



ID1-Developer-Guide

Technical Description

Document Release Version:

V1.0



1 Table of Content

Contents

Introduction.....	7
Document scope.....	7
References.....	7
Terms and operators.....	8
Chip and card application	9
Card Platform.....	9
Answer to reset (Contact interface)	9
ATS (Contactless interface).....	10
PKI application	11
PKI Data Structure.....	11
Instance – ID-A.....	12
Document Number File.....	12
EF.DIR.....	12
EF.ATR/INFO	13
EF.CardAccess.....	13
PIN – PIN1 (Authentication)	14
PIN – PUK.....	14
AES – Police Key	15
ADF AWP.....	16
PKCS#15 CIAInfo	16
PKCS#15 OD.....	19
PKCS#15 AOD.....	19
PKCS#15 PrKD	20
PKCS#15 PuKD.....	21
PKCS#15 CD.....	22
PKCS#15 DCOD	22
Authentication key 1 (Private Part).....	23
Authentication key 1 (Public Part).....	23



Certificate Authentication 1	23
Authentication key 2 (Private Part).....	24
Authentication key 2 (Public Part).....	24
Certificate Authentication 2	24
Minidriver files	24
EF.KSP	24
EF.CardId.....	25
EF.CardApps.....	25
EF.CardCf	26
DF – mscp.....	26
ADF QSCD	26
PIN – PIN2 (Qualified Signature).....	26
PKCS#15 CIAInfo	27
PKCS#15 OD.....	29
PKCS#15 AOD.....	30
PKCS#15 PrKD	32
PKCS#15 PuKD.....	32
PKCS#15 CD	33
PKCS#15 DCOD	33
Signature key 1 (Private Part).....	34
Signature key 1 (Public Part)	34
Certificate Signature 1	34
Signature key 2 (Private Part).....	34
Signature key 2 (Public Part)	34
Certificate Signature 2	35
APDU structure and contents.....	35
C-APDU structure.....	35
C-APDU contents	36
R-APDU structure.....	39
R-APDU contents	39
R-APDU indicating the operation was successful	39
Command APDUs	40
SELECT FILE	40



READ BINARY	44
GET RESPONSE.....	46
GET DATA.....	48
VERIFY	61
CHANGE REFERENCE DATA	64
RESET RETRY COUNTER	67
MANAGE SECURITY ENVIRONMENT (Set)	70
PERFORM SECURITY OPERATION - COMPUTE DIGITAL SIGNATURE	72
PERFORM SECURITY OPERATION - DECIPHER.....	74
INTERNAL AUTHENTICATE for client/server authentication.....	77
Card application objects and general operations	79
Document number.....	79
Reading document number from card application.....	79
Personal data file.....	80
ID card Personal data file example	80
Digital Identity Card (eResident) Personal data file example	81
Residence Permit Card Personal data file example	82
Diplomatic Identity Card Personal data file example	83
Reading personal info from card application Cosmo v8.1, Cosmo v8.2 and CosmoX	85
PIN1, PIN2 and PUK code operations.....	86
Using the VERIFY command to read the remaining tries counter for PIN1, PIN2 and PUK codes	86
Changing PIN1, PIN2 or PUK code.	87
Unblocking PIN1 and PIN2 code	89
Reading certificates.....	90
Computing digital signature.....	92
Computing digital signature for pre-calculated hash	92
Calculating response for TLS challenge	95
Decrypting public key encrypted data	96
Technical implementation of NFC connectivity.....	97
MRTD application	97
Basic Access Control (BAC)	97
Establish PACE tunnel.....	97



2.	Set MSE Authentication Template	98
3.	Get Global Authentication GetNonce	98
4.	Get Global Authentication MapNonce.....	98
5.	Get Global Authentication KeyAgreement.....	98
6.	Get Global Authentication MutualAuthentication	98
	Set MSE Authentication Template	98
	Get Global Authentication GetNonce	99
	Get Global Authentication MapNonce	100
	Get Global Authentication KeyAgreement.....	100
	Get Global Authentication MutualAuthentication.....	101
	Java code examples for general operations.....	102
	Establishing a channel.....	102
	Establishing a PACE tunnel.....	102
	NFC Helper functions	103
	Helper functions.....	103
	Reading document number from card application.....	104
	Reading personal info from card application Cosmo v8.1 and Cosmo v8.2	104
	Reading personal info from card application Cosmo X.....	105
	Read remaining tries counter for PIN1, PIN2, PUK.....	106
	PIN1	106
	PUK.....	106
	PIN2	107
	Changing PIN1, PIN2 or PUK code.	107
	PIN1	107
	PUK.....	107
	PIN2	107
	Unblocking PIN1 and PIN2 code	108
	PIN1	108
	PIN2	108
	Reading certificates.....	109
	Read authentication certificate.....	109
	Read digital signature certificate	109
	Computing digital signature for pre-calculated hash	110



Calculating response for TLS challenge	111
Decrypting public key encrypted data	112
Appendix 1: Detailed ATR description	113
Answer to reset (Contact interface) Cosmo v8.1 and Cosmo v8.2.....	113
Answer to reset (Contact interface) Cosmo X	114



2 Introduction

The main aim of this document is to empower software engineers and developers to create applications that make use of the interface of the security chip of the latest iteration of EstEID smart card. It would be useful for the reader of this document to be familiar with smart card and chip application related topics but it's not strictly required. By exploring the general operations section and code examples together, readers with a background in software engineering should be able to follow along.

3 Document scope

This document describes the basic use cases of the default application available on the chip such as:

- reading data from the chip
- changing and blocking/unblocking PINs
- signature computation
- authentication
- decryption

These operations are described as step by step instructions with code examples written in the Java programming language to accompany them. The commands that are used in examples are also explained in detail to provide context and to enable the reader to use variations of them to arrive at the same outcome. In its current iteration it does not however cover personalisation, configuration and maintenance. Also the reaction and behaviour of the application and the card to experiments, tests and attack attempts is out of scope of this document.

4 References

- ISO/IEC 7816-3 (2006): "Identification cards - Integrated circuit cards - Part 3: Cards with contacts — Electrical interface and transmission protocols".
- ISO/IEC 7816-4 (2013): "Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange".
- ISO/IEC 7816-8 (2014): "Identification cards - Integrated circuit cards - Part 8: Commands for security operations".
- EstEID v. 3.5 (2013): "Estonian Electronic ID-card application specification".
- NIST Special Publication 800-56A Revision 3 (2018): "Recommendation for Pair-Wise KeyEstablishment Schemes Using Discrete Logarithm Cryptography - Chapter 6: Key agreement schemes"
- FIPS PUB 186-4 (2013): "Digital Signature Standard (DSS) - Chapter 6: The Elliptic Curve Digital Signature Algorithm (ECDSA)"
- "ID-One Cosmo V8.1 – Public Security Target"



"ID-A v1.0 on Cosmo X - Public Security Target"

"ESTONIA eID-eRP cards on CosmoX. Differences with CosmoV8"

4.1 Terms and operators

- [||](#) - concatenation operation
- [0x](#) - Marks that the following number is presented in hexadecimal format.
- [ADF](#) - Application Directory File
- [AID](#) - Application identifier
- [APDU](#) - Application protocol data unit
- [ASCII](#) - The standard 7-bit code table to present digitally the English alphabet and other keyboard symbols.
- [AT](#) - Authentication template
- [bit](#) - Marks that the number is presented in binary format.
- [CSE](#) - Current security environment
- [CRT](#) - Control reference template
- [CT](#) - Confidentiality template
- [dec](#) - Marks that the number is presented in decimal format.
- [DF](#) - Directory file
- [DST](#) - Digital signature template
- [ECDH](#) - Elliptic curve Diffie-Hellman is an encryption or more precisely a key-agreement protocol used in elliptic curve cryptography
- [ECDSA](#) - Elliptic curve digital signature algorithm
- [EF](#) - Elementary file
- [FCP](#) - File control parameter
- [FID](#) - File id
- [hex](#) - Marks that the number is presented in hexadecimal format.
- [MF](#) - The DF at the root is called the master file (MF). The MF is mandatory.
- [PIN](#) - Personal identification code
- [SDO](#) - Security data object
- [SHA](#) - Secure hash algorithm
- [TLV](#) - Binary data structure of Tag, Length and Value



5 Chip and card application

5.1 Card Platform

ID1 is the 7-th Estonian eID platform that is implemented on top of ID-One™ Cosmo v8.1 (issued before 01.08.2021), which is certified as an open platform CC EAL5+.

ID1 is the 8-th Estonian eID platform that is implemented on top of ID-One™ Cosmo v8.2 (issued before 07.2025), which is certified as an open platform CC EAL5+.

ID1 is the 9-th Estonian eID platform that is implemented on top of ID-One™ Cosmo X (issued after 07.2025), which is certified as an open platform CC EAL5+ on top of the Infineon SLC37 security controller.

The platform includes an application loading mechanism, which has also CC EAL5+ level certification. ID-One™ Cosmo is installed on a separate domain on the Java global platform. Additional applications are loaded onto a separate domain. Domains are protected by a firewall between them which ensures compliance with CC EAL5+ certification. As a result, even if a non-evaluated applet is loaded the security is not compromised and the certificate remains valid. The certification of an external application is also strongly simplified by this existing certificate by simple composition on the platform. The Cosmo platform is compliant with the latest international standards:

- [JavaCard™ 3.0.4 Classic Edition](#)
- Global Platform v2.2.1 (ID Configuration v1.0)
- ISO/IEC 7816 parts 1, 2, 3, 4, 5, 6, 8 and 9
- ISO/IEC 14443 Type A

5.2 Answer to reset (Contact interface)

Every contact card responds to reset with sequence of bytes called Answer To Reset (ATR). The ATR gives information about the electrical communication protocol and the chip itself. It is mainly linked with the underlying integrated circuit (chip) and also the Java operating system on it. There is a block of historical bytes that can be used to indicate the purpose of the chip card. The ATR can be different depending on if the reset is the first since power-up (Cold ATR) or not (Warm ATR). The meaningful info of ATR can be read from historical bytes of Cold ATR.

[Cosmo v8.1 and Cosmo v8.2:](#)

Together with a category indicator byte the historical bytes form a string of 10 bytes with the value "00 12 23 3F 53 65 49 44 0F 90 00_{hex}".

The resulting specific ATR to Estonia is: 3B DB 96 00 80 B1 FE 45 1F 83 00 12 23 3F 53 65 49 44 0F 9000 F1

[Cosmo X:](#)

Together with a category indicator byte the historical bytes form a string of 10 bytes with the value "00 12 23 3F 54 65 49 44 32 0F 90 00_{hex}".



The resulting specific ATR to Estonia is: 3B DC 96 00 80 B1 FE 45 1F 83 00 12 23 3F 54 65 49
44 32 0F 9000 C3

5.3 ATS (Contactless interface)

[\(ATS Cosmo v8.1 and Cosmo v8.2\):](#)

- speed rate (kbit/s): 848 / 424 / 212 / 106
- Historical bytes: Default (same as contact) / Other (between 0 and 15 bytes)

Default parameters are:

- Baud rate = symmetrical 848 kb/s
- FWI + CID:
 - FWI = 'C', FWT = 1.237s
 - CID supported
- Historical Bytes: 0012233053654944 0F 9000
 - Category Indicator: 0x00
 - Country Code (ISO 3166-1): 0x233F (Estonia)
 - Card's issuer data: 0x654944 ("eID")
 - LCS: 0x0F (Termination State)
 - SW: 0x9000

VHBR (Very High Baud Rate) is activated.

The resulting specific ATR to Estonia is: 3B 8B 80 01 00 12 23 3F 53 65 49 44 0F 90 00 A0

[\(ATS Cosmo X\):](#)

- speed rate (kbit/s): 848 / 424 / 212 / 106
- Historical bytes: Default (same as contact) / Other (between 0 and 15 bytes)

Default parameters are:

- Baud rate = symmetrical 848 kb/s
- FWI + CID:
 - FWI = 'C', FWT = 1.237s
 - CID supported
- Historical Bytes: 001223305465494432 0F 9000
 - Category Indicator: 0x00
 - Country Code (ISO 3166-1): 0x233F (Estonia)
 - Card's issuer data: 0x65494432 ("eID2")
 - LCS: 0x0F (Termination State)
 - SW: 0x9000

VHBR (Very High Baud Rate) is activated.

The resulting specific ATR to Estonia is: 3B 8B 80 01 00 12 23 3F 54 65 49 44 32 0F 90 00 A0



5.4 PKI application

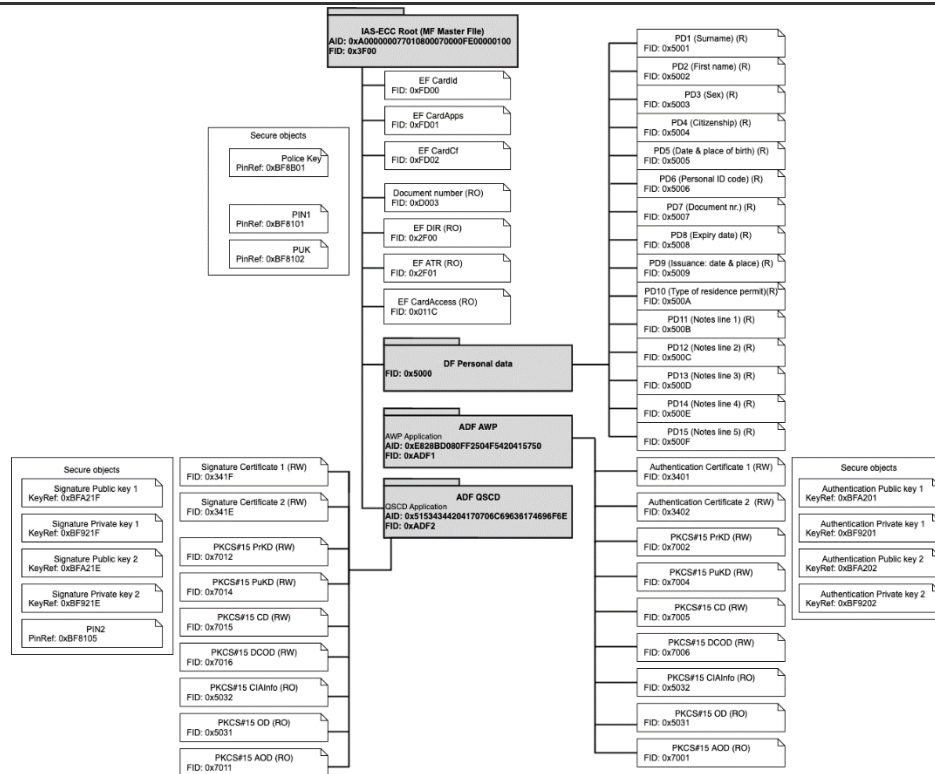
The application enabling PKI functionalities in Estonian eID Documents is IAS-ECC, a sophisticated but standardised solution conforming to CEN TS 15480-2 (European eID) with extra features. Everything detailed in the inter-industry standard “EUROPEAN CARD FOR e-SERVICES AND NATIONAL e-ID APPLICATIONS – Technical Specifications” (rev. 1.0.1). IAS-ECC, which stands for Identification Authentication Signature - European Citizen Card, is a PKI application which is QSCD certified according to the following Protection Profiles:

- CEN/EN 14169-2 (EN 419211-2) – Device with key generation
- CEN/EN 14169-3 (EN 419211-3) – Device with key import
- CEN/EN 14169-4 (EN 419211-4) – Extension for device with key generation and trusted communication with certificate generation application
- CEN/EN 14169-5 (EN 419211-5) – Extension for device with key generation and trusted communication with signature creation application
- CEN/EN 14169-6 (EN 419211-6) – Extension for device with key import and trusted communication with signature creation application

The several features available in IAS-ECC are for final user or for securing the usage on field.

5.5 PKI Data Structure

[Cosmo v8.1, Cosmo v8.2 and Cosmo X:](#)
[Filesystem diagram:](#)



6 Instance – ID-A

Root Master file

(A) ID FID : '3F00'

AID : 'A00000077030C60000000FE00000500'

Access Conditions: Nothing allowed

6.1 Document Number File

ID: 'D003'

SFI: No SFI

Length: 11 bytes

Access Conditions: Read

Content: Document number stored in a DER encoded TLV.

