Einführung der Gesundheitskarte

Spezifikation des Card Operating System (COS)

**Elektrische Schnittstelle**

|  |  |
| --- | --- |
| Version: | 3.10.0 |
| Revision: | \main\rel\_opb1\6 |
| Stand: | 21.04.2017 |
| Status: | freigegeben |
| Klassifizierung: | öffentlich |
| Referenzierung: | [gemSpec\_COS\_G2.1] |

Dokumentinformationen

Es handelt sich hier um eine veränderte Fassung des Dokumentes für Karten der Generation 2.1.

Die wichtigsten Änderungen gegenüber der Betriebssystemspezifikation der Generation 2 sind:

1. Der DES Algorithmus wurde aus dem normativen Umfang in ein optionales Funktionspaket verschoben, siehe (N000.030), entsprechende Performanzvorgaben wurden entfernt.
2. CV-Zertifikate basierend auf dem RSA Algorithmus wurden in ein optionales Funktionspaket verschoben, siehe (N000.032) entsprechende Performanzvorgaben wurden entfernt.
3. Performanzvorgaben angepasst, das bedeutet:
   1. Dort, wo die zugelassenen G2 COS deutlich performanter sind, wurden strengere Werte gesetzt und
   2. dort, wo die Performanzvorgaben viel zu streng waren, wurden sie an den Stand der Technik angepasst.
4. Anforderungen zur Kürzung von *keyReferenceList* im Kommando List Public Key präzisiert, siehe 14.9.7.3.

Die wichtigsten Änderungen gegenüber den Betriebssystemspezifikationen der Generation 1 sind:

1. Alle Betriebssystemspezifikationen der Generation 1 wurden in diesem Dokument zusammengefasst.
2. Alle SRQs zu den Betriebssystemspezifikationen wurden integriert.
3. Die kontaktlose Datenübertragung und die Datenübertragung gemäß [ISO/IEC 7816-12] wurden aufgenommen.
4. Die Kryptographie gemäß [FIPS 197] (AES) und [BSI-TR-03111] (elliptische Kurven) wurden dem normativen Teil hinzugefügt.
5. Für RSA-Schlüssel ist die Moduluslänge von 3072 bit zu unterstützen.

Dokumentenhistorie

| **Version** | **Stand** | **Kap./ Seite** | **Grund der Änderung, besondere Hinweise** | **Bearbeitung** |
| --- | --- | --- | --- | --- |
| 2.2.0 | R 0.5.2  R 0.5.3 |  | Die Dokumentenhistorie zu freigegebenen Dokumenten im Release 0.5.2 bzw. 0.5.3 und Vorgängerversionen ist in den dort gültigen Dokumenten ausführlich dargestellt. | gematik |
| 3.0.0 | 20.09.12 |  | Zusammenfügen aller Generation 1 Betriebssystemspezifikationen  Einarbeitung aller SRQs zu Betriebssystemspezifikationen der Generation 1  Einarbeitung der Generation 2 Anforderungen, unter anderem: Kontaktlose Datenübertragung, Datenübertragung gemäß [ISO/IEC 7816-12], AES, elliptische Kurven, RSA Moduluslänge 3072 bit | gematik |
| 3.1.0 | 30.01.13 |  | Anpassung des normativen Umfangs, Fehlerkorrekturen | gematik |
|  | 16.05.13 |  | Fehlerkorrekturen | PL P71 |
|  | 20.09.13 |  | Präzisierung PACE  Einarbeitung von Kommentaren | PL P71 |
| 3.2.0 | 22.10.13 |  | Einarbeitung von Kommentaren | gematik |
| 3.2.1 | 11.12.13 |  | List Public Key Kommando, Cachen von importierten Schlüsseln, Einarbeitung Kommentare | gematik |
| 3.4.0 | 21.02.14 |  | Fehlerkorrekturen | gematik |
| 3.5.0 | 01.04.14 |  | Fehlerbereinigung | gematik |
| 3.6.0 | 06.06.14 |  | Öffentliche RSA 3072 aus normativem Umfang entfernt, Fehlerbereinigungen | gematik |
| 3.7.0 | 26.08.14 |  | Einarbeitung Iteration 4 | gematik |
| 3.8.0 | 17.07.15 |  | Folgende Errata eingearbeitet:  R1.4.2 | Technik / SPE |
| 3.9.0 | 24.08.16 |  | Anpassungen zum Online-Produktivbetrieb (Stufe 1) | gematik |
|  | 02.03.17 |  | DES aus dem normativen Umfang entfernt | gematik |
| 3.10.0 | 21.04.17 |  | freigegeben | gematik |

Inhaltsverzeichnis

[1 Einordnung des Dokuments 21](#_Toc488231442)

[1.1 Zielsetzung 21](#_Toc488231443)

[1.2 Zielgruppe 22](#_Toc488231444)

[1.3 Geltungsbereich 22](#_Toc488231445)

[1.4 Abgrenzung des Dokuments 23](#_Toc488231446)

[1.5 Methodik 23](#_Toc488231447)

[1.5.1 Nomenklatur der Präfixe 23](#_Toc488231448)

[1.5.2 Nomenklatur Verschiedenes 24](#_Toc488231449)

[1.5.3 Normative und informative Abschnitte 24](#_Toc488231450)

[1.5.4 Komponentenspezifische Anforderungen 25](#_Toc488231451)

[1.5.5 Verwendung von Schüsselworten 26](#_Toc488231452)

[2 Optionen (normativ) 27](#_Toc488231453)

[3 Systemüberblick (informativ) 29](#_Toc488231454)

[4 Lebenszyklus von Karte und Applikation (informativ) 30](#_Toc488231455)

[5 Datentypen und Datenkonvertierung (normativ) 31](#_Toc488231456)

[5.1 BitLength Anzahl Bit in einem Bitstring 31](#_Toc488231457)

[5.2 OctetLength Anzahl Oktett in einem Oktettstring 32](#_Toc488231458)

[5.3 I2OS Integer nach Oktettstring 32](#_Toc488231459)

[5.4 OS2I Oktettstring nach Integer 33](#_Toc488231460)

[5.5 OS2P Oktettstring nach Punkt (uncompressed encoding) 33](#_Toc488231461)

[5.6 P2OS Endlicher Punkt nach Oktettstring 34](#_Toc488231462)

[5.7 Extrahiere führende Elemente 35](#_Toc488231463)

[5.7.1 Extrahiere führende Bits 35](#_Toc488231464)

[5.7.2 Extrahiere führende Oktette 35](#_Toc488231465)

[5.8 PaddingIso 35](#_Toc488231466)

[5.9 TruncateIso 36](#_Toc488231467)

[5.10 MGF Mask Generation Function 37](#_Toc488231468)

[5.11 RAND Zufälliger Oktettstring 38](#_Toc488231469)

[5.12 ceiling Aufrunden einer reellen Zahl 38](#_Toc488231470)

[5.13 floor Abrunden einer reellen Zahl 38](#_Toc488231471)

[6 Kryptographische Algorithmen (normativ) 39](#_Toc488231472)

[6.1 Hash-Algorithmen 39](#_Toc488231473)

[6.1.1 SHA-1 39](#_Toc488231474)

[6.1.2 SHA-256 39](#_Toc488231475)

[6.1.3 SHA-384 40](#_Toc488231476)

[6.1.4 SHA-512 40](#_Toc488231477)

[6.2 Schlüsselvereinbarung 40](#_Toc488231478)

[6.2.1 Verhalten für 3TDES-Schlüssel, Option\_DES 40](#_Toc488231479)

[6.2.2 Vereinbarung von AES-128-Schlüsseln 41](#_Toc488231480)

[6.2.3 Vereinbarung von AES-192-Schlüsseln 41](#_Toc488231481)

[6.2.4 Vereinbarung von AES-256-Schlüsseln 42](#_Toc488231482)

[6.2.5 Schlüsselableitung aus einer Card Access Number 42](#_Toc488231483)

[6.3 Symmetrischer Basisalgorithmus für Vertraulichkeit 43](#_Toc488231484)

[6.3.1 Symmetrische Verschlüsselung eines Datenblocks, Option\_DES 43](#_Toc488231485)

[6.3.1.1 Verschlüsselung mittels DES, Option\_DES 43](#_Toc488231486)

[6.3.1.2 Verschlüsselung mittels 3TDES, Option\_DES 44](#_Toc488231487)

[6.3.2 Symmetrische Entschlüsselung eines Datenblocks, Option\_DES 44](#_Toc488231488)

[6.3.2.1 Entschlüsselung mittels DES, Option\_DES 44](#_Toc488231489)

[6.3.2.2 Entschlüsselung mittels 3TDES, Option\_DES 44](#_Toc488231490)

[6.3.3 Symmetrische Verschlüsselung eines Datenblocks, AES 45](#_Toc488231491)

[6.3.4 Symmetrische Entschlüsselung eines Datenblocks, AES 45](#_Toc488231492)

[6.4 Asymmetrischer Basisalgorithmus RSA 46](#_Toc488231493)

[6.5 Asymmetrischer Basisalgorithmus elliptische Kurven 47](#_Toc488231494)

[6.6 Datenauthentisierung 47](#_Toc488231495)

[6.6.1 MAC-Generierung 47](#_Toc488231496)

[6.6.1.1 Generierung Retail-MAC, Option\_DES 47](#_Toc488231497)

[6.6.1.2 Generierung CMAC ohne internes Padding vor der CMAC-Berechnung 48](#_Toc488231498)

[6.6.1.3 Generierung CMAC mit internem ISO-Padding vor der CMAC-Berechnung 49](#_Toc488231499)

[6.6.2 MAC-Prüfung 49](#_Toc488231500)

[6.6.2.1 Prüfung Retail-MAC, Option\_DES 49](#_Toc488231501)

[6.6.2.2 Prüfung CMAC ohne internes Padding vor der CMAC-Berechnung 49](#_Toc488231502)

[6.6.2.3 Prüfung CMAC mit internem Padding vor der CMAC Berechnung 50](#_Toc488231503)

[6.6.3 Signaturberechnung 50](#_Toc488231504)

[6.6.3.1 Signaturberechnung mittels RSA 50](#_Toc488231505)

[6.6.3.1.1 RSA, ISO9796–2, DS1, SIGN, Option\_RSA\_CVC 50](#_Toc488231506)

[6.6.3.1.2 RSA, SSA, PKCS1–V1\_5 51](#_Toc488231507)

[6.6.3.1.3 RSA, SSA, PSS 52](#_Toc488231508)

[6.6.3.1.4 RSA, ISO9796–2, DS2, SIGN 53](#_Toc488231509)

[6.6.3.1.5 RSASSA–PSS–SIGN 53](#_Toc488231510)

[6.6.3.2 Signaturberechnung mittels ELC 54](#_Toc488231511)

[6.6.4 Signaturprüfung 54](#_Toc488231512)

[6.6.4.1 RSA, ISO9796–2, DS1, VERIFY, Option\_RSA\_CVC 54](#_Toc488231513)

[6.6.4.2 Signaturprüfung mittels elliptischer Kurven 55](#_Toc488231514)

[6.7 Vertraulichkeit von Daten, symmetrischer Fall 56](#_Toc488231515)

[6.7.1 Symmetrische Verschlüsselung 56](#_Toc488231516)

[6.7.1.1 Verschlüsselung 3TDES, Option\_DES 56](#_Toc488231517)

[6.7.1.2 Verschlüsselung AES 57](#_Toc488231518)

[6.7.2 Symmetrische Entschlüsselung 57](#_Toc488231519)

[6.7.2.1 Entschlüsselung 3TDES, Option\_DES 57](#_Toc488231520)

[6.7.2.2 Entschlüsselung AES 58](#_Toc488231521)

[6.8 Vertraulichkeit von Daten, asymmetrischer Fall 59](#_Toc488231522)

[6.8.1 Asymmetrische Verschlüsselung 59](#_Toc488231523)

[6.8.1.1 RSA, ES, PKCS1 V1.5 59](#_Toc488231524)

[6.8.1.2 RSA, OAEP, Verschlüsselung 59](#_Toc488231525)

[6.8.1.3 Elliptic Curve Key Agreement 60](#_Toc488231526)

[6.8.1.4 ELC Verschlüsselung 61](#_Toc488231527)

[6.8.2 Asymmetrische Entschlüsselung 62](#_Toc488231528)

[6.8.2.1 RSA, ES, PKCS1 V1.5, Decrypt 62](#_Toc488231529)

[6.8.2.2 RSA, OAEP, Decrypt 63](#_Toc488231530)

[6.8.2.3 Asymmetrische Entschlüsselung mittels ELC 64](#_Toc488231531)

[7 CV-Zertifikat 65](#_Toc488231532)

[7.1 CV-Zertifikat für RSA-Schlüssel, Option\_RSA\_CVC 65](#_Toc488231533)

[7.1.1 Bestandteile eines CV-Zertifikats für RSA-Schlüssel 65](#_Toc488231534)

[7.1.1.1 Certificate Profile Identifier (CPI) 65](#_Toc488231535)

[7.1.1.2 Certification Authority Reference (CAR) 65](#_Toc488231536)

[7.1.1.3 Certificate Holder Reference (CHR) 65](#_Toc488231537)

[7.1.1.4 Certificate Holder Autorisation (CHA) 66](#_Toc488231538)

[7.1.1.5 Object Identifier (OID) 66](#_Toc488231539)

[7.1.1.6 Öffentlicher Schlüssel 66](#_Toc488231540)

[7.1.1.6.1 Modulus 66](#_Toc488231541)

[7.1.1.6.2 Öffentlicher Exponent 66](#_Toc488231542)

[7.1.2 Zertifikatsprofile für RSA-Schlüssel 66](#_Toc488231543)

[7.1.2.1 CV-Zertifikat für CA-Schlüssel 66](#_Toc488231544)

[7.1.2.2 CV–Zertifikat für Authentisierungsschlüssel 67](#_Toc488231545)

[7.1.3 Struktur und Inhalt eines CV-Zertifikats für RSA-Schlüssel 67](#_Toc488231546)

[7.2 CV-Zertifikate für ELC-Schlüssel (informativ) 68](#_Toc488231547)

[8 Objekte 69](#_Toc488231548)

[8.1 Diverse Attribute (normativ) 69](#_Toc488231549)

[8.1.1 File Identifier 69](#_Toc488231550)

[8.1.2 Short File Identifier 69](#_Toc488231551)

[8.1.3 Life Cycle Status 69](#_Toc488231552)

[8.1.4 Zugriffsregelliste 70](#_Toc488231553)

[8.1.5 Rekord 71](#_Toc488231554)

[8.1.6 SE–Identifier 72](#_Toc488231555)

[8.1.7 PIN 72](#_Toc488231556)

[8.1.8 Datum 72](#_Toc488231557)

[8.2 Schlüsselmaterial (normativ) 73](#_Toc488231558)

[8.2.1 Symmetrische Schlüssel 73](#_Toc488231559)

[8.2.1.1 3TDES-Schlüssel, Option\_DES 73](#_Toc488231560)

[8.2.1.2 AES-128-Schlüssel 73](#_Toc488231561)

[8.2.1.3 AES-192-Schlüssel 73](#_Toc488231562)

[8.2.1.4 AES-256-Schlüssel 74](#_Toc488231563)

[8.2.2 Domainparameter für elliptische Kurven 74](#_Toc488231564)

[8.2.3 Privater Schlüssel 75](#_Toc488231565)

[8.2.3.1 Privater RSA-Schlüssel 75](#_Toc488231566)

[8.2.3.2 Privater ELC-Schlüssel 76](#_Toc488231567)

[8.2.4 Öffentlicher Schlüssel 76](#_Toc488231568)

[8.2.4.1 Öffentlicher RSA-Schlüssel 76](#_Toc488231569)

[8.2.4.2 Öffentlicher ELC-Schlüssel 76](#_Toc488231570)

[8.2.5 Transportschutz für ein Passwort 76](#_Toc488231571)

[8.3 File (normativ) 77](#_Toc488231572)

[8.3.1 Ordner 77](#_Toc488231573)

[8.3.1.1 Applikation 79](#_Toc488231574)

[8.3.1.2 Dedicated File 80](#_Toc488231575)

[8.3.1.3 Application Dedicated File 80](#_Toc488231576)

[8.3.2 Datei 80](#_Toc488231577)

[8.3.2.1 Transparentes Elementary File 81](#_Toc488231578)

[8.3.2.2 Strukturiertes Elementary File 82](#_Toc488231579)

[8.3.2.2.1 Linear variables Elementary File 84](#_Toc488231580)

[8.3.2.2.2 Linear fixes Elementary File 84](#_Toc488231581)

[8.3.2.2.3 Zyklisches Elementary File 85](#_Toc488231582)

[8.3.3 File Control Parameter 85](#_Toc488231583)

[8.4 Reguläres-Passwort (normativ) 87](#_Toc488231584)

[8.5 Multireferenz-Passwort (normativ) 91](#_Toc488231585)

[8.6 Schlüsselobjekt (normativ) 92](#_Toc488231586)

[8.6.1 Symmetrisches Authentisierungsobjekt 93](#_Toc488231587)

[8.6.2 Symmetrisches Kartenverbindungsobjekt 95](#_Toc488231588)

[8.6.3 Privates Schlüsselobjekt 97](#_Toc488231589)

[8.6.4 Öffentliches Schlüsselobjekt 102](#_Toc488231590)

[8.6.4.1 Öffentliches Signaturprüfobjekt 103](#_Toc488231591)

[8.6.4.2 Öffentliches Authentisierungsobjekt 104](#_Toc488231592)

[8.6.4.3 Öffentliches Verschlüsselungsobjekt 105](#_Toc488231593)

[8.7 Datenobjekte (informativ) 106](#_Toc488231594)

[8.8 Security Environment (informativ) 106](#_Toc488231595)

[8.9 Sicherheitsstatus (informativ) 108](#_Toc488231596)

[9 Objektsystem (normativ) 109](#_Toc488231597)

[9.1 Aufbau und Strukturtiefe 109](#_Toc488231598)

[9.2 Objektsuche 113](#_Toc488231599)

[9.2.1 Filesuche 113](#_Toc488231600)

[9.2.2 Passwortsuche 113](#_Toc488231601)

[9.2.3 Suche nach einem Schlüsselobjekt 114](#_Toc488231602)

[9.2.3.1 Suche nach einem geheimen Schlüsselobjekt mittels keyReference 115](#_Toc488231603)

[9.2.3.2 Suche nach einem öffentlichen Schlüsselobjekt 116](#_Toc488231604)

[9.3 Cache für öffentliche Schlüsselobjekte 117](#_Toc488231605)

[10 Zugriffskontrolle (normativ) 120](#_Toc488231606)

[10.1 Zugriffsart 120](#_Toc488231607)

[10.2 Zugriffsbedingung 121](#_Toc488231608)

[10.3 Zugriffsregel 123](#_Toc488231609)

[10.4 Zugriffsregelauswertung 124](#_Toc488231610)

[10.5 Auslesen von Zugriffsregeln 125](#_Toc488231611)

[11 Kommunikation (normativ) 126](#_Toc488231612)

[11.1 Request – Response 126](#_Toc488231613)

[11.2 Elektrische Schnittstellen 126](#_Toc488231614)

[11.2.1 Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] 126](#_Toc488231615)

[11.2.2 Übertragungsprotokoll gemäß [ISO/IEC 7816-12] 129](#_Toc488231616)

