 0 (01h): Session object: The key EF is automatically removed during next reset. bit 1 (02h): RFU bit 2 (04h): Auto size flag – the file grows dynamically when writing past EOF bit 3 (08h): Extractable: The key can be wrapped. bit 4 (10h): Admin generate key (RSA/EC key EF) bit 5 (20h): Admin delete file bit 6 (40h): Admin update (Transparent EF), Admin PUT DATA / Unwrap (Key EF) bit 7 (80h): Admin read (Transparent EF), Admin use / wrap key (Key EF)
Life Cycle Status tag	1	8Ah
Length	1	01h
Life Cycle Status	1	X1h: creation state X7h: operational state – activated The high nibble is RFU

* 0800h is supported from MyEID applet version 2.2.0 onwards.
** Only supported on some of the MyEID platforms
*** a Generic Secret is a secret key which is not associated with an algorithm and cannot perform crypto-operations. I.e. it is a secret file that is treated like a key.
**** deprecated and not available in MyEID 5.

13.2.1 File Security Attributes

The operations that can be performed on a file, such as reading and updating, are controlled by File Security Attributes. Every file has six Security Attributes, coded in four bits each, and totalling three bytes. A given Security Attribute indicates which Access Condition must be fulfilled before the operation in question can be performed. The Access Conditions can be fulfilled by executing the VERIFY command with the correct data.

The exact meaning of the six File Security Attributes depends on the file type, as listed in tables Table 25 to Table 29. Each Security Attribute is coded in four bits as follows:

Table 25 - File Security Attribute values

Value	Meaning
0h	The operation is allowed at all times
1h…Eh	Indicates the reference number of the PIN that must be valid before the operation can be performed
Fh	The operation is forbidden

Table 26 - MF Security Attributes

Position	Meaning
X00000h	Create DF's
0X0000h	Create EF's
00X000h	Recreate MF (removes all existing files)
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

Table 27 - DF Security Attributes

Position	Meaning
X00000h	Create DF's
0X0000h	Create EF's
00X000h	Delete File (this file)
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

Table 28 - Transparent EF Security Attributes

Position	Meaning
X00000h	Read
0X0000h	Update / Erase
00X000h	Delete File (this file)
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

Table 29 – Key EF Security Attributes

Position	Meaning
X00000h	Use / Wrap
0X0000h	Put Data / Unwrap
00X000h	Delete File (this file)
000X00h	Generate (RSA and EC key EF's)
0000X0h	RFU
00000Xh	RFU

Please see section 17.1 for guidance on configuring secure values for security attributes.

13.3 GET RESPONSE

The GET RESPONSE command returns the response of the APDU when T=0 protocol is used, and to retrieve subsequent blocks of a multi-part response with T=1 protocol, when using short APDUs. The response can only be retrieved once, and the GET RESPONSE command must be executed immediately after the command that generated the response.

This command retrieves the response data of the following commands:

- SELECT FILE,
- PSO: COMPUTE DIGITAL SIGNATURE, and
- PSO: DECIPHER

Table 30 - GET RESPONSE command APDU

Byte	Value
CLA	00h
INS	C0h
P1	00h

P2	00h
LC	Empty
Data	Empty
LE	Data length of the response to retrieve

Table 31 - GET RESPONSE response APDU

Byte	Value
Data	Previous APDU commands response data. Empty on error
SW1 – SW2	Status Bytes

13.4 READ BINARY

The READ BINARY command reads data from transparent EF’s. The file to read must be selected first with the SELECT FILE command.

Table 32 - READ BINARY command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	B0h
P1	00-7Fh: MSB of the 15 bit offset to the first byte to read
P2	00-FFh: LSB of the 15 bit offset to the first byte to read
LC	Empty
Data	Empty
LE	Number of bytes to read.

Table 33 - READ BINARY response APDU

Byte	Value
Data	Bytes read, or empty on error
SW1 – SW2	Status Bytes. See Table 38 for related response values.

13.5 UPDATE BINARY

The UPDATE BINARY command updates data in transparent EF’s. The file to update must be selected first with the SELECT FILE command.

Table 34 - UPDATE BINARY command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	D6h
P1	00-7Fh: MSB of the 15 bit offset to the first byte to update
P2	00-FFh: LSB of the 15 bit offset to the first byte to update
LC	Length of data to update
Data	Data to be written
LE	Empty

Table 35 - UPDATE BINARY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes. See Table 38 for related response values.

13.6 ERASE BINARY

The ERASE BINARY command erases bytes starting at the requested offset until the end of the transparent EF. Erasing means removing the affected bytes from the file, so the file shrinks the number of bytes erased. The file to erase must be selected first with the SELECT FILE command.

Table 36 - ERASE BINARY command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	0Eh
P1	00-7Fh: MSB of the 15 bit offset to the first byte to erase
P2	00-FFh: LSB of the 15 bit offset to the first byte to erase
LC	Empty
Data	Empty
LE	Empty

Table 37 - ERASE BINARY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes. See Table 38 for related response values.

Table 38 - Response Status Byte values for file I/O commands

SW1 – SW2	Description
9000h	The command succeeded
6B00h	Offset exceeds file length
6700h	Wrong length
6982h	Security status not satisfied (access condition required for the operation is not in authenticated state).

Note: File I/O commands can return also other status byte values defined in ISO 7816-4. This table includes the values for successful execution and typical error situations.

13.7 VERIFY

The VERIFY command is used to authenticate the user. The verification data (PIN) is compared internally with the reference data stored in the applet. A successful verification sets the Access Condition (AC) in parameter P2, which enables the execution of commands that are restricted by that AC in the File Security Attributes of the file in question. This command can also be used just to inquire the status of the given Access Condition. This is achieved by executing the command without any reference data.

Verification can fail if the presented PIN is incorrect, or the PIN is blocked or locked.

13.7.1 Blocked PIN

Each time the PIN is verified with this command, a retry counter connected to this PIN is updated. If the verification was successful, the counter is reset to its original value of 5 (or optionally a value that has been configured when the PIN was created). If the verification fails, the counter will be decremented. If the counter reaches zero, the PIN will be blocked. To unblock it, a Reset Retry Counter command must be executed with the correct PUK (PIN Unblocking Code) and a new PIN. This also resets the retry counter to the original value.

13.7.2 PIN locking

Besides blocking, it is also possible to configure the PIN in such a way that it can be locked. The locking can be set to occur in two situations: After the creation of the PIN, and after unblocking the PIN with the Reset Retry Counter command. A locked PIN can be unlocked by changing it with the Change Reference Data command. This optional feature can be used to enforce the changing of an initial PIN when user receives his card. This can be desirable if all the cards initially have a constant PIN. A locked PIN will cause the Verify command to fail with the error code SW1SW2 = 0x6985 (*CONDITIONS NOT SATISFIED*).

13.7.3 Admin and Global Unblocker states

If the verified PIN is an admin PIN or a global unblocker PIN, the relevant state is authenticated, when PIN is verified or changed successfully. Admin state and global unblocker state remain authenticated until one of the following events occur:

- Selecting the applet, either explicitly or due to card reset.

- An explicit DEAUTHENTICATE command with P1 = A0/B0 is issued.
- DEAUTHENTICATE command for all PINs (P1 = 00) or all PINs that enable the respective state is issued, or the PINs get deauthenticated indirectly, e.g. after a crypto-operation (clear AC after crypto) or unblocking.
- VERIFY command with any PIN that does not enable the respective state is issued. (does not apply to MyEID 5)

Table 39 - VERIFY command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	20h
P1	00h: Verify PIN FFh: Deauthenticate PIN (MyEID 5)
P2	PIN reference number 01h to 0Eh
LC	00h: No data, just query the status of the PIN 08h: PIN reference data present in the data field C/R pins: >= 08h: Response for a challenge present in the data field
Data	Normal PINs: Empty or PIN reference data (verification data) padded to 8 bytes. Byte value 00h or FFh can be used for padding. C/R PINs: response to a challenge. The data length must match the length of the challenge acquired using GET CHALLENGE command, and the challenge/response length must be a multiple of the block size of the algorithm. The response is calculated using either 3DES or AES algorithm in CBC mode according to the PIN type. IV of zero bytes is used.
LE	Empty

Table 40 - VERIFY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 41 - Response Status Byte values

SW1 – SW2	Description
9000h	Verification successful and/or AC acquired
6985h	PIN locked – *CONDITIONS NOT SATISFIED*
6983h	Verification failed and no number of retries left PIN blocked.
63CXh	Verification failed and X number of retries left.

13.8 CHANGE REFERENCE DATA

The CHANGE REFERENCE DATA command replaces the current internal reference data with a new value. The current reference data is first validated with the verification data. Successful validation also removes optional reference data locking.

Table 42 - CHANGE REFERENCE DATA command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	24h
P1	00h: exchange reference data
P2	PIN reference number 01h to 0Eh
LC	PIN change using verification reference data: 10h PIN change using admin state: 08h For C/R pins, LC depends on the length of the challenge requested and the C/R pin algorithm.
Data	PIN change using verification reference data: Verification reference data padded to 8 bytes, followed by the new reference data padded to 8 bytes. C/R pins: response to a challenge followed by a new C/R key. Length depends on challenge length and C/R key algorithm. PIN change using admin state: Only 8 bytes of new reference data or new C/R key for C/R pins. A PIN with admin state flag must be verified for the operation to succeed. Byte value 00h or FFh can be used for padding.
LE	Empty

Table 43 - CHANGE REFERENCE DATA response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes: 9000h – Command successful 6983h – Verification failed and/or no number of retries left PIN blocked. 63CXh – Verification failed and/or X number of retries left.