D003-Content ::= [APPLICATION 26] UTF8String -- Document number

6.2 EF.DIR

ID: '2F00'

SFI: '1E'

Length: 43 bytes



Access Conditions: Read (PACE is not enforced)

Content:

Application template (tag '61')

Application aid (Tag '4F') {adfPki.aid}

Application label (Tag '50') "OT AWP Application"

Application CIODDO (tag '73')

| Application aid (tag '4F') {adfPki.aid}

Application template (tag '61')

Application aid (Tag '4F') {adfQscd.aid}

Application label (Tag '50') "QSCD Application"

Application CIODDO (tag '73')

| Application aid (tag '4F') {adfQscd.aid}

6.3 EF.ATR/INFO

ID: '2F01'

SFI: Default from file ID

Length: Fixed size, adapted to file content

Access Conditions: Read

Content: As defined in [ISO 7816-4]

Tag '80' empty

Tag '43' 'B8'

Tag '46' 04B0ECC1

Tag '47' Card capabilities 940180

Tag '4F' {mfAid} # ref §6.2

Tag 'E0'

Tag '02' '0104'

Tag '02' '0104'

Tag '02' '0100'

Tag '02' '0100'

Tag '78'

Tag '06' '2B8122F87802' (1.3.162.15480.2)

Tag '82' '9000'

6.4 EF.CardAccess

ID: '011C'

SFI: Default from file ID

Length: 131 bytes

Access Conditions: Read

Content: As defined in [ICAO 9303] and BSI TR-03110.

The Distinguished Encoding Rules (DER) according to [X.690] shall be used to encode both ASN.1 data structures and (application specific) data objects.

SecurityInfos

PACEInfo



protocol OID configured for PACE
version 2
parameterId Domain parameter ID configured for PACE, for ID-A
PasswordInfo
Protocol id-MRZ
requiredPwdData
 pwdId MRZ identifier '01'
OptionalPwdData
 Bit_string empty
PasswordInfo
Protocol id-CAN
requiredPwdData
 pwdId CAN identifier '02'
OptionalPwdData
 Bit_string empty

6.5 PIN – PIN1 (Authentication)

PIN Authentication Password

The value of the PIN is padded, to accommodate the middleware management and PIN pad devices. As such, the value will always need to be padded and to be considered as such. Not doing so would bring inconsistency, among other with the PKCS15 EF.AOD. Therefore, the min and max size of the value are put to the padded length.

The unblocking is protected with either:

- a User Auth. verification using the PUC code
- a Secure Messaging using the Police Key. The secure messaging brings the assurance of a point-to-point communication between the card and the PIN management application. When the security level requires it, this mode could be used, for example when the PIN directly comes from the PIN management application, before being handled to the holder.

ID: 'BF8101'

Length: 12 bytes (the value of the PIN is padded)

Access Conditions: Change reference data, Verify, Reset retry counter, Get Data (public)

Content: Random value on 12 bytes, generated by the personalization system. 4 digits padded with 'FF' character up to 12 bytes.

Retry Counter: 3 tries.

Usage Counter: No usage counter

Life cycle: '05' (operational mode)

6.6 PIN – PUK

PUC Authentication Password

The value of the PIN is padded, to accommodate the middleware management and PIN pad devices. As such, the value will always need to be padded and to be considered as such. Not



doing so would bring inconsistency, among other with the PKCS15 EF.AOD. Therefore, the min and max size of the value are put to the padded length.

The usage counter allows a possible monitoring on the number of times authentication has occurred with this code. The value does not bring any concrete limitation, as it is well above the number of times it will be used during the lifetime of the document.

The unblocking is protected with either:

- a Secure Messaging the Issuer Key. The secure messaging brings the assurance of a point-to-point communication between the card and the PIN management application. When the security level requires it, this mode could be used, for example when the PIN directly comes from the PIN management application, before being handled to the holder.

ID: 'BF8102'

Length: 12 bytes (the value of the PIN is padded)

Access Conditions: Change reference data, Verify, Get Data (public)

Content: Random value on 12 bytes, generated by the personalization system. 8 digits padded with 'FF' character up to 12 bytes.

Retry Counter: 3 tries.

Usage Counter: '7FFF'

Life cycle: '05' (operational mode)

6.7 AES – Police Key

AES Issuer key

The usage counter allows a possible monitoring on the number of times authentication has occurred with this code. The value does not bring any concrete limitation, as it is well above the number of times it will be used during the lifetime of the document.

ID: 'BF8B01'

Length: 256 bits

Access Conditions: Mutual Auth, Get Data (public)

Content: The keyset values are directly loaded from HSM:

Enc Key = GAK.5994806F.AES256.ISSUER.KENC.00000001

Mac Key = GAK.5994806F.AES256.ISSUER.KMAC1.00000001

Retry Counter: 5 tries.

Usage Counter: '7FFF'



7 ADF AWP

FID : 'ADF1'

AID : 'E828BD080FF2504F5420415750'

- 'E8' (Standard AID)
- '28BD080F' (OID 7816-15)
- 'F2504F5420415750' (Application-specific identifier extension; "OT AWP")

7.1 PKCS#15 CIAInfo

ID: '5032'

SFI: Default from file ID

Length: Fixed size, adapted to file content

Access Conditions: Read

Content: CIAInfo content as defined in [ISO 7816-15]

CIAInfo

version : 01

manufacturerID : "IDEMIA"

label : "Estonia eID"

cardflags : readOnly | authRequired | prnGeneration # 05E0

seInfo

SecurityEnvironmentInfo

se : 2

aid : A0 00 00 00 77 01 08 00 07 00 00 FE 00 00 01 00

SecurityEnvironmentInfo

se : 3

aid : A0 00 00 00 77 01 08 00 07 00 00 FE 00 00 01 00

SecurityEnvironmentInfo

se : 2

aid : E8 28 BD 08 0F F2 50 4F 54 20 41 57 50

SecurityEnvironmentInfo

se : 3

aid : E8 28 BD 08 0F F2 50 4F 54 20 41 57 50

SecurityEnvironmentInfo

se : 4

aid : E8 28 BD 08 0F F2 50 4F 54 20 41 57 50

SecurityEnvironmentInfo

se : 5

aid : E8 28 BD 08 0F F2 50 4F 54 20 41 57 50

supportedAlgorithms

AlgorithmInfo

reference : 1

algorithm : CKM_SHA1_RSA_PKCS # 6

parameters : null



supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : sha1withRSAEncryption
algRef : 0x12
AlgorithmInfo
reference : 2
algorithm : CKM_SHA256_RSA_PKCS # 0x40
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : sha256withRSAEncryption
algRef : 0x42
AlgorithmInfo
reference : 3
algorithm : CKM_SHA384_RSA_PKCS # 0x41
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : sha384withRSAEncryption
algRef : 0x52
AlgorithmInfo
reference : 4
algorithm : CKM_SHA512_RSA_PKCS # 0x42
parameters : null
supportedOperations : compute-signature | generate-key # 41
objId : sha512withRSAEncryption
algRef : 0x62
AlgorithmInfo
reference : 5
algorithm : CKM_RSA_PKCS # 0x1
parameters : null
supportedOperations : compute-signature | verify-signature | encipher | decipher | generate-
key # 5D
objId : rsaEncryption
algRef : 0x02
AlgorithmInfo
reference : 6
algorithm : # 0x3
parameters : null
supportedOperations : compute-signature | verify-signature | encipher | decipher | generate-
key # 5D
objId : rsaEncryption
algRef : 0x1A
AlgorithmInfo
reference : 7
algorithm : CKM_ECDSA # 0x1041
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51



```
objId : ecPublicKey
algRef : 0xFF200800 # 0x04
AlgorithmInfo
reference : 8
algorithm : CKM_ECDSA_SHA1 # 0x1042
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA1
algRef : 0xFF110800 # 0x14
AlgorithmInfo
reference : 9
algorithm : CKM_ECDSA_SHA224 # 0x1043
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA224
algRef : 0xFF130800 # 0x34
AlgorithmInfo
reference : 10
algorithm : CKM_ECDSA_SHA256 # 0x1044
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA256
algRef : 0xFF140800 # 0x44
AlgorithmInfo
reference : 11
algorithm : CKM_ECDSA_SHA384 # 0x1045
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA384
algRef : 0xFF150800 # 0x54
AlgorithmInfo
reference : 12
algorithm : CKM_ECDSA_SHA512 # 0x1046
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA512
algRef : 0xFF160800 # 0x64
AlgorithmInfo
reference : 13
algorithm : CKM_ECDH1_DERIVE # 0x1050
parameters : null
supportedOperations : generate-key | derive-key # 80
objId : ecdh
algRef : 0xFF300400 # 0x0B
lastUpdate : Date encoded as a DER ASN.1 GeneralizedTime.
```



7.2 PKCS#15 OD

ID: '5031'
SFI: Default from file ID
Length: Fixed size, adapted to file content
Access Conditions: Read
Content: OD content as defined in [ISO 7816-15]
authObjects/path
 efidOrPath : FID of EF.AOD
privateKeys/path
 efidOrPath : FID of EF.PrKD
publicKeys/path
 efidOrPath : FID of EF.PuKD
certificates/path
 efidOrPath : FID of EF.CD
dataContainerObjects/path
 efidOrPath : FID of EF.DCOD

7.3 PKCS#15 AOD

ID: '7001'
SFI: Default from file ID
Length: Fixed size, adapted to file content
Access Conditions: Read
Content: AOD content as defined in [ISO 7816-15]
(30) pwd
 (30) commonObjectAttributes
 (0C) label : "User PIN"
 (03) flags : modifiable # 0640
 (04) authId : "02" #not relevant but consistant with IAS/CosmoV8
 (30) accessControlRules
 (30) AccessControlRule
 (03) accessMode : execute # 0520
 (05) securityCondition/always
 (30) AccessControlRule
 (03) accessMode : update # 0640
 (05) securityCondition/always
 (30) AccessControlRule
 (03) accessMode : attribute # 0308
 (04) securityCondition/authId : "04" # ref AOD "PUK"
 (30) CommonAuthenticationObjectAttributes
 (04) authId : "01"
 (A1) typeAttributes
 (30) PasswordAttributes



(03) pwdFlags : case-sensitive | initialized | needs-padding | exchangeRefData # 048C10

(0A) pwdType : utf8

(02) minLength : 04

(02) storedLength : 0C

(02) maxLength : 0C

(80) pwdReference : 01

(04) padChar : FF

pwd

commonObjectAttributes

label : "PUK"

flags : modifiable # 0640

accessControlRules

AccessControlRule

accessMode : execute # 0520

securityCondition/always

AccessControlRule

accessMode : update # 0640

securityCondition/always

CommonAuthenticationObjectAttributes

authId : "04"

typeAttributes

PasswordAttributes

pwdFlags : case-sensitive | unblock-disabled | initialized | needs-padding |
unblockingPassword | exchangeRefData # 049E10

pwdType : utf8

minLength : 08

storedLength : 0C

maxLength : 0C

pwdReference : 02

padChar : FF

7.4 PKCS#15 PrKD

ID: '7002'

SFI: No SFI

Length: Resizable, adapted to file content

Access Conditions: Read, Update

Content: PrKD content as defined in [ISO 7816-15]:

privateECKey

commonObjectAttributes

label : "Authentication 01"

flags : private # 0780

authId : "01"

accessControlRules

AccessControlRule



```

    accessMode : read # 0780
    securityCondition/always
  AccessControlRule
    accessMode : int-auth # 060040
    securityCondition/authId : "01" # ref AOD "User PIN"
  AccessControlRule
    accessMode : pso-dec # 01
    securityCondition/authId : "01" # ref AOD "User PIN"
  CommonKeyAttributes
    iD : "xx...xx" # UID on 20 bytes to be randomly generated at perso
    usage : sign | signRecover | derive #073080
    accessFlags : sensitive | alwaysSensitive | neverExtractable | cardGenerated # 03B8
    keyReference : 81
    algReference # ref CIAinfo
      Reference : 7
  typeAttributes
    PrivateECKKeyAttributes
      value
        efidOrPath :
          KeyInfo/paramsAndOps:
            oid : secp384r1

```

7.5 PKCS#15 PuKD

```

ID: '7004'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: PuKD content as defined in [ISO 7816-15]:
publicECKKey
  commonObjectAttributes
    label : "Authentication 01"
    flags : 0700
    authId : "01"
  CommonKeyAttributes
    iD : "xx...xx" # UID on 20 bytes to be randomly generated at perso
    usage : 070380 # verify | verifyRecover | derive
    native : TRUE # FF
    accessFlags : extractable | cardGenerated # 0348
    keyReference : 81
  typeAttributes
    PublicECKKeyAttributes
      value/ReferencedValue/path
        efidOrPath :
          KeyInfo/paramsAndOps:

```



oid : secp384r1

7.6 PKCS#15 CD

D: '7005'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: CD content as defined in [ISO 7816-15]:
x509Certificate
commonObjectAttributes
label : "Authentication 01"
flags : modifiable # 0640
accessControlRules
AccessControlRule
accessMode : read # 0780
securityCondition/always # no access control rule for update
CommonCertificateAttributes
iD : "xx...xx" # UID on 20 bytes to be randomly generated at perso
typeAttributes
X509CertificateAttributes
value/ReferencedValue/path
efidOrPath : 3401

7.7 PKCS#15 DCOD

ID: '7006'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: DCOD content as defined in [ISO 7816-15]:
opaqueDO
commonObjectAttributes
label : "cardid"
flags : modifiable
accessControlRules
AccessControlRule
accessMode : read
securityCondition/always
AccessControlRule
accessMode : update
securityCondition/authId : "01" # ref AOD "User PIN"
CommonDataContainerObjectAttributes
applicationName : "MiniDriver"
typeAttributes



OpaqueDOAttributes/indirect/path
efidOrPath : FD00
opaqueDO
commonObjectAttributes
label : "cardcf"
flags : modifiable
accessControlRules
AccessControlRule
accessMode : read
securityCondition/always
AccessControlRule
accessMode : update
securityCondition/authId : "01" # ref AOD "User PIN"
CommonDataContainerObjectAttributes
applicationName : "MiniDriver"
typeAttributes
OpaqueDOAttributes/indirect/path
efidOrPath : FD02

7.8 Authentication key 1 (Private Part)

ID: 'BF9201'
AlgoID: [secp384r1](#)
Length: 384 bits
Access Conditions: Internal Authentication, PSO Decipher, Get Data (public info)
Content: Private key components, Generated onboard
Usage Counter: '7FFF'

7.9 Authentication key 1 (Public Part)

ID: 'BFA201'
AlgoID: [secp384r1](#)
Length: 384 bits
Access Conditions: Get Data (public info)
Content: Public key components, Generated onboard

7.10 Certificate Authentication 1

ID: '3401'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read
Content: If available during personalization, the content is the X.509 signature certificate, as delivered by the Certification Authority. Otherwise, the file content is not updated.



7.11 Authentication key 2 (Private Part)

ID: 'BF9202'

AlgoID: secp384r1

Length: 384 bits

Access Conditions: Internal Authentication, PSO Decipher, Get Data (public info)

Content: Empty (reserved for future use)

Usage Counter: '7FFF'

7.12 Authentication key 2 (Public Part)

ID: 'BFA202'

AlgoID: secp384r1

Length: 384 bits

Access Conditions: Get Data (public info)

Content: Empty (reserved for future use)

7.13 Certificate Authentication 2

ID: '3402'

SFI: No SFI

Length: Resizable, adapted to file content

Access Conditions: Read

Content: Empty (reserved for future use)

7.14 Minidriver files

IAS/CosmoV8 EF.Container AWP/QCSD files (ID 'F001'/'F01F') are no more needed on ID-A/CosmoX

7.14.1 EF.KSP

ID: 'FF00'

SFI: No SFI

Length: Resizable, adapted to file content

Access Conditions: Read, Update

Content:

SEQUENCE

SEQUENCE

UTF8String : "cardid"

OCTET STRING : FID of EF.CardId

ENUMERATED : 03

BIT STRING : 00



```

INTEGER : 14
SEQUENCE
  UTF8String : "cardapps"
  OCTET STRING : FID of EF.CardApps
  ENUMERATED : 01
  BIT STRING : 00
  INTEGER : 0C
SEQUENCE
  UTF8String : "cardcf"
  OCTET STRING : FID of EF.CardCf
  ENUMERATED : 01
  BIT STRING : 00
  INTEGER : 0A
SEQUENCE
  SEQUENCE
    UTF8String : "mscp"
    OCTET STRING : FID of DF.Mscp
    ENUMERATED : 01
    SEQUENCE
      SEQUENCE
        UTF8String : "cmapfile"
        OCTET STRING : FD00
        ENUMERATED : 01
        BIT STRING : 00
        INTEGER : 035C

```

7.14.2 EF.CardId

```

ID: 'FD00'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update (User Auth), Delete (User Auth)
Content:
SEQUENCE
  OCTET STRING: Document Number left-padded with zeros to 16 bytes

```

7.14.3 EF.CardApps

```

ID: 'FD01'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update (User Auth), Delete (User Auth)
Content:
SEQUENCE
  OCTET STRING : {'mscp',0,0,0,0}

```



7.14.4 EF.CardCf

ID: 'FD02'

SFI: No SFI

Length: Resizable, adapted to file content

Access Conditions: Read, Update (User Auth), Delete (User Auth)

Content:

SEQUENCE

OCTET STRING : 6 bytes of zeros

7.14.5 DF – mscp

FID: 'FC02'

Access Conditions: Delete DF (itself), Create EF

Life Cycle: '05'

8 ADF QSCD

FID : 'ADF2'

AID : '51534344204170706C69636174696F6E' ("QSCD Application")

8.1 PIN – PIN2 (Qualified Signature)

PIN QSC Signature Password

The value of the PIN is padded, to accommodate the middleware management and PIN pad devices. As such, the value will always need to be padded and to be considered as such. Not doing so would bring inconsistency, among other with the PKCS15 EF.AOD. Therefore, the min and max size of the value are put to the padded length.