[11.2.3 Kontaktlose Datenübertragung gemäß ISO/IEC 14443 129](#_Toc488231617)

[11.3 OSI-Referenzmodell (informativ) 130](#_Toc488231618)

[11.4 Kommandobearbeitung 131](#_Toc488231619)

[11.5 Kommando-APDU 132](#_Toc488231620)

[11.5.1 Class Byte 132](#_Toc488231621)

[11.5.2 Instruction Byte 133](#_Toc488231622)

[11.5.3 Parameter P1 133](#_Toc488231623)

[11.5.4 Parameter P2 133](#_Toc488231624)

[11.5.5 Datenfeld 133](#_Toc488231625)

[11.5.6 LeFeld 134](#_Toc488231626)

[11.6 Antwort-APDU 135](#_Toc488231627)

[11.6.1 Datenfeld 135](#_Toc488231628)

[11.6.2 Trailer 135](#_Toc488231629)

[11.7 Zulässige Kommando-Antwort-Paare 136](#_Toc488231630)

[11.7.1 Case 1 Kommando-Antwort-Paar 136](#_Toc488231631)

[11.7.2 Case 2 Kommando-Antwort-Paar 136](#_Toc488231632)

[11.7.2.1 Case 2 Short Kommando 136](#_Toc488231633)

[11.7.2.2 Case 2 Extended Kommando 137](#_Toc488231634)

[11.7.2.3 Case 2 Response 138](#_Toc488231635)

[11.7.3 Case 3 Kommando 138](#_Toc488231636)

[11.7.3.1 Case 3 Short Kommando 138](#_Toc488231637)

[11.7.3.2 Case 3 Extended Kommando 139](#_Toc488231638)

[11.7.3.3 Case 3 Response 139](#_Toc488231639)

[11.7.4 Case 4 Kommando 140](#_Toc488231640)

[11.7.4.1 Case 4 Short Kommando 140](#_Toc488231641)

[11.7.4.2 Case 4 Extended Kommando 141](#_Toc488231642)

[11.7.4.3 Case 4 Response 141](#_Toc488231643)

[11.8 Command Chaining 142](#_Toc488231644)

[11.9 Längenbeschränkung von APDU 143](#_Toc488231645)

[12 Kanalkontext (normativ) 145](#_Toc488231646)

[12.1 Attribute eines logischen Kanals 145](#_Toc488231647)

[12.2 Reset-Verhalten 148](#_Toc488231648)

[12.3 Setzen eines Sicherheitsstatus 149](#_Toc488231649)

[12.4 Löschen eines Sicherheitsstatus 150](#_Toc488231650)

[12.4.1 Löschen des Sicherheitszustandes eines Schlüssels 150](#_Toc488231651)

[12.4.2 Löschen der Sicherheitszustände eines Ordners 151](#_Toc488231652)

[12.4.3 Löschen von Sessionkeys 151](#_Toc488231653)

[12.5 Setzen eines Passwortstatus 152](#_Toc488231654)

[12.6 Löschen eines Passwortstatus 152](#_Toc488231655)

[13 Gesicherte Kommunikation (normativ) 154](#_Toc488231656)

[13.1 Secure Messaging Layer 154](#_Toc488231657)

[13.1.1 Ableitung von Sessionkeys 154](#_Toc488231658)

[13.1.2 Bearbeitung einer Kommando-APDU 156](#_Toc488231659)

[13.2 Sicherung einer Kommando-APDU 158](#_Toc488231660)

[13.3 Sicherung einer Antwort-APDU 161](#_Toc488231661)

[14 Kommandos (normativ) 164](#_Toc488231662)

[14.1 Roll-Verhalten 165](#_Toc488231663)

[14.1.1 Roll-Back 166](#_Toc488231664)

[14.1.2 Roll-Forward 166](#_Toc488231665)

[14.2 Management des Objektsystems 166](#_Toc488231666)

[14.2.1 Activate 166](#_Toc488231667)

[14.2.1.1 Use Case Aktivieren eines Ordners oder einer Datei 167](#_Toc488231668)

[14.2.1.2 Use Case Aktivieren eines privaten oder symmetrischen Schlüsselobjektes 167](#_Toc488231669)

[14.2.1.3 Use Case Aktivieren eines öffentlichen Schlüsselobjektes 168](#_Toc488231670)

[14.2.1.4 Use Case Aktivieren eines Passwortobjektes 168](#_Toc488231671)

[14.2.1.5 Antwort der Karte auf Aktivieren eines Files 169](#_Toc488231672)

[14.2.1.6 Kommandoabarbeitung innerhalb der Karte 169](#_Toc488231673)

[14.2.2 Create 170](#_Toc488231674)

[14.2.3 Deactivate 171](#_Toc488231675)

[14.2.3.1 Use Case Deaktivieren eines Ordners oder einer Datei 171](#_Toc488231676)

[14.2.3.2 Use Case Deaktivieren eines privaten oder symmetrischen Schlüsselobjektes 171](#_Toc488231677)

[14.2.3.3 Use Case Deaktivieren eines öffentlichen Schlüsselobjektes 172](#_Toc488231678)

[14.2.3.4 Use Case Deaktivieren eines Passwortobjektes 172](#_Toc488231679)

[14.2.3.5 Antwort der Karte auf Deaktivieren eines Files 173](#_Toc488231680)

[14.2.3.6 Kommandoabarbeitung innerhalb der Karte 173](#_Toc488231681)

[14.2.4 Delete 175](#_Toc488231682)

[14.2.4.1 Use Case Löschen eines Ordners oder einer Datei 175](#_Toc488231683)

[14.2.4.2 Use Case Löschen eines privaten oder symmetrischen Schlüsselobjektes 175](#_Toc488231684)

[14.2.4.3 Use Case Löschen eines öffentlichen Schlüsselobjektes 176](#_Toc488231685)

[14.2.4.4 Use Case Löschen eines Passwortobjektes 176](#_Toc488231686)

[14.2.4.5 Antwort der Karte auf Löschen eines Files 177](#_Toc488231687)

[14.2.4.6 Kommandoabarbeitung innerhalb der Karte 177](#_Toc488231688)

[14.2.5 Load Application 180](#_Toc488231689)

[14.2.5.1 Use Case Anlegen neues Objekt, nicht Ende der Kommandokette 180](#_Toc488231690)

[14.2.5.2 Use Case Anlegen neues Objekt, Ende der Kommandokette 181](#_Toc488231691)

[14.2.5.3 Antwort der Karte auf Anlegen neues Objekt 181](#_Toc488231692)

[14.2.5.4 Kommandoabarbeitung innerhalb der Karte 182](#_Toc488231693)

[14.2.6 Select 184](#_Toc488231694)

[14.2.6.1 Use Case Selektieren ohne AID, first, keine Antwortdaten 184](#_Toc488231695)

[14.2.6.2 Use Case Selektieren ohne AID, first, Antwortdaten mit FCP 185](#_Toc488231696)

[14.2.6.3 Use Case Selektieren ohne AID, next, keine Antwortdaten 186](#_Toc488231697)

[14.2.6.4 Use Case Selektieren ohne AID, next, Antwortdaten mit FCP 186](#_Toc488231698)

[14.2.6.5 Use Case Selektieren per AID, first, keine Antwortdaten 187](#_Toc488231699)

[14.2.6.6 Use Case Selektieren per AID, first, Antwortdaten mit FCP 188](#_Toc488231700)

[14.2.6.7 Use Case Selektieren per AID, next, keine Antwortdaten 189](#_Toc488231701)

[14.2.6.8 Use Case Selektieren per AID, next, Antwortdaten mit FCP 189](#_Toc488231702)

[14.2.6.9 Use Case Selektieren, DF oder ADF, keine Antwortdaten 190](#_Toc488231703)

[14.2.6.10 Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP 191](#_Toc488231704)

[14.2.6.11 Use Case Selektieren übergeordnetes Verzeichnis ohne FCP 192](#_Toc488231705)

[14.2.6.12 Use Case Selektieren übergeordnetes Verzeichnis mit FCP 192](#_Toc488231706)

[14.2.6.13 Use Case Selektieren einer Datei, keine Antwortdaten 193](#_Toc488231707)

[14.2.6.14 Use Case Selektieren einer Datei, Antwortdaten mit FCP 193](#_Toc488231708)

[14.2.6.15 Zusammenfassung der Select-Kommando-Varianten 194](#_Toc488231709)

[14.2.6.16 Antwort der Karte auf Selektieren eines Files 195](#_Toc488231710)

[14.2.6.17 Kommandoabarbeitung innerhalb der Karte 196](#_Toc488231711)

[14.2.7 Terminate Card Usage 198](#_Toc488231712)

[14.2.7.1 Use Case Terminieren der Karte 199](#_Toc488231713)

[14.2.7.2 Antwort der Karte auf terminieren der Karte 199](#_Toc488231714)

[14.2.7.3 Kommandoabarbeitung innerhalb der Karte 199](#_Toc488231715)

[14.2.8 Terminate DF 200](#_Toc488231716)

[14.2.8.1 Use Case Terminieren eines Ordners 201](#_Toc488231717)

[14.2.8.2 Antwort der Karte auf terminieren der Karte 201](#_Toc488231718)

[14.2.8.3 Kommandoabarbeitung innerhalb der Karte 201](#_Toc488231719)

[14.2.9 Terminate 202](#_Toc488231720)

[14.2.9.1 Use Case Terminieren einer Datei 202](#_Toc488231721)

[14.2.9.2 Use Case Terminieren eines privaten oder symmetrischen Schlüsselobjektes 203](#_Toc488231722)

[14.2.9.3 Use Case Terminieren eines öffentlichen Schlüsselobjektes 203](#_Toc488231723)

[14.2.9.4 Use Case Terminieren eines Passwortobjektes 204](#_Toc488231724)

[14.2.9.5 Antwort der Karte auf terminieren von Datei, Schlüssel- oder Passwortobjekt 204](#_Toc488231725)

[14.2.9.6 Kommandoabarbeitung innerhalb der Karte 205](#_Toc488231726)

[14.3 Zugriff auf Daten in transparenten EF 206](#_Toc488231727)

[14.3.1 Erase Binary 206](#_Toc488231728)

[14.3.1.1 Use Case Löschen ohne shortFileIdentifier in transparenten EF 207](#_Toc488231729)

[14.3.1.2 Use Case Löschen mit shortFileIdentifier in transparenten EF 207](#_Toc488231730)

[14.3.1.3 Antwort der Karte auf Löschen in transparenten EF 208](#_Toc488231731)

[14.3.1.4 Kommandoabarbeitung innerhalb der Karte 208](#_Toc488231732)

[14.3.2 Read Binary 210](#_Toc488231733)

[14.3.2.1 Use Case Lesen ohne shortFileIdentifier in transparenten EF 210](#_Toc488231734)

[14.3.2.2 Use Case Lesen mit shortFileIdentifier in transparenten EF 211](#_Toc488231735)

[14.3.2.3 Antwort der Karte auf Lesen in transparenten EF 211](#_Toc488231736)

[14.3.2.4 Kommandoabarbeitung innerhalb der Karte 212](#_Toc488231737)

[14.3.3 Search Binary 213](#_Toc488231738)

[14.3.4 Set Logical Eof 214](#_Toc488231739)

[14.3.4.1 Use Case Setzen logical EOF ohne shortFileIdentifier 214](#_Toc488231740)

[14.3.4.2 Use Case Setzen logical EOF mit shortFileIdentifier 214](#_Toc488231741)

[14.3.4.3 Antwort der Karte auf Setzen logical EOF in transparenten EF 215](#_Toc488231742)

[14.3.4.4 Kommandoabarbeitung innerhalb der Karte 215](#_Toc488231743)

[14.3.5 Update Binary 217](#_Toc488231744)

[14.3.5.1 Use Case Schreiben ohne shortFileIdentifier in transparenten EF 217](#_Toc488231745)

[14.3.5.2 Use Case Schreiben mit shortFileIdentifier in transparenten EF 218](#_Toc488231746)

[14.3.5.3 Antwort der Karte auf Schreiben in transparenten EF 219](#_Toc488231747)

[14.3.5.4 Kommandoabarbeitung innerhalb der Karte 219](#_Toc488231748)

[14.3.6 Write Binary 221](#_Toc488231749)

[14.3.6.1 Use Case Anfügen ohne shortFileIdentifier in transparenten EF 221](#_Toc488231750)

[14.3.6.2 Use Case Anfügen mit shortFileIdentifier in transparenten EF 222](#_Toc488231751)

[14.3.6.3 Antwort der Karte auf Anfügen in transparenten EF 222](#_Toc488231752)

[14.3.6.4 Kommandoabarbeitung innerhalb der Karte 223](#_Toc488231753)

[14.4 Zugriff auf strukturierte Daten 225](#_Toc488231754)

[14.4.1 Activate Record 225](#_Toc488231755)

[14.4.1.1 Use Case Aktivieren eines Rekords ohne shortFileIdentifier 225](#_Toc488231756)

[14.4.1.2 Use Case Aktivieren eines Rekords mit shortFileIdentifier 226](#_Toc488231757)

[14.4.1.3 Use Case Aktivieren aller Rekords ab P1 ohne shortFileIdentifier 226](#_Toc488231758)

[14.4.1.4 Use Case Aktivieren aller Rekords ab P1 mit shortFileIdentifier 227](#_Toc488231759)

[14.4.1.5 Antwort der Karte auf Aktivieren eines Rekords 227](#_Toc488231760)

[14.4.1.6 Kommandoabarbeitung innerhalb der Karte 228](#_Toc488231761)

[14.4.2 Append Record 230](#_Toc488231762)

[14.4.2.1 Use Case Anlegen neuer Rekord, ohne shortFileIdentifier 230](#_Toc488231763)

[14.4.2.2 Use Case Anlegen neuer Rekords, mit shortFileIdentifier 230](#_Toc488231764)

[14.4.2.3 Antwort der Karte auf Anlegen eines neuen Rekords 231](#_Toc488231765)

[14.4.2.4 Kommandoabarbeitung innerhalb der Karte 232](#_Toc488231766)

[14.4.3 Deactivate Record 234](#_Toc488231767)

[14.4.3.1 Use Case Deaktivieren eines Rekords ohne shortFileIdentifier 234](#_Toc488231768)

[14.4.3.2 Use Case Deaktivieren eines Rekords mit shortFileIdentifier 235](#_Toc488231769)

[14.4.3.3 Use Case Deaktivieren aller Rekords ab P1 ohne shortFileIdentifier 235](#_Toc488231770)

[14.4.3.4 Use Case Deaktivieren aller Rekords ab P1 mit shortFileIdentifier 236](#_Toc488231771)

[14.4.3.5 Antwort der Karte auf Deaktivieren eines Rekords 237](#_Toc488231772)

[14.4.3.6 Kommandoabarbeitung innerhalb der Karte 237](#_Toc488231773)

[14.4.4 Delete Record 239](#_Toc488231774)

[14.4.4.1 Use Case Löschen eines Rekords ohne shortFileIdentifier 239](#_Toc488231775)

[14.4.4.2 Use Case Löschen eines Rekords mit shortFileIdentifier 240](#_Toc488231776)

[14.4.4.3 Antwort der Karte auf Entfernen eines Rekords 240](#_Toc488231777)

[14.4.4.4 Kommandoabarbeitung innerhalb der Karte 241](#_Toc488231778)

[14.4.5 Erase Record 242](#_Toc488231779)

[14.4.5.1 Use Case Löschen eines Rekordinhaltes ohne shortFileIdentifier 243](#_Toc488231780)

[14.4.5.2 Use Case Löschen eines Rekordinhaltes mit shortFileIdentifier 243](#_Toc488231781)

[14.4.5.3 Antwort der Karte auf Löschen des Inhaltes eines Rekords 244](#_Toc488231782)

[14.4.5.4 Kommandoabarbeitung innerhalb der Karte 244](#_Toc488231783)

[14.4.6 Read Record 246](#_Toc488231784)

[14.4.6.1 Use Case Lesen ohne shortFileIdentifier in strukturierten EF 246](#_Toc488231785)

[14.4.6.2 Use Case Lesen mit shortFileIdentifier in strukturierten EF 247](#_Toc488231786)

[14.4.6.3 Antwort der Karte auf Lesen in strukturierten EF 248](#_Toc488231787)

[14.4.6.4 Kommandoabarbeitung innerhalb der Karte 248](#_Toc488231788)

[14.4.7 Search Record 250](#_Toc488231789)

[14.4.7.1 Use Case Suchen ohne shortFileIdentifier in strukturierten EF 250](#_Toc488231790)

[14.4.7.2 Use Case Suchen mit shortFileIdentifier in strukturierten EF 251](#_Toc488231791)

[14.4.7.3 Antwort der Karte auf Suchen in strukturierten EF 252](#_Toc488231792)

[14.4.7.4 Kommandoabarbeitung innerhalb der Karte 252](#_Toc488231793)

[14.4.8 Update Record 254](#_Toc488231794)

[14.4.8.1 Use Case Rekordinhalt schreiben, ohne shortFileIdentifier 254](#_Toc488231795)

[14.4.8.2 Use Case Rekordinhalt schreiben, mit shortFileIdentifier 255](#_Toc488231796)

[14.4.8.3 Antwort der Karte auf Schreiben in strukturierten EF 255](#_Toc488231797)

[14.4.8.4 Kommandoabarbeitung innerhalb der Karte 256](#_Toc488231798)

[14.4.9 Write Record 259](#_Toc488231799)

[14.5 Zugriff auf Datenobjekte 259](#_Toc488231800)

[14.5.1 Get Data 259](#_Toc488231801)

[14.5.2 Put Data 259](#_Toc488231802)

[14.6 Benutzerverifikation 259](#_Toc488231803)

[14.6.1 Change Reference Data 260](#_Toc488231804)

[14.6.1.1 Use Case Ändern eines Benutzergeheimnisses 260](#_Toc488231805)

[14.6.1.2 Use Case Setzen eines Benutzergeheimnisses 260](#_Toc488231806)

[14.6.1.3 Antwort der Karte auf Ändern eines Benutzergeheimnisses 261](#_Toc488231807)

[14.6.1.4 Kommandoabarbeitung innerhalb der Karte 262](#_Toc488231808)

[14.6.2 Disable Verification Requirement 263](#_Toc488231809)

[14.6.2.1 Use Case Abschalten der Benutzerverifikation mit Benutzergeheimnis 264](#_Toc488231810)

[14.6.2.2 Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis 264](#_Toc488231811)

[14.6.2.3 Antwort der Karte auf Abschalten der Benutzerverifikation 265](#_Toc488231812)

[14.6.2.4 Kommandoabarbeitung innerhalb der Karte 265](#_Toc488231813)

[14.6.3 Enable Verification Requirement 267](#_Toc488231814)

[14.6.3.1 Use Case Einschalten der Benutzerverifikation mit Benutzergeheimnis 267](#_Toc488231815)

[14.6.3.2 Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis 267](#_Toc488231816)

[14.6.3.3 Antwort der Karte auf Einschalten der Benutzerverifikation 268](#_Toc488231817)

[14.6.3.4 Kommandoabarbeitung innerhalb der Karte 268](#_Toc488231818)

[14.6.4 Get Pin Status 270](#_Toc488231819)

[14.6.4.1 Use Case Auslesen des Status eines Passwortobjektes 270](#_Toc488231820)

[14.6.4.2 Antwort der Karte auf Auslesen des PIN Status 271](#_Toc488231821)