13.9 RESET RETRY COUNTER

The RESET RETRY COUNTER command unblocks a PIN which has been blocked after too many unsuccessful verification trials. Successful unblocking requires the correct unblocking reference data (PUK).

Successful execution of this command sets the reference data into locked state if the relocking after reset retry counter option is used. The relocking flag is optionally set with the PUT DATA: INITIALISE PIN command.

When using admin or global unblocker state to unblock a PIN, the state is deauthenticated in case only one PIN with admin or global unblocker state has been authenticated. If several PINs, which authenticate admin or global unblocker state are validated, only the one with the lowest index is deauthenticated.

Table 44 - RESET RETRY COUNTER command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	2Ch
P1	00h: reset retry counter and set reference data
P2	PIN reference number 01h to 0Eh
LC	00h: query status of unblocking reference data 10h: reset retry counter 08h (normal pins) or new C/R key length: reset retry counter using admin state XXh: reset retry counter of a challenge/response PIN. Length depends on C/R key length.
Data	Using PUK: Empty or unblocking reference data padded to 8 bytes, followed by the new reference data padded to 8 bytes (alphanumeric PIN), 24 bytes 3DES key or 16-, 24- or 32-byte AES key (challenge/response PIN). Using admin state: New reference data padded to 8 bytes (alphanumeric PIN), 24 bytes 3DES key (challenge/response PIN) or 16-, 24- or 32-byte AES key. Byte value 00h or FFh can be used for padding.
LE	Empty

Table 45 - RESET RETRY COUNTER response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 46 - Response Status Byte values

SW1 – SW2	Description
9000h	Command successful
6985h	PIN locked – *CONDITIONS NOT SATISFIED*
6983h	Verification failed and no number of retries, PUK blocked.
63CXh	Verification failed and X number of retries left. This status code is returned also when querying number of tries left, but the counter is not decreased in this case.

13.10 DEAUTHENTICATE

Deauthenticate command can be used to deauthenticate all PINs without resetting the card, to deauthenticate a specific PIN or to deauthenticate Admin state or Global Unblocker state.

Table 47 – DEAUTHENTICATE command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	2Eh
P1	00h: Deauthenticate PIN A0h: Deauthenticate admin state B0h: Deauthenticate global unblocker state
P2	P1 = 00h: Pin reference (01h-0Eh) or 00h to deauthenticate all PINs and states P1 <> 00h: P2 must be 00h
LC	Length of data field
Data	Empty
LE	Empty

Table 48 – DEAUTHENTICATE response APDU

Byte	Value
Data	Empty
9000h	The PINs requested to deauthenticate by P2 are in deauthenticated state after the command.
6985h	*CONDITIONS NOT SATISFIED* – the state requested in P1 to deauthenticate was not in authenticated state. *

* With P1=00h, the command always returns 9000h, regardless of whether any of the PINs requested to deuthenticate were in authenticated state. With P1=A0h and P1=B0h, the command returns 6985h, in case the state requested to deauthenticate was not in authenticated state.

13.11 MANAGE SECURITY ENVIRONMENT: RESTORE

The MANAGE SECURITY ENVIRONMENT: RESTORE command restores an empty or predefined Security Environment (SE).

Table 49 - MANAGE SECURITY ENVIRONMENT: RESTORE command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	22h
P1	F3h: Restore SE
P2	00h: SE reference number (00h denotes an empty SE)*
LC	Empty
Data	Empty
LE	Empty

* In MyEID 5 P2 must be 00h. MyEID 5 maintains only one SE at time.

Table 50 - MANAGE SECURITY ENVIRONMENT: RESTORE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

13.12 MANAGE SECURITY ENVIRONMENT: SET

The MANAGE SECURITY ENVIRONMENT: SET command sets attributes in the current Security Environment (SE). MyEID 5 requires the complete SE to be set in a single MSE: SET command. See section 17 for security implications of the options (e.g. padding modes and AES operation modes) selectable in the CRDO structure.

Table 51 - MANAGE SECURITY ENVIRONMENT: SET command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	22h
P1	41h: computation, decipherment or key agreement 81h: encipher

P2	B6h: attributes of DST in data field B8h: attributes of CT in data field A4h: authentication template in the data field (use before GENERAL AUTHENTICATE) See Table 53 for allowed P2 values for each P1 value.
LC	Empty or length of Data field
Data	Concatenation of Control Reference Data Objects (CRDO)
LE	Empty

Table 52 - MANAGE SECURITY ENVIRONMENT: SET response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 53 - MANAGE SECURITY ENVIRONMENT: P1 and P2 mapping

		P2		
		B6h	B8h	A4h
P1	41h	x	x	x
	81h		x	

Table 54 - Control Reference Data Objects (CRDO)

Tag	Meaning
80h	XXh: Algorithm Reference (see Table 55 for details). Use 00h for symmetric key operations, algorithm is selected according to key type.
81h	XXXXh: File Reference; File Identifier (FID) - only FIDs shall be used - identifies key EF to be used in PSO commands value according to PKCS#15: private key objects PKCS15Path.path **Note:** DF containing the referenced key file must be selected when executing the MSE command and during the following PSO commands associated with the SE.

83h	Key wrapping and unwrapping P1=41h: Key unwrapping P1=81h: Key wrapping XXXXh: Target File Reference; File Identifier (FID) - only FIDs shall be used - Key unwrapping: identifies the file which the unwrapped key material will be placed into. - Key wrapping: identifies the file which contains the key to be wrapped. Symmetric operations P1=41h: Decipher P1=81h: Encipher 00h: Symmetric key reference MyEID supports only one key in each key file Other use cases Not used, must not be present
84h	P1=41h: Asymmetric operations: 00h: Key Reference (references the private key in a key file) Key unwrapping with an asymmetric key: Not present. MyEID supports only one key in each key file P1=81h: 00h: Not used, must not be present

87h	P1=41h: Symmetric key operation: Initialisation vector.
	Asymmetric key operation: Not used, must not be present.
	P1=81h: Initialisation vector

Table 55 - Values for Algorithm Reference

Algorithm Reference	**Meaning**
00h	AES keys: symmetric encipherment or decipherment
0Xh	No hash algorithm
1Xh	SHA-1 hash algorithm. Applicable only with OAEP and PKCS#1 padding.
2Xh	RFU
3Xh	SHA-224
4Xh	SHA-256
5Xh	SHA-384

6Xh	SHA-512
8Xh	Symmetric key wrapping operations: PKCS#7 padding
X0h	RSA keys: Raw RSA operation * No input or output data formatting, padding, or hash encapsulation is applied. PSO: COMPUTE DIGITAL SIGNATURE: 1. Input data must be equal to the RSA key modulus length. 2. PKCS#1 RSASP1 signature primitive is applied (private key operation). PSO: DECIPHER 1. PKCS#1 RSADP decryption primitive is applied (private key operation).
X1h	RFU
XAh	Key wrapping or unwrapping. Deprecated and not required for key wrapping or unwrapping as of MyEID 4.9.9 or newer. **
X2h	RSASSA-PKCS1-v1-5 signature scheme According to PKCS#1 v2.0 with RSA algorithm, compatible with PKCS#15 v1.5 PSO: COMPUTE DIGITAL SIGNATURE: - The data (hash code) is encapsulated into a DigestInfo ASN.1 structure according to the selected hash algorithm. If no algorithm is selected (algorithm reference = 02h) then no encapsulation is done by the applet. - DigestInfo is padded to RSA key modulus length according to PKCS#1 v1.5, block type 01h. Size of the DigestInfo must not exceed 40% of the RSA key modulus length. - PKCS#1 RSASP1 signature primitive is applied (private key operation). RSAAES-PKCS1-v1-5 encryption scheme According to PKCS#1 v2.0 with RSA algorithm, compatible with PKCS#15 v1.5 PSO: DECIPHER: - PKCS#1 RSADP decryption primitive is applied (private key operation). PKCS#1 v1.5 padding is removed
X3h	RFU (DSA)
X4h	Elliptic curve operation (ECDSA or ECDH)
X5h	PSO: COMPUTE DIGITAL SIGNATURE: - The data to be signed is padded to RSA key modulus length using PSS padding scheme. Input data shall be a precomputed digest, the length of which must match the selected hash algorithm. The same algorithm is used to generate the mask with MGF1. PSO: DECIPHER: - OAEP padding is expected and removed by the card.

* Raw RSA operation is not supported on 2048 bit keys when computing a digital signature. (MyEID v < 4.5)

** Since MyEID 4.9.9, the applet uses the same algorithm references for encryption and wrapping, and decipherment and unwrapping. The applet identifies the operation from PSO command's parameters. XAh is supported for backward compability.

13.13 PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE

The PSO: COMPUTE DIGITAL SIGNATURE command computes a digital signature. The Secure Environment (SE) attributes Algorithm and File Reference must be previously set with MSE: SET command.