The unblocking is protected with either :

- a User Auth. verification using the PUC code (refer in §6.2.9).
- a Secure Messaging using the Issuer key (refer in §6.2.10). The secure messaging brings the assurance of a point-to-point communication between the card and the PIN management application.

ID: 'BF8105'

Length: 12 bytes (the value of the PIN is padded)

Access Conditions: Change reference data, Verify, Reset retry counter, Get Data (public)

Content: Random value on 12 bytes, generated by the personalization system. 5 digits padded with 'FF' character up to 12 bytes.

Retry Counter: 3 tries.

Usage Counter: No usage counter

Life cycle: '05' (operational mode)



8.2 PKCS#15 CIAInfo

ID: '5032'

SFI: Default from file ID

Length: Fixed size, adapted to file content

Access Conditions: Read

Content: CIAInfo content as defined in [ISO 7816-15]

CIAInfo

version : 01

manufacturerID : "IDEMIA "

label : "OT QSCD"

cardflags : readOnly | authRequired | prnGeneration # E0

seInfo

SecurityEnvironmentInfo

se : 2

aid : A0 00 00 00 77 01 08 00 07 00 00 FE 00 00 01 00

SecurityEnvironmentInfo

se : 3

aid : A0 00 00 00 77 01 08 00 07 00 00 FE 00 00 01 00

SecurityEnvironmentInfo

se : 2

aid : 51 53 43 44 20 41 70 70 6C 69 63 61 74 69 6F 6E

SecurityEnvironmentInfo

se : 3

aid : 51 53 43 44 20 41 70 70 6C 69 63 61 74 69 6F 6E

SecurityEnvironmentInfo

se : 4

aid : 51 53 43 44 20 41 70 70 6C 69 63 61 74 69 6F 6E

SecurityEnvironmentInfo

se : 5

aid : 51 53 43 44 20 41 70 70 6C 69 63 61 74 69 6F 6E

supportedAlgorithms

AlgorithmInfo

reference : 1

algorithm : CKM_SHA1_RSA_PKCS # 6

parameters : null

supportedOperations : compute-signature | verify-signature | generate-key # 51

objId : sha1withRSAEncryption

algRef : 0x12

AlgorithmInfo

reference : 2

algorithm : CKM_SHA256_RSA_PKCS # 0x40

parameters : null

supportedOperations : compute-signature | verify-signature | generate-key # 51

objId : sha256withRSAEncryption



algRef : 0x42
AlgorithmInfo
reference : 3
algorithm : CKM_SHA384_RSA_PKCS # 0x41
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : sha384withRSAEncryption
algRef : 0x52
AlgorithmInfo
reference : 4
algorithm : CKM_SHA512_RSA_PKCS # 0x42
parameters : null
supportedOperations : compute-signature | generate-key # 41
objId : sha512withRSAEncryption
algRef : 0x62
AlgorithmInfo
reference : 5
algorithm : CKM_RSA_PKCS # 0x1
parameters : null
supportedOperations : compute-signature | verify-signature | encipher | decipher | generate-
key # 5D
objId : rsaEncryption
algRef : 0x02
AlgorithmInfo
reference : 6
algorithm : # 0x3
parameters : null
supportedOperations : compute-signature | verify-signature | encipher | decipher | generate-
key # 5D
objId : rsaEncryption
algRef : 0x1A
AlgorithmInfo
reference : 7
algorithm : CKM_ECDSA # 0x1041
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecPublicKey
algRef : 0xFF200800 # 0x04
AlgorithmInfo
reference : 8
algorithm : CKM_ECDSA_SHA1 # 0x1042
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA1
algRef : 0xFF110800 # 0x14



AlgorithmInfo

reference : 9
algorithm : CKM_ECDSA_SHA224 # 0x1043
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA224
algRef : 0xFF130800 # 0x34

AlgorithmInfo

reference : 10
algorithm : CKM_ECDSA_SHA256 # 0x1044
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA256
algRef : 0xFF140800 # 0x44

AlgorithmInfo

reference : 11
algorithm : CKM_ECDSA_SHA384 # 0x1045
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA384
algRef : 0xFF150800 # 0x54

AlgorithmInfo

reference : 12
algorithm : CKM_ECDSA_SHA512 # 0x1046
parameters : null
supportedOperations : compute-signature | verify-signature | generate-key # 51
objId : ecdsaWithSHA512
algRef : 0xFF160800 # 0x64

AlgorithmInfo

reference : 13
algorithm : CKM_ECDH1_DERIVE # 0x1050
parameters : null
supportedOperations : generate-key | derive-key # 80
objId : ecdh
algRef : 0xFF300400 # 0x0B

lastUpdate

Date encoded as a DER ASN.1 GeneralizedTime.

8.3 PKCS#15 OD

ID: '5031'

SFI: Default from file ID

Length: Fixed size, adapted to file content

Access Conditions: Read

Content: OD content as defined in [ISO 7816-15]



authObjects/path
 efidOrPath : FID of EF.AOD
 privateKeys/path
 efidOrPath : FID of EF.PrKD
 publicKeys/path
 efidOrPath : FID of EF.PuKD
 certificates/path
 efidOrPath : FID of EF.CD
 dataContainerObjects/path
 efidOrPath : FID of EF.DCOD

8.4 PKCS#15 AOD

ID: '7011'
 SFI: Default from file ID
 Length: Fixed size, adapted to file content
 Access Conditions: Read
 Content: AOD content as defined in [ISO 7816-15]
 (30) pwd
 (30) commonObjectAttributes
 (0C) label : "User PIN"
 (03) flags : modifiable # 0640
 (04) authId : "02" #not relevant but consistant with IAS/CosmoV8
 (30) accessControlRules
 (30) AccessControlRule
 (03) accessMode : execute # 0520
 (05) securityCondition/always
 (30) AccessControlRule
 (03) accessMode : update # 0640
 (05) securityCondition/always
 (30) AccessControlRule
 (03) accessMode : attribute # 0308
 (04) securityCondition/authId : "04" # ref AOD "PUK"
 (30) CommonAuthenticationObjectAttributes
 (04) authId : "01"
 (A1) typeAttributes
 (30) PasswordAttributes
 (03) pwdFlags : case-sensitive | initialized | needs-padding | exchangeRefData # 048C10
 (0A) pwdType : utf8
 (02) minLength : 04
 (02) storedLength : 0C
 (02) maxLength : 0C
 (80) pwdReference : 01
 (04) padChar : FF
 pwd



```
commonObjectAttributes
label : "PUK"
flags : modifiable # 0640
accessControlRules
  AccessControlRule
    accessMode : execute # 0520
    securityCondition/always
  AccessControlRule
    accessMode : update # 0640
    securityCondition/always
CommonAuthenticationObjectAttributes
  authId : "04"
typeAttributes
  PasswordAttributes
    pwdFlags : case-sensitive | unblock-disabled | initialized | needs-padding |
unblockingPassword | exchangeRefData # 049E10
    pwdType : utf8
    minLength : 08
    storedLength : 0C
    maxLength : 0C
    pwdReference : 02
    padChar : FF
pwd
commonObjectAttributes
label : "Signature PIN"
flags : modifiable # 0640
authId : "06" #not relevant but consistant with IAS/CosmoV8
accessControlRules
  AccessControlRule
    accessMode : execute # 0520
    securityCondition/always
  AccessControlRule
    accessMode : update # 0640
    securityCondition/always
  AccessControlRule
    accessMode : attribute # 0308
    securityCondition/authId : "04" # ref AOD "PUK"
CommonAuthenticationObjectAttributes
  authId : "05"
typeAttributes
  PasswordAttributes
    pwdFlags : case-sensitive | local | initialized | needs-padding | integrity-protected |
exchangeRefData # 04CC50
    pwdType : utf8
    minLength : 05
```



storedLength : 0C
maxLength : 0C
pwdReference : 85 # ref §6.2.13.2
padChar : FF

8.5 PKCS#15 PrKD

ID: '7012'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: PrKD content as defined in [ISO 7816-15]:
privateECKey
label : "Signature 1F"
flags : private # 0780
authId : "01" #not relevant but consistant with IAS/CosmoV8
userConsent : 1
accessControlRules
AccessControlRule
accessMode : pso-cds # 0204
securityCondition/authId : "05" # ref AOD "Signature PIN"
CommonKeyAttributes
id : "xx...xx" # UID on 16 bytes to be randomly generated at perso
usage : sign | signRecover | nonRepudiation # 063040
accessFlags : sensitive | alwaysSensitive | neverExtractable # 03B1
keyReference : 9F
algReference # ref CIAinfo
Reference : B
typeAttributes
PrivateECKKeyAttributes
value
efidOrPath :
KeyInfo/paramsAndOps:
oid : secp384r1

8.6 PKCS#15 PuKD

ID: '7014'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: PuKD content as defined in [ISO 7816-15]:
publicECKey
commonObjectAttributes
label : "Signature 1F"



flags : 0600
authId : "01" #not relevant but consistant with IAS/CosmoV8
CommonKeyAttributes
iD : "xx...xx" # UID on 20 bytes to be randomly generated at perso
usage : 0102 # verify
native : TRUE # FF
accessFlags : extractable | cardGenerated # 0348
keyReference : 9F
typeAttributes
PublicEckKeyAttributes
value/ReferencedValue/path
efidOrPath :
KeyInfo/paramsAndOps:
oid : secp384r1

8.7 PKCS#15 CD

ID: '7015'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: CD content as defined in [ISO 7816-15]:
x509Certificate
commonObjectAttributes
label : "Signature 1F"
flags : modifiable # 0640
accessControlRules
AccessControlRule
accessMode : read # 0780
securityCondition/always
access control for update deleted
CommonCertificateAttributes
iD : "xx...xx" # UID on 20 bytes to be randomly generated at perso
typeAttributes
X509CertificateAttributes
value/ReferencedValue/path
efidOrPath : 341F

8.8 PKCS#15 DCOD

ID: '7016'
SFI: No SFI
Length: Resizable, adapted to file content
Access Conditions: Read, Update
Content: empty



8.9 Signature key 1 (Private Part)

ID: 'BF921F'

AlgoID: Restrict to SHA-384

Length: 384 bits (secp384r1)

Access Conditions: Compute Digital Signature, Get Data (public info)

Content: Private key components, Generated onboard

Non repudiation: true

Usage Counter: '7FFF'

8.10 Signature key 1 (Public Part)

ID: 'BFA21F'

AlgoID: No restrictions

Length: 384 bits (secp384r1)

Content: Public key components, Generated onboard

8.11 Certificate Signature 1

ID: '341F'

SFI: No SFI

Length: Resizable, adapted to file content

Access Conditions: Read

Content: If available during personalization, the content is the X.509 signature certificate, as delivered by the Certification Authority. Otherwise, the file content is not updated.

8.12 Signature key 2 (Private Part)

ID: 'BF921E'

AlgoID: Restrict to SHA-384

Length: 384 bits (secp384r1)

Access Conditions: Compute Digital Signature, Get Data (public info)

Content: Empty (reserved for future use)

Non repudiation: true

Usage Counter: '7FFF'

8.13 Signature key 2 (Public Part)

ID: 'BFA21E'

AlgoID: No restrictions

Length: 384 bits (secp384r1)

Content: Empty (reserved for future use)



8.14 Certificate Signature 2

ID: '341E'
 SFI: No SFI
 Length: Resizable, adapted to file content
 Access Conditions: Read
 Content: Empty (reserved for future use)

APDU protocol

Communication between a chip card and a host application is performed over application-level APDU protocol. Current chapter and subchapters gives the basics of APDU protocol usage. APDU protocol itself is specified in ISO 7816-4 standard.

APDU messages compromise two structures: one used by the host application to send commands to the card whereas one is used by the chip to send command response back to the host application. Data transmission between two ends is performed as master-slave communication where a host application is the master and a chip is the slave.

Command sent by a host application is called Command APDU (C-APDU) or simply APDU. Command sent by the chip as a response to C-APDU is called Response APDU (R-APDU). APDU messages can be transmitted with two different transmission-level Transmission Protocol Data Units (TPDU) which are T0 and T1. The T0 and T1 protocols are used to support APDU protocols transmission between chip reader and chip itself. APDU protocol is used between the chip application and the chip reader.

T1 is block oriented protocol which enables blocks or grouped collections of data to be transferred. These data groups are transferred as a whole between chip and reader. The theoretical maximum length of T1 grouped collections for C-APDU is 65535_{dec} and for R-APDU is 65536_{dec} bytes. The practical maximum length depends on the used chip platform on which the application runs.

T0 is byte oriented protocol which means that the minimum data that can be transferred has a length of one byte. The maximum length of data structure that can be transferred with this protocol for C-APDU is 255_{dec} and for R-APDU is 256_{dec} bytes.

8.15 APDU structure and contents

APDU structure defined in ISO 7816-4 standard is very similar to TDPU structure used in T0. When APDU is transmitted with T0, the elements of APDU exactly overlay the elements of TPDU.

8.15.1 C-APDU structure

Header				Body		
CLA	INS	P1	P2	Lc	Data	Le
				Optional for T1		



	Optional for T0
--	-----------------

8.15.2 C-APDU contents

Code	Name	Length	Description
CLA	Class	1	Instruction class - indicates the type of command, e.g. interindustry or proprietary
INS	Instruction	1	Instruction code - indicates the specific command, e.g. "write data"
P1	Parameter 1	1	Instruction parameter 1
P2	Parameter 2	1	Instruction parameter 2



Lc	Length	0, 1 or 3	<p>Encodes the number (Nc) of bytes of command data to follow</p> <ul style="list-style-type: none"> • 0 bytes denotes Nc=0 • 1 byte with a value from 1 to 255 denotes Nc with the same value • Extended APDU - 3 bytes, the first of which must be 0, denotes Nc in the range 1 to 65 535 (all three bytes may not be zero) (is case of Cosmo X is not supported)
Data	Command data	Variable, equal to Lc	String of bytes sent in the data field of the command



Le	Length	0, 1, 2 or 3	<p>Encodes the maximum number (Ne) of response bytes expected</p> <ul style="list-style-type: none"> • 0 bytes denotes Ne=0 • 1 byte in the range 1 to 255 denotes that value of Ne, or 0 denotes Ne=256 • 2 bytes (if extended Lc was present in the command) in the range 1 to 65 535 denotes Ne of that value, or two zero bytes denotes 65 536 • Extended APDU - 3 bytes (if Lc was not present in the command), the first of which must be 0, denote Ne in the same way
----	--------	--------------	--



			as two-byte Le
--	--	--	----------------

Keep in mind that by using T1 protocol either Le or data field has to be present always. When there is no specific value for Le or data field while using T1, then Le field must be set to value 00_{hex}.