[14.6.4.3 Kommandoabarbeitung innerhalb der Karte 271](#_Toc488231822)

[14.6.5 Reset Retry Counter 272](#_Toc488231823)

[14.6.5.1 Use Case Entsperren mit PUK, mit neuem Geheimnis 273](#_Toc488231824)

[14.6.5.2 Use Case Entsperren mit PUK, ohne neues Geheimnis 273](#_Toc488231825)

[14.6.5.3 Use Case Entsperren ohne PUK, mit neuem Geheimnis 274](#_Toc488231826)

[14.6.5.4 Use Case Entsperren ohne PUK, ohne neues Geheimnis 274](#_Toc488231827)

[14.6.5.5 Antwort der Karte auf Entsperren eines Benutzergeheimnisses 275](#_Toc488231828)

[14.6.5.6 Kommandoabarbeitung innerhalb der Karte 275](#_Toc488231829)

[14.6.6 Verify 277](#_Toc488231830)

[14.6.6.1 Use Case Vergleich eines Benutzergeheimnisses 277](#_Toc488231831)

[14.6.6.2 Antwort der Karte auf Vergleich eines Benutzergeheimnisses 278](#_Toc488231832)

[14.6.6.3 Kommandoabarbeitung innerhalb der Karte 278](#_Toc488231833)

[14.7 Komponentenauthentisierung 280](#_Toc488231834)

[14.7.1 External Authenticate / Mutual Authenticate 280](#_Toc488231835)

[14.7.1.1 Use Case externe Authentisierung ohne Antwortdaten 280](#_Toc488231836)

[14.7.1.2 Use Case externe Authentisierung mit Antwortdaten 281](#_Toc488231837)

[14.7.1.3 Antwort der Karte auf externe Authentisierung 281](#_Toc488231838)

[14.7.1.4 Kommandoabarbeitung innerhalb der Karte 282](#_Toc488231839)

[14.7.2 General Authenticate 288](#_Toc488231840)

[14.7.2.1 Gegenseitige Authentisierung mittels PACE für Endnutzerkarten 288](#_Toc488231841)

[14.7.2.1.1 Use Case PACE für Endnutzerkarten, Schritt 1a 288](#_Toc488231842)

[14.7.2.1.2 Use Case PACE für Endnutzerkarten, Schritt 2a 289](#_Toc488231843)

[14.7.2.1.3 Use Case PACE für Endnutzerkarten, Schritt 3a 290](#_Toc488231844)

[14.7.2.1.4 Use Case PACE für Endnutzerkarten, Schritt 4a 290](#_Toc488231845)

[14.7.2.2 Gegenseitige Authentisierung mittels ELC Schlüsseln 291](#_Toc488231846)

[14.7.2.2.1 Use Case gegenseitige ELC-Authentisierung, Schritt 1 291](#_Toc488231847)

[14.7.2.2.2 Use Case gegenseitige ELC-Authentisierung, Schritt 2 291](#_Toc488231848)

[14.7.2.3 Authentisierung für asynchrone, symmetrische Kartenadministration 292](#_Toc488231849)

[14.7.2.3.1 Use Case Authentisierung für asynchrone, sym. Administration, Schritt 1 292](#_Toc488231850)

[14.7.2.3.2 Use Case Authentisierung für asynchrone, sym. Administration, Schritt 2 293](#_Toc488231851)

[14.7.2.4 Gegenseitige Authentisierung mittels PACE für Sicherheitsmodule 293](#_Toc488231852)

[14.7.2.4.1 Use Case PACE für Sicherheitsmodule, Schritt 1b 293](#_Toc488231853)

[14.7.2.4.2 Use Case PACE für Sicherheitsmodule, Schritt 2b 294](#_Toc488231854)

[14.7.2.4.3 Use Case PACE für Sicherheitsmodule, Schritt 3b 295](#_Toc488231855)

[14.7.2.4.4 Use Case PACE für Sicherheitsmodule, Schritt 4b 295](#_Toc488231856)

[14.7.2.4.5 Use Case PACE für Sicherheitsmodule, Schritt 5b 296](#_Toc488231857)

[14.7.2.5 Authentisierung für asynchrone, asymmetrische Kartenadministration 297](#_Toc488231858)

[14.7.2.5.1 Use Case Authentisierung für asynchrone, asym. Administration, Schritt 1 297](#_Toc488231859)

[14.7.2.5.2 Use Case Authentisierung für asynchrone, asym. Administration, Schritt 2 297](#_Toc488231860)

[14.7.2.6 Antwort der Karte auf generelle Authentisierung 297](#_Toc488231861)

[14.7.2.7 Kommandoabarbeitung innerhalb der Karte 298](#_Toc488231862)

[14.7.3 Get Security Status Key 309](#_Toc488231863)

[14.7.3.1 Use Case Auslesen Sicherheitsstatus symmetrischer Schlüssels, Option\_DES 309](#_Toc488231864)

[14.7.3.2 Use Case Auslesen des Sicherheitsstatus einer Rolle, Option\_RSA\_CVC 310](#_Toc488231865)

[14.7.3.3 Use Case Auslesen des Sicherheitsstatus einer Bitliste 310](#_Toc488231866)

[14.7.3.4 Antwort der Karte auf Auslesen Sicherheitsstatus eines Schlüssels 311](#_Toc488231867)

[14.7.3.5 Kommandoabarbeitung innerhalb der Karte 311](#_Toc488231868)

[14.7.4 Internal Authenticate 312](#_Toc488231869)

[14.7.4.1 Use Case interne Authentisierung 312](#_Toc488231870)

[14.7.4.2 Antwort der Karte auf interne Authentisierung 313](#_Toc488231871)

[14.7.4.3 Kommandoabarbeitung innerhalb der Karte 314](#_Toc488231872)

[14.8 Kryptoboxkommandos 317](#_Toc488231873)

[14.8.1 PSO Compute Cryptographic Checksum 317](#_Toc488231874)

[14.8.1.1 Use Case Berechnen einer kryptographischen Checksumme 317](#_Toc488231875)

[14.8.1.2 Antwort der Karte auf Berechnen einer kryptographischen Checksumme 318](#_Toc488231876)

[14.8.1.3 Kommandoabarbeitung innerhalb der Karte 319](#_Toc488231877)

[14.8.2 PSO Compute Digital Signature 320](#_Toc488231878)

[14.8.2.1 Use Case Signieren des Datenfeldes, ohne „message recovery“ 320](#_Toc488231879)

[14.8.2.2 Use Case Signieren des Datenfeldes, mit „message recovery“ 321](#_Toc488231880)

[14.8.2.3 Antwort der Karte auf Signieren von Daten 322](#_Toc488231881)

[14.8.2.4 Kommandoabarbeitung innerhalb der Karte 322](#_Toc488231882)

[14.8.3 PSO Decipher 324](#_Toc488231883)

[14.8.3.1 Use Case Entschlüsseln mittels RSA 324](#_Toc488231884)

[14.8.3.2 Use Case Entschlüsseln mittels ELC 324](#_Toc488231885)

[14.8.3.3 Use Case Entschlüsseln mittels symmetrischer Schlüssel 325](#_Toc488231886)

[14.8.3.4 Antwort der Karte auf Entschlüsseln von Daten 326](#_Toc488231887)

[14.8.3.5 Kommandoabarbeitung innerhalb der Karte 326](#_Toc488231888)

[14.8.4 PSO Encipher 328](#_Toc488231889)

[14.8.4.1 Use Case Verschlüsseln von Daten mittels übergebenem RSA-Schlüssel 329](#_Toc488231890)

[14.8.4.2 Use Case Verschlüsseln von Daten mittels übergebenem ELC-Schlüssel 330](#_Toc488231891)

[14.8.4.3 Use Case Verschlüsseln mittels gespeichertem RSA-Schlüssel 331](#_Toc488231892)

[14.8.4.4 Use Case Verschlüsseln mittels gespeichertem ELC-Schlüssel 332](#_Toc488231893)

[14.8.4.5 Use Case Verschlüsseln mittels symmetrischem Schlüssel 332](#_Toc488231894)

[14.8.4.6 Antwort der Karte auf Verschlüsseln von Daten 333](#_Toc488231895)

[14.8.4.7 Kommandoabarbeitung innerhalb der Karte 333](#_Toc488231896)

[14.8.5 PSO Hash 336](#_Toc488231897)

[14.8.6 PSO Transcipher 336](#_Toc488231898)

[14.8.6.1 Use Case Umschlüsseln von Daten mittels RSA-Schlüssel 336](#_Toc488231899)

[14.8.6.2 Use Case Umschlüsseln von Daten von RSA-Schlüssel nach ELC-Schlüssel 337](#_Toc488231900)

[14.8.6.3 Use Case Umschlüsseln von Daten mittels ELC 338](#_Toc488231901)

[14.8.6.4 Use Case Umschlüsseln von Daten von ELC-Schlüssel nach RSA-Schlüssel 340](#_Toc488231902)

[14.8.6.5 Antwort der Karte auf Umschlüsseln von Daten 341](#_Toc488231903)

[14.8.6.6 Kommandoabarbeitung innerhalb der Karte 341](#_Toc488231904)

[14.8.7 PSO Verify Certificate 344](#_Toc488231905)

[14.8.7.1 Use Case Import RSA-Schlüssels mittels Zertifikat, Option\_RSA\_CVC 344](#_Toc488231906)

[14.8.7.2 Use Case Import ELC-Schlüssels mittels Zertifikat 345](#_Toc488231907)

[14.8.7.3 Antwort der Karte auf Vergleich eines Benutzergeheimnisses 346](#_Toc488231908)

[14.8.7.4 Kommandoabarbeitung innerhalb der Karte 346](#_Toc488231909)

[14.8.8 PSO Verify Cryptographic Checksum 351](#_Toc488231910)

[14.8.8.1 Use Case Prüfung einer kryptographischen Checksumme 351](#_Toc488231911)

[14.8.8.2 Antwort der Karte auf Berechnen einer kryptographischen Checksumme 351](#_Toc488231912)

[14.8.8.3 Kommandoabarbeitung innerhalb der Karte 352](#_Toc488231913)

[14.8.9 PSO Verify Digital Signature 353](#_Toc488231914)

[14.8.9.1 Use Case Prüfen einer ELC-Signatur 353](#_Toc488231915)

[14.8.9.2 Antwort der Karte auf Prüfen einer digitalen Signatur 354](#_Toc488231916)

[14.8.9.3 Kommandoabarbeitung innerhalb der Karte 355](#_Toc488231917)

[14.9 Verschiedenes 355](#_Toc488231918)

[14.9.1 Envelope 355](#_Toc488231919)

[14.9.2 Fingerprint 355](#_Toc488231920)

[14.9.2.1 Use Case Fingerprint über das COS berechnen. 356](#_Toc488231921)

[14.9.2.2 Antwort der Karte auf Fingerprintberechnung 356](#_Toc488231922)

[14.9.2.3 Kommandoabarbeitung innerhalb der Karte 357](#_Toc488231923)

[14.9.3 Generate Asymmetric Key Pair 357](#_Toc488231924)

[14.9.3.1 Use Case Generierung, ohne Überschreiben, ohne Referenz, ohne Ausgabe 358](#_Toc488231925)

[14.9.3.2 Use Case Generierung, ohne Überschreiben, mit Referenz, ohne Ausgabe 358](#_Toc488231926)

[14.9.3.3 Use Case Generierung, ggf. Überschreiben, ohne Referenz, ohne Ausgabe 359](#_Toc488231927)

[14.9.3.4 Use Case Generierung, ggf. Überschreiben, mit Referenz, ohne Ausgabe 359](#_Toc488231928)

[14.9.3.5 Use Case Auslesen vorhandener Schlüssel, ohne Referenz 360](#_Toc488231929)

[14.9.3.6 Use Case Auslesen vorhandener Schlüssel, mit Referenz 360](#_Toc488231930)

[14.9.3.7 Use Case Generierung, ohne Überschreiben, ohne Referenz, mit Ausgabe 361](#_Toc488231931)

[14.9.3.8 Use Case Generierung, ohne Überschreiben, mit Referenz, mit Ausgabe 361](#_Toc488231932)

[14.9.3.9 Use Case Generierung, ggf. Überschreiben, ohne Referenz, mit Ausgabe 362](#_Toc488231933)

[14.9.3.10 Use Case Generierung, ggf. Überschreiben, mit Referenz, mit Ausgabe 363](#_Toc488231934)

[14.9.3.11 Zusammenfassung der Generate Asymmetric Key Pair-Kommando-Varianten 363](#_Toc488231935)

[14.9.3.12 Antwort der Karte auf Schlüsselgenerierung 364](#_Toc488231936)

[14.9.3.13 Kommandoabarbeitung innerhalb der Karte 364](#_Toc488231937)

[14.9.4 Get Challenge 366](#_Toc488231938)

[14.9.4.1 Use Case Zufallszahl für DES oder RSA Authentisierung 366](#_Toc488231939)

[14.9.4.2 Use Case Zufallszahl für AES oder ELC Authentisierung 367](#_Toc488231940)

[14.9.4.3 Antwort der Karte auf Erzeugen einer Zufallszahl 367](#_Toc488231941)

[14.9.4.4 Kommandoabarbeitung innerhalb der Karte 368](#_Toc488231942)

[14.9.5 Get Random 368](#_Toc488231943)

[14.9.5.1 Use Case Erzeugen kryptographisch sicherer Zufallszahl 368](#_Toc488231944)

[14.9.5.2 Antwort der Karte Erzeugen kryptographisch sichere Zufallszahl 369](#_Toc488231945)

[14.9.5.3 Kommandoabarbeitung innerhalb der Karte 369](#_Toc488231946)

[14.9.6 Get Response 370](#_Toc488231947)

[14.9.7 List Public Key 370](#_Toc488231948)

[14.9.7.1 Use Case Auslesen der Liste öffentlicher Schlüsselobjekte 370](#_Toc488231949)

[14.9.7.2 Antwort der Karte auf Auslesen einer Schlüsselliste 371](#_Toc488231950)

[14.9.7.3 Kommandoabarbeitung innerhalb der Karte 371](#_Toc488231951)

[14.9.8 Manage Channel 373](#_Toc488231952)

[14.9.8.1 Use Case Öffnen eines logischen Kanals 373](#_Toc488231953)

[14.9.8.2 Use Case Schließen eines logischen Kanals 374](#_Toc488231954)

[14.9.8.3 Use Case Zurücksetzen eines logischen Kanals 374](#_Toc488231955)

[14.9.8.4 Use Case logischer Reset der Applikationsebene 375](#_Toc488231956)

[14.9.8.5 Antwort der Karte auf Kanalmanagementoperationen 375](#_Toc488231957)

[14.9.8.6 Kommandoabarbeitung innerhalb der Karte 375](#_Toc488231958)

[14.9.9 Manage Security Environment 377](#_Toc488231959)

[14.9.9.1 Use Case Ändern des SE-Identifiers 377](#_Toc488231960)

[14.9.9.2 Use Case Schlüsselauswahl zur internen, symmetrischen Authentisierung 378](#_Toc488231961)

[14.9.9.3 Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung 379](#_Toc488231962)

[14.9.9.4 Use Case Schlüsselauswahl zur externen, symmetrischen Authentisierung 379](#_Toc488231963)

[14.9.9.5 Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung 380](#_Toc488231964)

[14.9.9.6 Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung 381](#_Toc488231965)

[14.9.9.7 Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe 382](#_Toc488231966)

[14.9.9.8 Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe 383](#_Toc488231967)

[14.9.9.9 Use Case Schlüsselauswahl für Signierschlüssel 384](#_Toc488231968)

[14.9.9.10 Use Case Schlüsselauswahl zum Prüfen von CV–Zertifikaten 385](#_Toc488231969)

[14.9.9.11 Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung 385](#_Toc488231970)

[14.9.9.12 Use Case Schlüsselauswahl für Verschlüsselung 386](#_Toc488231971)

[14.9.9.13 Antwort der Karte auf Management des Security Environments 387](#_Toc488231972)

[14.9.9.14 Kommandoabarbeitung innerhalb der Karte 388](#_Toc488231973)

[15 Authentisierungsprotokolle (normativ) 390](#_Toc488231974)

[15.1 Externe Authentisierung 391](#_Toc488231975)

[15.1.1 Externe Authentisierung mittels symmetrischer Schlüssel 391](#_Toc488231976)

[15.1.2 RSA, asymmetrische Rollenauthentisierung, Option\_RSA\_CVC 392](#_Toc488231977)

[15.1.3 ELC, asymmetrische Berechtigungsnachweis 392](#_Toc488231978)

[15.2 Interne Authentisierung 392](#_Toc488231979)

[15.3 Card-2-Card-Authentisierung ohne Sessionkey-Aushandlung 393](#_Toc488231980)

[15.4 Aushandlung von Sessionkey 393](#_Toc488231981)

[15.4.1 Sessionkeys mittels symmetrischer Authentisierungsobjekte 393](#_Toc488231982)

[15.4.2 Sessionkeys mittels symmetrischer Kartenverbindungsobjekte 394](#_Toc488231983)

[15.4.3 Sessionkeyaushandlung mittels RSA-Schlüssel, Option\_DES 396](#_Toc488231984)

[15.4.4 Sessionkeys mittels ELC-Schlüssel 397](#_Toc488231985)

[15.5 Statisches Secure Messaging 400](#_Toc488231986)

[16 Verschiedenes (normativ) 402](#_Toc488231987)

[16.1 Identifier 402](#_Toc488231988)

[16.2 Codierungen für Trailer 405](#_Toc488231989)

[17 – Anhang A Hinweise zur Sicherheitsevaluierung (informativ) 407](#_Toc488231990)

[18 Anhang B – Vorgaben zur Performanz 408](#_Toc488231991)

[18.1 Einführung (informativ) 408](#_Toc488231992)

[18.2 Messaufbau (normativ) 408](#_Toc488231993)

[18.3 Anforderungen an die Steuersoftware (normativ) 409](#_Toc488231994)

[18.4 Anforderungen an das Interface Device (IFD) (normativ) 409](#_Toc488231995)

[18.4.1 Anforderungen an das IFD bezüglich T=1 409](#_Toc488231996)

[18.4.2 Anforderungen an das IFD für [ISO/IEC 7816-12] Datenübertragung 410](#_Toc488231997)

[18.4.3 Anforderungen an das IFD bezüglich kontaktloser Datenübertragung 410](#_Toc488231998)

[18.5 Allgemeines (normativ) 410](#_Toc488231999)

[18.5.1 Normale Zeitmessung 410](#_Toc488232000)

[18.5.1.1 Normale Zeitmessung für das Übertragungsprotokoll T=1 410](#_Toc488232001)

[18.5.1.2 Normale Zeitmessung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12] 411](#_Toc488232002)

[18.5.1.3 Normale Zeitmessung für die kontaktlose Datenübertragung 411](#_Toc488232003)

[18.5.2 Reguläre Aktivierung der Smartcard 412](#_Toc488232004)

[18.5.2.1 Reguläre Aktivierung für das Übertragungsprotokoll T=1 412](#_Toc488232005)

[18.5.2.2 Reguläre Aktivierung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12] 412](#_Toc488232006)

[18.5.2.3 Reguläre Aktivierung für die kontaktlose Datenübertragung 412](#_Toc488232007)

[18.5.3 Punkteermittlung 412](#_Toc488232008)