Table 56 - PSO: COMPUTE DIGITAL SIGNATURE command APDU

Byte	Value
CLA	00h: single or last data block 0Ch: single block or last data block, secure messaging 10h: chained APDU, more data to follow 1Ch: chained APDU with secure messaging
INS	2Ah
P1	9Eh: return digital signature data
P2	9Ah: data field contains data to be signed
LC	Length of data field
Data	SE Algorithm Reference = 00h: * - Raw data, data length must be equal to the RSA key modulus length. SE Algorithm Reference = 02h: - DigestInfo data, data is padded by the applet to match the length of the RSA key modulus. SE Algorithm Reference = 12h: - SHA-1 Hash data, the hash is encapsulated into a DigestInfo structure and then padded to the full length of the RSA key modulus. SE Algorithm Reference = 04h: - Raw data, data length must be equal to the ECC key length. SE Algorithm Reference = X5h: - X = 03h – 06h. A digest according to the selected hash algorithm. Data length must match the selected hash algorithm, e.g. SHA-256 = 256 bits.
LE	Empty or length of response data

* 00h algorithm reference is not supported on 2048 bit keys. (MyEID v < 4.5)

Table 57 - PSO: COMPUTE DIGITAL SIGNATURE response APDU

Byte	Value

Data	Digital signature
	With ECC keys, the digital signature is returned in the following format:
	EC Signature:: = SEQUENCE OF {
	r INTEGER
	s INTEGER
	}

SW1 – SW2	Status Bytes

13.14 PERFORM SECURITY OPERATION: ENCIPHER

The PSO: ENCIPHER operation enciphers data transmitted in the command data field using symmetric encryption. The Secure Environment (SE) attributes Algorithm and File Reference must be previously set with MSE: SET command to select an AES or DES key file. The PSO: ENCIPHER command can also be used for key wrapping. In key wrapping the command data field is left empty and the card encrypts key material from a key file defined in SE's Target File attribute.

When using the ENCIPHER command with Secure Messaging and command chaining for symmetric encryption, each APDU in the chain is encrypted and MACed as a separate message, in contrary to other commands where the whole command chain is encrypted and MACed as a single message. In ENCIPHER command, the SM packet counter is increased for each message in the chain.

Table 58 - PSO: ENCIPHER command APDU

Byte	Value
CLA	00h: DO FINAL (finalises the cipher) 0Ch: DO FINAL (finalises the cipher), secure messaging 10h: UPDATE CIPHER (more data is coming in subsequent APDUs) 1Ch: UPDATE CIPHER, secure messaging
INS	2Ah
P1	84h: return encrypted data
P2	80h: data field contains plaintext 00h: key wrapping: the data field is absent
LC	Length of data field, must be a multiple of the cipher block size
Data	Plain text data to encipher

LE	Empty or length of response data

Table 59 - PSO: ENCIPHER response APDU

Byte	Value
Data	Encrypted data
SW1 – SW2	Status Bytes

13.15 PERFORM SECURITY OPERATION: DECIPHER

The PSO: DECIPHER command decrypts an encrypted data. The Secure Environment (SE) attributes Algorithm and File Reference must be set previously with MSE: SET command. PSO: DECIPHER command can also be used for unwrapping keys into the card. In this case, the command does not return the deciphered data, but places it into a key EF defined in SE's Target File attribute.

With MyEID v<4.5.0 the cryptogram must be presented in two parts when a 2048 bit RSA key is used.

When using the DECIPHER command with Secure Messaging and command chaining for symmetric decryption, each APDU in the chain is encrypted and MACed as a separate message, in contrary to other commands where the whole command chain is encrypted and MACed as a single message. In DECIPHER command, the SM packet counter is increased for each message in the chain.

Table 60 - PSO: DECIPHER command APDU

Byte	Value
CLA	RSA and AES keys: 00h: single or final block of data (DO FINAL) 0Ch: as above, with Secure Messaging 10h: chained APDU, more data to follow (UPDATE CIPHER) 1Ch: as above, with Secure Messaging
INS	2Ah
P1	80h: return decrypted data 00h: the response data field is absent
P2	86h: data field contains padding indicator concatenated with the data to be decrypted (RSA keys) 84h: data field contains encrypted data (AES and DES keys)
LC	Length of the data field
Data	Data for smaller than 2048 bit keys: 00h (pad indicator) || cryptogram Data for 2048 bit keys: 81h (pad and first half indicator) || first half of cryptogram 82h (pad and second half indicator) || second half of cryptogram

	MyEID > 4.5, 2048 bit and longer keys: 00 (pad indicator) || cryptogram following subsequent parts of the cryptogram split into APDUs with maximum 255 bytes of data. * MyEID >= 5: extended APDUs are can be used alternatively to submit the complete cryptogram in one command. AES and DES keys: encrypted data, must be aligned with the cipher block size
LE	Empty or length of response data

* The old method of splitting a 2048 bit cryptogram can also be used with MyEID 4.5.

Table 61 - PSO: DECIPHER response APDU

Byte	Value
Data	SE Algorithm Reference = 00h: - Decrypted data is returned including formatting. SE Algorithm Reference = 02h: - Decrypted data, PKCS#1 v1.5 padding is removed by the applet.
SW1 – SW2	Status Bytes

13.16 GENERAL AUTHENTICATE

In MyEID applet GENERAL AUTHENTICATE command from ISO 7816 is used to perform Elliptic Curve Diffie-Hellman (ECDH) key agreement. It takes the other party's public point as input and returns the shared secret in the response APDU.

With INS code 87h, GENERAL AUTHENTICATE command works as specified in PIV Card Application Card Command Interface. INS 87h requires the PIV application to be selected first.

Table 62– GENERAL AUTHENTICATE command APDU

Byte	Value
CLA	00h 0Ch: Secure Messaging in use
INS	86h: ECDH key agreement 87h: GENERAL AUTHENTICATE PIV

P1	INS = 86h: 00h
	INS = 87h: Refer to PIV Card Specification

P2	INS = 86h: 00h
	INS = 87h: Refer to PIV Card Specification

LC	Length of data field
Data	Data Objects in the Dynamic Authentication Template (Tag '7C'), see Table 64.
LE	Empty

Table 63 – GENERAL AUTHENTICATE response APDU

Byte	Value
Data	Shared secret
SW1 – SW2	Status Bytes

Table 64 – Dynamic Authentication Template for GENERAL AUTHENTICATE

Element	Length	Value
Dynamic Authentication Template tag	1	7Ch
Length	1	Length of the structure
Optional "Witness" tag. (RFU for nonce)	0-1	80h
Length	0-1	RFU, must be 00h if tag 80h is present.
Nonce	0	RFU, must be empty
Exponential tag	1	85h
Length	1	Length of public point
Public point	1	PUBLIC POINT in uncompressed format [04h, x, y]

14 Personalisation and Management Commands

14.1 PUT DATA: INITIALISE APPLET

The PUT DATA: INITIALISE APPLET command initialises the applet's file system. The first time this command is called after the applet has been installed; it also allocates the requested memory space for the file system.

After a successful execution, the applet will be in the Creation State, and all files will be in the Created State. In the Creation State all Access Conditions are disabled, i.e. the applet does not check them against the File Security Attributes in the files. In other words, the applet behaves as if all the Security Attributes were set to ALLOWED ALWAYS (00h).

PUT DATA: INITIALISE APPLET deletes all files and PINs on the card. In addition, the command resets existing file mappings of the PIV interface and resets secure messaging parameters. Running PUT DATA: INITIALISE APPLET command for a card which is in Operational State returns the card to Creation State. Running this command in Operational State requires authenticating PIN associated with Recreate MF access condition first.

Table 65 – PUT DATA: INITIALISE APPET command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	DAh
P1	01h
P2	E0h
LC	08h or 0Ah
Data	File system initialisation parameters, see Table 67 for details.
LE	Empty

Table 66 – PUT DATA: INITIALISE APPLET response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 67 – File system initialisation parameters

Bytes	Length	Value
File system size MyEID3: Maximum number of files	2	XXXXh MyEID v>=3: 0080h to 0200h (128 to 2048 files), values outside the range will be changed to the closest allowed value by the applet before execution.
MF ACL	3	Security Attributes of the Master File. See SELECT FILE command for details. Security attribute Recreate MF cannot be zero.
DF 5015 ACL	3	Security Attributes of DF 5015. See SELECT FILE command for details.
Admin MF ACL	1	Optional: Admin state ACL for MF bits 0-4: RFU bit 5 (20h): Admin state allows to reinitialise applet bit 6 (40h): Admin state allows to create EFs bit 7 (80h): Admin state allows to create DFs Default: 00h
Admin DF 5015 ACL	1	Optional: Admin state ACL for DF 5015. bits 0-4: RFU bit 5 (20h): Admin state allows to delete this file bit 6 (40h): Admin state allows to create EFs bit 7 (80h): Admin state allows to create DFs Default: 00h

14.2 ACTIVATE APPLET

The ACTIVATE APPLET command causes the applet to exit the Creation State and enter the Operational State. All the files are changed from Created Status to Activated Status. All Access Conditions become active and fully functional.

The PIN that has been associated with MF’s Recreate MF access condition must be initialised before executing the ACTIVATE APPLET command. Since MyEID 5, all PINs referenced in EF or DF ACLs must be initialised, or this command returns an error.

Table 68 – ACTIVATE APPLET command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	44h
P1	04h: applet AID in data field
P2	00h
LC	0Ch: Length of AID (data field)
Data	Applet AID (A000000063504B43532D3135)
LE	Empty

Table 69 – ACTIVATE APPLET response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

14.3 PUT DATA: INITIALISE PIN

The PUT DATA: INITIALISE PIN command initialises the requested PIN and its unblocking reference data (PUK). When the card is in Operational State, The PIN that has been associated with MF's Recreate MF access condition must be authenticated to initialise PINs. A PIN which has already been initialised, cannot be initialised again without executing PUT DATA: INITIALISE APPLET first.