8.15.3 R-APDU structure

Body	Trailer	
Data	SW1	SW2

8.15.4 R-APDU contents

Code	Name	Length	Description
Data	Data	Variable (at most Le if was present in C-APDU)	Sequence of bytes received in the data field of the response (Optional field)
SW1	Status byte 1	1	Command processing status
SW2	Status byte 2	1	Command processing qualifier

8.15.5 R-APDU indicating the operation was successful

The application responds with the following R-APDU to indicate the successful processing of C-APDU:

Code	Name	Value
Data	Data	Optional - depending on the command this may or may not be present
SW1	Status byte 1	90 _{hex}
SW2	Status byte 2	00 _{hex}



8.16 Command APDUs

The following is a list and description of all the C-APDUs that are used within the context of this document in Card application objects and general operations. Card application APDU commands are derived from ISO 7816-4 but might not implement all given specification requirements.

8.16.1 SELECT FILE

This command is used to select a file (EF, DF), the MF or an application (ADF). After a successful selection the file selected becomes the current file. After reset the current DF is the MF and no EF is selected.

The following rules shall apply:

- After a successful selection of a DF there is no selected EF
- After a successful selection of an ADF, the associated application is selected and becomes the current application, there is no selected EF
- If the selection is aborted due to an error, the current files selection is unchanged
- When selecting an EF, the current DF becomes the parent DF of the selected EF
- Following an ADF selection, the current DF is the ADF, and there is no current EF
- Upon IFD request, the command may return file FCP

SELECT	
Command Parameter	Meaning
CLA	00 _{hex}
INS	A4 _{hex}
P1	Selection control - See table 1 below
P2	Selection control - See table 2 below
Lc field	Absent or length of data field <ul style="list-style-type: none"> • 02 - to pass a FID • 'xx' - to pass DF name or relative path
Data field	FID, DF name or relative path
Le field	empty or 00 _{hex} or maximum length of data expected in response

Table 1: Selection, file and life cycle commands P1 possible bit values

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	Command data field



0	0	0	0	0	0	0	0	x	x	Selection by file identifier			
								0	0	Select MF	MF identifier or empty		
								0	1	Select child DF	DF identifier		
								1	0	Select EF under current DF	EF identifier		
								1	1	Select parent DF of the current DF. Upper limit = ADF or MF	None		
								0	1	x	x	Selection by name	
										0	0	Select by DF name (ADF or MF)	AID
								1	0	x	x	Selection by path	
										0	1	Select from the current DF	Path without the current DF identifier

Table 2: Selection, file and life cycle commands P2 possible bit values

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
----	----	----	----	----	----	----	----	---------



0	0	0	0	-	-	x	x	File occurrence
				-	-	0	0	First only occurrence
				x	x	-	-	File control parameters (FCP)
				0	0	-	-	Not supported
				0	1	-	-	Return FCP template, mandatory use of FCP tag and length
				1	1	-	-	No data in response field
SELECT response								
Response parameter				Meaning				
Data field				Absent or FCP (See table 3 below)				



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none"> • '6283' - Warning Selected file deactivated • '6285' - The Selected file is in Terminate state • '6985' - Conditions of use not satisfied • '6A82' - File or application not found • '6A86' - Incorrect parameters P1-P2 • '6A87' - Lc inconsistent with parameters P1-P2
---------	--

Table 3: FCP returned upon file selection

Template	Length	Value field			Presence		
		Tag	Length	Content	ADF	DF	EF
62	L62	80 _{hex}	02 _{hex}	File length	-	-	<u>M</u>
		82 _{hex}	01 _{hex}	File descriptor byte <ul style="list-style-type: none"> • EF - 01_{hex} • DF - 38_{hex} • ADF/MF - 38_{hex} 	<u>M</u>	<u>M</u>	<u>M</u>
		83 _{hex}	02 _{hex}	File identifier	<u>M</u>	<u>M</u>	<u>M</u>



		84 _{hex}	05 _{hex} to 10 _{hex}	DF name (AID)	<u>M</u>	=	=
		88 _{hex}	00 _{hex} or 01 _{hex}	Short file identifier	=	=	<u>O</u>
		8A _{hex}	01 _{hex}	Life cycle status byte <ul style="list-style-type: none"> activated - 05_{hex} deactivated - 04_{hex} terminated - 0C_{hex} 	<u>M</u>	<u>M</u>	<u>M</u>
		A1 _{hex}	Var.	Security attributes in proprietary format	<u>M</u>	<u>M</u>	<u>M</u>
		A5 _{hex}	Var.	Issuer discretionary data in BER-TLV format	<u>O</u>	<u>O</u>	<u>O</u>
		85 _{hex}	Var.	Issuer discretionary data in NON BER-TLV format	<u>O</u>	<u>O</u>	<u>O</u>

Legend:

- - means that the parameters is not returned
- M means the parameter is present
- O means the parameter is present if it was set at creation

8.16.2 READ BINARY

The READ BINARY command is used to read binary data from Transparent EF.

READ BINARY	
Command Parameter	Meaning
CLA	00 _{hex}
INS	B0 _{hex}
P1-P2	Offset to start reading from file - See table 1 below
Le field	number of bytes to read

Table 1: P1 & P2 management - possible bit values

P1								P2							
b8	b7	b6	b5	b4	b3	b2	b1	b8	b7	b6	b5	b4	b3	b2	b1
0	Offset in the currently selected file over 15 bits 00 _{hex} <= Offset <= 7FFF _{hex}														
1	0	0	Short File Identifier 1 <= SFI <= 30					Offset in the file over 8 bits							



READ BINARY response	
Response parameter	Meaning
Data field	Binary data
SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none"> • 6282 - End of file reached before reading 'Ne' bytes • 6981 - Command incompatible with file structure • 6982 - Security status not satisfied • <u>6985</u> - Conditions of use not satisfied • 6A80 - Wrong data • 6A82 - File not found (no current EF) • 6B00 - Wrong parameters P1-P2 : Offset + length is beyond the end of file • 6700 - Wrong length (wrong Le field)



8.16.3 GET RESPONSE

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

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

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

GET RESPONSE	
Command Parameter	Meaning
CLA	00h - no secure messaging 0Ch - secure messaging
INS	C0h
P1	00h
P2	00h
Lc field	Empty
Data field	Empty
Le field	Maximum length of data expected in response
GET RESPONSE response	
Response parameter	Meaning
Data field	Absent



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none">• 6A84 - The chaining mechanism is not available for the command.• 6A86 - Incorrect parameters P1-P2.• 9000 - Successful execution of the command (Le = Licc)• 61XX - Normal processing: more data bytes are available ('XX' indicates a number of extra data bytes still available by subsequent GET RESPONSE).• 6281 - Part of returned data may be corrupted.• 6700 - Wrong length (the Lc field is incorrect).• 6CXX - Wrong length (wrong Le field, 'XX')
---------	---



	indicates the exact length).
--	------------------------------

8.16.4 GET DATA

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

- Public key elements
- PIN information

The data in the response is in TLV format.

GET DATA	
Command Parameter	Meaning
CLA	00h - no secure messaging 0Ch - secure messaging
INS	CBh
P1	(See Table 1b)
P2	(See Table 2b)
Lc field	Length of subsequent data field
Data field	See Table 1b and Table 2
Le field	00h

Get application data

- When the value of P1-P2 lies in the range from '0100' to '01FF', the value of P1-P2 shall be an identifier reserved for card internal tests and for proprietary services meaningful within a given application context.

Get data objects

- When the value of P1-P2 lies in the range from '0040' to '00FF', the value of P2 shall be a BER-TLV tag on a single byte. The value '00FF' is reserved for obtaining all the common BER-TLV data objects readable in the context.
- When the value of P1-P2 lies in the range from '0200' to '02FF', the value of P2 shall be a SIMPLE-TLV tag. The value '0200' is RFU. The value '02FF' is reserved for obtaining all the common SIMPLE-TLV data objects readable in the context.
- When the value of P1-P2 lies in the range from '4000' to 'FFFF', the value of P1-P2 shall be a BER-TLV tag on two bytes. The values '4000' and 'FFFF' are RFU.

When a primitive data object is requested, the data field of the response message shall contain the value of the corresponding primitive data object.



When a constructed data object is requested, the data field of the response message shall contain the value of the constructed data object, i.e. data objects including their tag, length and value.

Value	Meaning
'0000'-'003F'	RFU
'0040'-'00FF'	BER-TLV tag (1 byte) in P2
'0100'-'01FF'	Application data (proprietary coding)
'0200'-'02FF'	SIMPLE-TLV tag in P2
'0300'-'03FF'	RFU
'4000'-'FFFF'	BER-TLV tag (2 bytes) in P1-P2

Template	Length	Value field				
'4D'	'08'	Template	Length	Value field		
		'70'	'06'	Tag	Length	Content
				'BFyyxx'	'02'	'A080'

[yy](#) indicates the type of SDO. It shall take the value indicated in the table below:

Type of SDO	Value of yy
PIN object	'81'
Biometric template object	'82'
DES symmetric key object	'8A'
AES symmetric key object	'8B'
RSA private key object	'90'
RSA public key object	'A0'
EC private key object	'91'
EC public key object	'A2'



Diffie Hellmann domain parameters object	'A1'
Security environment object	'FB'

[xx](#) identifies of the SDO from which the object data shall be returned. It shall take a value between 1 (included) and 31 (included).

Upon successful execution, the command [GET DATA](#), returns the Data Object, nested in the following data structure:

GET DATA response	
Response parameter	Meaning
Data field	Value of retrieved Data Object. View Table 3.



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none"> • 6281 - Part of returned data may be corrupted • 6700 - Wrong length (wrong Le field) • 6982 - Security status not satisfied • 6985 - Conditions of use not satisfied • 6A81 - Function not supported • 6A88 - Referenced data (data objects) not found • 6CXX - Wrong length (wrong Le field: 'XX' indicates the exact length)
---------	---

The Data Object is a BER-TLV structure. The structure and content of object depends on the type of SDO.

Table 3: GET DATA response Data field structure				
Template	Length	Value field		
'70'	L ₇₀	Tag	Length	Content
		'BFyyxx'	L _{object}	Object



8.16.4.1 PIN object

Table 4: PIN object data structure				
Template	Length	Value field		
'A0'	L _{A0}	<u>Tag</u>	<u>Length</u>	<u>Content</u>
		'84'	'01' to '10'	Data Object name
		'9A'	'01'	Maximum number of tries
		'9B'	'01'	Remaining number of tries
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.2 Biometric template object



Table 4: Biometric template object data structure

Template	Length	Value field		
		Tag	Length	Content
'A0'	L _{A0}	'84'	'01' to '10'	Data Object name
		'9A'	'01'	Maximum number of tries
		'9B'	'01'	Remaining number of tries
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.3 DES symmetric key object



Table 5: DES symmetric key object data structure

Template	Length	Value field		
		Tag	Length	Content
'A0'	L _{A0}	'84'	'01' to '10'	Data Object name
		'9A'	'01'	Maximum number of tries
		'9B'	'01'	Remaining number of tries
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.4 AES symmetric key object



Table 6: AES symmetric key object data structure

Template	Length	Value field		
		Tag	Length	Content
'A0'	L _{A0}	'84'	'01' to '10'	Data Object name
		'9A'	'01'	Maximum number of tries
		'9B'	'01'	Remaining number of tries
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.5 RSA private key object



Table 7: RSA private key object data structure				
Template	Length	Value field		
'A0'	L _{AO}	<u>Tag</u>	<u>Length</u>	<u>Content</u>
		'84'	'01' to '10'	Data Object name
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'9E'	'01'	Non repudiation counter
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.6 RSA public key object

Table 8: RSA public key object data structure		
Template	Length	Value field



'A0'	L _{A0}	Tag	Length	Content
		'84'	'01' to '10'	Data Object name
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.7 EC private key object

Table 9: EC private key object data structure				
Template	Length	Value field		
'A0'	L _{A0}	Tag	Length	Content
		'84'	'01' to '10'	Data Object name
		'9C'	'02'	Maximum number of use



		'9D'	'02'	Remaining number of use
		'9E'	'01'	Non repudiation counter
		'80'	'02'	Object length '00A0' - 160 bits, '00C0' - 192 bits, '00E0' - 224 bits, '0100' - 256 bits, '0140' - 320 bits, '0180' - 384 bits, '0200' - 512 bits, '0209' - 521 bits
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.8 EC public key object

Table 10: EC public key object data structure		
Template	Length	Value field



		<u>Tag</u>	<u>Length</u>	<u>Content</u>
'A0'	L _{AO}	'84'	'01' to '10'	Data Object name
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length '00A0' - 160 bits, '00C0' - 192 bits, '00E0' - 224 bits, '0100' - 256 bits, '0140' - 320 bits, '0180' - 384 bits, '0200' - 512 bits, '0209' - 521 bits
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.9 Diffie Hellmann domain parameters object



Table 11: Diffie Hellmann domain parameters object data structure

Template	Length	Value field		
		Tag	Length	Content
'A0'	L _{AO}	'84'	'01' to '10'	Data Object name
		'9C'	'02'	Maximum number of use
		'9D'	'02'	Remaining number of use
		'80'	'02'	Object length '0080' - 1024 bits, '00C0' - 1536 bits, '0100' - 2048 bits
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.4.10 Security environment object



Table 11: Diffie Hellmann domain parameters object data structure				
Template	Length	Value field		
'A0'	L _{A0}	Tag	Length	Content
		'84'	'01' to '10'	Data Object name
		'80'	'02'	Object length
		'A1'	Var.	Security attributes in proprietary format
		'A5'	Var.	Issuer discretionary data in BER-TLV format
		'85'	Var.	Issuer discretionary data in NON BER-TLV format

8.16.5 VERIFY

The VERIFY command is used to authenticate cardholder through PIN1, PIN2 or PUK code or to devalidate PIN1, PIN2, PUK. Codes must be provided in communication as ASCII character numbers.

VERIFY	
Command Parameter	Meaning
CLA	00 _{hex}
INS	20 _{hex}
P1	00 _{hex} for verification or FF _{hex} for devalidation
P2	See table 1 below



Lc field	0C _{hex} or Absent or 00 _{hex}
Data field	Candidate PIN/PUK (P1 = '00 _{hex} ' and the command is used to submit a PIN/PUK) Or Empty (P1 = 'FF _{hex} ' or P1 = '00 _{hex} ' and the command is used to audit the validation status) If present The candidate PIN/PUK must be padded with FF _{hex} on the right side so that the total length of the data field is 0C _{hex} e.g. if Code is 3132333435 _{hex} then the data field value will be 3132333435FFFFFFFFFFFFFFFF _{hex}
Le field	Absent

Table 1: P2 encoding - possible bit values

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Local reference data Application
0	-	-	-	-	-	-	-	Global reference data (Card)
-	-	x	x	x	x	x	x	User authentication DO reference (Code reference)
0	0	0	0	0	0	0	0	Forbidden

VERIFY response

Response parameter	Meaning
Data field	Absent



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none"> • 6A86 - P1 ≠ '00' and P1 ≠ 'FF' • 6700 - PIN length is out of valid boundaries [2*Lmin - 2*Lmax]" • 6A88 - Referenced PIN not found • 6982 - Security Status not satisfied • 6983 - Referenced PIN not successfully verified AND no subsequent tries are allowed (remaining tries counter reached 0) • 6984 - Referenced PIN usage counter reached 0 • 6300 - No retry limit : User authentication failed (if Pin verification) or PIN is not validated (if Lc=0)
---------	---



	<ul style="list-style-type: none"> • 63Cx - x = remaining tries : User authentication failed (if Pin verification) or PIN is not validated (if Lc=0). • 9000 - user authentication successful.
--	--

8.16.6 CHANGE REFERENCE DATA

This command allows changing PIN1, PIN2 and PUK codes. To change PIN1, PIN2 or PUK codes you need to know the currently active code. PIN1, PIN2 and PUK codes must be provided in communication as ASCII character numbers.