[18.5.4 Gesamtbewertung 413](#_Toc488232009)

[A.1 Übertragungsgeschwindigkeit 417](#_Toc488232010)

[A.1.1 Übertragungsgeschwindigkeit für das Übertragungsprotokoll T=1 417](#_Toc488232011)

[A.1.2 Übertragungsgeschwindigkeit für das Protokoll [ISO/IEC 7816-12] 418](#_Toc488232012)

[A.1.3 Übertragungsgeschwindigkeit für kontaktlose Datenübertragung 418](#_Toc488232013)

[A.2 Startsequenz für das Übertragungsprotokoll T=1 418](#_Toc488232014)

[A.3 Messverfahren für Einzelkommandos (normativ) 419](#_Toc488232015)

[A.3.1 Activate, Deactivate, Delete, Load Application, Terminate 419](#_Toc488232016)

[A.3.2 Select Datei 424](#_Toc488232017)

[A.3.3 Fingerprint 425](#_Toc488232018)

[A.3.4 Terminate Card Usage 425](#_Toc488232019)

[A.3.5 Set Logical Eof, Write Binary 426](#_Toc488232020)

[A.3.6 Erase Binary, Update Binary 427](#_Toc488232021)

[A.3.7 Read Binary 429](#_Toc488232022)

[A.3.8 Rekord orientierte Kommandos 430](#_Toc488232023)

[A.3.9 Search Record 432](#_Toc488232024)

[A.3.10 Symmetrische Sessionkeyaushandlung für Secure Messaging 433](#_Toc488232025)

[A.3.11 Schlüsselimport und asymmetrische Authentisierungsprotokolle 434](#_Toc488232026)

[A.3.11.1 ELC 256 434](#_Toc488232027)

[A.3.11.2 ELC 384 436](#_Toc488232028)

[A.3.11.3 ELC 512 438](#_Toc488232029)

[A.3.11.4 RSA 2048 440](#_Toc488232030)

[A.3.11.5 Testablauf Schlüsselimport und asymmetrische Authentisierung 440](#_Toc488232031)

[A.3.12 Internal Authenticate zur Rollenauthentisierung 443](#_Toc488232032)

[A.3.13 PSO Compute Digital Signature mittels signPSS 444](#_Toc488232033)

[A.3.14 Signaturerzeugung und –verifikation mittles signECDSA 445](#_Toc488232034)

[A.3.15 PSO Encipher und PSO Decipher mittels rsaDecipherOaep 446](#_Toc488232035)

[A.3.16 PSO Encipher und PSO Decipher mittels elcSharedSecretCalculation 448](#_Toc488232036)

[A.3.17 Selektieren von Ordnern und Logical Channel Reset 449](#_Toc488232037)

[A.3.18 Manage Security Environment 450](#_Toc488232038)

[A.3.19 General Authenticate, PACE 451](#_Toc488232039)

[A.3.20 Symmetrische Sessionkeyaushandlung für Trusted Channel 451](#_Toc488232040)

[A.3.21 Sessionkeynutzung im Trusted Channel 453](#_Toc488232041)

[A.3.22 Get Random 455](#_Toc488232042)

[A.3.23 Öffnen und Schließen logischer Kanäle 456](#_Toc488232043)

[A.4 Kartenkonfiguration für Performanztests (normativ) 457](#_Toc488232044)

[A.4.1 Attribute des Objektsystems 457](#_Toc488232045)

[A.4.1.1 Answer To Reset 457](#_Toc488232046)

[A.4.2 Allgemeine Festlegungen zu Attributstabellen 458](#_Toc488232047)

[A.4.3 *Root*, die Wurzelapplikation 459](#_Toc488232048)

[A.4.3.1 / MF / EF.ATR 459](#_Toc488232049)

[A.4.3.2 / MF / EF.DIR 460](#_Toc488232050)

[A.4.3.3 / MF / EF.GDO 460](#_Toc488232051)

[A.4.4 Anwendung für Authentisierungsprotokolle, DF.Auth 461](#_Toc488232052)

[A.4.4.1 / MF/ DF.Auth / PrK.Auth\_ELC256 462](#_Toc488232053)

[A.4.4.2 / MF/ DF.Auth / PrK.Auth\_ELC384 462](#_Toc488232054)

[A.4.4.3 / MF/ DF.Auth / PrK.Auth\_ELC512 462](#_Toc488232055)

[A.4.4.4 / MF/ DF.Auth / PrK.Auth\_RSA2048 463](#_Toc488232056)

[A.4.4.5 / MF/ DF.Auth / PuK.RCA\_ELC256 463](#_Toc488232057)

[A.4.4.6 / MF/ DF.Auth / PuK.RCA\_ELC384 463](#_Toc488232058)

[A.4.4.7 / MF/ DF.Auth / PuK.RCA\_ELC512 464](#_Toc488232059)

[A.4.4.8 / MF/ DF.Auth / PuK.RCA\_RSA2048 464](#_Toc488232060)

[A.4.4.9 / MF / DF.Auth / SK.AES128 464](#_Toc488232061)

[A.4.4.10 / MF / DF.Auth / SK.AES192 465](#_Toc488232062)

[A.4.4.11 / MF / DF.Auth / SK.AES256 465](#_Toc488232063)

[A.4.4.12 / MF / DF.Auth / TC.AES128 465](#_Toc488232064)

[A.4.4.13 / MF / DF.Auth / TC.AES192 466](#_Toc488232065)

[A.4.4.14 / MF / DF.Auth / TC.AES256 466](#_Toc488232066)

[A.4.5 Anwendung für IAS Services, DF.IAS 466](#_Toc488232067)

[A.4.5.1 / MF/ DF.IAS / PrK.X509\_ELC256 467](#_Toc488232068)

[A.4.5.2 / MF/ DF.IAS / PrK.X509\_ELC384 467](#_Toc488232069)

[A.4.5.3 / MF/ DF.IAS / PrK.X509\_ELC512 468](#_Toc488232070)

[A.4.5.4 / MF/ DF.IAS / PrK.X509\_RSA2048 468](#_Toc488232071)

[A.4.5.5 / MF/ DF.IAS / PrK.X509\_RSA3072 469](#_Toc488232072)

[A.4.6 Anwendung für LCS Use Cases, DF.LCS 470](#_Toc488232073)

[A.4.6.1 / MF/ DF.LCS / CAN\_256 471](#_Toc488232074)

[A.4.6.2 / MF / DF.LCS / EF.LCS 471](#_Toc488232075)

[A.4.6.3 / MF / DF.LCS / PIN.LCS 471](#_Toc488232076)

[A.4.6.4 / MF / DF.LCS / PrK.LCS 472](#_Toc488232077)

[A.4.6.5 / MF / DF.LCS / PuK.LCS 472](#_Toc488232078)

[A.4.6.6 / MF / DF.LCS / SK.LCS 473](#_Toc488232079)

[18.5.5 Anwendung für Select EF Use Cases, DF.SelectEF 473](#_Toc488232080)

[A.4.6.7 / MF / DF.SelectEF / EF.xx 473](#_Toc488232081)

[18.5.6 Anwendung für rekordorientierte Dateioperationen, DF.strukturiert 474](#_Toc488232082)

[18.5.6.1 / MF / DF.strukturiert / EF.strukturiert 474](#_Toc488232083)

[18.5.7 Anwendung für transparente Dateioperationen, DF.transparent 475](#_Toc488232084)

[A.4.6.8 / MF / DF.transparent / EF.transparent 475](#_Toc488232085)

[Anhang B Anhang C – Domainparameter elliptischer Kurven (informativ) 476](#_Toc488232086)

[18.6 ansix9p256r1, OID = {1.2.840.10045.3.1.7} = ´2A8648CE3D030107´ 476](#_Toc488232087)

[18.7 ansix9p384r1, OID = {1.3.132.0.34} = ´2B81040022´ 476](#_Toc488232088)

[18.8 brainpoolP256r1, OID={1.3.36.3.3.2.8.1.1.7}=´2B2403030208010107´ 477](#_Toc488232089)

[18.9 brainpoolP384r1, OID={1.3.36.3.3.2.8.1.1.11}=´2B240303020801010B´ 477](#_Toc488232090)

[18.10 brainpoolP512r1, OID={1.3.36.3.3.2.8.1.1.13}=´2B240303020801010D´ 477](#_Toc488232091)

[19 Anhang D – Erläuterungen zu Wertebereichen (informativ) 478](#_Toc488232092)

[20 Anhang E – Verzeichnisse (informativ) 479](#_Toc488232093)

[20.1 Abkürzungen 479](#_Toc488232094)

[20.2 Glossar 480](#_Toc488232095)

[20.3 Abbildungsverzeichnis 480](#_Toc488232096)

[20.4 Tabellenverzeichnis 480](#_Toc488232097)

[20.5 Referenzierte Dokumente 489](#_Toc488232098)

[20.5.1 Dokumente der gematik 489](#_Toc488232099)

[20.5.2 Weitere Dokumente 490](#_Toc488232100)

[21 Anhang F – Asynchrone, gesicherte APDU-Sequenzen (informativ) 493](#_Toc488232101)

[21.1 Einleitung 493](#_Toc488232102)

[21.2 CMS-Kommunikationsmuster 493](#_Toc488232103)

[21.2.1 Synchrone Kommunikation zwischen CMS und Smartcard 493](#_Toc488232104)

[21.2.2 Asynchrone Kommunikation zwischen CMS und Smartcard 493](#_Toc488232105)

[21.3 Anforderungen an die asynchrone Kommunikation 494](#_Toc488232106)

[21.4 Lösungskonzept 495](#_Toc488232107)

[21.5 Kryptographie 497](#_Toc488232108)

[21.5.1 CMS Aktivitäten 497](#_Toc488232109)

[21.5.2 Puffer-Aktivitäten 498](#_Toc488232110)

[21.5.3 Smartcard-Aktivitäten 498](#_Toc488232111)

[21.6 Auswirkungen auf die Kommandoschnittstelle 499](#_Toc488232112)

[22 Anhang G – CV-Zertifikate für RSA-Schlüssel, Option\_RSA\_CVC 500](#_Toc488232113)

[23 Anhang F – CV-Zertifikate für ELC-Schlüssel (normativ) 501](#_Toc488232114)

[24 Anhang F – Speichern öffentlicher Schlüssel (informativ) 502](#_Toc488232115)

[24.1 Definitionen 502](#_Toc488232116)

[24.2 Perspektiven 503](#_Toc488232117)

# Einordnung des Dokuments

## Zielsetzung

Diese Spezifikation definiert die Anforderungen an die Funktionalität einer Betriebssystemplattform (COS COS-Plattform) für elektronische Karten im Gesundheitswesen (eGK, HBA, …), die internationalen Standards entsprechen und die internationale sowie europäische Interoperabilität sicherstellen.

Im Einzelnen werden auf der Basis von [ISO/IEC 7816-4], [ISO/IEC 7816-8] und [ISO/IEC 7816-9] Kommandos und Optionen beschrieben, die vom COS unterstützt werden müssen.

Der Aufbau dieses Dokumentes gliedert sich wie folgt:

* Kapitel 1 Einordnung des Dokuments enthält Aussagen zum Umgang mit diesem Dokument.
* Kapitel 2 Optionen wird in einer späteren Version Eingangsanforderungen enthalten, welche die Basis für die normativen Aussagen in späteren Kapiteln bilden werden.
* Kapitel 3 Systemüberblick (informativ) wird in einer späteren Version Grundlagen zu Betriebssystemen für Smartcards enthalten, die auf der Normenreihe ISO/IEC 7816 basieren.
* Kapitel 4 Lebenszyklus von Karte und Applikation (informativ) grenzt den Gültigkeitsbereich der Spezifikation aus zeitlicher Sicht ab.
* Kapitel 5 Datentypen und Datenkonvertierung definiert einige grundlegende Datentypen, welche die normativen Beschreibungen in späteren Kapiteln vereinfachen.
* Kapitel 6 Kryptographische Algorithmen (normativ) definiert einige grundlegende kryptographische Funktionen, welche die normativen Beschreibungen in späteren Kapiteln vereinfachen.
* Kapitel 7 CV-Zertifikat beschreibt Zertifikate, welche zu nutzen sind, um öffentliche Schlüssel in eine Smartcard zu importieren.
* Kapitel 8 Objekte enthält eine Art Klassendiagramm in textueller Form. Die dort definierten Objekte und Attribute vereinfachen die normativen Beschreibungen in nachfolgenden Kapiteln.
* Kapitel 9 Objektsystem (normativ) beschreibt, wie sich Informationen persistent auf einer Smartcard hierarchisch speichern lassen.
* Kapitel 10 Zugriffskontrolle (normativ) beschreibt den Schutz von Informationen auf einer Smartcard vor unberechtigtem Zugriff.
* Kapitel 11 Kommunikation (normativ) beschreibt, wie die Smartcard Informationen mit anderen Systemen austauscht.
* Kapitel 12 Kanalkontext (normativ) beschreibt die Informationen, welche volatil und kanalspezifisch (zu logischen Kanälen siehe auch [ISO/IEC 7816-4]) von der Smartcard gespeichert werden.
* Kapitel 13 Gesicherte Kommunikation (normativ) beschreibt, wie die Smartcard Informationen kryptographisch geschützt mit anderen Systemen austauscht.
* Kapitel 14 Kommandos (normativ) enthält normative Aussagen zu Kommandos, welche an eine Smartcard geschickt werden und normative Aussagen, wie diese Kommandos von der Smartcard zu bearbeiten sind. Im Wesentlichen ist dies das wichtigste Kapitel des Dokumentes, weil es die äußere Sichtweise auf das Verhalten der Smartcard an der elektrischen Schnittstelle am umfassendsten beschreibt.
* Kapitel 15 Authentisierungsprotokolle (normativ) beschreibt Sequenzen, die aus mehr als einem Kommando bestehen.
* Kapitel 16 Verschiedenes (normativ) spezifiziert konkrete Werte für eine Reihe von Platzhaltern.

Das Dokument ist „bottom up“ aufgebaut, das bedeutet, Artefakte werden zunächst beschrieben und definiert, bevor sie verwendet werden. Für eine „top down“ Herangehensweise empfiehlt es sich, mit Kapitel 14 zu beginnen. Dort werden, wenn möglich, Verweise auf andere Kapitel gesetzt, wenn Dinge dort ausführlicher beschrieben werden. Wegen der besonderen Bedeutung des Kapitels 14 wird dessen Aufbau im Folgenden näher beleuchtet:

Kapitel 14 enthält alle in der Normenreihe ISO/IEC 7816 standardisierten Kommandos. Der besseren Übersichtlichkeit halber ist Kapitel 14 unterteilt in die Abschnitte Management des Objektsystems, Zugriff auf Daten in transparenten EF, Zugriff auf strukturierte Daten, Benutzerverifikation, Komponentenauthentisierung, Kryptoboxkommandos und Verschiedenes. Jeder Abschnitt enthält eine Reihe von Unterabschnitten mit Kommandos in alphabetischer Reihenfolge.

## Zielgruppe

Das Dokument richtet sich an Hersteller von Smartcard-Betriebssystemen und an Hersteller von Anwendungen, welche unmittelbar mit einer Smartcard kommunizieren.

## Geltungsbereich

Dieses Dokument enthält normative Festlegungen zur Telematikinfrastruktur des deutschen Gesundheitswesens. Der Gültigkeitszeitraum der vorliegenden Version und deren Anwendung in Zulassungs- oder Abnahmeverfahren wird durch die gematik GmbH in gesonderten Dokumenten (z.B. Dokumentenlandkarte, Produkttypsteckbrief, Leistungsbeschreibung) festgelegt und bekannt gegeben.

Der Inhalt des Dokumentes ist verbindlich für die Erstellung elektronischer Karten im Gesundheitswesen.

**Schutzrechts-/Patentrechtshinweis**

*Die nachfolgende Spezifikation ist von der gematik allein unter technischen Gesichtspunkten erstellt worden. Im Einzelfall kann nicht ausgeschlossen werden, dass die Implementierung der Spezifikation in technische Schutzrechte Dritter eingreift. Es ist allein Sache des Anbieters oder Herstellers, durch geeignete Maßnahmen dafür Sorge zu tragen, dass von ihm aufgrund der Spezifikation angebotene Produkte und/oder Leistungen nicht gegen Schutzrechte Dritter verstoßen und sich ggf. die erforderlichen Erlaubnisse/Lizenzen von den betroffenen Schutzrechtsinhabern einzuholen. Die gematik GmbH übernimmt insofern keinerlei Gewährleistungen.*

## Abgrenzung des Dokuments

Dieses Dokument spezifiziert das Verhalten an der elektrischen Schnittstelle zu einem Smartcard-Betriebssystem (Card Operating System, COS). Dieses Dokument spezifiziert NICHT die Architektur des COS. Der einfacheren Darstellung wegen, wird in diesem Dokument von einer modularen Aufteilung des COS ausgegangen. Die hier beschriebene Aufteilung ist nicht verpflichtend. Es wird aber empfohlen, sich an dieser Aufteilung zu orientieren, weil bei künftigen Ergänzungen und Erweiterungen die hier beschriebene Aufteilung zu Grunde gelegt wird.

Die Konfiguration einer Smartcard, also die Festlegung, welche Applikationen, Ordner, Dateien, Schlüssel und Passwörter auf einer Versichertenkarte zu finden sind, ist nicht Gegenstand dieses Dokumentes. Diese finden sich den kartenspezifischen Festlegungen zum Objektsystem (z. B. in [gemSpec\_eGK\_ObjSys], [gemSpec\_HBA\_ObjSys], [gemSpec\_SMC-B\_ObjSys]).

In Absprache mit den Verantwortlichen des Dokumentes [gemSpec\_Krypt] werden in diesem Dokument bewusst Redundanzen zum vorgenannten Dokument akzeptiert. Trotzdem ist [gemSpec\_Krypt] relevant für eine konkrete Karte, da dort, anders als in diesem Dokument, normative Vorgaben für die Nutzungsdauer gewisser kryptographischer Verfahren getroffen werden, die hier beschrieben werden.

## Methodik

### Nomenklatur der Präfixe

Tabelle 1: Präfixe, die auf Vielfachen von Zehnerpotenzen beruhen:

| **Name** | **Symbol** | **Wert gemäß SI** | **nächstliegende Zweierpotenz** |
| --- | --- | --- | --- |
| kilo | k | 103 = 1.000 | 210 = 1.024 |
| mega | M | 106 = 1.000.000 | 220 = 1.048.576 |
| giga | G | 109 = 1.000.000.000 | 230 = 1.073.741.824 |
| tera | T | 1012 = 1.000.000.000.000 | 240 = 1.099.511.627.776 |
| peta | P | 1015 = 1.000.000.000.000.000 | 250 = 1.125.899.906.842.624 |
| exa | E | 1018 | 260 |
| zetta | Z | 1021 | 270 |
| yotta | Y | 1024 | 280 |

Die folgende Tabelle basiert auf [BinPrefix].