Table 70 – PUT DATA: INITIALISE PIN command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	DAh
P1	01h
P2	01h: PIN #1 02h: PIN #2 03h: PIN #3 MyEID3 onwards: 01h to 0Eh
LC	10h to 37h
Data	PIN initialisation parameters, see Table 72 for details.
LE	Empty

Table 71 – PUT DATA: INITIALISE PIN response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 72 – PIN initialisation parameters

Bytes	Length	Value
PIN reference data	8	Padded with 00h or FFh
Unblocking reference data	8	Padded with 00h or FFh
* PIN retry counter	1	Lower nibble indicates the number of failed PIN verifications before the PIN is blocked. For example: X3h = 3 trials. X = RFU. Default value: 03h = 3 trials
* PUK retry counter	1	Lower nibble indicates the number of failed PUK verifications before the PUK is blocked. For example: XAh = 10 trials. X = RFU. Default value: 0Ah = 10 trials
* PIN flags	1	bit0: 0 = normal mode (default), 1 = PIN is locked after initialisation. PIN will be unlocked after successful execution of the Change Reference Data command. bit1: 0 = normal mode (default), 1 = PIN will be locked after a Reset Retry Counter command. bit2: Allow global unblocking (default = 0, not allowed) bit3: Global unblocker (default = 0, no global unblocker) bit4: Allow admin changing and unblocking (default = 0, not allowed) bit5: Administrator PIN (default = 0, is not admin PIN) (bit6: PIN authenticated, not applicable in PUT DATA: INITIALISE PIN) bit7: Allow PIN type change (default = 0, not allowed) **
* PIN type	1	00h = Alphanumeric PIN 01h = Grid PIN (Deprecated in MyEID 5) 02h = Challenge / response PIN, 3DES key *** 03h = Challenge / response PIN, AES key
* GridPIN grid size	1	MyEID 5: Deprecated, must be zero
* Minimum length of PIN	1	4-8. Takes effect when creating new PINs and when a PIN is changed.
* Minimum length of PUK	1	4-8. Default = 4

C/R pins only: Challenge / response key	16-32	Triple-DES key (192 bits) or AES key (128, 192 or 256 bits)

* Optional field
** Deprecated in MyEID 5, ignored
*** For CC EAL4+ compliant security with MyEID 5, do not user C/R PINs with 3DES algorithm. Use AES-based C/R PINs instead.

14.4 PUT DATA: INITIALISE PIV EMULATION

The PUT DATA: INITIALISE PIV EMULATION command activates the PIV/CIV interface and maps selected keys and certificates to corresponding PIV objects.

Before issuing this command, the keys and certificates to be used must be loaded on the card using MyEID command interface. Mapping objects is optional. For example, if only authentication key and certificate are needed, other FIDs can be set to 0000 in the PIV Initialisation Parameters structure.

After execution of this command, PIV compatible applications can recognize the cards as a PIV card. The PIV interface can be selected with the SELECT command, using the following AID: A00000030800001000.

Table 73 – PUT DATA: INITIALISE PIV EMULATION command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	DAh
P1	01h
P2	50h: Initialise PIV emulation
LC	14h
Data	PIV Initialisation Parameters, see Table 75 for details.
LE	Empty

Table 74 – PUT DATA: INITIALISE PIV EMULATION response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 75 – PIV Initialisation Parameters / PIV State

Description	Length	Value
State	1	80h = PIV emulation activated 00h = PIV emulation deactivated 0x7F = RFU
ACL	3	Access Control List. See table Table 76 for details.
PIV Authentication Key FID	2	FID of the key EF that is accessible as PIV Authentication Key via the PIV/CIV interface
PIV Authentication Certificate FID	2	FID of the EF that is accessible as PIV Authentication Certificate via the PIV/CIV interface
Card Authentication Key FID	2	FID of the key EF that is accessible as Card Authentication Key via the PIV/CIV interface
Card Authentication Certificate FID	2	FID of the EF that is accessible as Card Authentication Certificate via the PIV/CIV interface
Signature Key FID	2	FID of the key EF that is accessible as Signature Key via the PIV/CIV interface
Signature Certificate FID	2	FID of the EF that is accessible as Signature Certificate via the PIV/CIV interface
Management Key FID	2	FID of the key EF that is accessible as Management Key via the PIV/CIV interface
Management Certificate FID	2	FID of the EF that is accessible as Management Certificate via the PIV/CIV interface.

Table 76 – PIV emulation security attributes

Position	Meaning
X00000h	RFU
0X0000h	RFU
00X000h	Manage
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

14.5 PUT DATA: ACTIVATE OR DEACTIVATE PIV EMULATION

This command is used to switch between MyEID command interface and PIV command interface. PUT DATA: INITIALISE PIV EMULATION command must be issued before using this command. Manage access condition set in PUT DATA: INITIALISE PIV EMULATION must be authenticated for this command to succeed.

Table 77 – PUT DATA: ACTIVATE OR DEACTIVATE PIV EMULATION command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	DAh
P1	01h
P2	5Ah: Activate PIV emulation 5Dh: Deactivate PIV emulation
LC	Empty
Data	Empty
LE	Empty

Table 78 – PUT DATA: ACTIVATE OR DEACTIVATE PIV EMULATION response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

14.6 PUT DATA: SET SECURE MESSAGING PARAMETERS

This command is used to set up PIV Secure Messaging by selecting the cipher suite and mapping a private key and a file containing a Card Verifiable Certificate on the card to be used with Secure Messaging. This command requires Recreate MF access condition unless it is issued in Creation State. If the Exclusive PIN Mode flag is set, authentication state of the secure channel is isolated as follows: If the card receives a plain APDU in between secure APDUs, it invalidates all PINs. This flag can be used to ensure that when a process verifies a PIN on the secure channel, another process cannot do crypto-operations with the keys the PIN gives access to.

While by the PIV specification the Secure Messaging Certificate and Secure Messaging Certificate Signer should be Card Verifiable Certificates, MyEID supports using also X.509 certificates for these purposes. MyEID does not validate the certificates mapped to the secure messaging parameters in any way. It is a responsibility of the application using MyEID to verify them.

Table 79 – PUT DATA: SET SECURE MESSAGING PARAMETERS command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	DAh
P1	01h
P2	55h: Set Secure messaging parameters
LC	11h
Data	Secure messaging parameters, see Table 81.
LE	Empty

Table 80 – PUT DATA: SET SECURE MESSAGING PARAMETERS response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 81 – Secure Messaging Parameters

Description	Length	Value
Secure Messaging Certificate Signer tag	1	80h
Length	1	2
Signer Certificate FID	2	FID of the Secure Messaging Certificate Signer certificate
Secure Messaging Flags tag *	1	81h
Length	1	1
Secure Messaging Flags value	1	00h = no flags 01h = Exclusive PIN Mode
Cipher suite parameters tag	1	A0h
Length	1	Length of the inner structure
Algorithm identifier tag	1	80h
Length	1	1
Algorithm identifier	1	27h for CS2 (128 bit channel strength, AES 128) or 2Eh for CS7 (192 bit channel strength, AES 256)
Secure Messaging Certificate tag	1	81h

Length	1	2
Secure Messaging Certificate FID	2	FID of the Secure Messaging CVC Certificate. READ access condition is ignored in the SM key agreement.
Secure Messaging Key tag	1	82h
Length	1	2
Secure Messaging Key FID	2	FID of the Secure Messaging Key (C_{ICC} key pair, ECC P-256 or P-384 according to the selected cipher suite). USE access condition must be set to forbidden (F) for the referenced key EF. This allows using it only for SM key agreement.

* Secure Messaging Flags is an optional parameter. The value is 00h by default.

14.7 CREATE FILE

The CREATE FILE command creates EF's and DF's into the current DF. After a successful creation the created file is set as the current file. If the applet is in the Operational State, the created file is set to Created State. In Created State, the security attributes of the file are not yet enforced. This allows updating data to the file after creation, even though the access condition required to update the file is not authenticated. The file exits Created State when another file is selected, another file is created, the card is reset or the applet is reselected.

Table 82 – CREATE FILE command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	E0h
P1	00h
P2	00h
LC	19h – 31h: Length of data field
Data	File Control Parameters (FCP), see Table 84 for details.
LE	Empty

Table 83 – CREATE FILE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes: 0x6A89 – File already exists

MyEID 5: The following structure is interpreted as DER encoded, ordering does not matter and length of elements may vary. Tags 84h and 8Ah are optional, and 81h is optional and ignored if present for DFs.

MyEID 4 and earlier: The following table shows the mandatory position and length of each parameter; they must not be rearranged, and the only optional parameter is the DF name that can be used when creating DF's.