CHANGE REFERENCE DATA	
Command Parameter	Meaning
CLA	00 _{hex}
INS	24 _{hex}
P1	00 _{hex}
P2	See table 1 below
Lc field	18 _{hex}
Data field	Current Code New Code The current code and new code must be padded with FF _{hex} on the right side so that the total length of the data field is 18 _{hex} e.g. if Code is 3132333435 _{hex} and the new code is 35343231 _{hex} then the data field value will be 3132333435FFFFFFFFFFFFFFFF35343231FFFFFFFFFFFFFFFF _{hex}
Le field	Absent

Table 1: P2 encoding - possible bit values

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Local reference data Application



0	-	-	-	-	-	-	-	Global reference data (Card)
-	-	x	x	x	x	x	x	User authentication DO reference (Code reference)
0	0	0	0	0	0	0	0	Forbidden
CHANGE REFERENCE DATA response								
Response parameter		Meaning						
Data field		Absent						



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none">• 6A81 - Command not supported (state selectable)• 6A86 - P1 \neq '00'• 6A88 - Referenced PIN not found• 63Cx - Referenced PIN not successfully verified AND subsequent tries are allowed (error counter not null), x = remaining tries allowed• 6700 - Lc \neq '00' – PIN length is out of valid boundaries.• 6983 - Referenced PIN not successfully verified AND no subsequent tries are allowed (remaining tries counter reached 0)• 6984 - Referenced
---------	--



	PIN usage counter reached 0 <ul style="list-style-type: none"> • 6982 - Security status not satisfied
--	--

8.16.7 RESET RETRY COUNTER

The RESET RETRY COUNTER command is used to unblock, devalidate or unblock and change PIN1 or PIN2 code. In order to unblock code PUK verification operation must be performed first.

RESET RETRY COUNTER	
Command Parameter	Meaning
CLA	00 _{hex}
INS	2C _{hex}
P1	02 _{hex} (to unblock and change PIN1/PIN2) or 03 _{hex} (to unblock only) or FF _{hex} (to devalidate PIN1/PIN2)
P2	See table 1 below
Lc field	Variable
Data field	Absent (P1 = '03 _{hex} ' or 'FF _{hex} ') or new reference data (P1 = '02 _{hex} ') If present the new code must be padded with FF _{hex} on the right side so that the total length of the data field is 0C _{hex} e.g. if new code is 3132333435 _{hex} then the data field value will be 3132333435FFFFFFFFFFFFFFFF _{hex}
Le field	Absent

Table 1: P2 encoding - possible bit values

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Local reference data Application



0	-	-	-	-	-	-	-	Global reference data (Card)
-	-	x	x	x	x	x	x	User authentication DO reference (Code reference)
0	0	0	0	0	0	0	0	Forbidden
RESET RETRY COUNTER response								
Response parameter		Meaning						
Data field		Absent						



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none">• 6A81 - Command not supported (state selectable)• 6A86 - P1 \neq '02', P1 \neq '03' and P1 \neq 'FF'• 6A88 - Referenced PIN not found• 6700 - The length of the new reference data doesn't match with the length of the PIN reference container length (P1 = '02') or Lc \neq '00' (P1 = '03' or 'FF').• 6984 - Reference data not usable – Usage counter of referenced PIN raised 0.• 6982 - Security status not satisfied
---------	--



8.16.8 MANAGE SECURITY ENVIRONMENT (Set)

This command is used to change the currently active pointers to keys for security operations. At any time, a current cryptographic context – named current security environment (CSE) - is available. This context gathers all the informations required to perform any possible cryptographic operation offered by the application:

- Authentication
- Signature/verification of signature
- Encryption/decryption
- Hashing
- Key agreement

For each cryptographic operation, the current security environment contains:

- The security object reference
- The algorithm identifier indicating the usage
- The mode of operation (signature/verification of signature, ...)

Before performing any cryptographic operations such as digital signature, authentication protocols,...the application looks into the current security environment to find the data needed (security object reference, algorithm identifier indicating the usage, mode of operation)

MANAGE SECURITY ENVIRONMENT (Set)	
Command Parameter	Meaning
CLA	00 _{hex}
INS	22 _{hex}
P1	41 _{hex}
P2	See CRT chapter below
Lc field	Variable
Data field	Control reference template (CRT) See CRT chapter below
Le field	Absent



8.16.8.1 Control Reference Template (CRT)

The Control Reference Template is used in the MANAGE SECURITY ENVIRONMENT Set command. Following is a description of CRT's and the corresponding P1-P2 parameters that must be used when security environment is set for signature computation, internal authentication/calculating response to TLS challenge and decryption. CRT's covered within this chapter are:

- AT - Authentication template (internal authentication/calculating response to TLS challenge)
- DST - Digital signature template (signature computation)
- CT - Confidentiality template (encryption key decipherment)

8.16.8.1.1 Digital signature template (DST) for computing digital signature with EC private key

P1	P2	Length	Tag	Length	Value	Presence
41 _{hex}	B6 _{hex}	Variable	80 _{hex}	Variable	Algorithm identifier	Mandatory
			84 _{hex}	01 _{hex}	Asymmetric key object (private portion) reference (EC private key)	Mandatory

Algorithm identifier values in DST for computing digital signature

Algorithm identifier	Usage
14 _{hex} or FF110800 _{hex}	Digital signature with ECDSA SHA-1
34 _{hex} or FF130800 _{hex}	Digital signature with ECDSA SHA-224
44 _{hex} or FF140800 _{hex}	Digital signature with ECDSA SHA-256
54 _{hex} or FF150800 _{hex}	Digital signature with ECDSA SHA-384
64 _{hex} or FF160800 _{hex}	Digital signature with ECDSA SHA-512

8.16.8.1.2 Authentication template (AT) for calculating response to TLS challenge (client/server authentication)

P1	P2	Length	Tag	Length	Value	Presence
41 _{hex}	A4 _{hex}	Variable	80 _{hex}	Variable	Algorithm identifier	Mandatory
			84 _{hex}	01 _{hex}	Asymmetric key object (private portion) reference (EC private key)	Mandatory

Algorithm identifier values in AT for calculating response to TLS challenge

Algorithm identifier	Usage
----------------------	-------



04 _{hex} or FF200800 _{hex}	Authentication with ECDSA without any data hashing
--	--

8.16.8.1.3 Confidentiality template (CT) for encryption key decipherment

P1	P2	Length	Tag	L	Value	Presence
41 _{hex}	B8 _{hex}	Variable	80 _{hex}	Variable	Algorithm identifier	Mandatory
			84 _{hex}	01 _{hex}	Asymmetric key object (private portion) reference (EC private key)	Mandatory
Algorithm identifier values in AT for calculating response to TLS challenge						
Algorithm identifier		Usage				
0B _{hex} or FF300400 _{hex}		Encryption key decipherment with ECDH				

8.16.9 PERFORM SECURITY OPERATION - COMPUTE DIGITAL SIGNATURE

This command performs the digital signature creation using the signature EC private key on the card. To use this command the security environment must be set to use ECDSA operation with signature private key on QSCD application. Also PIN2 needs to be verified.

PERFORM SECURITY OPERATION - COMPUTE DIGITAL SIGNATURE	
Command Parameter	Meaning
CLA	00 _{hex}
INS	2A _{hex}
P1	9E _{hex}
P2	9A _{hex}



Lc field	Data field length (in case of hash off card): 0x30 (SHA-384 hash length) For last round hashing and on card hashing: empty
Data field	Hash of data or absent Hash algorithm outputs that have shorter bitlength than the EC key bitlength (384) - 0x30 need to be padded with zeroes from the left until they have a length of 0x30 including padding. Hash algorithm outputs that have longer bitlength than the EC key bitlength (384) - 0x30 need to be truncated so that they have a length of 0x30. See https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf (chapter 6.4) for information on truncating a longer hash.
Le field	00 _{hex} or absent
PERFORM SECURITY OPERATION - COMPUTE DIGITAL SIGNATURE response	
Response parameter	Meaning
Data field	Digital signature



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none"> • 6984 - Security Data Object (SDO) not usable • 6985 - No hash available / Conditions of use not satisfied • 6A81 - Command not supported (state selectable) • 6A88 - Current Security environment problem • 6A86 - Incorrect P1-P2 • 6982 - Security status not satisfied
---------	---

8.16.10 PERFORM SECURITY OPERATION - DECIPHER

This command performs the encryption key decipherment - deriving the shared secret using the authentication EC private key on card and public key in the command data field. To use this command the security environment must be set to use encryption key decipherment with ECDH with authentication private key. Also PIN1 needs to be verified.

PERFORM SECURITY OPERATION - DECIPHER
--



Command Parameter	Meaning
CLA	00 _{hex}
INS	2A _{hex}
P1	80 _{hex}
P2	86 _{hex}
Lc field	Variable (length of data field)
Data field	00 _{hex} Public key or Ephemeral public key
Le field	00 _{hex} or absent
PERFORM SECURITY OPERATION - DECIPHER response	
Response parameter	Meaning
Data field	Shared secret



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none">• 6A80 - Padding verification error. Length of data is too big or the public key is not in uncompressed format, its value is inconsistent or has a wrong length• 6A81 - Command not supported (state selectable)• 6A88 - Current Security environment problem• 6A86 - Incorrect P1-P2• 6982 - Decipher key access conditions not fulfilled, or extraction error• 6984 - Security Data Object (SDO) not usable• 6985 - Conditions of use not satisfied
---------	--



	<ul style="list-style-type: none"> 6982 - Security status not satisfied
--	--

8.16.11 INTERNAL AUTHENTICATE for client/server authentication

This command is used to authenticate the cardholder by the host side. Data field in C-APDU must contain challenge that will be encrypted with private key stored in the card. Response can be verified by using public key of the same key pair.

INTERNAL AUTHENTICATE	
Command Parameter	Meaning
CLA	00 _{hex}
INS	88 _{hex}
P1	00 _{hex}
P2	00 _{hex}
Lc field	Variable (length of data field)
Data field	TLS Challenge For ECDSA scheme the length of data shall not exceed the length in bits of the order of the generator
Le field	00 _{hex} or absent

INTERNAL AUTHENTICATE response	
Response parameter	Meaning
Data field	Authentication cryptogram



SW1-SW2	<p>All SW values are in hexadecimal format</p> <ul style="list-style-type: none">• 6700 - Wrong length; no further indication.• 6982 - Security status not satisfied• 6984 - Security data object (SDO) not usable• 6985 - Security environment content doesn't allow processing the command.• 6A81 - Command not supported (state selectable)• 6A86 - P1P2 ≠ '0000'• 6A88 - Reference data needed for internal authenticate not found



9 Card application objects and general operations

In this chapter we'll cover card application objects and main use cases for interacting with them. For more detailed information on the command- and response APDUs used in this chapter take a look at the [Command APDUs](#) section in this document. Some of the operations can be performed in multiple ways which are not all covered in this section. For more information on these ways you can again refer to the [Command APDUs](#) section or to ISO-7816-4. A good example of this is using the SELECT operation to navigate through the filesystem. Depending on your intent you can use different parameters to select DF, ADF or EF files or control whether the card application should or should not return file control parameters (FCP).

In case of communication via NFC interface [PACE tunnel creation](#) is required.

All the operations described here also have corresponding java code samples included in the [Java code examples for general operations](#) chapter.

9.1 Document number

It is a transparent file holding the document number as defined by Estonian legislation: two prefix letters and a seven digits unique number for the given prefix.

The Document Number is generated during personalisation phase.

Example: AS9991044

The document number is stored in a binary file as a TLV value, where the tag has [04_{oct}](#) value and document number value is ASCII encoded.

9.1.1 Reading document number from card application

Let the document number in our case be AS9991044_{ASCII} - 415339393931303434_{hex}

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select document number EF by issuing the following SELECT FILE command.

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	02 _{hex}	0C _{hex}	02 _{hex}	D003 _{hex}

3. To read data from the selected file issue a READ BINARY command. P1 and P2 are used to specify an offset in the selected file to start reading from.



CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	00 _{hex}	00 _{hex}	00 _{hex}

4. In case of a successful read the card responds with Response APDU holding a TLV value:

Data (Document number)	SW1	SW2
0409415339393931303434 _{hex}	90 _{hex}	00 _{hex}

NB! The document number can also be read from Personal Data file (See next chapter).

9.2 Personal data file

The same personal information that is visible on the card can also be read from the card application (except photo and signature image). Cardholders personal information is held in a Dedicated File (DF) with file identifier 5000_{hex}. All personal data records are Elementary files (EF) with a transparent structure (Transparent EF) which means the EF is seen as a sequence of data units. Personal data files are all mandatorily present on the card but some fields may be empty. In case the personal data field is empty, the corresponding Data File exists, has a one-byte size and contains the 0x00 terminator byte. In case the personal data field is present, the data length defines personal data file size.

9.2.1 ID card Personal data file example

File	Content	Object ID	Encoding	Sample value	Data format
PD1	Surname	5001 _{hex}	ASCII/UTF-8	JÕEORG	Xn
PD2	First name	5002 _{hex}	ASCII/UTF-8	JAAK-KRISTJAN	Xn
PD3	Sex	5003 _{hex}	ASCII/UTF-8	M	X
PD4	Citizenship	5004 _{hex}	ASCII/UTF-8	EST	XXX
PD5	Date and place of birth	5005 _{hex}	ASCII/UTF-8	08 01 1980 EST	DD MM YYYY XXX
PD6	Personal identification code	5006 _{hex}	ASCII/UTF-8	38001085718	99999999999



PD7	Document number	5007 _{hex}	ASCII/UTF-8	AS9991044	XX9999999
PD8	Expiry date	5008 _{hex}	ASCII/UTF-8	18 10 2023	DD MM YYYY
PD9 (before 01.08.2021)	Date of issuance	5009 _{hex}	ASCII/UTF-8	18 10 2018	DD MM YYYY
PD9 (from 01.08.2021)	Issuance date and authority	5009 _{hex}	ASCII/UTF-8	18 10 2018 PPA/PBGB	DD MM YYYY Xn
PD10	Type of residence permit	500A _{hex}	ASCII/UTF-8		
PD11	Notes line 1	500B _{hex}	ASCII/UTF-8		
PD12	Notes line 2	500C _{hex}	ASCII/UTF-8		
PD13	Notes line 3	500D _{hex}	ASCII/UTF-8		
PD14	Notes line 4	500E _{hex}	ASCII/UTF-8		
PD15	Notes line 5	500F _{hex}	ASCII/UTF-8		

9.2.2 Digital Identity Card (eResident) Personal data file example

File	Content	Object ID	Encoding	Sample value	Data format
PD1	Surname	5001 _{hex}	ASCII/UTF-8	JÕEORG	Xn
PD2	First name	5002 _{hex}	ASCII/UTF-8	JAAK-KRISTJAN	Xn
PD3	Sex	5003 _{hex}	ASCII/UTF-8		
PD4	Citizenship	5004 _{hex}	ASCII/UTF-8		



PD5	Date and place of birth	5005 _{hex}	ASCII/UTF-8		
PD6	Personal identification code	5006 _{hex}	ASCII/UTF-8	38001085718	9999999999
PD7	Document number	5007 _{hex}	ASCII/UTF-8	NS0000009	XX9999999
PD8	Expiry date	5008 _{hex}	ASCII/UTF-8	18 10 2023	DD MM YYYY
PD9 (before 01.08.2021)	Date of issuance	5009 _{hex}	ASCII/UTF-8	18 10 2018	DD MM YYYY
PD9 (from 01.08.2021)	Issuance date and authority	5009 _{hex}	ASCII/UTF-8	18 10 2018 PPA/PBGB	DD MM YYYY Xn
PD10	Type of residence permit	500A _{hex}	ASCII/UTF-8		
PD11	Notes line 1	500B _{hex}	ASCII/UTF-8		
PD12	Notes line 2	500C _{hex}	ASCII/UTF-8		
PD13	Notes line 3	500D _{hex}	ASCII/UTF-8		
PD14	Notes line 4	500E _{hex}	ASCII/UTF-8		
PD15	Notes line 5	500F _{hex}	ASCII/UTF-8		

9.2.3 Residence Permit Card Personal data file example

File	Content	Object ID	Encoding	Sample value	Data format
PD1	Surname	5001 _{hex}	ASCII/UTF-8	JÕEORG	Xn
PD2	First name	5002 _{hex}	ASCII/UTF-8	JAAK-KRISTJAN	Xn