Tabelle 2: Präfixe, die auf Vielfachen von Zweierpotenzen beruhen:

|  |  |  |
| --- | --- | --- |
| **Name** | **Symbol** | **Wert** |
| kibi | Ki | 210 = 10241 = 1.024 |
| mebi | Mi | 220 = 10242 = 1.048.576 |
| gibi | Gi | 230 = 10243 = 1.073.741.824 |
| tebi | Ti | 240 = 10244 = 1.099.511.627.776 |
| pebi | Pi | 250 = 10245 = 1.125.899.906.842.624 |
| exbi | Ei | 260 = 10246 = 1.152.921.504.606.846.976 |
| zebi | Zi | 270 = 10247 = 1.180.591.620.717.411.303.424 |
| yobi | Yi | 280 = 10248 = 1.208.925.819.614.629.174.706.176 |

1. Beispiel: 300 GB ≈ 279,4 GiB, sprich 300 Gigabyte sind ungefähr 279,4 Gibibyte

### Nomenklatur Verschiedenes

In diesem Dokument wird eine objektorientierte Sichtweise verfolgt. Dazu werden etwa die Artefakte Datei (EF in der Nomenklatur nach [ISO/IEC 7816-4]) oder Schlüssel als Objekte aufgefasst und die Eigenschaften als Attribute des Objektes. Wenn Attribute eines Objektes angesprochen werden, dann wird die Notation *obj.attribute* verwendet. Falls das Attribut wieder ein Objekt mit weiteren Attributen ist, dann sind auch längere Bezeichnungen möglich.

|  |  |
| --- | --- |
| G1 | Abkürzung für Generation 1, bezeichnet die vorherige Version des Dokumentes, in der Regel ergänzt um den Zusatz „normativ“. |
| G2 | Abkürzung für Generation 2, bezeichnet diese Version des Dokumentes, in der Regel ergänzt um den Zusatz „normativ“. |
| ´1D´ | Hexadezimale Zahlen und Oktettstrings werden in Hochkommata eingeschlossen. |
| ´XX´ | Ein Oktett mit beliebigem Inhalt. Obwohl für das obere und untere Nibble dasselbe Symbol verwendet wird ist es möglich, dass die Nibble unterschiedlich sind. |
| ´XX…XX´ | Ein Oktettstring beliebiger Länge und beliebigen Inhalts. |
| x || y | Das Symbol || steht für die Konkatenierung von Oktettstrings oder Bitstrings  ´1234´   ||   ´5678´   =   ´12345678´ |
| y = x | Der Variablen y wird der Wert von x zugewiesen (Standardnotation in gängigen Programmiersprachen). |

### Normative und informative Abschnitte

Abschnitte mit normativen Inhalten tragen hinter der Kapitelüberschrift den Hinweis:

(normativ)

Generell gilt, dass lediglich die Gliederungen, welche durch eine Nummer (N4711) gekennzeichnet sind, zulassungsrelevante Eigenschaften enthalten SOLLEN und somit im Rahmen der Zulassung getestet werden SOLLEN. Falls dies in einem speziellen Fall nicht so ist, handelt es sich höchstwahrscheinlich um einen editorischen Fehler.

### Komponentenspezifische Anforderungen

Da es sich beim vorliegenden Dokument um die Spezifikation einer Schnittstelle zwischen mehreren Komponenten handelt, ist es unvermeidlich, dass dieses Dokument Anforderungen für jede der Komponenten enthalten mag. Die normativen Abschnitte tragen deshalb eine Kennzeichnung, auf welche Komponente sich die Anwendung primär bezieht. Dabei gelten natürlicherweise folgende Zusammenhänge:

* Alle normativen Anforderungen an die Komponenten K\_Anwendungs­spe­zifikation und K\_externeWelt sind auch normative Anforderungen an diejenigen Komponenten, die in den jeweiligen Mengenklammern genannt sind.
* Für eine in der Mengenklammer aufgeführte Komponente ist es zulässig, mehr zu unterstützen als durch K\_Anwendungsspezifikation oder K\_exter­neWelt gefordert.
* Für eine in der Mengenklammer aufgeführte Komponente ist es zulässig, die Unterstützung von Dingen abzulehnen, die durch K\_Anwendungsspezifikation oder K\_externeWelt nicht gefordert werden.

Die obigen Aussagen werden im Folgenden durch Beispiele verdeutlicht:

* In (N007.900) wird von K\_Anwendungsspezifikation {K\_Karte} gefordert, nur bestimmte *seIdentifier* zu verwenden, wenn die Anwendungsspezifikation für die Komponente K\_COS bestimmt ist. Damit ist auch die maximale Anzahl von möglichen Security Environments in der Anwendung beschränkt. Einer Anwendungsspezifikation ist es nicht erlaubt, mehr SEs zu verwenden. Für das COS der Komponente K\_COS bedeutet dies, dass es mindestens diese maximale Anzahl an SE zu unterstützen hat. Für das COS der Komponente K\_COS ist es sowohl zulässig, mehr SEs zu unterstützen als auch zusätzliche SEs abzulehnen.
* In den 14.3.2.1 und 14.3.2.2 werden im Zusammenhang mit dem Kommando Read Binary Anforderungen an K\_externeWelt {K\_Karte} gestellt. Für die externe Welt ist es unzulässig, andere Read Binary-Varianten zu verwenden. Für das COS bedeutet dies, dass es mindestens diese Varianten zu unterstützen hat. Für das COS ist es sowohl zulässig. mehr Read Binary-Varianten zu unterstützen als auch zusätzliche Read Binary-Varianten abzulehnen.

Tabelle 3: Liste der Komponenten, an welche dieses Dokument Anforderungen stellt

| **Komponente** | **Beschreibung** |
| --- | --- |
| K\_Anwendungs­spezifikation {…} | Instanz, welche eine Anwendung spezifiziert; damit gilt diese Anforderung auch für jede Anwendungsspezifikation, die für eine in der Mengenklammer genannte Komponente bestimmt ist |
| K\_COS | Betriebssystem einer Smartcard |
| K\_COS\_G1 | Betriebssystem einer Smartcard in der Generation 1, es ist denkbar, dass derartig gekennzeichnete Anforderungen in späteren Versionen dieses Dokumentes entfallen |
| K\_IC | Das IC einer Smartcard |
| K\_Karte | beliebiger Kartentyp, Oberbegriff für die Menge {eGK, HBA, …} |
| K\_externeWelt {…} | Instanz, welche Nachrichten generiert, um diese an eine in der Mengenklammer genannte Komponente zu senden |

### Verwendung von Schüsselworten

Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID in eckigen Klammern sowie die dem RFC 2119 [RFC2119] entsprechenden, in Großbuchstaben geschriebenen deutschen Schlüsselworte MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet. Abwandlungen von „MUSS“ zu „MÜSSEN“ etc. sind der Grammatik geschuldet.

Da im Beispielsatz „*Eine leere Liste DARF NICHT ein Element besitzen*.“ die Phrase „DARF NICHT“ semantisch irreführend wäre (wenn nicht ein, dann vielleicht zwei?), wird in diesem Dokument stattdessen „*Eine leere Liste DARF KEIN Element besitzen*.“ verwendet.

In diesem Dokument werden Aussagen mit dem Schlüsselwort KANN generell sowohl positiv als auch negativ formuliert. (N002.200) ist im Zusammenhang mit (N002.100)a dafür ein gutes Beispiel. In (N002.100)a wird eine normative Forderung erhoben, die offen lässt, ob zusätzliche Werte fakultativ verboten sind oder nicht. Diese Lücke wird durch (N002.200) geschlossen.

# Optionen (normativ)

Dieses Unterkapitel listet Funktionspakete auf, die nicht zwingend erforderlich sind für eine Zulassung des COS einer Generation 2 Smartcard.

1. (N000.020) K\_IC, Option\_USB\_Schnittstelle:
   1. Das IC und das COS einer Smartcard KÖNNEN die Option\_USB\_Schnittstelle unterstützen.
   2. Falls das IC und das COS einer Smartcard die Option\_USB\_Schnittstelle
      1. unterstützen, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_USB\_Schnittstelle gekennzeichnet sind.
      2. nicht unterstützen, dann DÜRFEN mit Option\_USB\_Schnittstelle gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
2. (N000.022) K\_IC, Option\_kontaktlose\_Schnittstelle:
   1. Das IC und das COS einer Smartcard KÖNNEN die Option\_kontaktlose\_Schnittstelle unterstützen.
   2. Falls das IC und das COS einer Smartcard die Option\_kontaktlose\_Schnittstelle
      1. unterstützen, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_kontaktlose\_Schnittstelle gekennzeichnet sind.
      2. nicht unterstützt, dann DÜRFEN mit Option\_kontaktlose\_Schnittstelle gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
3. (N000.024) K\_COS {K\_Karte}, Option\_logische\_Kanäle:
   1. Das COS KANN die Option\_logische\_Kanäle unterstützen.
   2. Falls das COS die Option\_logische\_Kanäle
      1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_logische\_Kanäle gekennzeichnet sind.
      2. nicht unterstützt, dann DÜRFEN mit Option\_logische\_Kanäle gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
4. (N000.026) K\_COS {K\_Karte}, Option\_Kryptobox:
   1. Das COS KANN die Option\_Kryptobox unterstützen.
   2. Falls das COS die Option\_Kryptobox
      1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_Kryptobox gekennzeichnet sind.
      2. nicht unterstützt, dann DÜRFEN mit Option\_Kryptobox gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
5. (N000.028) K\_COS {K\_Karte}, Option\_PACE\_PCD:
   1. Das COS KANN die Option\_PACE\_PCD unterstützen.
   2. Falls das COS die Option\_PACE\_PCD
      1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_PACE\_PCD gekennzeichnet sind.
      2. nicht unterstützt, dann DÜRFEN mit Option\_PACE\_PCD gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
6. (N000.030) K\_COS {Karte}, Option\_DES:
   1. Das COS KANN die Option\_DES unterstützen.
   2. Falls das COS die Option\_DES
      1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_DES und Option\_RSA\_CVC gekennzeichnet sind.
      2. nicht unterstützt, dann DÜRFEN mit Option\_DES gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
7. Die Option\_DES wird derzeit lediglich zur Implementierung von Karten der Generation 1 verwendet, also eGK G1 und eGK G1+. Die Option\_DES ist für Karten der Generation 2 und später irrelevant.
8. (N000.032) K\_C K\_COS {Karte}, Option\_RSA\_CVC:
   1. Das COS KANN die Option\_RSA\_CVC unterstützen.
   2. Falls das COS die Option\_RSA\_CVC
      1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option\_RSA\_CVC gekennzeichnet sind.
      2. nicht unterstützt, dann DÜRFEN mit Option\_RSA\_CVC gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.
9. Die Option\_RSA\_CVC wird derzeit zur Implementierung folgendere Karten benötigt:
   1. Karten der Generaion 1, also eGK G1 und eGK G1+,
   2. HBA und SMC-B der Generation 2 und Generation G2.1.

# Systemüberblick (informativ)

1. Smartcards sind sichere Datenspeicher.
   1. Smartcards speichern Daten in Dateien oder Rekords.
   2. Smartcards speichern personenbezogene Schlüssel für IAS-Services.
2. Smartcards kontrollieren den Zugriff mittels Zugriffsregeln.
   1. Ein personenbezogener Zugriff wird durch Passwörter ermöglicht.
   2. Ein rollenbezogener Zugriff wird durch Authentisierungsschlüssel ermöglicht.
   3. Eine sichere Kommunikation wird durch „Trusted Channel" ermöglicht.
3. Eine Smartcard kann sowohl als Start- als auch als Endpunkt eines Trusted Channels fungieren. D. h. diese Smartcard kann einerseits Sessionkeys im Applikationslayer für PSO-Kommandos verwenden, als auch im Secure Messaging Layer. Wo genau hängt von der algId bei der gegenseitigen Authentisierung ab.

# Lebenszyklus von Karte und Applikation (informativ)

In der Literatur finden sich verschiedene Beschreibungen für den Lebenszyklus von Karten. Dieses Kapitel stellt eine vereinfachte Sicht dar und legt dabei einen Gültigkeitsbereich für diese Spezifikation fest. Grob lässt sich der Lebenszyklus einer Karte in drei Phasen einteilen:

1. **Vorbereitungsphase:** Diese Phase umfasst aus Sicht der Produktion alle Schritte, die erforderlich sind, um eine Karte für die Nutzungsphase vorzubereiten. Dazu zählen im Wesentlichen die Entwicklung des Betriebssystems, dessen Test, Abnahme und gegebenenfalls auch Evaluierung und Zulassung. Entsprechende Chips werden anschließend produziert, initialisiert und personalisiert. Die Chips werden in einen Kartenkörper implantiert und an einen Kartennutzer ausgeliefert. Die Reihenfolge der Produktionsschritte weicht unter bestimmten Umständen von der genannten Reihenfolge ab und ist hier lediglich beispielhaft skizziert.   
   Dieses Dokument gilt nicht für die Vorbereitungsphase der Karte.   
   Die Vorbereitungsphase der Karte endet spätestens mit der Übergabe der Karte an einen Kartennutzer. Dann beginnt die Nutzungsphase der Karte.
2. **Nutzungsphase:** Diese Phase umfasst den elektrischen Gebrauch der Karte.   
   Dieses Dokument gilt für die Nutzungsphase der elektrischen Kartenschnittstelle.   
   Die Nutzungsphase der Karte endet, wenn sämtliche Business Use Cases irreversibel gesperrt sind, mithin also auch, wenn die Karte physikalisch zerstört wird.
3. **Terminierungsphase:** Befindet sich die Karte in der Terminierungsphase, dann sind typischerweise alle intendierten Nutzungen der Karte irreversibel gesperrt. In der Regel lassen sich also weder Daten auslesen noch speichern und es ist keine Benutzerverifikation und auch keine Komponentenauthentisierung mehr möglich. Dies ist erreichbar durch eine physikalische Zerstörung des Chips, oder etwa auch durch die Unterstützung des Kommandos Terminate Card Usage (siehe 14.2.7). Da nach Ausführung eines solchen Kommandos herstellerspezifisch noch gewisse Kommandos möglich sind (Select, Get Challenge, …) oder die Übertragungsschicht T=1 möglicherweise noch aktiv ist, ist es nicht möglich, hier von einer Karte zu sprechen, die völlig inaktiv ist.

Analog zu den Phasen einer Karte ist es möglich, auch für Applikationen oder deren Bestandteile (Dateien, Passwörter, Schlüssel, …) die Phasen Vorbereitung, Nutzung und Terminierung zu definieren. Die Aussagen zur physikalischen Zerstörung der Karte gehen dann über in ein Löschen der Applikation oder deren Bestandteile.

Dieses Dokument gilt nicht für die Vorbereitungsphase von Applikationen oder deren Bestandteile. Sie beschreibt lediglich den Zustand des Objektsystems in der Nutzungsphase.

Die Nutzungsphase einer Applikation oder eines Applikationsbestandteils beginnt, sobald sich ein derartiges Objekt, wie in der Spezifikation der Anwendung definiert, verwenden lässt. Die Nutzungsphase einer Applikation oder eines Applikationsbestandteils endet, wenn das entsprechende Objekt gelöscht wird.

# Datentypen und Datenkonvertierung (normativ)

Dieses Dokument verwendet die folgenden Datentypen äquivalent zu [BSI-TR-03111#3]:

1. Oktettstring (OS),
2. Bitstring (BS),
3. Integer (I),
4. Körperelement (Field Element FE) und
5. elliptischen Kurvenpunkt (ECP).

Definition: Das höchstwertige Bit (most significant bit, MSBit) eines Bitstrings ist das am weitesten links stehende.

Definition: Das niedrigstwertige Bit (least significant bit, LSBit) eines Bitstrings ist das am weitesten rechts stehende.

Definition: Das höchstwertige Oktett (most significant byte, MSByte) eines Oktettstrings ist das am weitesten links stehende.

Definition: Das niedrigstwertige Oktett (least significant byte, LSByte) eines Oktettstrings ist das am weitesten rechts stehende.

Dieses Dokument verwendet die folgenden Konvertierungsfunktionen äquivalent zum Dokument [BSI-TR-03111#3.1]:

1. Bitstring nach Oktettstring BS2OS,
2. Oktettstring nach Bitstring OS2BS,
3. Körperelement nach Oktettstring FE2OS,
4. Oktettstring nach Körperelement OS2FE.

## BitLength Anzahl Bit in einem Bitstring

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Bitstring mit beliebigem Inhalt und beliebiger Länge |
| Output: | *out* | Integer, Anzahl der Bits aus denen *in* besteht |
| Errors: | – | Keine |
| Notation: |  | out = BitLength( in ) |

Beispiele:

* 1. BitLength( ´´ ) = 0
  2. BitLength( ´0´ ) = 1, BitLength( ´1´ ) = 1,
  3. BitLength( ´00´ ) = 2, BitLength( ´01´ ) = 2,
  4. BitLength( ´10´ ) = 2, BitLength( ´11´ ) = 2, …

## OctetLength Anzahl Oktett in einem Oktettstring

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Oktettstring mit beliebigem Inhalt und beliebiger Länge, oder  nicht–negative ganze Zahl |
| Output: | *out* | Integer, Anzahl der Oktette aus denen *in* besteht, oder  Anzahl Oktette, die mindestens nötig sind, um eine nicht–negative ganze Zahl zu codieren. |
| Errors: | – | Keine |
| Notation: |  | out = OctetLength( in ) |

1. Beispiele:
   1. OctetLength( ´´ ) = 0
   2. OctetLength( ´0034´ ) = 2
   3. OctetLength( 0 ) = 1, weil die Zahl 0 in einem Oktett codierbar ist, 0 = ´00´.
   4. OctetLength( 127 ) = 1, weil 127 = ´7F`
   5. OctetLength( 255 ) = 1, weil 255 = ´FF´
   6. OctetLength( 256 ) = 2, weil 256 = ´0100´.
2. ACHTUNG: Der Oktettstring ´0000FFFF´ lässt sich als Repräsentant der Zahl 65535 interpretieren. Ohne führende Nullen lautet die hexadezimale Repräsentation der Zahl 65535=´FFFF´. Daraus folgt:   OctetLength( ´0000FFFF´ ) = 4, aber OctetLength(65535) = 2.

## I2OS Integer nach Oktettstring

Dieser Abschnitt beschreibt die Konvertierung einer nicht-negativen ganzen Zahl in einen Oktettstring. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *x* | Integer, nicht-negative ganze Zahl |
| *n* | Integer, Anzahl der Oktette in out, *n* darf null sein, dann ist *out* leer |
| Output: | *out* | Oktettstring der Länge *n* Oktett |
| Errors: | – | Keine, ACHTUNG: Im Unterschied zu [BSI-TR-03111#3.1.2] wird hier keine Fehlermeldung erzeugt, falls *x* größer gleich 256*n* ist |
| Notation: |  | *out* = I2OS( *x*, *n* ) |

1. In gewissen Grenzen ist dies die Umkehrfunktion zu (N000.200).
2. (N000.100) K\_COS   
   Das Prinzip ist, die nicht-negative ganze Zahl *x* als Ziffernfolge zur Basis 256 zu notieren und dann nur die niedrigwertigsten Ziffern für die Ausgabe zu verwenden:
   1. Schritt 1: *x* = 2560 *x*0 + 2561 *x*1 + 2562 *x*2 + … + 256*i* *xi* + …
   2. Schritt 2: *Mi* = *xi*.   
      Anmerkung: Jede Ziffer wird vorzeichenlos in einem Oktett codiert.
   3. Schritt 3: *out* = *Mn*–1   ||   *Mn*–2   ||   …   ||   M2   ||   *M*1   ||   *M*0.
3. Beispiele:
   1. I2OS(30010, 1) = ´3A´,
   2. I2OS(30010, 2) = ´753A´,
   3. I2OS(30010, 3) = ´00753A´.