Table 84 – File Control Parameters (FCP)

Byte	Length	Value
FCP tag	1	62h
Length	1	17h: EF's 17h - 2Fh: DF's
File Size tag	1	80h: transparent EF 81h: DF and key EF
Length	1	02h
File Size	2	Transparent EF's: XXXXh: File size in bytes RSA key EF's: 0400h or 1000h*: size of modulus in bits (in 0040h increments) ECC key EF's: 0100h to 0209h: key size in bits (field size) DES key EF's: 0040h: 64 bit DES key 0080h: 128 bit 3DES key 00C0h: 192 bit 3DES key ** AES key EF's: 0080h: 128 bit key 00C0h: 192 bit key 0100h: 256 bit key DF's: 0000h: Not applicable
File Description tag	1	82h
Length	1	01h

File Descriptor	1	01h: transparent EF 11h: RSA key EF 19h: DES key EF ** 22h: ECC key EF 29h: AES key EF 38h: DF 41h: Generic secret EF
File Identifier tag	1	83h
Length	1	02h
File Identifier	2	XXXXh: FID
Security Attributes Tag	1	86h
Length	1	03h
Security Attributes	3	See SELECT FILE command for details
Proprietary Information tag	1	85h: File status byte
Length	1	02h
Proprietary Information	2	**First status byte:** X0h: for private RSA and EC key EF's X = AC to clear after security operation. The remaining bits are RFU for EF Key, and for other file types the whole byte is RFU. If zero, no AC is cleared after security operation. 0Yh: for AES and Generic Secret key EF See Table 85 for details. **Second status byte:** Transparent EF's: version < 3.5: 00h (RFU) version >= 3.5: Transparent EF's and key EF's: See table 86
Life Cycle Status tag	1	8Ah
Length	1	01h
Life Cycle Status	1	00h
DF Name tag	1	84h: Optional tag, can only be used when creating DF's.
Length	1	01h – 10h
DF Name	1-16	A byte or character string which identifies the DF.

* 0800h is supported from MyEID applet version 2.2.0 onwards. RSA keys shorter than 2048 bits (0800h) are deprecated and not supported in MyEID 5.

** Deprecated and not supported in MyEID 5.

Table 85 – FCP proprietary information flags, first status byte (bits 0-3)

Bits	Hex value	Description
0	01h	RFU
1	02h	RFU
2	04h	The key is **trusted** (AES only). Corresponds with the PKCS#11 attribute CKA_TRUSTED. *
3	08h	The key requires a **trusted** key for wrapping (Symmetric and generic secret keys). Corresponds with the PKCS#11 attribute CKA_WRAP_WITH_TRUSTED.

* The **trusted** flag can be set only in Creation State.

Table 86 – FCP proprietary information flags, second status byte (bits 0-7)

Bit	Hex value	Description
0	01h	Session object: The key EF is automatically removed during next reset. (Symmetric and generic secret key EF's)
1	02h	RFU
2	04h	Auto size flag – the file grows dynamically when writing past end of file. New in MyEID 4.
3	08h	Extractable: The key can be wrapped (Symmetric and generic secret keys)
4	10h	Admin generate key (RSA/EC key EF)
5	20h	Admin recreate MF (MF), admin delete file (other file types)
6	40h	Admin create EF's (MF/DF), Admin update (Transparent EF), Admin PUT DATA / Unwrap (Key EF)
7	80h	Admin create DF's (MF/DF), Admin read (Transparent EF), Admin use / wrap key (Key EF)

14.8 DELETE FILE

The DELETE FILE command removes the currently selected file if the required Access Condition is fulfilled. In case that the file to be deleted is a DF, the DF must be empty for the command to succeed. After a successful file removal the file space is de-allocated and can be reused for new files.

Table 87 – DELETE FILE command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	E4h
P1	00h
P2	00h
LC	Empty
Data	Empty
LE	Empty

Table 88 – DELETE FILE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

14.9 GENERATE PUBLIC KEY PAIR

The GENERATE PUBLIC KEY PAIR command generates and stores a new key into an existing key EF. The key length is given when the key EF is created; see CREATE FILE command for details. Before the command can be executed successfully, the related Access Condition must be fulfilled. The command returns the modulus or public point of the new key.

Table 89 – GENERATE PUBLIC KEY PAIR command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	46h
P1	00h
P2	00h
LC	Length of the data field
Data	RSA keys: TLV with value of public exponent, see Table 91 for details.

	ECC keys: Empty or DER-encoded OID of the elliptic curve. See Table 92. If empty, a NIST prime curve is used for key sizes 256, 384 and 521, and a regular Brainpool curve is used for key sizes 320 and 512. The curve must match the key length set in the CREATE FILE command.
LE	Empty, 00h or length of public key in bytes

Table 90 – GENERATE PUBLIC KEY PAIR response APDU

Byte	Value
Data	RSA keys: Modulus ECC keys: a TLV structure containing the public point in uncompressed format: [tag 86h, length, 04h, X, Y]
SW1 – SW2	Status Bytes

Table 91 – Generate public key pair command data parameters for RSA keys

Bytes	Length	Value
Outer sequence Tag	1	30h
Length	1	03h - 06h
Public exponent tag	1	02h or 81h
Length	1	01h-04h
Public Exponent	1 – 4	Public exponent value* **MyEID3: supports only 010001h**

* In MyEID 5, public exponent values up to 4 bytes are accepted. Values less than 010001h are not allowed.

Table 92 – Generate public key pair command data parameters for EC keys

Bytes	Length	Value
Outer sequence Tag	1	30h
Length	1	Length of the structure
Object Identifier tag	1	06h
Length	1	05h-08h
Object Identifier	5-8	See Table 93 for supported values.

Table 93 – Object identifiers of the supported named elliptic curves

Curve name	OID	Encoded OID
NIST P-256	1.2.840.10045.3.1.7	2A8648CE3D030107
NIST P-384	1.3.132.0.34	2B81040022
NIST P-521	1.3.132.0.35	2B81040023
brainpoolP256r1	1.3.36.3.3.2.8.1.1.7	2B2403030208010107
brainpoolP256t1	1.3.36.3.3.2.8.1.1.8	2B2403030208010108
brainpoolP320r1	1.3.36.3.3.2.8.1.1.9	2B2403030208010109
brainpoolP320t1	1.3.36.3.3.2.8.1.1.10	2B240303020801010A
brainpoolP384r1	1.3.36.3.3.2.8.1.1.11	2B240303020801010B
brainpoolP384t1	1.3.36.3.3.2.8.1.1.12	2B240303020801010C
brainpoolP512r1	1.3.36.3.3.2.8.1.1.13	2B240303020801010D
brainpoolP512t1	1.3.36.3.3.2.8.1.1.14	2B240303020801010E

14.10 PUT DATA: LOAD KEY

The PUT DATA: LOAD KEY command stores an externally generated key to the applet. A key EF must be selected first with the SELECT FILE command.

All the key components must be loaded to create a valid key. If a single new key component is loaded into an existing complete key, all other key components are reset.

All MyEID versions support the loading of private RSA keys in the CRT format. The MyEID3 version and newer also supports the private exponent and modulus format. Only one private key can be loaded into a key file together with a public key.

Security notice: Follow cryptography standards and best practices on generating the keys that you load to card. Ensure that the key is handled in secure way before loading to card. Use FIPS 140-2 compliant random number generator.

Table 94 – PUT DATA command APDU

Byte	Value
CLA	00h: single or final block of data 10h: chained APDU, more data to follow 0Ch and 1Ch: as above, with secure messaging
INS	DAh
P1	01h
P2	RSA key components: 80h: modulus, N *4 81h: public exponent, E 83h: prime 1, P 84h: prime 2, Q 85h: *d* mod (*p* - 1), DP1 86h: *d* mod (*q* - 1), DQ1 87h: *q*-1 mod *p*, PQ 88h: first half of 2048 bit modulus 89h: second half of 2048 bit modulus *1 MyEID3 adds support for: 82h: private exponent, D *3, *4 8Ah: first half of 2048 bit key private exponent, D 8Bh: second half of 2048 bit key private exponent, D ECC key components: 86h: Public key (a point denoted as PP on the curve, equal to x times PB where x is the private key, coded on 2z bytes, prefixed by 04h) 87h: Private key 88h: Elliptic curve OID. See Table 93 for supported values. If not set, a NIST prime curve or a regular Brainpool curve (only for sizes 320 and 512) corresponding to the key length set in CREATE FILE is used by default. Symmetric keys (AES and DES): A0h: load symmetric key. Data field contains the symmetric key data.

LC	Length of Data field
	
	P2 = 80h: **80h**
	P2 = 81h: **01h** or **03h**
	P2 = 83h, 84h, 85h, 86h, 87h: **40h or 80h** *2
	P2 = 88h, 89h: **80h**
	
	MyEID3:
	P2 = 80h, 82h: **40h to C1h**
	P2 = 81h: **01h** or **03h**
	P2 = 83h, 84h, 85h, 86h, 87h: **20h or C1h**
	P2 = 88h, 89h: **81h**
	P2 = 8Ah, 8Bh: 81h
	
	MyEID 5:
	Length of the key component. APDU chaining or extended APDUs can be used to load components longer than 255 bytes.
	
	ECC-keys:
	Length of ECC key component
	
	Symmetric keys:
	Key length. Must be equal to the file size (in bytes) set in CREATE FILE command. Note that bits is used as unit in CREATE FILE.

Data	Component
LE	Empty

*1 The second half of 2048 bit modulus must be preceded by the first half and the order of the commands must not be reversed.
*2 80h is used for the 2048 bit key components.
*3 If a private key is loaded as a modulus and a private exponent, then only the public exponent is required. The private exponent must not be loaded with a CRT formatted key, loading it will remove the CRT components.
*4 With MyEID 4.5 and newer, for 2048 bit and longer keys use APDU chaining and P2=80h for modulus and P2=82h for private exponent. For 2048 bit keys also the old MyEID3 method is available for compatibility.