PD3	Sex	5003 _{hex}	ASCII/UTF-8	M	X
PD4	Citizenship	5004 _{hex}	ASCII/UTF-8	UKR	XXX
PD5	Date and place of birth	5005 _{hex}	ASCII/UTF-8	08 01 1980 UKR	DD MM YYYY XXX
PD6	Personal identification code	5006 _{hex}	ASCII/UTF-8	38001085718	9999999999
PD7	Document number	5007 _{hex}	ASCII/UTF-8	PS0000038	XX9999999
PD8	Expiry date	5008 _{hex}	ASCII/UTF-8	18 10 2023	DD MM YYYY
PD9 (before 01.08.2021)	Date and place of Issuance	5009 _{hex}	ASCII/UTF-8	18 10 2018 PPA	DD MM YYYY XXX
PD9 (from 01.08.2021)	Issuance date and authority	5009 _{hex}	ASCII/UTF-8	18 10 2018 PPA/PBGB	DD MM YYYY X _n
PD10	Type of residence permit	500A _{hex}	ASCII/UTF-8	PIKAAJALINE ELANK	X _n
PD11	Notes line 1	500B _{hex}	ASCII/UTF-8		
PD12	Notes line 2	500C _{hex}	ASCII/UTF-8		
PD13	Notes line 3	500D _{hex}	ASCII/UTF-8		
PD14	Notes line 4	500E _{hex}	ASCII/UTF-8		
PD15	Notes line 5	500F _{hex}	ASCII/UTF-8		

9.2.4 Diplomatic Identity Card Personal data file example

File	Content	Object ID	Encoding	Sample value	Data format
------	---------	-----------	----------	--------------	-------------



PD1	Surname	5001 _{hex}	ASCII/UTF-8	THOMPSON	Xn
PD2	First name	5002 _{hex}	ASCII/UTF-8	STEVEN PAUL	Xn
PD3	Sex	5003 _{hex}	ASCII/UTF-8		
PD4	Citizenship	5004 _{hex}	ASCII/UTF-8		
PD5	Date of birth	5005 _{hex}	ASCII/UTF-8	11 08 1975	DD MM YYYY
PD6	Personal identification code	5006 _{hex}	ASCII/UTF-8	37508110387	9999999999
PD7	Document number	5007 _{hex}	ASCII/UTF-8	A19000195	XX9999999
PD8	Expiry date	5008 _{hex}	ASCII/UTF-8	18 10 2023	DD MM YYYY
PD9 (before 01.08.2021)	Date and place of Issuance	5009 _{hex}	ASCII/UTF-8	18 10 2018	DD MM YYYY
PD9 (from 01.08.2021)	Issuance date and authority	5009 _{hex}	ASCII/UTF-8	18 10 2018 PPA/PBGB	DD MM YYYY Xn
PD10	Type of residence permit	500A _{hex}	ASCII/UTF-8		
PD11	Notes line 1	500B _{hex}	ASCII/UTF-8		
PD12	Notes line 2	500C _{hex}	ASCII/UTF-8		
PD13	Notes line 3	500D _{hex}	ASCII/UTF-8		
PD14	Notes line 4	500E _{hex}	ASCII/UTF-8		
PD15	Notes line 5	500F _{hex}	ASCII/UTF-8		



9.2.5 Reading personal info from card application Cosmo v8.1, Cosmo v8.2 and CosmoX

To read the personal information a combination of SELECT FILE and READ BINARY operations must be performed.

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select Personal data DF by issuing the following SELECT FILE command. The P1=01_{hex} value means we are selecting a child DF.

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	02 _{hex}	5000 _{hex}

3. Select the personal data transparent EF that you wish to read data from by issuing another SELECT FILE command. The P1=02_{hex} value means we are selecting a child EF. In the example below we are selecting the first name record but by replacing the object ID the same command can be used to select any personal file record.

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	02 _{hex}	0C _{hex}	02 _{hex}	5002 _{hex}

4. To read data from the selected file issue a READ BINARY command. P1 and P2 are used to specify an offset in the selected file to start reading from.

CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	00 _{hex}	00 _{hex}	00 _{hex}

- 5.
6. In case of a successful read the card responds with Response APDU:

Data	SW1	SW2
4A41414B2D4B524953544A414E _{hex}	90 _{hex}	00 _{hex}

7. **NB!** You can repeat 3. and 4. in a loop to read all personal data from the chip - Just increment the Data portion of the Command APDU in part 3 from 5001_{hex} to 500F_{hex}



9.3 PIN1, PIN2 and PUK code operations

For PIN and PUK examples let's omit the following code values

Type	Text value	Hex value
PIN1	1234 _{ASCII}	31323334 _{hex}
PIN2	12345 _{ASCII}	3132333435 _{hex}
PUK	12345678 _{ASCII}	3132333435363738 _{hex}

9.3.1 Using the VERIFY command to read the remaining tries counter for PIN1, PIN2 and PUK codes

By issuing the VERIFY command without the actual code value (empty data field) the retry count can be read from the status bytes (SW1-SW2) of the Response APDU. When the body is empty, the command may be used either to retrieve the number 'X' of further allowed retries (SW1-SW2='63CX_{hex}') or to check whether the verification is not required (SW1-SW2='9000_{hex}').

9.3.1.1 PIN1 or PUK tries left

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Issue the following VERIFY command

to verify [PIN1](#)

CLA	INS	P1	P2
00 _{hex}	20 _{hex}	00 _{hex}	01 _{hex}

to verify [PUK](#)

CLA	INS	P1	P2
00 _{hex}	20 _{hex}	00 _{hex}	02 _{hex}



- The card should respond with the following response APDU where **X** is the number of further allowed retries. For example if the retry count is 2 then SW1-SW2=63C2_{hex} and if the retry count is 3 then SW1-SW2=63C3_{hex} etc.

Data	SW1	SW2
empty	63 _{hex}	C X _{hex}

9.3.1.2 PIN2 tries left

PIN2 is local PIN on QSCD ADF unlike **PIN1 and PUK** (are global PIN-s) so it requires navigating to QSCD application using extra SELECT command

- Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

- Select QSCD application DF by issuing the following SELECT FILE command. The data field is the file id (FID) of the QSCD application DF.

CLA	INS	P1	P2	Lc	Data (FID of QSCD ADF)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	10 _{hex}	ADF2 _{hex}

- Issue the following VERIFY command

CLA	INS	P1	P2
00 _{hex}	20 _{hex}	00 _{hex}	85 _{hex}

- The card should respond with the following response APDU where **X** is the number of further allowed retries. For example if the retry count is 2 then SW1-SW2=63C2_{hex} and if the retry count is 3 then SW1-SW2=63C3_{hex} etc.

Data	SW1	SW2
empty	63 _{hex}	C X _{hex}

9.3.2 Changing PIN1, PIN2 or PUK code.

The values of PIN1, PIN2 and PUK codes can be replaced by issuing the CHANGE REFERENCE DATA command if the given code is not blocked.



NB! As can be seen in the tables below the length of the codes must be 12 bytes. To achieve this the codes have to be padded with FF_{hex} on the right side. So in our PIN1 example 1234_{ASCII} is 31323334_{hex} and 31323334FFFFFFFFFFFFFFFF_{hex} including padding. Since we also need to add the new code value of 4321_{ASCII} which is 34333231_{hex} and 34333231FFFFFFFFFFFFFFFF_{hex} including padding. The combined command data value becomes 31323334FFFFFFFFFFFFFFFF34333231FFFFFFFFFFFFFFFF_{hex}. The same principle applies to PIN2 and PUK codes.

9.3.2.1 Change PIN1 or PUK codes

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Issue the following CHANGE REFERENCE DATA command

to change [PIN1](#) from 1234 to 4321

CLA	INS	P1	P2	Lc	Data (existing PIN1 new PIN1)
00 _{hex}	24 _{hex}	00 _{hex}	01 _{hex}	18 _{hex}	31323334FFFFFFFFFFFFFFFF34333231FFFFFFFFFFFFFFFF _{hex}

to change [PUK](#) from 12345678 to 87654321

CLA	INS	P1	P2	Lc	Data (existing PUK new PUK)
00 _{hex}	24 _{hex}	00 _{hex}	02 _{hex}	18 _{hex}	3132333435363738FFFFFFFF3837363534333231FFFFFFFF _{hex}

9.3.2.2 Change PIN2 code

PIN2 unlike PIN1 and PUK is located on the QSCD ADF so navigating to it requires an extra SELECT command

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select QSCD application DF by issuing the following SELECT FILE command. The data field is the file id (FID) of the QSCD application DF.



CLA	INS	P1	P2	Lc	Data (FID of QSCD ADF)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	10 _{hex}	ADF2 _{hex}

- Issue the following CHANGE REFERENCE DATA command to change [PIN2](#) from 12345 to 54321

CLA	INS	P1	P2	Lc	Data (existing PIN2 new PIN2)
00 _{hex}	24 _{hex}	00 _{hex}	01 _{hex}	18 _{hex}	3132333435FFFFFFFFFFFFFFFF3534333231FFFFFFFFFFFFFFFF _{hex}

9.3.3 Unblocking PIN1 and PIN2 code

When PIN codes retry counter values has decremented to value 0 and gets blocked, it is possible to unblock them by resetting the retry counter. This operation is possible with command RESET RETRY COUNTER.

9.3.3.1 Unblocking PIN1

- Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

- Verify PUK code

CLA	INS	P1	P2	Lc	Data (PUK code '12345678' padded)
00 _{hex}	20 _{hex}	00 _{hex}	02 _{hex}	0C _{hex}	3132333435363738FFFFFFFF _{hex}

- After given command is executed and successful response (SW1-SW2=9000) returned it is possible to reset retry counter of PIN codes
- Reset PIN1 with RESET RETRY COUNTER command. This resets the PIN1 with the new given value.

CLA	INS	P1	P2	Lc	Data (new PIN1 code '1234' padded)
00 _{hex}	2C _{hex}	02 _{hex}	01 _{hex}	0C _{hex}	31323334FFFFFFFFFFFFFFFF _{hex}

9.3.3.2 Unblocking PIN2



PIN2 unlike PIN1 and PUK is located on the QSCD ADF. This means that after PUK verification it's needed to navigate to QSCD application DF before issuing the RESET RETRY COUNTER COMMAND.

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Verify PUK code

CLA	INS	P1	P2	Lc	Data (PUK code '12345678' padded)
00 _{hex}	20 _{hex}	00 _{hex}	02 _{hex}	0C _{hex}	3132333435363738FFFFFFFF _{hex}

3. After given command is executed and successful response (SW1-SW2=9000) returned it is possible to reset retry counter of PIN codes
4. Select QSCD application DF by issuing the following SELECT FILE command. The data field is the file id (FID) of the QSCD application DF.

CLA	INS	P1	P2	Lc	Data (FID of QSCD ADF)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	10 _{hex}	ADF2 _{hex}

5. Reset PIN2 with RESET RETRY COUNTER command. This resets the PIN2 with the new given value.

CLA	INS	P1	P2	Lc	Data (new PIN2 code '12345' padded)
00 _{hex}	2C _{hex}	02 _{hex}	85 _{hex}	0C _{hex}	3132333435FFFFFFFFFFFFFFFF _{hex}

9.4 Reading certificates

The chip contains 2 certificates. One for cardholder authentication, encrypt and decrypt operations and the second for cardholder digital signing operations. The certificates are located on different application definition files (ADF) with the authentication certificate on AWP application (FID = ADF1_{hex}) and the signing certificate on QSCD application (FID = ADF2_{hex}). Certificate files are transparent files and can be read with the READ BINARY command. First navigate to the correct file

1. Select Master file by issuing a SELECT FILE command



CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select the correct ADF for certificate by issuing a SELECT FILE command
 1. For signing certificate select QSCD application

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	02 _{hex}	ADF2 _{hex}

2. For authentication certificate select AWP application

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	02 _{hex}	ADF1 _{hex}

3. Select the certificate file by issuing another SELECT FILE command
 1. For signing certificate the object ID is 341F_{hex}

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	02 _{hex}	0C _{hex}	02 _{hex}	341F _{hex}

2. For authentication certificate the object ID is 3401_{hex}

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	02 _{hex}	0C _{hex}	02 _{hex}	3401 _{hex}

- 3.

Now the certificate is ready for reading operations. The method described next is reading the file by sending multiple READ BINARY commands and changing the file reading offset for every command until the whole file has been read. Data of the result has to be concatenated together.

1. Send the first READ BINARY command in sequence

CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	00 _{hex}	00 _{hex}	00 _{hex}

2. Response:

Data	SW1	SW2
part1 of certificate	90 _{hex}	00 _{hex}

3. Continue sending READ BINARY commands while increasing the file reading offset by the data size of the previous read binary response every time until the chip responds with



an error SW1-SW2=6B00_{hex} (Wrong parameter(s) P1-P2) indicating that our pointer is farther than the file length.

CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	00 _{hex}	E7 _{hex}	00 _{hex}
CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	01 _{hex}	CE _{hex}	00 _{hex}
CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	02 _{hex}	B5 _{hex}	00 _{hex}
CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	03 _{hex}	9C _{hex}	00 _{hex}
CLA	INS	P1	P2	Le
00 _{hex}	B0 _{hex}	04 _{hex}	07 _{hex}	00 _{hex}

- Error response (Wrong parameter(s) P1-P2) indicating that the pointer is farther than the file length and the file has been read:

Data	SW1	SW2
empty	6B _{hex}	00 _{hex}

9.5 Computing digital signature

Card application enables the calculation of the electronic signature using the EC key in two ways:

- Providing card application with already calculated hash for signing procedure.
- Providing card application with data to be hashed before the signature procedure.

9.5.1 Computing digital signature for pre-calculated hash

This chapter describes the signing method where hash is calculated by the host application and provided to card application prior to signing operation. For authorising cardholder for given operation it is needed to authenticate the user with PIN2.

For this chapter let PIN2 code be 12345_{ascii} - 3132333435_{hex}

- Select Master file by issuing a SELECT FILE command



CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select QSCD Application DF by issuing SELECT FILE command

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	02 _{hex}	ADF2 _{hex}

3. Verify PIN2 using the VERIFY command

CLA	INS	P1	P2	Lc	Data (PIN2 code '12345' padded)
00 _{hex}	20 _{hex}	00 _{hex}	85 _{hex}	0C _{hex}	3132333435FFFFFFFFFFFFFFFF _{hex}

4. Prepare the card security environment for ECDSA operation with sign key by executing command `MANAGE SECURITY ENVIRONMENT`

CLA	INS	P1	P2	Lc	Data (Cryptographic mechanism ref len value (ECDSA SHA- 384) Private key ref len value)
00 _{hex}	22 _{hex}	41 _{hex}	B6 _{hex}	09 _{hex}	80 _{hex} 04 _{hex} FF150800 _{hex} 84 _{hex} 01 _{hex} 9F _{hex}

5. Compute digital signature by executing command `COMPUTE DIGITAL SIGNATURE`

CLA	INS	P1	P2	Lc	Data (Hash)	Le
00 _{hex}	2A _{hex}	9E _{hex}	9A _{hex}	30 _{hex}	Hash algorithm output	00 _{hex}

9.5.1.1 Using hash algorithms with shorter output than 30_{hex}

Hash algorithm outputs that have shorter bit length than the EC key bit-length (384) need to be padded with zeroes from the left.



CLA	INS	P1	P2	Lc	Data (Hash)
00 _{hex}	2A _{hex}	9E _{hex}	9A _{hex}	30 _{hex}	3D5D6073666A36A7CD68A1B1DD0A4CBEF3197DDE32AFEE5DF6001F

9.6 Calculating response for TLS challenge

This chapter covers the use of the INTERNAL AUTHENTICATE command to calculate the response for a TLS challenge. This command is used for the client/server authentication. To authorise cardholder to perform this operation authentication with PIN1 is required. In the context of this chapter let PIN1 code be 1234_{ascii} - 31323334_{hex}.

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select AWP Application DF by issuing SELECT FILE command

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	02 _{hex}	ADF1 _{hex}

3. Authenticate cardholder with PIN1 using the VERIFY command to authorise executing command INTERNAL AUTHENTICATE

CLA	INS	P1	P2	Lc	Data (PIN1 code '1234' padded)
00 _{hex}	20 _{hex}	00 _{hex}	01 _{hex}	0C _{hex}	31323334FFFFFFFFFFFFFFFF _{hex}

4. Prepare the card security environment for ECDSA operation with auth key by executing command `MANAGE SECURITY ENVIRONMENT`

CLA	INS	P1	P2	Lc	Data (Cryptographic mechanism ref len value Private key ref len value)
00 _{hex}	22 _{hex}	41 _{hex}	A4 _{hex}	09 _{hex}	80 _{hex} 04 _{hex} <u>FF200800</u> _{hex} 84 _{hex} 01 _{hex} 81 _{hex}

5. Calculate the response for TLS challenge by executing command INTERNAL AUTHENTICATE

CLA	INS	P1	P2	Lc	Data	Le
-----	-----	----	----	----	------	----



00 _{hex}	88 _{hex}	00 _{hex}	00 _{hex}	TLS challenge length _{hex}	TLS challenge	00 _{hex}
-------------------	-------------------	-------------------	-------------------	-------------------------------------	---------------	-------------------

9.7 Decrypting public key encrypted data

Current chapter describes deriving the shared secret necessary for decryption by combining (ephemeral) public key with the authentication private key on the card. This is performed by the DECIPHER operation. To authorise cardholder to perform this operation authentication with PIN1 is required. In the context of this chapter let PIN1 code be 1234_{ascii} - 31323334_{hex}.