## OS2I Oktettstring nach Integer

Dieser Abschnitt beschreibt die Konvertierung eines Oktettstrings in eine nicht–negative ganze Zahl. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Oktettstring beliebiger Länge und beliebigen Inhalts |
| Output: | *out* | Integer, nicht–negative ganze Zahl |
| Errors: | – | Keine |
| Notation: |  | *out* = OS2I( *in* ) |

1. In gewissen Grenzen ist dies die Umkehrfunktion zu (N000.100).
2. (N000.200) K\_COS   
   Das Prinzip ist, jedes Oktett als Ziffer zur Basis 256 einer nicht-negativen ganzen Zahl im Big-Endian-Format aufzufassen.
   1. Schritt 1: *n* = OctetLength( *in* )
   2. Falls *n*
      1. gleich null ist, dann ist *out* = 0.
      2. ungleich null ist, dann wähle *out* so, dass gilt I2OS( *out*, *n* ) = *in*.

## OS2P Oktettstring nach Punkt (uncompressed encoding)

Dieser Abschnitt beschreibt die Konvertierung eines Oktettstrings in einen Punkt auf einer elliptischen Kurve. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *PO* | Oktettstring, codiert einen Punkt auf einer elliptischen Kurve |
| *dP* | Domainparameter gemäß (N008.600) |
| Output: | *P* | Punkt auf einer elliptischen Kurve mit den Koordinaten *P* = ( *x*, *y* ) |
| Errors: | ERROR | Falls PO nicht im Format „uncompressed encoding“ vorliegt |
| Notation: |  | *P* = OS2P( *PO*, *dP* ) |

1. (N000.300) K\_COS   
   Die Decodierung von *PO* erfolgt gemäß [BSI-TR-03111#3.2.1]:
   1. Schritt 1: Falls OctetLength( *PO* ) ungleich (2 *dP*.*L* + 1) ist, dann gebe den Fehler „*ERROR*“ zurück und beende diesen Algorithmus.
   2. Schritt 2: Teile *PO* auf gemäß:   
        *PO* = *PC*   ||   *X*   ||   *Y*,   
        1 = OctetLength( *PC* ),   
        *dp.L* = OctetLength( *X* ) = OctetLength( *Y* ).
   3. Schritt 3: Falls *PC* ungleich ´04´ ist, dann gebe den Fehler „*ERROR*“ zurück und beende diesen Algorithmus.
   4. Schritt 4: *P* = ( *x*, *y* ) = ( OS2I( *X* ) mod *dP.p*, OS2I( *Y* ) mod *dP.p* ).
   5. Schritt 5: Falls *P* nicht auf der durch *dP* definierten Kurve liegt, dann gebe den Fehler „*ERROR*“ zurück und beende diesen Algorithmus.

## P2OS Endlicher Punkt nach Oktettstring

Dieser Abschnitt beschreibt die Konvertierung eines Punktes auf einer elliptischen Kurve in einen Oktettstring. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *P* | Punkt auf einer elliptischen Kurve mit den Koordinaten *P* = ( *x*, *y* ) |
| *n* | Integer, Anzahl Oktette pro Koordinate |
| Output: | *PO* | Oktettstring, codiert einen Punkt auf einer elliptischen Kurve |
| Errors: | – |  |
| Notation: |  | *PO* = P2OS( *P*, *n* ) |

1. (N000.400) K\_COS   
   Die Codierung von *P* erfolgt gemäß [BSI-TR-03111#3.2.1]:   
     *PO* = ´04´   ||   I2OS( *x*, *n* )   ||   I2OS( *y*, *n* ).

## Extrahiere führende Elemente

### Extrahiere führende Bits

Dieser Abschnitt beschreibt, wie aus einem Bitstring oder Oktettstring führende Bits extrahiert werden.

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Entweder Bitstring der Länge *s* Bit, oder Oktettstring der Länge *s* Bit |
| *n* | Integer, Anzahl der zu extrahierenden Bit |
| Output: | *out* | Bitstring der Länge *n* Bit |
| Errors: | – | Keine |
| Notation: |  | *out* = Extract\_MSBit( *in*, *n* ) |

1. (N000.500) K\_COS   
   Es gilt die Vorbedingung *n* kleiner gleich *s*.
2. (N000.600) K\_COS   
   Der Bitstring *out* enthält die *n* MSBit von *in*.

### Extrahiere führende Oktette

Dieser Abschnitt beschreibt, wie aus einem Oktettstring führende Oktette extrahiert werden.

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Oktettstring der Länge *s* Oktett |
| *n* | Integer, Anzahl der zu extrahierenden Elemente |
| Output: | *out* | Oktettstring der Länge *n* Oktett |
| Errors: | – | Keine |
| Notation: |  | out = Extract\_MSByte( *in*, *n* ) |

1. (N000.700) K\_COS   
   Es gilt die Vorbedingung *n* kleiner gleich *s*.
2. (N000.800) K\_COS   
   Der Oktettstring *out* enthält die *n* MSByte von *in*.

## PaddingIso

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Es wird gemäß [ISO/IEC 7816-4#10.2.3.1] Abschnitt „Sequential stage“ Spiegelstrich 2 gepadded. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Oktettstring mit beliebigem Inhalt und beliebiger Länge |
| *n* | Integer, gibt die Blocklänge von *out* in Oktett an |
| Output: | *out* | Oktettstring mit einer Anzahl Oktette, die Vielfaches von *n* ist |
| Errors: | – | Keine |
| Notation: |  | *out* = PaddingIso( *in*, *n* ) |

1. Dies ist die Umkehrfunktion zu (N001.000).
2. (N000.900) K\_COS   
   Führe folgende Aktion durch:
   1. Schritt 1: *out* = *in*   ||   ´80´.
   2. Schritt 2: Falls 0 = OctetLength( *out* ) mod *n*, dann gebe *out* zurück und beende den Algorithmus, sonst fahre mit Schritt 3 fort.
   3. Schritt 3: *out* = *out*   ||   ´00´.
   4. Schritt 4: Fahre mit Schritt 2 fort.

## TruncateIso

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *in* | Oktettstring mit beliebigem Inhalt und einer Anzahl Oktette, die Vielfaches von n ist |
| *n* | Integer, gibt die Blocklänge in Oktett an, auf die gepadded wurde |
| Output: | *out* | Oktettstring mit beliebigem Inhalt und beliebiger Länge |
| Errors: | *paddingError* | Die Länge von in ist kein Vielfaches von n  Es sind zu viele Paddingbits vorhanden |
| Notation: |  | *out* = TruncateIso( *in*, *n* ) |

1. Dies ist die Umkehrfunktion (N000.900).
2. (N001.000) K\_COS   
   Führe folgende Aktion durch:
   1. Schritt 1: *len* = OctetLength( *in* )   
      Falls *len* kein Vielfaches von *n* ist, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
   2. Schritt 2: Falls *len* gleich null ist, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
   3. Schritt 3: Falls LSByte von *in* den Wert ´00´ hat, dann setze   
        *in* = Extract\_MSByte( *in*, OctetLength( *in* ) – 1)   
      und fahre mit Schritt 2 fort.
   4. Schritt 4: Falls LSByte von *in* nicht den Wert ´80´ hat, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
   5. Schritt 5: *out* = Extract\_MSByte( *in*, OctetLength( *in* ) – 1)
   6. Schritt 6: Falls ( *len* – OctetLength( *out* ) ) größer als *n* ist, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.

## MGF Mask Generation Function

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | Z | Bitstring mit beliebigem Inhalt und beliebiger Länge |
| LN | Integer, gibt die Bitlänge von N an |
| i | Integer, Startwert der Iteration |
| Output: | N | Bitstring, Ergebnis der Mask Generation Function |
| Errors: | error | Gemäß [ISO/IEC 9796-2#C.3.2] Punkt 1 ist ein Fehler zu werfen, falls Z zu lang, oder LN zu groß ist.   1. Dieser Fehler ist für Smartcards derzeit nicht praxisrelevant, da sie erst dann auftreten, wenn Z oder N oberhalb mehrerer Gigabyte liegen. |
| Notation: |  | N = MGF( Z, LN , i ) |

1. Wenn diese Funktion mit i = 0 aufgerufen wird, dann ist das Ergebnis konform zu
   1. [ISO/IEC 9796-2#C], und
   2. [PKCS#1#B.2.1].
2. Wenn diese Funktion mit i = 1 aufgerufen wird, dann ist das Ergebnis konform zu
   1. [ANSI X9.63#5.6.3], falls dort der optionale Parameter SharedInfo leer ist, und
   2. [BSI-TR-03111#4.3.3], allerdings wird dort das Schlüsselmaterial anders aus N extrahiert.
3. (N001.100) K\_COS   
   Führe folgende Aktionen durch:
   1. Schritt 1: Setze *n* = ´´,     (Anmerkung: Leerer Oktettstring).
   2. Schritt 2: Setze *n* = *n* || SHA\_256( BS2OS( *Z* ) || I2OS( *i*, 4 ) ).
   3. Schritt 3: Falls die Hash-Operation mit einem Fehler terminierte, dann breche diesen Algorithmus mit der Fehlermeldung *error* ab.
   4. Schritt 4: Setze *i* = *i* + 1.
   5. Schritt 5: Falls i größer gleich 232 = ´1 0000 0000´ ist, dann breche diesen Algorithmus mit der Fehlermeldung „*error*“ ab.
   6. Schritt 6: Falls das Achtfache von OctetLength( *n* ) kleiner als *LN* ist, dann setze diesen Algorithmus mit Schritt 2 fort.
   7. Schritt 7: Setze *N* = Extract\_MSBit( *n*, *LN* ).

## RAND Zufälliger Oktettstring

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | n | Positive ganze Zahl, welche die Anzahl Oktette in out angibt |
| Output: | out | Zufälliger Oktettstring der Länge n Oktett |
| Errors: | – | Keine |
| Notation: |  | out = RAND( n ) |

1. (N001.200) K\_COS   
   Führe folgende Aktionen durch: Erzeuge einen Oktettstring *out* der Länge *n*, wobei jedes Bit von *out* zufällig erzeugt wird.

## ceiling Aufrunden einer reellen Zahl

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | x | beliebige positive oder negative reelle Zahl |
| Output: | n | kleinste ganze Zahl, die nicht kleiner ist als x |
| Errors: | – | Keine |
| Notation: |  | n = ceiling( x ) |

1. (N001.210) K\_COS   
   Zu x MUSS die kleinste ganze Zahl *n* bestimmt werden, die nicht kleiner ist als *x*, das heißt: ceiling( *x* ) = min {*n* є Z | *n* ≥ *x*}.

## floor Abrunden einer reellen Zahl

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | x | beliebige positive oder negative reelle Zahl |
| Output: | n | größte ganze Zahl, die nicht größer ist als x |
| Errors: | – | Keine |
| Notation: |  | n = floor( x ) |

1. (N001.220) K\_COS   
   Zu x MUSS die größte ganze Zahl *n* bestimmt werden, die nicht größer ist als *x*, das heißt: floor( *x* ) = max {*n* є Z | *n* ≤ *x*}.

# Kryptographische Algorithmen (normativ)

## Hash-Algorithmen

Ein Hash-Algorithmus errechnet zu einer beliebigen Folge von Bits, von (beinahe) unbegrenzter Länge, einen Bitstring fester Länge. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird ein Hash-Algorithmus als Funktion, wie in den folgenden Unterkapiteln gezeigt, verwendet.

### SHA-1

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring, der zu hashen ist |
| Output: | *H* | Oktettstring, Hash-Wert der Länge 20 Oktett = 160 Bit |
| Errors: | „M too long“ | Zu hashende Nachricht *M* enthält zu viele Bits. Dies ist der Fall wenn die Nachricht *M* länger ist als 264 bit = 2 EiB.   1. Dieser Fehler ist für Smartcards nicht praxisrelevant. Um eine Nachricht mit einer Länge von 264 Bit innerhalb von 10 Jahren zu hashen, wäre der Hash-Algorithmus mit einer Geschwindigkeit von 6,8 GiB pro Sekunde auszuführen. |
| Notation: |  | *H* = SHA\_1( *M* ) |

1. (N001.290) K\_COS   
   Das COS MUSS *H* gemäß [FIPS 180-4#6.1] aus *M* berechnen.

### SHA-256

|  |  |  |
| --- | --- | --- |
| Input: | M | Beliebiger Oktettstring, der zu hashen ist |
| Output: | H | Oktettstring, Hash-Wert der Länge 32 Oktett = 256 Bit |
| Errors: | „M too long“ | Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 264 bit = 2 EiB   1. Dieser Fehler ist für Smartcards nicht praxisrelevant. Um eine Nachricht mit einer Länge von 264 Bit innerhalb von 10 Jahren zu hashen, wäre der Hash-Algorithmus mit einer Geschwindigkeit von 6,8 GiB pro Sekunde auszuführen. |
| Notation: |  | H = SHA\_256( M ) |

1. (N001.300) K\_COS   
   Das COS MUSS *H* gemäß [FIPS 180-4#6.2] aus *M* berechnen.

### SHA-384

|  |  |  |
| --- | --- | --- |
| Input: | M | Beliebiger Oktettstring, der zu hashen ist |
| Output: | H | Oktettstring, Hash-Wert der Länge 48 Oktette = 384 bit |
| Errors: | „M too long“ | Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 2128 bit = 2125 Byte   1. Dieser Fehler ist für Smartcards weniger relevant, als der unter Hinweis (15): beschriebene. |
| Notation: |  | H = SHA\_384( M ) |

1. (N001.310) K\_COS   
   Das COS MUSS *H* gemäß [FIPS 180-4#6.5] aus *M* berechnen.

### SHA-512

|  |  |  |
| --- | --- | --- |
| Input: | M | Beliebiger Oktettstring, der zu hashen ist |
| Output: | H | Oktettstring, Hash-Wert der Länge 64 Oktette = 512 Bit |
| Errors: | „M too long“ | Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 2128 Bit = 2125 Byte   1. Dieser Fehler ist für Smartcards weniger relevant, als der unter Hinweis (15): beschriebene. |
| Notation: |  | H = SHA\_512( M ) |

1. (N001.320) K\_COS   
   Das COS MUSS *H* gemäß [FIPS 180-4#6.4] aus *M* berechnen.

## Schlüsselvereinbarung

In diesem Abschnitt wird eine Funktion beschrieben, die aus einem Eingabewert Schlüsselmaterial für symmetrische Algorithmen berechnet. Derartiges Schlüsselmaterial wird vorwiegend im Rahmen von sicherer Transportverschlüsselung eingesetzt. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Schlüsselvereinbarung als Funktion, wie in den folgenden Unterkapiteln gezeigt, verwendet.

### Verhalten für 3TDES-Schlüssel, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | KD | Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung. Prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts |
| Output: | *Kenc* | Oktettstring der Länge 24 Oktette, der als 3TDES-Schlüssel für Ver- und Entschlüsselung verwendet wird |
| *T1* | Nicht-negative ganze Zahl, die im Zusammenhang mit *Kenc* als Send Sequence Counter verwendet wird (siehe Funktionen in 6.7) |
| *Kmac* | Oktettstring der Länge 24 Oktette, der als 3TDES-Schlüssel für          MAC-Generierung und MAC-Prüfung verwendet wird |
| *T2* | Nicht-negative ganze Zahl, die im Zusammenhang mit *Kmac* als Send Sequence Counter verwendet (siehe (N032.800) und (N034.100)) |
| Errors: | – | Keine |
| Notation: |  | ( *Kenc*, *T1*, *Kmac*, *T2* ) = KeyDerivation\_3TDES( *KD* ) |

1. (N001.400) K\_COS\_G1, Option\_DES   
   Das COS MUSS ( *Kenc*, *T*1, *Kmac*, *T*2 ) gemäß [ANSI X9.63#5.6.3] wie folgt aus *KD* berechnen.
   1. Schritt 1: *km* = BS2OS( MGF( OS2BS( *KD* ), 512, 1 )).
   2. Schritt 2: Teile *km* so auf *Kenc*, *t*1, *Kmac* und *t*2 auf, dass gilt:
      1. OctetLength( *Kenc* ) = OctetLength( *Kmac* ) = 24
      2. OctetLength( *t*1 ) = OctetLength( *t*2 ) = 8
      3. *km* = *Kenc*   ||   *t*1   ||   *Kmac*   ||   *t*2
   3. Schritt 3: *T*1 = OS2I( *t*1 )   und   *T*2 = OS2I( *t*2 )

### Vereinbarung von AES-128-Schlüsseln

|  |  |  |
| --- | --- | --- |
| Input: | *KD* | Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung. Prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts |
| Output: | *Kenc* | Oktettstring der Länge 16 Oktette, der als AES-128-Schlüssel für Ver- und Entschlüsselung verwendet wird |
| *Kmac* | Oktettstring der Länge 16 Oktette, der als AES-128-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird |
| *T2* | Nicht-negative ganze Zahl, die im Zusammenhang mit *Kmac* als Send Sequence Counter verwendet wird (siehe (N032.800) und (N034.100)) |
| Errors: | – | keine |
| Notation: |  | ( *Kenc*, *Kmac*, *T2* ) = KeyDerivation\_AES128( *KD* ) |

1. (N001.500) K\_COS   
   Das COS MUSS (*Kenc*, *Kmac*, *T*2 ) analog zu [BSI-TR-03110-3#A.2.3.2] wie folgt aus *KD* berechnen:
   1. *Kenc* = Extract\_MSByte(SHA\_1(KD || ´00000001´), 16)
   2. *Kmac* = Extract\_MSByte(SHA\_1(KD || ´00000002´), 16)
   3. *T2* = 0

### Vereinbarung von AES-192-Schlüsseln

|  |  |  |
| --- | --- | --- |
| Input: | *KD* | Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung, prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts |
| Output: | *Kenc* | Oktettstring der Länge 24 Oktette, der als AES-192-Schlüssel für Ver- und Entschlüsselung verwendet wird |
| *Kmac* | Oktettstring der Länge 24 Oktette, der als AES-192-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird |
| *T2* | Nicht-negative ganze Zahl, die im Zusammenhang mit *Kmac* als Send Sequence Counter verwendet wird (siehe (N032.800) und (N034.100)) |
| Errors: | – | keine |
| Notation: |  | ( *Kenc*, *Kmac*, *T2* ) = KeyDerivation\_AES192( *KD* ) |

1. (N001.510) K\_COS   
   Das COS MUSS (*Kenc*, *Kmac*, *T*2 ) analog zu [BSI-TR-03110-3#A.2.3.2] wie folgt aus *KD* berechnen:
   1. *Kenc* = Extract\_MSByte(SHA\_256(KD || ´00000001´), 24)
   2. *Kmac* = Extract\_MSByte(SHA\_256(KD || ´00000002´), 24)
   3. *T2* = 0

### Vereinbarung von AES-256-Schlüsseln

|  |  |  |
| --- | --- | --- |
| Input: | *KD* | Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung, prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts |
| Output: | *Kenc* | Oktettstring der Länge 32 Oktette, der als AES-256-Schlüssel für Ver- und Entschlüsselung verwendet wird |
|  | *Kmac* | Oktettstring der Länge 32 Oktette, der als AES-256-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird |
|  | *T2* | Nicht-negative ganze Zahl, die im Zusammenhang mit *Kmac* als Send Sequence Counter verwendet wird (siehe (N032.800) und (N034.100)) |
| Errors: | – | keine |
| Notation: |  | ( *Kenc*, *Kmac*, *T2* ) = KeyDerivation\_AES256( *KD* ) |

1. (N001.520) K\_COS   
   Das COS MUSS (*Kenc*, *Kmac*, *T*2 ) analog zu [BSI-TR-03110-3#A.2.3.2] wie folgt aus *KD* berechnen:
   1. *Kenc* = SHA\_256(KD || ´00000001´)
   2. *Kmac* = SHA\_256(KD || ´00000002´)
   3. *T2* = 0

### Schlüsselableitung aus einer Card Access Number

|  |  |  |
| --- | --- | --- |
| Input: | *KD* | Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselableitung, prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts |
| *n* | natürliche Zahl, die bei der Schlüsselableitung berücksichtigt wird |
| *OID* | Algorithmuskennung mit einem Wert aus einer Menge in (N102.440) |
| Output: | *K* | Oktettstring, je nach OID mit einer Länge von 16 oder 24 oder 32 Oktette, der als AES-Schlüssel verwendet wird |
| Errors: | – | keine |
| Notation: |  | *K* = KDF( *KD*, *n*, *OID* ) |

1. (N001.530) K\_COS   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD   
   Das COS MUSS *K* analog zu [BSI-TR-03110-3#A.2.3] wie folgt aus *KD* und *OID* berechnen: Falls *OID* aus der Menge
   1. {id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128} ist, gilt:   
      *K* = Extract\_MSByte(SHA\_1( *KD* || I2OS(*n*, 4) ), 16)
   2. {id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192} ist, gilt:   
      *K* = Extract\_MSByte(SHA\_256( *KD* || I2OS(*n*, 4) ), 24)
   3. {id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256} ist, gilt:   
      *K* = SHA\_256( *KD* || I2OS(*n*, 4) )

## Symmetrischer Basisalgorithmus für Vertraulichkeit

Ein Verschlüsselungsalgorithmus berechnet zu einem beliebigen Oktettstring (plaintext) mit Hilfe eines geheimen Schlüssels ein Chiffrat (ciphertext). Ein entsprechender Entschlüsselungsalgorithmus berechnet zu einem Chiffrat (ciphertext) mit Hilfe desselben symmetrischen Schlüssels den ursprünglichen Oktettstring (plaintext). In diesem Kapitel wird lediglich die Bearbeitung eines Blocks mittels eines Blockverschlüsselungsalgorithmus behandelt. Die Behandlung von Inputdaten mit beliebiger Länge wird in 6.7 spezifiziert.