Table 95 – PUT DATA: LOAD KEY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

14.11 PUT DATA: SET SESSION FLAGS

PUT DATA: SET SESSION FLAGS sets parameters to the card which are effective until the card is reset. This command is used for enabling crypto-operations in Creation State in since MyEID 5.

Table 96 – PUT DATA command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	DAh
P1	01h
P2	E5h = Set Session Flags: Enable features until next reset
LC	02h
Data	Flags, 2 bytes: 0001h = Enable Crypto Operations Description: Enables crypto-operations in Creation State until next reset.
LE	Empty

Table 97 – PUT DATA: SET SESSION FLAGS response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

14.12 GET CHALLENGE

The GET CHALLENGE command can be used to generate a random challenge to be used for authenticating a challenge/response PIN or to generate random data for any purpose by using the card’s random number generator. Since MyEID 5, the challenge is valid only when a response for the challenge is verified in the next command, with the exception that GET DATA command does not invalidate the challenge.

Table 98 – GET CHALLENGE command APDU

Byte	Value
CLA	00h or 0Ch for secure messaging
INS	84h
P1	00h: Get random data 02h: Get challenge data for authenticating a

	challenge/response PIN.

P2	P1=00h: 00h
	P1=02h: PIN number

LC	Empty
Data	Empty
LE	Number of bytes to return. Must be multiple of the cipher block size of the C/R pin, if P2=02h. With P1=00h, the maximum length is 512 bytes. With P1<>00h, the maximum challenge length is 128 bytes.

Table 99 – GET CHALLENGE response APDU

Byte	Value
Data	Requested number of random bytes
SW1 – SW2	Status Bytes

15 Secure Messaging

Since version 5, MyEID supports Secure Messaging as specified in NIST Special Publication SP 800-73-4. Secure Messaging can be used with both MyEID and PIV command interfaces. The Secure Messaging Key is created as any EC key using the MyEID Command interface, and selected for secure messaging using PUT DATA: SET SECURE MESSAGING KEYS command. ACL_USE must be set to forbidden (F) for the SM key. This command is also used to select the cipher suite and to select the EF which contains the CVC used in SM.

When transmitting chained commands, there are two different scenarios how SM is used. Besides PSO: Encipher and PSO: Decipher commands when used for symmetric encryption, the complete chained message is encrypted as one cryptogram. The packet counter is increased by one for the complete message. In symmetric encryption, however, each APDU is processed as a separate encrypted packet. The reason for this design is that symmetric encryption can be used to process an in-principle unlimited data stream, where the number of APDUs is not known in advance. In asymmetric encryption, the number of commands is related to the key size used.

16 Card genuineness verification

MyEID 5 includes a method to cryptographically verify that a card is a genuine MyEID card originating from Aventra.

Each card contains a unique on-card generated ECDSA key pair, which is signed with a MyEID version specific private key proven to be only in Aventra’s ownership. Using INTERNAL AUTHENTICATE command, the card signs random data provided by both the card and the user. The signature proves ownership of the public key, and signature of the public key proves that it is signed using the version key. The public key of the version key is signed using the root MyEID signing key. The public key of the root key is published on Aventra's web site. The root key and the version key are both stored in Aventra's HSM.

The public key of the card key pair can be obtained using GET DATA command with P1 = 01h, P2 = E6h: "Get card authentication public key."

The card contains an ASN.1 formatted DER-encoded structure called Card Authentication Signature Chain. This structure contains the digital signatures needed for verifying the chain trust-chain from the root key Aventra publishes on its web site. The structure can be obtained using GET DATA command with P1 = 01h, P2 = E7h: "Get card authentication signature chain."

NIST P-384 curve is used in Card genuineness verification.

16.1 INTERNAL AUTHENTICATE

INTERNAL AUTHENTICATE command uses the Card Authentication Key (CAK) to sign data which consists of the following concatenated values: ASCII string "GenuineMyEID" + 16 byte Client Nonce provided by user + 16 byte Card Nonce. User can verify the signature with the public part of CAK to confirm that the card is genuine.

Table 100 – INTERNAL AUTHENTICATE command APDU

Byte	Value
CLA	00h 0Ch: Secure Messaging in use
INS	88h
P1	00h
P2	00h
LC	10h
Data	Client nonce: 16 random bytes generated using a FIPS 140-2 compliant RNG.
LE	00h

Table 101 – INTERNAL AUTHENTICATE response APDU

Byte	Value
Data	Data Objects in the Dynamic Authentication Template (Tag '7C'), see Table 102.
SW1 – SW2	Status Bytes

Table 102 – Dynamic Authentication Template for INTERNAL AUTHENTICATE response APDU

Element	Length	Value
Dynamic Authentication Template tag	1	7Ch
Length	1	Length of the structure
Witness tag: card nonce	1	80h
Length	1	10h
Card Nonce	10h	Random nonce 16 bytes
Response tag	1	82h
Length	1	Length of response: 67-69h
Card response	Length of the DER endcoded signature	ECDSA signature of SHA384 ["GenuineMyEID" + Client Nonce + Card Nonce] signed with the Card Authentication Key.

16.2 Card authentication signature chain

Card authentication signature chain (CASC) is a DER encoded ASN.1 structure defined as follows:

```
MyEIDCardAuthSignature ::= SEQUENCE {
        SignedData cardSignature,
        SignedData versionSignature
}
```

SignedData is defined in https://datatracker.ietf.org/doc/html/rfc5652

cardSignature contains an ECDSA signature of SHA384 hash of concatenated elements [Version major][Version minor][Version revision][Security certifications][Card public key]. The first four elements are one byte each.

versionSignature contains an ECDSA signature of SHA384 hash of concatenated elements [Version major][Version minor][Version revision][Security certifications][Version public key]. The first four elements are one byte each.

encapContentInfo field of versionSignature contains the Version Public Key.

16.3 Verification steps

1. Get Factory Root Public Key from Aventra's web site.

2. Read Card Authentication Signature Chain (CASC) data structure from the card, using GET DATA: Card Authentication Signature Chain (P2=E7h) command.

3. Get Card Authentication Public Key (CAPub) from the card using a GET DATA command. Alternatively, CAPub can be extracted from the CASC.

4. Build version information by taking bytes 5 to 7 from response to GET DATA (Applet Information) and adding the last byte (Security certifications) from GET DATA (Card Capabilities)

5. Generate 16-byte random Client Nonce using a cryptography-capable (FIPS 140-2 level) random number generator.

6. Issue an INTERNAL AUTHENTICATE command with the Client Nonce as a parameter.

7. Verify that the response to INTERNAL AUTHENTICATE is signed by Card Auth Public Key (CAPub):

    - Extract Card nonce from the INTERNAL AUTHENTICATE response data.
    - Concatenate "GenuineMyEID" + Client Nonce + Card Nonce and compute a SHA384 hash.
    - Extract the signature from the Response field (tag 82h) of the INTERNAL AUTHENTICATE response data.
    - Verify the signature using CAPub and the hash.

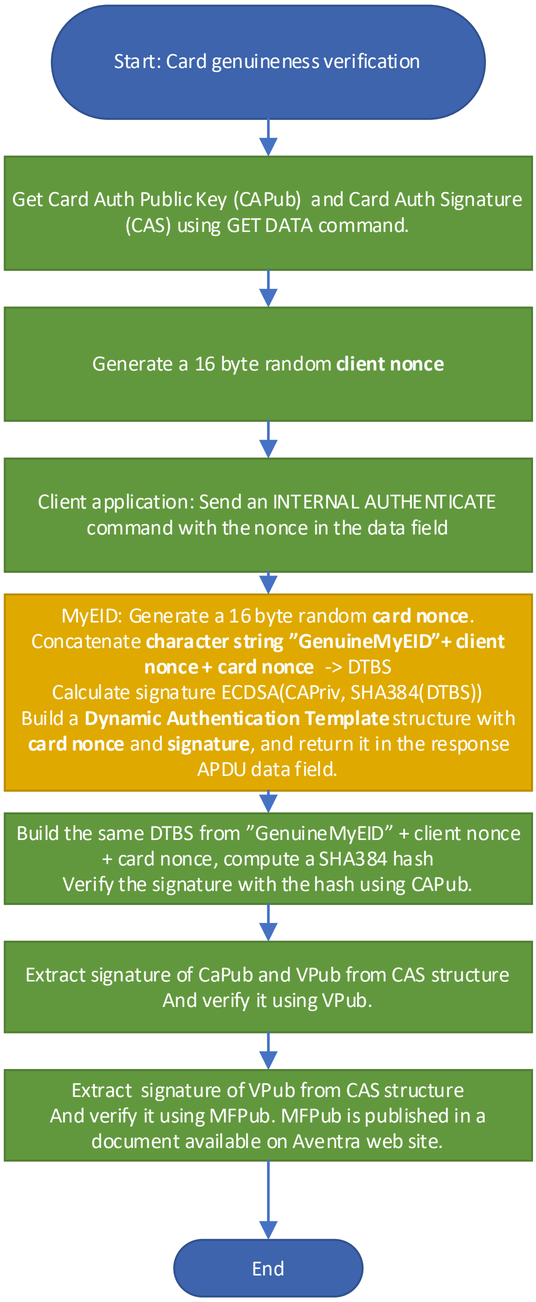
2. VERIFY VERSION SIGNATURE

    - extract SubjectKeyIdentifier from versionSignature and verify that it matches with Factory Root Public Key
    - extract SignatureValue from the SignerInfo element of versionSignature.
    - extract Version Public Key from encapContentInfo
    - calculate SHA384 of "MVmvVRCC" + Version public key where MV = Major Version, mv = minor version, VR = version revision, CC = Security Certifications (01 = Common Criteria Certified, 00 = non-certified version).
    - verify the signature value with the hash using Factory Root Public Key

3. VERIFY CARD SIGNATURE

    - extract SubjectKeyIdentifier from cardSignature and verify that it matches with Version Public Key
    - extract SignatureValue from cardSignature's SignerInfo
    - calculate SHA384 of "MVmvVRCC" + Card public key where MV = Major Version, mv = minor version, VR = version revision, CC = Security Certifications (01 = Common Criteria Certified, 00 = non-certified version).
    - verify the signature value with the hash using Version Public Key.