1. Select Master file by issuing a SELECT FILE command

CLA	INS	P1	P2
00 _{hex}	A4 _{hex}	00 _{hex}	0C _{hex}

2. Select AWP Application DF by issuing SELECT FILE command

CLA	INS	P1	P2	Lc	Data (FID)
00 _{hex}	A4 _{hex}	01 _{hex}	0C _{hex}	02 _{hex}	ADF1 _{hex}

3. Authenticate cardholder with PIN1 using the VERIFY command to authorise executing command DECIPHER

CLA	INS	P1	P2	Lc	Data (PIN1 code '1234' padded)
00 _{hex}	20 _{hex}	00 _{hex}	01 _{hex}	0C _{hex}	31323334FFFFFFFFFFFFFFFF _{hex}

4. Prepare the card security environment for ECDH operation with auth private key by executing command `MANAGE SECURITY ENVIRONMENT`

CLA	INS	P1	P2	Lc	Data (Cryptographic mechanism ref len value Private key ref len value)
00 _{hex}	22 _{hex}	41 _{hex}	B8 _{hex}	09 _{hex}	80 _{hex} 04 _{hex} FF300400_{hex} 84 _{hex} 01 _{hex} 81 _{hex}

5. Obtain shared secret by executing command DECIPHER

CLA	INS	P1	P2	Lc	Data	Le
00 _{hex}	2A _{hex}	80 _{hex}	86 _{hex}	Data length _{hex}	<u>Padding indicator: 0x00_{hex} + Public key (Ephemeral) used when deriving shared secret</u>	00 _{hex}



10 Technical implementation of NFC connectivity

10.1 MRTD application

Estonia eID Document “MRTD” part is implemented by LDS applet, which is an ISO 7816-4 compliant application, with reserved Master File (MF), not shared with other applications. In compliance with ICAO specifications, which define the MRTD structure, mechanisms and implementation, this application is selected by default on the card ; therefore the presented MF after power on is the one managed by this LDS applet.

LDS Application presents a Dedicated File (ADF) under the MF for storage of identity information, with standard ICAO security mechanisms :

- BAC
- AA
- PA
- SAC (PACE V2)
- PACE-CAM (Refinement of PACE protocol to allow in the same time as PACE an authentication of the document)
- EAC, bound to SAC

10.2 Basic Access Control (BAC)

Basic Access Control is the mechanism that provides Secure Messaging establishment for all data exchanges between the MRTD and the reader (Inspection System), based on symmetric cryptography (3DES and MAC), using session keys derived from a root value produced from printed information on the document (part of the MRZ). This makes that knowledge of this “secret value” is only accessible when the document is explicitly produced for reading, preventing non-legitimate reading or eavesdropping.

10.3 Establish PACE tunnel

PACE is a secure authentication protocol designed to establish a secure channel between an NFC device (like a smartphone) and an ePassport.

The NFC standard for the PACE protocol (Password Authenticated Connection Establishment) is defined in the ICAO Document 9303 (Technical Report on Physical Characteristics of Machine Readable Travel Documents) for secure communication with ePassports, using a combination of a shared password (derived from the Machine Readable Zone) and Diffie-Hellman key exchange to establish a secure channel.

PACE tunnel creation flow:





1. Select root IAS-ECC Root (MF Master File) AID.
 1. (for authentication with PKI application, MRTD application authentication is not in scope of this document.)
2. Set MSE Authentication Template.
3. Get Global Authentication GetNonce.
4. Get Global Authentication MapNonce.
5. Get Global Authentication KeyAgreement.
6. Get Global Authentication MutualAuthentication.

10.4 Set MSE Authentication Template

SetMSEAuthenticationTemplate	
Command Parameter	Meaning
CLA	00 _{hex}
INS	22 _{hex}
P1	C1 _{hex}
P2	A4 _{hex}
Lc field	Absent or length of data field



Data field	<p>template data bytes:</p> <ol style="list-style-type: none"> 1. 800A04007F00070202040204_{hex} (tag: 80_{hex} len: 0A_{hex}) <ol style="list-style-type: none"> 1. value: 4007F00070202040204_{hex}, value from GENERAL MAPPING protocol. id-PACE-ECDH-GM-AES-CBC-CMAC-256 2. 830102_{hex} (tag: 83_{hex} len: 01_{hex}) <ol style="list-style-type: none"> 1. value: 01_{hex} for MRZ authentication, value 02_{hex} for CAN authentication. 3. 84010C_{hex} (tag: 84_{hex} len: 01_{hex}) <ol style="list-style-type: none"> 1. value: 0C_{hex} for NIST256 algorithm, value 10_{hex} for BP384 algorithm. <p>This information can be obtained from EF.CardAccess file. Example of EF.CardAccess file response: SET { -- SecurityInfos SEQUENCE { -- Security Info OBJECT IDENTIFIER '0 4 0 127 0 7 2 2 4 2 4' -- PACEInfo, id-PACE-ECDH-GM-AES-CBC-CMAC-256 INTEGER 2 -- version: 2 INTEGER 16 -- parameterId : BrainpoolP384r1 } }</p>
Le field	00 _{hex}

10.5 Get Global Authentication GetNonce

1. Getting GetNonce response from global authentication.

GetGAGetNonce	
Command Parameter	Meaning
CLA	10 _{hex}
INS	86 _{hex}
P1	00 _{hex}
P2	00 _{hex}
Lc field	Absent or length of data field
Data field	bytes: 7C00 _{hex} (tag: 7C _{hex} len: 00 _{hex})
Le field	00 _{hex}



2. Validating Global Authentication GetNonce response header:

header: 7C228020_{hex}

3. Decrypting received nonce:

Padding nonce decryption: 03_{hex}. Create key with card CAN number and [cipher](#) transformation: AES/CBC/NoPadding. Decrypt with decrypt mode: 2, algorithm: AES and block size: 10_{hex}.

10.6 Get Global Authentication MapNonce

1. Generate an EC keypair and exchange public keys with the chip (curve parameter spec: secp256r1).

GetGAMapNonce	
Command Parameter	Meaning
CLA	10 _{hex}
INS	86 _{hex}
P1	00 _{hex}
P2	00 _{hex}
Lc field	Absent or length of data field
Data field	prefix: 7C438141 _{hex} (tag: 7C _{hex} len: 43 _{hex} , tag: 81 _{hex} len: 41 _{hex}) + generated publicKey bytes
Le field	00 _{hex}

2. Validating Global Authentication MapNonce response header:

header: 7C438241_{hex}

3. Extract bytes from R-APDU to represent card public key.

10.7 Get Global Authentication KeyAgreement

1. Calculate the new base point, use it to generate a new keypair and exchange public keys.

GetGAKeyAgreement	
Command Parameter	Meaning



CLA	10 _{hex}
INS	86 _{hex}
P1	00 _{hex}
P2	00 _{hex}
Lc field	Absent or length of data field
Data field	prefix: 7C438341 _{hex} (tag: 7C _{hex} len: 43 _{hex} , tag: 83 _{hex} len: 41 _{hex}) + generated publicKey bytes
Le field	00 _{hex}

2. Extract 65 bytes from R-APDU to represent card public key.

3. Validating Global Authentication MapNonce response header:

header: 7C438441_{hex}

10.8 Get Global Authentication MutualAuthentication

1. Generate the session keys and exchange MACs to verify them.
2. Get MAC.
3. GetGAMutualAuthentication:

GetGAMutualAuthentication	
Command Parameter	Meaning
CLA	00 _{hex}
INS	86 _{hex}
P1	00 _{hex}
P2	00 _{hex}
Lc field	Absent or length of data field
Data field	prefix: 7C0A8508 _{hex} (tag: 7C _{hex} len: 0A _{hex} , tag: 85 _{hex} len: 08 _{hex}) + MAC bytes
Le field	00 _{hex}

4. Validating Global Authentication MutualAuthentication response header:

header: 7C0A8608_{hex}



5. verify chip's MAC and return session keys.

11 Java code examples for general operations

These are code examples written in Java. All of them use classes from `javax.smartcardio.*` package to establish a channel and communicate with the smartcard. The first chapter shows how to establish a channel to communicate with the smartcard. All other examples assume that the channel is established and we can use its *transmit* function.

11.1 Establishing a channel

```
//Get a list of available card terminals
List<CardTerminal> terminals =
TerminalFactory.getDefault().terminals().list();

//You can select the correct terminal by filtering the list by name or
whether it has a card present
//For the purposes of this example let's just choose the first one from the
list
CardTerminal terminal = terminals.get(0);

//Connect with card (using the T=1 protocol)
Card card = terminal.connect("T=1");

//Establish a channel
CardChannel channel = card.getBasicChannel();
```

11.2 Establishing a PACE tunnel

See steps: Establish PACE tunnel

Java based NFC lib Android implementation with open source code and demo application is accessible via the following GitHub repo: <https://github.com/open-eid/nfc-Android-lib>

Swift based NFC lib iOS implementation with open source code and demo application is accessible via the following GitHub repo: <https://github.com/open-eid/nfc-iOS-lib>

```
try {
    byte[][] keys = establishPace(can.getBytes(StandardCharsets.UTF_8));

    keyEnc = keys[0];
    keyMAC = keys[1];

    // In case we were successful we notify the card that from now on
    // everything is encrypted
    nfcReader.setAduEncryptor(this);
} catch (SmartCardReaderException ex) {
    if (ex instanceof AduResponseException aex) {
        if ((aex.sw1 == (byte) 0x63) && (aex.sw2 == 0x00)) {
```



```

        throw new PaceTunnelException(ex);
    }
    }
    throw ex;
} catch (Exception ex) {
    throw new SmartCardReaderException("Could not establish tunnel", ex);
}

```

11.3 NFC Helper functions

we will be using the following helper functions for validating header, generating random private key and getting MAC:

```

private static void validateHeader(byte[] response, byte[] header) throws
    SmartCardReaderException {

    if (header.length > response.length) {
        throw new SmartCardReaderException(
            "Response length is shorter than expected header length");
    }

    for (int i = 0; i < header.length; i++) {
        if (header[i] != response[i]) {
            throw new SmartCardReaderException(
                "Unexpected byte at index " + i + " in the APDU header");
        }
    }
}

private static BigInteger generateRandomPrivateKey(BigInteger upperBound) {
    SecureRandom random = new SecureRandom();

    // Generate a random number in the range [1, upperBound-1]
    BigInteger randomValue = new BigInteger(upperBound.bitLength(),
        random).add(BigInteger.ONE);

    // Ensure the generated value is within the specified range
    return randomValue.min(upperBound.subtract(BigInteger.ONE));
}

private byte[] getMAC(byte[] data, byte[] keyMAC) {
    BlockCipher blockCipher = new AESEngine();
    CMac cmac = new CMac(blockCipher);
    cmac.init(new KeyParameter(keyMAC));
    cmac.update(data, 0, data.length);
    byte[] MAC = new byte[cmac.getMacSize()];
    cmac.doFinal(MAC, 0);
    return Arrays.copyOf(MAC, MAC_LENGTH);
}

```

11.4 Helper functions

we will be using 2 helper functions for concatenation and padding codes:



```
private static byte[] concat(byte[] ... byteArrays) throws IOException {
    ByteArrayOutputStream stream = new ByteArrayOutputStream();
    for (byte[] byteArray : byteArrays) {
        stream.write(byteArray);
    }
    return stream.toByteArray();
}

private static byte[] padCode(byte[] code) {
    byte[] padded = Arrays.copyOf(code, 12);
    Arrays.fill(padded, code.length, padded.length, (byte) 0xFF);
    return padded;
}
```

11.5 Reading document number from card application

```
CardChannel channel = card.getBasicChannel();

//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Select document number EF
CommandAPDU selectDocumentNumberEF = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C,
new byte[]{(byte)0xD0, 0x03});
channel.transmit(selectDocumentNumberEF);
//Read binary and convert the response data into a UTF-8 string
CommandAPDU readBinary = new CommandAPDU(new byte[]{0x00, (byte)0xB0, 0x00,
0x00, 0x00});
ResponseAPDU response = channel.transmit(readBinary);

String documentNumber = new String(response.getData(), Charset.forName("UTF-
8")).trim();
```

11.6 Reading personal info from card application Cosmo v8.1 and Cosmo v8.2

Reading first name value:

```
CardChannel channel = card.getBasicChannel();

//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Select personal data DF
CommandAPDU selectPersonalDataFile = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C,
new byte[]{0x50, 0x00});
channel.transmit(selectPersonalDataFile);
//Select first name record
CommandAPDU selectFirstNameRecord = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C,
new byte[]{0x50, 0x02});
channel.transmit(selectFirstNameRecord);
```



```
//Read binary and convert the response data into an UTF-8 string
CommandAPDU readBinary = new CommandAPDU(new byte[]{0x00, (byte)0xB0, 0x00,
0x00, 0x00});
ResponseAPDU response = channel.transmit(readBinary);
String firstName = new String(response.getData(), Charset.forName("UTF-
8")).trim();
```

Reading all personal data records

```
CardChannel channel = card.getBasicChannel();

//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Select personal data DF
CommandAPDU selectPersonalDataFile = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C,
new byte[]{0x50, 0x00});
channel.transmit(selectPersonalDataFile);

//Read all records and add them to a list of strings
List<String> allRecords = new ArrayList<>();
CommandAPDU readBinary = new CommandAPDU(new byte[]{0x00, (byte)0xB0, 0x00,
0x00, 0x00});
for (int i = 1; i <= 15; i++) {
    CommandAPDU selectChildeF = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C, new
byte[] {0x50, (byte) i});
    channel.transmit(selectChildeF);
    ResponseAPDU response = channel.transmit(readBinary);
    String record = new String(response.getData(), Charset.forName("UTF-
8")).trim();
    allRecords.add(record);
}
```

11.7 Reading personal info from card application Cosmo X

Reading first name value:

```
CardChannel channel = card.getBasicChannel();

//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Select personal data DF
CommandAPDU selectPersonalDataFile = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C,
new byte[]{0x50, 0x00});
channel.transmit(selectPersonalDataFile);
//Select first name record
CommandAPDU selectFirstNameRecord = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C,
new byte[]{0x50, 0x02});
channel.transmit(selectFirstNameRecord);

//Read binary and convert the response data into an UTF-8 string
CommandAPDU readBinary = new CommandAPDU(new byte[]{0x00, (byte)0xB0, 0x00,
0x00, 0x00});
ResponseAPDU response = channel.transmit(readBinary);
```



```
String firstName = new String(response.getData(), Charset.forName("UTF-8")).trim();
```

Reading all personal data records

```
CardChannel channel = card.getBasicChannel();

//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Select personal data DF
CommandAPDU selectPersonalDataFile = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C,
new byte[]{0x50, 0x00});
channel.transmit(selectPersonalDataFile);

//Read all records and add them to a list of strings
List<String> allRecords = new ArrayList<>();
CommandAPDU readBinary = new CommandAPDU(new byte[]{0x00, (byte)0xB0, 0x00,
0x00, 0x00});
for (int i = 1; i <= 15; i++) {
    CommandAPDU selectChildeF = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C, new
byte[] {0x50, (byte) i});
    channel.transmit(selectChildeF);
    ResponseAPDU response = channel.transmit(readBinary);
    String record = new String(response.getData(), Charset.forName("UTF-8")).trim();
    allRecords.add(record);
}
```