### Symmetrische Verschlüsselung eines Datenblocks, Option\_DES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Verschlüsselung als Funktion wie folgt verwendet:

#### Verschlüsselung mittels DES, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | Pj | beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Klartext |
| K | Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, der als Schlüssel verwendet wird |
| Output: | Cj | Oktettstring, verschlüsselte Daten der Länge 8 Oktett |
| Errors: | – | Keine |
| Notation: |  | Cj = DES\_ENC( K, Pj ) |

1. (N001.600) K\_COS\_G1, Option\_DES   
   Das COS MUSS *Cj* mittels *K* gemäß [ANSI X3.92] aus *Pj* berechnen.

#### Verschlüsselung mittels 3TDES, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | Pj | Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Klartext |
| K | Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. K setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: K = Ka   ||   Kb   ||   Kc |
| Output: | Cj | Oktettstring, verschlüsselte Daten der Länge 8 Oktett |
| Errors: | – | Keine |
| Notation: |  | Cj = 3TDES\_ENC( K, Pj ) |

1. (N001.700) K\_COS\_G1, Option\_DES   
   Das COS MUSS *Cj* mittels *K* wie folgt aus *Pj* berechnen:   
   *Cj*   =   DES\_ENC( *Kc*, DES\_DEC( *Kb*, DES\_ENC( *Ka*, *Pj* ) ) ).

### Symmetrische Entschlüsselung eines Datenblocks, Option\_DES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Entschlüsselung als Funktion wie folgt verwendet:

#### Entschlüsselung mittels DES, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | Cj | Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Chiffrat |
| K | Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, der als Schlüssel verwendet wird |
| Output: | Pj | Oktettstring, entschlüsselte Daten der Länge 8 Oktett |
| Errors: | – | Keine |
| Notation: |  | Pj = DES\_DEC( K, Cj ) |

1. (N001.800) K\_COS\_G1, Option\_DES   
   Das COS MUSS *Pj* mittels *K* gemäß [ANSI X3.92] aus *Cj* berechnen.

#### Entschlüsselung mittels 3TDES, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | Cj | Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Chiffrat |
| K | Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. K setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: K = Ka   ||   Kb   ||   Kc |
| Output: | Pj | Oktettstring, entschlüsselte Daten der Länge 8 Oktett |
| Errors: | – | Keine |
| Notation: |  | Pj = 3TDES\_DEC( K, Cj ) |

1. (N001.900) K\_COS\_G1, Option\_DES   
   Das COS MUSS *Pj* mittels *K* wie folgt aus *Cj* berechnen:   
   *Pi*   =   DES\_DEC( *Ka*, DES\_ENC( *Kb*, DES\_DEC( *Kc*, *Ci* ) ) ).

### Symmetrische Verschlüsselung eines Datenblocks, AES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Verschlüsselung als Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *Pj* | Beliebiger Oktettstring der Länge 16 Oktett = 128 Bit, Klartext |
| *K* | Beliebiger Oktettstring der Länge 16 oder 24 oder 32 Oktett, der als Schlüssel verwendet wird |
| Output: | *Cj* | Oktettstring, verschlüsselte Daten der Länge 16 Oktett |
| Errors: | – | Keine |
| Notation: |  | *Cj* = AES\_ENC( *K*, *Pj* ) |

1. (N002.000) K\_COS   
   Das COS MUSS *Cj* mittels *K* so aus *Pj* berechnen, dass sich derselbe funktionale Zusammenhang *fK*: *Pj* 🡪 *Cj* ergibt wie in [FIPS 197#Figure 5].

### Symmetrische Entschlüsselung eines Datenblocks, AES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Verschlüsselung als Funktion wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | Cj | Oktettstring, verschlüsselte Daten der Länge 16 Oktett |
| K | Beliebiger Oktettstring der Länge 16 oder 24 oder 32 Oktett, der als Schlüssel verwendet wird |
| Output: | Pj | Beliebiger Oktettstring der Länge 16 Oktett = 128 Bit, Klartext |
| Errors: | – | Keine |
| Notation: |  | Pj = AES\_DEC( K, Cj ) |

1. (N002.010) K\_COS   
   Das COS MUSS *Pj* mittels *K* so aus *Cj* berechnen, dass sich derselbe funktionale Zusammenhang *fK*: *Cj* 🡪 *Pj* ergibt, wie in [FIPS 197#Figure 12] oder [FIPS 197#Figure 15].

## Asymmetrischer Basisalgorithmus RSA

Als asymmetrischer Basisalgorithmus wird RSA verwendet. Bezüglich des mathema­tischen Hintergrundes wird an dieser Stelle lediglich auf [PKCS#1] verwiesen.

1. (N002.100) K\_COS   
   Das COS MUSS RSA–Schlüssel mit
   1. einer Moduluslänge *modulusLength* aus den Mengen
      1. {2048} Bit für private und öffentliche RSA‑Schlüssel unterstützen und
      2. {3072} Bit, falls es sich um einen privaten RSA‑Schlüssel handelt und
   2. öffentlichen Exponenten *e* aus folgendem Intervall unterstützen:   
      [216+1, 232–1] = [65.537, 4.294.967.295] = [´0001 0001´, ´FFFF FFFF´].
2. (N002.200) K\_COS   
   Das COS KANN weitere Moduluslängen
   1. unterstützen oder
   2. ablehnen.
3. (N002.300) K\_COS   
   Das COS KANN öffentliche Exponenten aus weiteren Intervallen
   1. unterstützen oder
   2. ablehnen.
4. (N002.400) K\_COS   
   Private RSA-Schlüssel, welche die in [BKryA#3.1] empfohlenen Schranken ε1 und ε2 verletzen, für die mithin NICHT gilt 0,1 < |*log*2(p) – *log*2 (q)| < 30,
   1. KÖNNEN vom COS akzeptiert werden.
   2. KÖNNEN vom COS abgelehnt werden.

Konform zu [PKCS#1#2] werden die RSA-Schlüsselparameter in diesem Dokument wie folgt dargestellt:

Tabelle 4: Liste der Schlüsselparameter eines RSA-Schlüssels

|  |  |
| --- | --- |
| **Parameter** | **Bedeutung** |
| *n* | RSA Modulus |
| *e* | RSA öffentlicher Exponent |
| *d* | RSA privater Exponent |

1. In diesem Dokument wird der Einfachheit halber nicht mit „chinese remainder theorem“ Parametern gearbeitet. Wegen der größeren Performanz wird aber empfohlen, innerhalb des COS für Operationen mit dem privaten Schlüssel „chinese remainder theorem“ Parameter zu verwenden.

## Asymmetrischer Basisalgorithmus elliptische Kurven

Als asymmetrischer Basisalgorithmus werden elliptische Kurven verwendet. Bezüglich des mathematischen Hintergrundes wird an dieser Stelle lediglich auf [BSI-TR-03111] verwiesen. Die Domainparameter der im folgenden genannten Kurven finden sich auch im Anhang C.

1. (N002.500) K\_COS   
   Das COS MUSS die folgenden elliptischen Kurven unterstützen:
   1. 256 Bit gemäß
      1. [RFC5639#3.4, brainpoolP256r1] (siehe C.3) und
      2. [ANSI X9.62#L.6.4.3, ansix9p256r1] (siehe C.1).
   2. 384 Bit gemäß
      1. [RFC5639#3.6, brainpoolP384r1] (siehe C.4) und
      2. [ANSI X9.62#L6.5.2, ansix9p384r1] (siehe C.2).
   3. 512 Bit gemäß [RFC5639#3.7, brainpoolP512r1] (siehe C.5).
2. (N002.609) K\_COS   
   Das COS KANN weitere elliptische Kurven
   1. unterstützen oder
   2. ablehnen.

## Datenauthentisierung

Im Rahmen einer Datenauthentisierung werden beliebigen Daten Informationen derart hinzugefügt, dass die Integrität und Authentizität der Daten überprüfbar ist.

### MAC-Generierung

Die MAC-Generierung ist eine Datenauthentisierung, welche auf einem symmetrischen Basisalgorithmus basiert. Dabei wird zu einem Oktettstring beliebigen Inhalts und Länge ein MAC berechnet, dessen Länge lediglich vom Algorithmus abhängt, nicht aber vom Oktettstring. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine MAC-Generierung als Funktion wie folgt verwendet:

#### Generierung Retail-MAC, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring beliebiger Länge für den ein MAC berechnet wird |
| *K* | Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von *K* beträgt 24 Oktett. *K* setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: *K* = Ka   ||   Kb   ||   Kc |
| Output: | *T* | Oktettstring, der zur Prüfung der Integrität und Authentizität von *M* verwendbar ist |
| Errors: | – | Keine |
| Notation: |  | *T* = CALCULATE\_Retail\_MAC( *K*, *M* ) |

1. (N002.700) K\_COS\_G1, Option\_DES   
   Das COS MUSS *T* mittels *K* aus *M* berechnen. Dabei werden folgende Schritte durchgeführt:
   1. Schritt 1: Berechne *X* = PaddingIso( *M*, 8 )
   2. Schritt 2: Teile *X* auf in Blöcke mit jeweils 8 Oktett *X* = *X*1 || *X*2 || … || *Xn*.
   3. Schritt 3: Setze *Y*0 = ´0000 0000 0000 0000´.
   4. Schritt 4: Berechne *Yi* = DES\_ENC( *Ka*, *Yi*–1 XOR *Xi* ) für *i* = 1, …, *n* – 1.
   5. Schritt 5: Berechne *T* = 3TDES\_ENC( *K*, *Yn*–1 XOR *Xn* )
2. Im Rahmen von Secure Messaging wird ein von „null“ verschiedener Initialisierungsvektor Y0 bei der Konstruktion der Nachricht M berücksichtigt (siehe dazu (N032.800) und (N034.100)).

#### Generierung CMAC ohne internes Padding vor der CMAC-Berechnung

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring, für den ein MAC berechnet wird |
| *K* | Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von *K* beträgt 16 oder 24 oder 32 Oktett |
| Output: | *T* | Oktettstring, der zur Prüfung der Integrität und Authentizität von *M* verwendbar ist |
| Errors: | – | Keine |
| Notation: |  | *T* = CalculateCMAC\_NoPadding( *K*, *M* ) |

1. (N002.810) K\_COS   
   Das COS MUSS *T* mittels *K* gemäß [CMAC#6.2] aus *M* berechnen. Dabei werden folgende Schritte durchgeführt:
   1. Definitionen: CIPH = Block Cipher Algorithmus (siehe (N002.000))   
        *b* = Blocklänge von CIPH in Bit hier gleich 128 Bit
   2. Schritt 1: Generiere Unterschlüssel *K1* und *K2* gemäß [CMAC#6.1].
   3. Schritt 2: Falls *Mlen* gleich null ist, dann setze *n* = 1,   
        sonst *r* = *Mlen* mod *b* und   
        falls *r* = 0, dann setze *n* = *Mlen* / *b*   
        sonst setze *n* = ( *Mlen* – *r* ) / *b* + 1.
   4. Schritt 3: Teile *M* wie folgt in Blöcke auf:   
        *M* = *M*1   ||   *M*2   ||   …   ||   *Mn*–1   ||   *M*\**n*.   
      Die Blöcke *M*1 bis *Mn*–1 besitzen die Länge *b*. Der Block *M*\**n* besitzt eine Länge kleiner gleich *b*.
   5. Schritt 4: Falls *r* = 0 ist, dann setze *Mn* = *K1*   XOR   *M*\**n*,   
        sonst setze *Mn* = *K2*   XOR   PaddingIso( *M*\**n*, *b / 8* ).
   6. Schritt 5: *C*0 = I2OS( 0, *b* / 8 )
   7. Schritt 6: *Ci* = AES\_ENC( *K*, *Ci*–1 XOR *Mi* ).
   8. Schritt 7: *T* = Extract\_MSByte( *Cn*, 8).

#### Generierung CMAC mit internem ISO-Padding vor der CMAC-Berechnung

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring, für den ein MAC berechnet wird |
| *K* | Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von *K* beträgt 16 oder 24 oder 32 Oktett |
| Output: | *T* | Oktettstring, der zur Prüfung der Integrität und Authentizität von *M* verwendbar ist |
| Errors: | – | Keine |
| Notation: |  | *T* = CalculateCMAC\_IsoPadding( *K*, *M* ) |

1. (N002.820) K\_COS   
   Das COS MUSS *T* mittels *K* gemäß [CMAC#6.2] aus *M* wie folgt berechnen.   
   *T* = CalculateCMAC\_NoPadding( K, PaddingIso( *M*, 16 ) ).

### MAC-Prüfung

Die MAC-Prüfung prüft die Konsistenz zwischen Daten und den hinzugefügten Informationen der Datenauthentisierung. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine MAC-Prüfung als Funktion wie folgt verwendet:

#### Prüfung Retail-MAC, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring, der durch einen MAC geschützt ist |
| *T´* | Oktettstring, der *M* zwecks Datenauthentisierung hinzugefügt wurde |
| *K* | Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von *K* beträgt 24 Oktett. *K* setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: *K* = Ka   ||   Kb   ||   Kc |
| Output: | *out* | Ergebnis der MAC-Prüfung, entweder VALID oder INVALID |
| Errors: | – | Keine |
| Notation: |  | *out* = VERIFY\_Retail\_MAC( *K*, *T´*, *M* ) |

1. (N002.900) K\_COS\_G1, Option\_DES   
   Das COS MUSS die Integrität der Daten *M* mittels *T*´ und *K* prüfen. Dabei werden folgende Schritte durchgeführt:
   1. Schritt 1: *T* = CALCULATE\_Retail\_MAC( *K*, *M* ).
   2. Schritt 2: Falls *T* identisch zu *T*´ ist, dann gebe VALID zurück   
        sonst gebe INVALID zurück.

#### Prüfung CMAC ohne internes Padding vor der CMAC-Berechnung

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring, der durch einen MAC geschützt ist |
| *T´* | Oktettstring, der *M* zwecks Datenauthentisierung hinzugefügt wurde |
| *K* | Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 16 oder 24 oder 32 Oktett |
| Output: | *out* | Ergebnis der MAC-Prüfung, entweder VALID oder INVALID |
| Errors: | – | Keine |
| Notation: |  | *out* = VerifyCMAC\_NoPadding( *K*, *T´*, *M* ) |

1. (N003.010) K\_COS   
   Das COS MUSS die Integrität der Daten *M* mittels *T*´ und *K* gemäß [CMAC#6.3] prüfen. Dabei werden folgende Schritte durchgeführt:
   1. Schritt 1: *T* = CalculateCMAC\_NoPadding( *K*, *M* ).
   2. Schritt 2: Falls *T* identisch zu *T*´ ist, dann gebe VALID zurück   
        sonst gebe INVALID zurück.

#### Prüfung CMAC mit internem Padding vor der CMAC Berechnung

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Beliebiger Oktettstring, der durch einen MAC geschützt ist |
| *T´* | Oktettstring, der *M* zwecks Datenauthentisierung hinzugefügt wurde |
| *K* | Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 16 oder 24 oder 32 Oktett |
| Output: | *out* | Ergebnis der MAC-Prüfung, entweder VALID oder INVALID |
| Errors: | – | Keine |
| Notation: |  | *out* = VerifyCMAC\_IsoPadding( *K*, *T´*, *M* ) |

1. (N003.020) K\_COS   
   Das COS MUSS die Integrität der Daten *M* mittels *T*´ und *K* gemäß [CMAC#6.3] prüfen. Dabei werden folgende Schritte durchgeführt:
   1. Schritt 1: *T* = CalculateCMAC\_IsoPadding( *K*, *M* ).
   2. Schritt 2: Falls *T* identisch zu *T*´ ist, dann gebe VALID zurück   
        sonst gebe INVALID zurück.

### Signaturberechnung

Unter der Berechnung einer Signatur wird in diesem Dokument lediglich die mit dem privaten Schlüssel durchgeführte Operation verstanden.