16.4 Card authentication flow



Start: Card genuineness verification

Get Card Auth Public Key (CAPub) and Card Auth Signature (CAS) using GET DATA command.

Generate a 16 byte random **client nonce**

Client application: Send an INTERNAL AUTHENTICATE command with the nonce in the data field

MyEID: Generate a 16 byte random **card nonce**.
Concatenate **character string "GenuineMyEID" + client nonce + card nonce** -> DTBS
Calculate signature ECDSA(CAPriv, SHA384(DTBS))
Build a **Dynamic Authentication Template** structure with **card nonce** and **signature**, and return it in the response APDU data field.

Build the same DTBS from "GenuineMyEID" + client nonce + card nonce, compute a SHA384 hash
Verify the signature with the hash using CAPub.

Extract signature of CaPub and VPub from CAS structure
And verify it using VPub.

Extract signature of VPub from CAS structure
And verify it using MFPub. MFPub is published in a document available on Aventra web site.

End

Legend

MFK = MyEID Factory Key which consists of MFPub and MFPriv. Gen. and stored on Aventra HSM

VK = Version Key, which consists of VPub and VPriv.

CAK = Card Authentication Key. It is generated on card, is unique for each card and consists of CAPub and CAPriv.

CAS = Card Authentication Signature: Contains VPub and signatures of CAPub and VPub. MyEID version is included in calculcation of the signatures.

17 Guidance, requirements and recommendations for secure usage

This section contains instructions on how to configure and use MyEID in a secure way. Each item is classified either as **GUIDANCE**, **REQUIREMENT** or **RECOMMENDATION**. All requirements must be followed when targeting EAL4+ certified level of security.

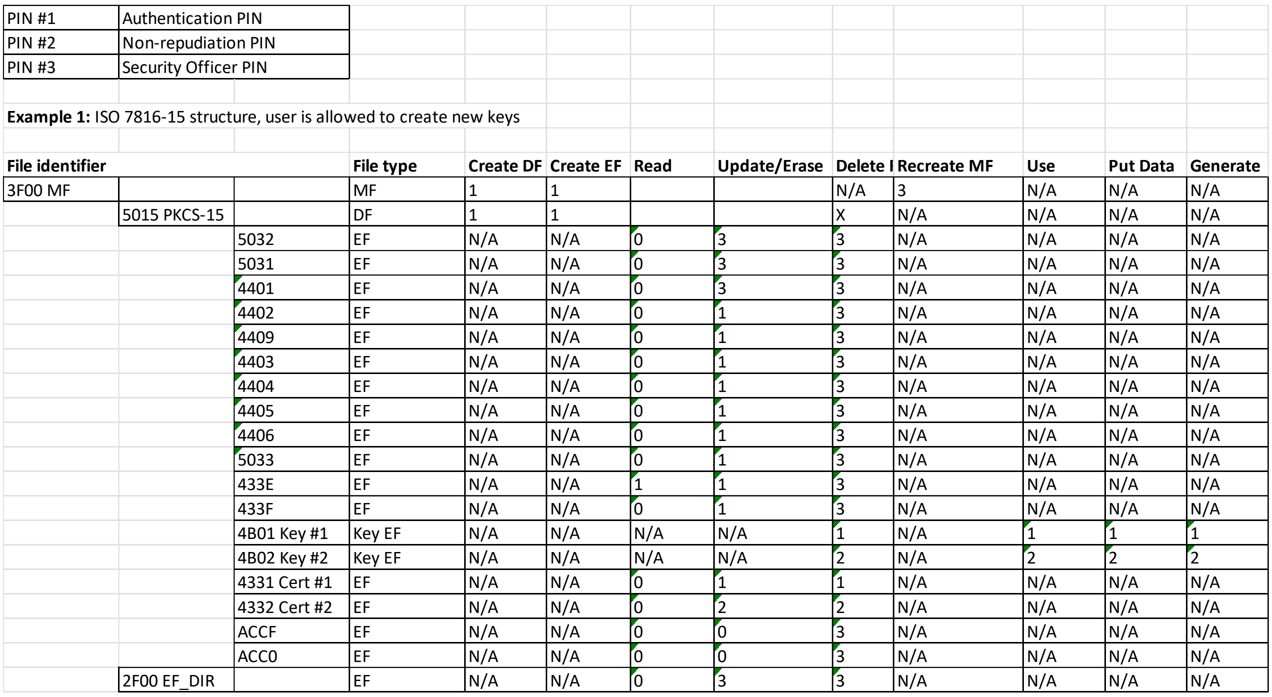
17.1 Security Attributes

- **REQUIREMENT:** Configure security attributes carefully. Be aware, that setting zero (ALLOWED ALWAYS) to a security attribute of a file allows the associated operation to be performed on the file without authentication.

- **GUIDANCE:** MyEID 5 or newer does not allow setting USE Security Attribute of a Key EF to ALLOWED ALWAYS (zero).

The following examples show typical configurations of security attributes. In both examples, PIN #3 is assigned for the Administrator role, and PINs #1 and #2 are assigned for the End User role. PINs #1 and #2 are associated with different keys. PIN #1 gives access to use key 4B01 and PIN#2 gives access to use key 4B02.

Example #1:



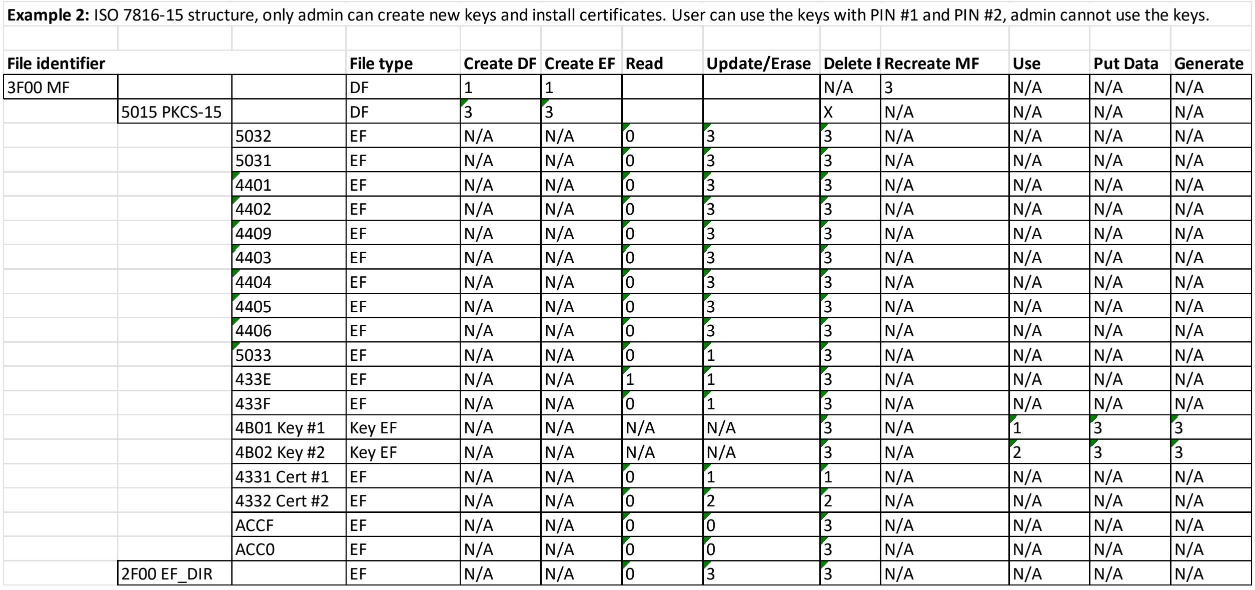
PIN #1	Authentication PIN
PIN #2	Non-repudiation PIN
PIN #3	Security Officer PIN

Example 1: ISO 7816-15 structure, user is allowed to create new keys

File identifier			File type	Create DF	Create EF	Read	Update/Erase	Delete I	Recreate MF	Use	Put Data	Generate
3F00 MF			MF	1	1			N/A	3	N/A	N/A	N/A
	5015 PKCS-15		DF	1	1			X	N/A	N/A	N/A	N/A
		5032	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		5031	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4401	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4402	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4409	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4403	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4404	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4405	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4406	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		5033	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		433E	EF	N/A	N/A	1	1	3	N/A	N/A	N/A	N/A
		433F	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4B01 Key #1	Key EF	N/A	N/A	N/A	N/A	1	N/A	1	1	1
		4B02 Key #2	Key EF	N/A	N/A	N/A	N/A	2	N/A	2	2	2
		4331 Cert #1	EF	N/A	N/A	0	1	1	N/A	N/A	N/A	N/A
		4332 Cert #2	EF	N/A	N/A	0	2	2	N/A	N/A	N/A	N/A
		ACCF	EF	N/A	N/A	0	0	3	N/A	N/A	N/A	N/A
		ACC0	EF	N/A	N/A	0	0	3	N/A	N/A	N/A	N/A
	2F00 EF_DIR		EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A

In this example, the End user is allowed to delete their own key, to generate or import a new key and load new certificates. In practice, this kind of structure allows users to renew their key and certificate and to add a new key/certificate pair for a new purpose.