11.8 Read remaining tries counter for PIN1, PIN2, PUK

11.8.1 PIN1

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Verify PIN1
CommandAPDU verifyPin1 = new CommandAPDU(0x00, 0x20, 0x00, 0x01);
ResponseAPDU response = channel.transmit(verifyPin1);
//Get retry count from SW
String sw = Integer.toHexString(response.getSW());
int retryCount = Integer.parseInt(sw.substring(sw.length() - 1));
```

11.8.2 PUK

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Verify PUK
CommandAPDU verifyPuk = new CommandAPDU(0x00, 0x20, 0x00, 0x02);
ResponseAPDU response = channel.transmit(verifyPuk);
//Get retry count from SW
String sw = Integer.toHexString(response.getSW());
```



```
int retryCount = Integer.parseInt(sw.substring(sw.length() - 1));
```

11.8.3 PIN2

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
//Select QSCD ADF
byte[] qscdFid = new byte[] {(byte)0xAD, (byte)0xF2};
CommandAPDU selectQSCDAdf = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C, qscdFid);
channel.transmit(selectQSCDAdf);
//Verify PIN2
CommandAPDU verifyPin2 = new CommandAPDU(0x00, 0x20, 0x00, 0x85);
ResponseAPDU response = channel.transmit(verifyPin2);

//Get retry count from SW
String sw = Integer.toHexString(response.getSW());
int retryCount = Integer.parseInt(sw.substring(sw.length() - 1));
```

11.9 Changing PIN1, PIN2 or PUK code.

The values of PIN1, PIN2 and PUK codes can be replaced by issuing the CHANGE REFERENCE DATA command if the given code is not blocked.

11.9.1 PIN1

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Change PIN1 from 1234 to 4321
CommandAPDU changePin1 = new CommandAPDU(0x00, 0x24, 0x00, 0x01,
concat(padCode("1234".getBytes()), padCode("4321".getBytes())));
channel.transmit(changePin1);
```

11.9.2 PUK

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Change PUK from 12345678 to 87654321
CommandAPDU changePuk = new CommandAPDU(0x00, 0x24, 0x00, 0x02,
concat(padCode("12345678".getBytes()), padCode("87654321".getBytes())));
channel.transmit(changePuk);
```

11.9.3 PIN2

```
CardChannel channel = card.getBasicChannel();
```



```
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
//Select QSCD ADF
byte[] qscdFid = new byte[] {(byte)0xAD, (byte)0xF2};
CommandAPDU selectQSCDAdf = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C, qscdFid);
channel.transmit(selectQSCDAdf);
//Change PIN2 from 12345 to 54321
CommandAPDU changePuk = new CommandAPDU(0x00, 0x24, 0x00, 0x85,
concat(padCode("12345".getBytes()), padCode("54321".getBytes())));
channel.transmit(changePuk);
```

11.10 Unblocking PIN1 and PIN2 code

When PIN codes retry counter values has decremented to value 0 and gets blocked, it is possible to unblock them by resetting the retry counter. This operation is possible with command RESET RETRY COUNTER.

11.10.1 PIN1

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Verify PUK code
CommandAPDU verifyPuk = new CommandAPDU(0x00, 0x20, 0x00, 0x02,
padCode("12345678".getBytes()));
ResponseAPDU response = channel.transmit(verifyPuk);
//If PUK verification was successful then reset PIN1 with value 1234
if ("9000".equalsIgnoreCase(Integer.toHexString(response.getSW()))) {
    CommandAPDU resetRetryCounter = new CommandAPDU(0x00, 0x2C, 0x02, 0x01,
padCode("1234".getBytes()));
    channel.transmit(resetRetryCounter);
}
```

11.10.2 PIN2

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Verify PUK code
CommandAPDU verifyPuk = new CommandAPDU(0x00, 0x20, 0x00, 0x02,
padCode("12345678".getBytes()));
ResponseAPDU response = channel.transmit(verifyPuk);
//If PUK verification was successful then select QSCD ADF and reset PIN2 with
value 12345
if ("9000".equalsIgnoreCase(Integer.toHexString(response.getSW()))) {
    byte[] qscdFid = new byte[] {(byte)0xAD, (byte)0xF2};
    CommandAPDU selectQSCDAdf = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C,
qscdFid);
    channel.transmit(selectQSCDAdf);
}
```



```

    CommandAPDU resetRetryCounter = new CommandAPDU(0x00, 0x2C, 0x02, 0x85,
padCode("12345".getBytes()));
    channel.transmit(resetRetryCounter);
}

```

11.11 Reading certificates

The chip contains 2 certificates. One for cardholder authentication operations and the second for cardholder digital signing operations. The certificates are located on different application definition files (ADF) with the authentication certificate on AWP application (ADF1) and the signing certificate on QSCD application (ADF2). Certificate files are transparent files and can be read with the READ_BINARY command.

11.11.1 Read authentication certificate

```

CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);

//Select AWP application (fid = ADF1)
CommandAPDU selectADF1 = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C, new
byte[] {(byte)0xAD, (byte)0xF1});
channel.transmit(selectADF1);
//Select auth certificate (fid=3401)
CommandAPDU selectCertificate = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C, new
byte[] {0x34, 0x01});
channel.transmit(selectCertificate);

//Read certificate
ByteArrayOutputStream stream = new ByteArrayOutputStream();
boolean doneReading = false;
while (!doneReading) {
    CommandAPDU readBinary = new CommandAPDU(new byte[] {0x00, (byte)0xB0,
(byte)(stream.size() >> 8), (byte)stream.size(), 0x00});
    ResponseAPDU response = channel.transmit(readBinary);
    stream.write(response.getData());
    String sw = Integer.toHexString(response.getSW());
    if ("6B00".equalsIgnoreCase(sw)) {
        doneReading = true;
    }
}
byte[] cert = stream.toByteArray();
//Optionally create an X509Certificate
X509Certificate x509Certificate = (X509Certificate)
CertificateFactory.getInstance("X.509")
    .generateCertificate(new ByteArrayInputStream(cert));

```

11.11.2 Read digital signature certificate

```

CardChannel channel = card.getBasicChannel();

```



```
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);

//Select QSCD application (fid = ADF2)
CommandAPDU selectADF1 = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C, new
byte[] {(byte)0xAD, (byte)0xF2});
channel.transmit(selectADF1);
//Select digital signature certificate (fid=341F)
CommandAPDU selectCertificate = new CommandAPDU(0x00, 0xA4, 0x02, 0x0C, new
byte[] {0x34, 0x1F});
channel.transmit(selectCertificate);

//Read certificate
ByteArrayOutputStream stream = new ByteArrayOutputStream();
boolean doneReading = false;
while (!doneReading) {
    CommandAPDU readBinary = new CommandAPDU(new byte[] {0x00, (byte)0xB0,
(byte)(stream.size() >> 8), (byte)stream.size(), 0x00});
    ResponseAPDU response = channel.transmit(readBinary);
    stream.write(response.getData());
    String sw = Integer.toHexString(response.getSW());
    if ("6B00".equalsIgnoreCase(sw)) {
        doneReading = true;
    }
}
byte[] cert = stream.toByteArray();
//Optionally create an X509Certificate
X509Certificate x509Certificate = (X509Certificate)
CertificateFactory.getInstance("X.509")
    .generateCertificate(new ByteArrayInputStream(cert));
```

11.12 Computing digital signature for pre-calculated hash

```
CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
//Select QSCD application
byte[] qscdFid = new byte[] {(byte)0xAD, (byte)0xF2};
CommandAPDU selectQSCDadf = new CommandAPDU(0x00, 0xA4, 0x01, 0x0C, qscdFid);
channel.transmit(selectQSCDadf);
//Verify PIN2
CommandAPDU verifyPin2 = new CommandAPDU(0x00, 0x20, 0x00, 0x85,
padCode("12345".getBytes()));
ResponseAPDU response = channel.transmit(verifyPin2);
//Proceed with signature calculation if pin2 verification was successful
if ("9000".equalsIgnoreCase(Integer.toHexString(response.getSW()))) {
    //Set Security environment
    CommandAPDU setEnvironment = new CommandAPDU(0x00, 0x22, 0x41, 0xB6, new
byte[] {(byte) 0x80, 0x04, (byte) 0xFF, 0x15, 0x08, 0x00, (byte) 0x84, 0x01,
(byte) 0x9F});
    channel.transmit(setEnvironment);
```



```

        //SHA-256 hash
        String text = "JÕEORG";
        byte[] sha256DigestValue = MessageDigest.getInstance("SHA-
256").digest(text.getBytes());
        //pad with zeroes
        byte[] padded = padWithZeroes(sha256DigestValue);
        //calculate digital signature
        CommandAPDU securityOperationComputeSignature =
            new CommandAPDU(concat(new byte[]{0x00, 0x2A, (byte)0x9E, (byte)0x9A,
(byte)padded.length}, padded, new byte[]{0x00}));
        byte[] signature =
channel.transmit(securityOperationComputeSignature).getData();
    }

//Helper function to pad hash with zeroes
private static byte[] padWithZeroes(byte[] hash) throws IOException {
    if (hash.length >= 48) {
        return hash;
    }
    try (ByteArrayOutputStream toSign = new ByteArrayOutputStream()) {
        toSign.write(new byte[48 - hash.length]);
        toSign.write(hash);
        return toSign.toByteArray();
    }
}

```

11.13 Calculating response for TLS challenge

```

CardChannel channel = card.getBasicChannel();
//Select master file
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);
channel.transmit(selectMF);
//Select AWP application
CommandAPDU selectADF1 = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C, new
byte[]{(byte)0xAD, (byte)0xF1});
channel.transmit(selectADF1);
//Verify PIN1
CommandAPDU verifyPin1 = new CommandAPDU(0x00, 0x20, 0x00, 0x01,
padCode("1234".getBytes()));
ResponseAPDU verifyResponse = channel.transmit(verifyPin1);
//proceed if pin verification was successful
if ("9000".equalsIgnoreCase(Integer.toHexString(verifyResponse.getSW()))) {
    //Set security environment
    CommandAPDU setEnvironment =
        new CommandAPDU(0x00, 0x22, 0x41, 0xA4, new byte[] {(byte) 0x80,
0x04, (byte) 0xFF, 0x20, 0x08, 0x00, (byte) 0x84, 0x01, (byte) 0x81});
    channel.transmit(setEnvironment);
    //Use JÕEORG as challenge
    byte[] challenge = "JÕEORG".getBytes();
    //Send the INTERNAL AUTHENTICATE command and read response from
ResponseAPDU
    CommandAPDU internalAuthenticate =
        new CommandAPDU(concat(new byte[]{0x00, (byte)0x88, (byte)0x00,
(byte)0x00, (byte)challenge.length}, challenge, new byte[]{0x00}));
}

```



```
    byte[] response = channel.transmit(internalAuthenticate).getData();  
}
```

11.14 Decrypting public key encrypted data

```
CardChannel channel = card.getBasicChannel();  
//Select master file  
CommandAPDU selectMF = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C);  
channel.transmit(selectMF);  
//Select AWP application  
CommandAPDU selectADF1 = new CommandAPDU(0x00, 0xA4, 0x00, 0x0C, new  
byte[]{(byte)0xAD, (byte)0xF1});  
channel.transmit(selectADF1);  
//Verify PIN1  
CommandAPDU verifyPin1 = new CommandAPDU(0x00, 0x20, 0x00, 0x01,  
padCode("1234".getBytes()));  
channel.transmit(verifyPin1);  
//Set security environment  
CommandAPDU setEnvironment =  
    new CommandAPDU(0x00, 0x22, 0x41, 0xB8, new byte[] {(byte) 0x80, 0x04,  
(byte) 0xFF, 0x30, 0x04, 0x00, (byte) 0x84, 0x01, (byte) 0x81});  
channel.transmit(setEnvironment);  
  
//The encrypted data transmitted to the card application must be pre-padded  
with 00hex  
byte[] prefix = new byte[] {0x00};  
  
//We need the ephemeral public key data for the card application to derive  
the shared secret  
byte[] publicKeyData = getEphemeralPublicKeyRawBitsFromSomewhere();  
byte[] data = concat(prefix, publicKeyData);  
  
//Derive shared secret  
byte[] header = new byte[]{0x00, 0x2A, (byte)0x80, (byte)0x86};  
byte[] headerWithLc = concat(header, new byte[]{(byte)data.length});  
byte le = 0x00;  
  
CommandAPDU deriveSharedSecret = new CommandAPDU(concat(headerWithLc, data,  
new byte[]{le}));  
  
byte[] sharedSecret = channel.transmit(deriveSharedSecret).getData();
```



12 Appendix 1: Detailed ATR description

12.1 Answer to reset (Contact interface) Cosmo v8.1 and Cosmo v8.2

Every contact card responds to reset with sequence of bytes called Answer To Reset (ATR). The ATR gives information about the electrical communication protocol and the chip itself. It is mainly linked with the underlying integrated circuit (chip) and also the Java operating system on it. There is a block of historical bytes that can be used to indicate the purpose of the chip card. The ATR can be different depending on if the reset is the first since power-up (Cold ATR) or not (Warm ATR). The meaningful info of ATR can be read from historical bytes of Cold ATR. Together with a category indicator byte the historical bytes form a string of 10 bytes with the value "00 12 23 3F 53 65 49 44 0F 90 00_{hex}".

TS	3B								
Protocol bytes _{hex}	T0	TA1	TC1	TD1	TD2	TA3	TB3	TD3	TA15
	DB	96	00	80	B1	FE	45	1F	83
Historical bytes _{hex}	T1	T2	T3	T4	T5	T6	T7	T8	
	00	31	C1	64	084021				
Additional bytes _{hex}	STATE	SW1	SW2	TCK					
	00	90	00	XX					

The ATR resulting from default choices is detailed below:

- TA1='96': F=512, D=32, i.e. 307 200 bauds
- TC1='00': No Extra Guard time (specific to T=0 protocol - character time = 12 etu)
- TD1='80': Card bi-protocol T=0/T=1
- TA3='FE': IFSC=254 bytes (specific to T=1 protocol)
- TB3='45': Waiting times BWI=4, CWI=5 (specific to T=1 protocol)
- TA15='83': Clock stop indicator state H (high) and class A & B (class C is not supported)
- Historical Bytes: 0012233053654944 0F 9000
 - Category Indicator: 0x00
 - Country Code (ISO 3166-1): 0x233F (Estonia)
 - Card's issuer data: 0x654944 ("eID")
 - LCS: 0x0F (Termination State)
 - SW: 0x9000
- TCK: 0xF1



The resulting specific ATR to Estonia is: 3B DB 96 00 80 B1 FE 45 1F 83 00 12 23 3F 53 65 49 44 0F 9000 F1

12.2 Answer to reset (Contact interface) Cosmo X

Every contact card responds to reset with sequence of bytes called Answer To Reset (ATR). The ATR gives information about the electrical communication protocol and the chip itself. It is mainly linked with the underlying integrated circuit (chip) and also the Java operating system on it. There is a block of historical bytes that can be used to indicate the purpose of the chip card. The ATR can be different depending on if the reset is the first since power-up (Cold ATR) or not (Warm ATR). The meaningful info of ATR can be read from historical bytes of Cold ATR. Together with a category indicator byte the historical bytes form a string of 10 bytes with the value "00 12 23 3F 54 65 49 44 32 0F 90 00_{hex}".

TS	3B								
Protocol bytes _{hex}	T0	TA1	TC1	TD1	TD2	TA3	TB3	TD3	TA15
	DC	96	00	80	B1	FE	45	1F	83
Historical bytes _{hex}	T1	T2	T3	T4	T5	T6	T7	T8	
	00	31	C1	64	084021				
Additional bytes _{hex}	STATE	SW1	SW2	TCK					
	00	90	00	XX					

The ATR resulting from default choices is detailed below:

The ATR resulting from default choices is detailed below:

- TA1='96': F=512, D=32, i.e. 307 200 bauds
- TC1='00': No Extra Guard time (specific to T=0 protocol - character time = 12 etu)
- TD1='80': Card bi-protocol T=0/T=1
- TA3='FE': IFSC=254 bytes (specific to T=1 protocol)
- TB3='45': Waiting times BWI=4, CWI=5 (specific to T=1 protocol)
- TA15='83': Clock stop indicator state H (high) and class A & B (class C is not supported)
- Historical Bytes: 001223305465494432 0F 9000
 - Category Indicator: 0x00
 - Country Code (ISO 3166-1): 0x233F (Estonia)
 - Card's issuer data: 0x65494432 ("eID2")
 - LCS: 0x0F (Termination State)
 - SW: 0x9000
- TCK: 0xC3

The resulting specific ATR to Estonia is: 3B DC 96 00 80 B1 FE 45 1F 83 00 12 23 3F 54 65 49 44 32 0F 9000 C3