Example #2:



Example 2: ISO 7816-15 structure, only admin can create new keys and install certificates. User can use the keys with PIN #1 and PIN #2, admin cannot use the keys.

File identifier			File type	Create DF	Create EF	Read	Update/Erase	Delete I	Recreate MF	Use	Put Data	Generate
3F00 MF			DF	1	1			N/A	3	N/A	N/A	N/A
	5015 PKCS-15		DF	3	3			X	N/A	N/A	N/A	N/A
		5032	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		5031	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4401	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4402	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4409	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4403	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4404	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4405	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		4406	EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A
		5033	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		433E	EF	N/A	N/A	1	1	3	N/A	N/A	N/A	N/A
		433F	EF	N/A	N/A	0	1	3	N/A	N/A	N/A	N/A
		4B01 Key #1	Key EF	N/A	N/A	N/A	N/A	3	N/A	1	3	3
		4B02 Key #2	Key EF	N/A	N/A	N/A	N/A	3	N/A	2	3	3
		4331 Cert #1	EF	N/A	N/A	0	1	1	N/A	N/A	N/A	N/A
		4332 Cert #2	EF	N/A	N/A	0	2	2	N/A	N/A	N/A	N/A
		ACCF	EF	N/A	N/A	0	0	3	N/A	N/A	N/A	N/A
		ACC0	EF	N/A	N/A	0	0	3	N/A	N/A	N/A	N/A
	2F00 EF_DIR		EF	N/A	N/A	0	3	3	N/A	N/A	N/A	N/A

In this example, the End user cannot delete a key, create a new key or load a certificate. Administrator’s access rights are required for these operations. However, the Administrator cannot use the keys (4B01 and 4B02) for computing signatures or other crypto-operations, because the Use security attribute is associated with the End user’s PINs.

17.2 Non-repudiation / User consent

- **GUIDANCE:** Be aware that by default access condition to use a specific private key remain open after using the key for a crypto-operation. If it is necessary to ensure that only one operation can be performed after one PIN verification, use the “AC to clear after crypto-operation” attribute in Key EF attributes. Typically, USE access condition is allowed to remain open for authentication keys to allow single-sign-on type operation, and “AC to clear” option is set for non-repudiation keys, to ensure that crypto-operations (such as signing a document) are never done without user’s consent.

- **RECOMMANDATION:** Set AC to clear parameter in the Proprietary Information / File Status byte field of the FCP of a Key EF used Non-repudiation purpose to index of the PIN associated with the key. For example, if USE security attribute of Key EF 4B02 is set to PIN #2, set the AC to clear bits to value 2. Configured this way, after a

PSO: Compute Digital Signature or other crypto-operation performed with Key EF 4B02, PIN #2 is invalidated. See Table 84 – File Control Parameters (FCP) for details.

17.3 Padding algorithms

MyEID offers several options for cryptographic operations. The following should be considered when selecting which option to use:

- **RECOMMENDATION:** When using RSA algorithm in the raw mode, apply a padding with a secure padding algorithm in software. RSA without padding is not considered secure. **GUIDANCE:** MyEID supports raw RSA to allow using padding algorithms not implemented on card, and for backward compatibility (older MyEID versions did not support PSS or OAEP and the newer padding methods were used in software).

- **RECOMMENDATION**: Be aware of PKCS#1 1.5 padding's vulnerability to Padding Oracle / Chosen Ciphertext Attacks, when used for encryption. In the smart card usage scenario, the fact that a PIN code is required to authenticate lowers the risk of such attacks. Another method to reduce risk is to authenticate ciphertext with a separate method, e.g. signature or a MAC, before passing it to the card for decryption.

- **RECOMMENDATION**: Be aware of PKCS#7 padding's vulnerability to padding oracle attacks at both ends of the encryption system. In the smart card usage scenario, the fact that a PIN code is required to authenticate lowers the risk of such attacks. However, when the smart card encrypts the data, at the decrypting party who shares the same key, there should not be possibility to use the decryption function in unrestricted manner. Signing and authenticating the ciphertext before passing it to the decryption function should be considered.

- **RECOMMENDATION**: In general, OAEP padding is recommended over PKCS#1 1.5 for RSA decryption. **GUIDANCE:** When decrypting OEAP padded RSA encrypted data, if the plain text data, i.e. the output when the padding is removed, is less than 8 bytes long, this may potentially make the data less secure considering side-channel attacks. Consider adding a random nonce to short plaintext before encryption.

17.4 AES encryption

- **RECOMMENDATION**: AES in ECB mode should generally be avoided and is considered safe for very limited scenarios. Particularly, AES-ECB should not be used to encrypt anything longer than a single AES-block (16 bytes), and the same data should not be encrypted with the same key more than once. Be aware of the potential weaknesses and if using ECB mode, ensure that the known vulnerabilities are considered and avoided in your usage scenario.

- **REQUIREMENT:** When using AES in CBC mode, ensure proper usage of the Initialisation Vector (IV). The IV should be random and a single IV should be used only once with the same key. Be aware of the potential attacks on AEC-CBC with PKCS#7 paddings and ensure that your usage scenario is appropriately protected from them.

- **REQUIREMENT:** If you are using AES without PKCS#7 padding, ensure that a secure padding method is used, if the encrypted data is not aligned with AES block size.

17.5 PIN management

GUIDANCE: Ensure secure creation and distribution of PINs, and train end users to handle their PINs in secure ways. It is recommended, and common practice, to use separate PINs for administrative purposes and crypto-operations. This is required to comply with Common Criteria and EIDAS requirements for SSCD. In addition, it is recommended to create separate PINs for different crypto-operations: authentication, non-repudiation and confidentiality. Administrator PIN is called Security Officer PIN (SO-PIN) in PKCS#11 and PKCS#15 terminology.

- **RECOMMENDATION:** It is recommended to use a maximal false acceptance probability of 5 × 10−6, corresponding to at most 5 trials of a 6-digit PIN. This can be enforced by setting the minimum PIN length to 6 and try counter to 5 in PUT DATA: INITIALISE PIN command’s parameters.

- **REQUIREMENT:** Create separate PINs for User and Administrator roles.

- **REQUIREMENT:** Create PINs using a cryptographically secure random number generator (unless they are not directly selected by the end users)

- **REQUIREMENT:** If users are allowed to select their PIN codes, ensure technically or by guidance that insecure PINs such as 123456, 111111 or cardholder’s birthday are not used. The preferred way is to validate PINs and reject insecure PINs in the software used to input user PINs.

- **REQUIREMENT**: Distribute PINs to end users in secure way. Secure methods include secure PIN envelopes, properly implemented electronic distribution and letting uses select their own PIN codes.

- **RECOMMENDATION:** If users are allowed to select their own PINs, distribute personal activation PINs to the users securely, and use “PIN is locked after initialisation” option to force users to change the activation PIN before the card can be used.

- **GUIDANCE:** Be aware, that allowing Administrator PIN or Global Unblock PIN to unblock users PINs could technically allow an administrator to perform security operations acting as the end user. Consider if this can be accepted.

17.6 Key management

GUIDANCE: When importing keys to the card, ensure that proper key management is used to ensure security of the keys outside the card. Key management consists of planning life cycle, access rights and protection methods for each key. Keys should be generated in a secure environment (no internet access or properly restricted internet connectivity) using certified key generation algorithm implementations and stored securely, for example in an HSM. Use secure messaging (or key wrapping for symmetric keys) for importing the keys to the card.

In most scenarios it is a good choice to generate keys for authentication and digital signature purposes on card, and to use the key import feature for keys intended for encryption. Keys intended for encryption can be archived using a CA system's key archival feature. Please note guidance about fulfilling OE.SCD_Secrecy in Annex I, when using MyEID as SSCD with imported keys.

17.7 Hashing algorithms

- **REQUIREMENT**: SHA-1 algorithm is not considered safe for hashing data to be signed. Do not use SHA-1 algorithm for hashing data to be signed in new implementations.

- **GUIDANCE:** The functionality to encapsulate a SHA-1 hash into a DigestInfo structure is maintained for backward compatibility only. Prefer SHA-2 (SHA256, SHA384 or SHA512) algorithm over SHA-1.

17.8 Key wrapping

RECOMMENDATION:
The key wrapping and unwrapping functions of MyEID do not have a built-in mechanism for verifying integrity of the wrapped key. Verifying integrity of wrapped keys should be considered. A recommended mechanism is using a Key Check Value (KCV) as follows:

- Calculate a Key Check Value (KCV) before wrapping the key, using the method described in the GlobalPlatform Card Specification.

- Sign the KCV and transmit it to the receiving end.

- After unwrapping the key in the receiving end, verify signature of the sender's KCV, calculate a KCV with the unwrapped key and compare the KCVs.

18 List of Annexes

ANNEX I – Common Criteria EAL4+ Compliance: Contains description how Common Criteria requirements are met, guidelines for compliant configuration where options are available, and terminology mapping.