#### Signaturberechnung mittels RSA

##### RSA, ISO9796–2, DS1, SIGN, Option\_RSA\_CVC

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos Internal Authenticate sichtbar (siehe (N086.900)c und (N086.900)d). Sie wird wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *PrK* | Privater RSA-Schlüssel gemäß 8.2.3 |
| *M* | Beliebiger Oktettstring, der die zu signierende Nachricht repräsentiert |
| Output: | *sig* | Oktettstring, welcher die Signatur repräsentiert |
| *M*2 | Oktettstring, welcher den „non recoverable part“ der Nachricht *M* repräsentiert |
| Errors: | – | – |
| Notation: |  | ( *sig*, *M2* ) = RSA\_ISO9796\_2\_DS1\_SIGN( *PrK*, *M* ) |

1. (N003.100) K\_COS\_G1, Option\_RSA\_CVC   
   Gemäß [ISO/IEC 9796-2#7, 8] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten: *n* = *PrK*.*n*, *d* = *PrK*.*d*
   1. Schritt 1: Message allocation: Aufspalten der Nachricht *M* in *M*1 und *M*2 gemäß [ISO/IEC 9796-2#7.2.2].
   2. Schritt 2: Message representative production: Berechnen des Repräsentanten *F* der Nachricht *M* gemäß [ISO/IEC 9796-2#8.3], wobei *t* = 1 gesetzt werden MUSS und als Hash-Funktion SHA-256 (siehe (N001.300)) verwendet werden MUSS.
   3. Schritt 3: Signature production: Gemäß [ISO/IEC 9796-2#7.2.4] und [ISO/IEC 9796-2#B.4] werden folgende Operationen ausgeführt:
      1. Schritt 3.1: *J* = OS2I( *F* )
      2. Schritt 3.2: *a* = *Jd* mod *n*
      3. Schritt 3.3: *b* = *n* – *a*
      4. Schritt 3.4: *c* = min{ *a*, *b* }
      5. Schritt 3.5: *sig* = I2OS( *c*, OctetLength( *n* ) )

##### RSA, SSA, PKCS1–V1\_5

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Compute Digital Signature sichtbar (siehe (N088.600)d). Sie wird wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *digestInfo* | Beliebiger Oktettstring, der als Digest Info verwendet wird (siehe [PKCS#1#A.2.4]) |
| *PrK* | Privater RSA–Schlüssel gemäß 8.2.3 |
| Output: | *S* | Oktettstring, welcher die Signatur repräsentiert |
| Errors: | DigestInfoTooLong | Der Inputparameter *digestInfo* enthält zu viele Oktette |
| Notation: |  | *S* = RSASSA\_PKCS1\_V1\_5\_SIGN( *PrK*, *digestInfo* ) |

1. (N003.200) K\_COS\_G1   
   Gemäß [PKCS#1#8.2.1, 9.2] MUSS das COS folgende Aktion durchführen, wobei folgende Definitionen gelten: *n* = *PrK*.*n*, *d* = *PrK*.*d*.
   1. Schritt 0: Falls OctetLength( *digestInfo* ) > 0,4 OctetLength( *n* ) ist, dann breche diesen Algorithmus mit der Fehlermeldung DigestInfoTooLong ab.
   2. Schritt 1: Setze *EM* 🡨 ´00´   ||   *digestInfo*.
   3. Schritt 2: Setze *EM* 🡨 ´FF´   ||   *EM*.
   4. Schritt 3: Falls OctetLength( *EM* ) kleiner als OctetLength( *n* ) – 2 ist, dann fahre mit Schritt 2 fort.
   5. Schritt 4: Setze *EM* 🡨 ´01´   ||   *EM*.
   6. Schritt 5: Setze *m* 🡨 OS2I( *EM* ).
   7. Schritt 6: Setze *s* 🡨 *md* mod *n*.
   8. Schritt 7: Setze *S* 🡨 I2OS( *s*, OctetLength( *n* ) ).

##### RSA, SSA, PSS

Diese Funktionalität ist an der physikalischen Schnittstelle nicht sichtbar. Sie wird im Rahmen interner Funktionen verwendet.

|  |  |  |
| --- | --- | --- |
| Input: | *M1* | Beliebiger Oktettstring, der den „recoverable part“ der zu signierenden Nachricht M repräsentiert |
| *h(M2)* | Beliebiger Oktettstring, der den Hash-Wert über den „non recoverable part“ der zu signierenden Nachricht M enthält |
| *PrK* | Privater RSA-Schlüssel gemäß 8.2.3 |
| Output: | *sig* | Oktettstring, welcher die Signatur repräsentiert |
| Errors: | – | Keine |
| Notation: |  | *sig* = RSA\_PSS\_SIGN( *PrK*, *M1*, *h*(*M2*) ) |

1. (N003.300) K\_COS   
   Gemäß [ISO/IEC 9796-2#7, 9] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten: *n* = *PrK*.*n*, *d* = *PrK*.*d*
   1. Schritt 1: Message representative production: Berechnen des Repräsentanten *F* der Nachricht *M* gemäß [ISO/IEC 9796-2#9.3], wobei *t* = 1 und *Ls* = *Lh* gesetzt werden MUSS und als Hash-Funktion SHA-256 verwendet werden MUSS. Im einzelnen:
      1. Setze *C* = I2OS( BitLength( OS2BS( *M*1 ) ), 8)
      2. Setze *S* = RAND( 32 )
      3. Berechne *H* = SHA\_256( *C* || *M*1 || *h(M*2*)* || *S* )
      4. Berechne *F* gemäß [ISO/IEC 9796-2#9.3.2].
   2. Schritt 2: Signature production: Gemäß [ISO/IEC 9796-2#7.2.4] und [ISO/IEC 9796-2#B.6] werden folgende Operationen ausgeführt:
      1. Schritt 2.1: *J* = OS2I( *F* )
      2. Schritt 2.2: *a* = *Jd* mod *n*
      3. Schritt 2.3: *sig* = I2OS( *a*, OctetLength( *n* ) )

##### RSA, ISO9796–2, DS2, SIGN

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Compute Digital Signature sichtbar (siehe (N088.600)b). Sie wird wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *M1* | Beliebiger Oktettstring, der den „recoverable part“ der zu signierenden Nachricht *M* repräsentiert |
| *h(M2)* | Beliebiger Oktettstring, der den Hash-Wert über den „non recoverable part“ der zu signierenden Nachricht M enthält |
| *PrK* | Privater RSA-Schlüssel gemäß 8.2.3 |
| Output: | *sig* | Oktettstring, welcher die Signatur repräsentiert |
| Errors: | – | Keine |
| Notation: |  | *sig* = RSA\_ISO9796\_2\_DS2\_SIGN( *PrK*, *M1*, *h*(*M2*) ) |

1. (N003.400) K\_COS   
   Gemäß [ISO/IEC 9796-2#7, 9] MUSS das COS folgende Aktionen durchführen, wobei folgende Definition gilt: *n* = *PrK*.*n*
   1. Schritt 1: *a* = OS2I( RSA\_PSS\_SIGN( *PrK, M*1*, h(M*2*)* )).
   2. Schritt 2: *b* = *n* – *a*
   3. Schritt 3: *c* = min{ *a*, *b* }
   4. Schritt 4: *sig* = I2OS( *c*, OctetLength( *n* ) )

##### RSASSA–PSS–SIGN

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen der Kommandos Internal Authenticate (siehe (N086.900)a) und PSO Compute Digital Signature sichtbar (siehe (N088.600)c). Sie wird wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *mHash* | Beliebiger Oktettstring, der den Hash-Wert über die zu signierenden Nachricht *M* repräsentiert |
| *PrK* | Privater RSA-Schlüssel gemäß 8.2.3 |
| Output: | *S* | Oktettstring, welcher die Signatur repräsentiert |
| Errors: | – | Keine |
| Notation: |  | S = RSASSA\_PSS\_SIGN( *PrK*, *mHash* ) |

1. (N003.500) K\_COS   
   Gemäß [PKCS#1#8.1.1, 9.1.1] MUSS das COS folgende Aktion durchführen:
   1. Schritt 1: *M*1 = ´´,     (Anmerkung: *M*1 ist ein leerer Oktettstring)
   2. Schritt 2: *S* = RSA\_PSS\_SIGN( *PrK, M*1*, mHash* ).

#### Signaturberechnung mittels ELC

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Compute Digital Signature sichtbar (siehe (N088.600)c). Sie wird wie folgt verwendet:

|  |  |  |
| --- | --- | --- |
| Input: | *H* | Oktettstring mit beliebigem Inhalt, der einen Hash-Wert repräsentiert |
| *PrK* | Privater ELC–Schlüssel gemäß 8.2.3 |
| Output: | *R* | Oktettstring, erster Teil der ECDSA-Signatur |
| *S* | Oktettstring, zweiter Teil der ECDSA-Signatur |
| Errors: | – | Keine |
| Notation: |  | ( *R*, *S* ) = ELC\_SIG( *PrK*, *H* ) |

1. (N003.600) K\_COS   
   Gemäß [BSI-TR-03111#4.2.1.1] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten:   
     *dA* = *PrK*.*d*, *G* = *PrK*.*domainParameter*.*G*,   
     *n* = *PrK*.*domainParameter*.*n*, τ = *PrK*.*domainParameter*.τ
   1. Schritt 0: *H*τ = BS2OS(Extract\_MSBit( *H*, τ)).
   2. Schritt 1: *k* = RNG({1,2, …, *n* – 1}), d.h. zufällig erzeugte ganze Zahl aus dem Intervall [1, *n* – 1].
   3. Schritt 2: *Q* = [*k*] *G*     mit     *Q* = ( *xQ*, *yQ* ).
   4. Schritt 3: *r* = OS2I( FE2OS(*xQ*) ) mod *n*.   
      Falls *r* gleich null ist, dann gehe zu Schritt 1.
   5. Schritt 4: *kinv* = *k*–1 mod *n*.
   6. Schritt 5: *s* = *kinv* ( *r* *dA* + OS2I( *H*τ ) ) mod *n*.   
      Falls *s* gleich null ist, dann gehe zu Schritt 1.
   7. Schritt 6: *R* = I2OS( *r*, ceiling(τ / 8) ).   
        *S* = I2OS( *s*, ceiling(τ / 8) ).
2. In (N003.600)a wird ein möglicherweise zu langer Hash-Wert auf die Bitlänge von n reduziert, vergleiche dazu auch [BSI-TR-03111#4.2]. Bei den in diesem Dokument verwendeten Kombinationen aus Hash-Algorithmus und Kurvenlänge ist Hτ stets identisch zu H, sodass die Berechnung in (N003.600)a nicht praxisrelevant ist.

### Signaturprüfung

Unter der Prüfung einer Signatur wird in diesem Dokument lediglich die mit dem öffentlichen Schlüssel durchgeführte Operation verstanden. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Signaturprüfung als Funktion wie folgt verwendet:

#### RSA, ISO9796–2, DS1, VERIFY, Option\_RSA\_CVC

|  |  |  |
| --- | --- | --- |
| Input: | PuK | Öffentlicher RSA-Schlüssel gemäß 8.2.4 |
| sig | Beliebiger Oktettstring, der eine Signatur repräsentiert |
| M2 | Beliebiger Oktettstring, der den „non recoverable part“ der Nachricht M repräsentiert. M2 KANN leer sein |
| Output: | out | Boolean, True, falls die Signatur gültig ist, andernfalls False |
| M | Oktettstring, die rekonstruierte Nachricht |
| Errors: | – | Keine |
| Notation: |  | ( out, M ) = RSA\_ISO9796\_2\_DS1\_VERIFY( PuK, sig, M2 ) |

1. (N003.700) K\_COS\_G1, Option\_RSA\_CVC   
   Gemäß [ISO/IEC 9796-2#7.2] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten: *n* = *PuK*.*n*, *e* = *PuK*.*e.*
   1. Schritt 1: Signature opening: Gemäß [ISO/IEC 9796-2#B.5]:
      1. Schritt 1.1: Falls die Bitlänge von *sig* ungleich der Bitlänge von *n* ist, dann gebe *False* zurück und breche diesen Algorithmus ab.
      2. Schritt 1.2: Falls das höchstwertige Bit von *sig* den Wert 1 hat, dann gebe  
         *False* zurück und breche diesen Algorithmus ab.
      3. Schritt 1.3: *s* = OS2I( *sig* ).
      4. Schritt 1.4: *J*\* = *se* mod *n*.
      5. Schritt 1.5: Falls *J*\* gerade ist, dann setze *I*\* = *J*\*,   
           sonst setze *I*\* = *n* – *J*\*.
      6. Schritt 1.6: Falls *I*\* mod 256 ungleich ´BC´ ist, dann gebe *False* zurück und breche diesen Algorithmus ab.
      7. Schritt 1.7: *F*\* = I2OS( *I*\*, OctetLength( *n* ) ).
      8. Schritt 1.8: Falls das höchstwertige Bit von *F*\* den Wert 1 hat, dann gebe  
         *False* zurück und breche diesen Algorithmus ab.
   2. Schritt 2: Message recovery: Gemäß [ISO/IEC 9796-2#8.4]:
      1. Schritt 2.1: Falls das zweithöchste Bit von *F*\* den Wert 0 hat, dann gebe *False* zurück und breche diesen Algorithmus ab.
      2. Schritt 2.2: Berechne *H*\* und *M*1\* gemäß [ISO/IEC 9796-2#8.4].
      3. Schritt 2.3: Berechne *M* = *M*1   ||   *M*2.
      4. Schritt 2.4: Berechne *H* = SHA\_256( *M* ).
      5. Schritt 2.5: Falls *H* identisch ist zu *H*\*, dann gebe *True* und *M* zurück, sonst gebe *False* ohne *M* zurück.
2. ACHTUNG: In (N003.700)b.1 ist F\* ein Oktettstring. Aus diesem F\* geht durch Abschneiden des „most significant bit“ ein Bitstring hervor, der F\* aus [ISO/IEC 9796-2#8.4] entspricht. Deshalb wird in (N003.700)b.1 das zweithöchste Bit geprüft und in [ISO/IEC 9796-2#8.4] das „leftmost“ Bit.

#### Signaturprüfung mittels elliptischer Kurven

|  |  |  |
| --- | --- | --- |
| Input: | *PuK* | Öffentlicher ELC-Schlüssel gemäß 8.2.4 |
| *H* | Beliebiger Oktettstring, der einen Hash-Wert repräsentiert |
| *R* | Oktettstring, erster Teil der ECDSA-Signatur |
| *S* | Oktettstring, zweiter Teil der ECDSA-Signatur |
| Output: | *out* | Boolean, True, falls die Signatur gültig ist, andernfalls False |
| Errors: | – | Keine |
| Notation: |  | *out* = ELC\_VER\_SIG( *PuK*, *R*, *S*, *H* ) |

1. (N003.800) K\_COS   
   Gemäß [BSI-TR-03111#4.2.1.2] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten:   
     *Pa* = *PuK*.*P,*  *G* = *PuK*.*domainParameter*.*G*,   *n* = *PuK*.*domainParameter*.*n*, τ = *PuK*.*domainParameter*.τ
   1. Schritt 0: *H*τ = BS2OS(Extract\_MSBit( *H*, τ)), siehe Hinweis (20):   
        *r* = OS2I( *R* ).   
        *s* = OS2I( *S* ).
   2. Schritt 1: Überprüfe, ob *r* und *s* Element der Menge {1, 2, …, *n* – 1} sind. Falls nicht, gebe *False* zurück und breche diesen Algorithmus ab.
   3. Schritt 2: *sinv* = *s*–1 mod *n*.
   4. Schritt 3: *u*1 = *sinv* OS2I (*H*τ ) mod *n*.   
        *u*2 = *sinv* *r* mod *n*.
   5. Schritt 4: *Q* = [*u*1] *G* + [*u*2] *Pa*     mit     *Q* = ( *xQ*, *yQ* ).
   6. Schritt 5: *v* = OS2I ( FE2OS(*xQ*) ) mod *n*.
   7. Schritt 6: Gebe *True* zurück, falls *v* gleich *r* ist, andernfalls *False*.

## Vertraulichkeit von Daten, symmetrischer Fall

### Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung überführt eine beliebige Nachricht *plaintext* (Oktettstring beliebigen Inhalts und Länge) in ein Chiffrat *ciphertext*. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine symmetrische Verschlüsselung als Funktion wie folgt verwendet:

#### Verschlüsselung 3TDES, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | *P* | Oktettstring, Klartext (plaintext), beliebiger Oktettstring beliebiger Länge, der verschlüsselt wird |
| *K* | Beliebiger Oktettstring der Länge 24 Oktett, der als Schlüssel verwendet wird |
| *T1* | Beliebige nicht-negative Zahl, die als Startwert verwendet wird |
| Output: | *C* | Oktettstring, Chiffrat (ciphertext) |
| Errors: | *lengthError* | Die Länge von *P* ist kein ganzzahliges Vielfaches der Blocklänge |
| Notation: |  | C = 3TDES\_CBC\_ENC( K, T1, P ) |

1. (N003.900) K\_COS\_G1, Option\_DES   
   Das COS MUSS *C* mittels *K* und *T*1 wie folgt aus *P* berechnen. Dabei sind folgende Aktionen durchzuführen:
   1. Schritt 1: Falls OctetLength( *P* ) mod 8 ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
   2. Schritt 2: Teile *P* auf in Blöcke mit jeweils 64 Bit *P* = *P*1   ||   *P*2   ||   …   ||   *Pn*.
   3. Schritt 3: Setze *C*0 = I2OS( *T*1, 8 )
   4. Schritt 4: Berechne *Ci* = 3TDES\_ENC( *K*, *Ci*–1   XOR   *Pi* ) für *i* = 1,…, *n*.
   5. Schritt 5: Berechne *C* = *C*1   ||   *C*2   ||   …   ||   *Cn*–1   ||   *Cn*.

#### Verschlüsselung AES

|  |  |  |
| --- | --- | --- |
| Input: | *P* | Oktettstring, Klartext (plaintext), beliebiger Oktettstring beliebiger Länge, der verschlüsselt wird |
| *K* | Beliebiger Oktettstring der Länge 16, 24 oder 32 Oktett, der als Schlüssel verwendet wird |
| *T1* | Beliebige nicht-negative Zahl, die als Startwert verwendet wird |
| Output: | *C* | Oktettstring, Chiffrat (ciphertext), das dieselbe Länge wie P besitzt |
| Errors: | *lengthError* | Die Länge von *P* ist kein ganzzahliges Vielfaches der Blocklänge |
| Notation: |  | C = AES\_CBC\_ENC( K, T1, P ) |

1. (N004.000) K\_COS   
   Das COS MUSS *C* mittels *K* und *T1* gemäß [NIST sp800-38a#6.2] aus *P* berechnen. Dabei sind folgende Aktionen durchzuführen:
   1. Schritt 1: Falls OctetLength( *P* ) mod 16 ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
   2. Schritt 2: Teile *P* auf in Blöcke mit jeweils 128 Bit *P* = *P*1   ||   *P*2   ||   …   ||   *Pn*.
   3. Schritt 3: Setze *C*0 = I2OS( *T*1, 16 )
   4. Schritt 4: Berechne *Ci* = AES\_ENC( *K*, *Ci*–1   XOR   *Pi* ) für *i* = 1,…, *n*.
   5. Schritt 5: Berechne *C* = *C*1   ||   *C*2   ||   …   ||   *Cn*–1   ||   *Cn*.

### Symmetrische Entschlüsselung

Die symmetrische Entschlüsselung überführt ein Chiffrat *ciphertext* (Oktettstring beliebigen Inhalts und Länge) in einen Klartext *plaintext*. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine symmetrische Verschlüsselung als Funktion wie folgt verwendet:

#### Entschlüsselung 3TDES, Option\_DES

|  |  |  |
| --- | --- | --- |
| Input: | *C* | Beliebiger Oktettstring, Chiffrat (ciphertext), der entschlüsselt wird, die Länge ist ein ganzzahliges Vielfaches der Blocklänge |
| *K* | Beliebiger Oktettstring der Länge 192 Bit, der als Schlüssel verwendet wird |
| *T1* | Beliebige nicht-negative Zahl, die als Startwert verwendet wird |
| Output: | *P* | Oktettstring, Klartext (plaintext) |
| Errors: | *lengthError* | Die Länge von C ist kein ganzzahliges Vielfaches der Blocklänge |
| Notation: |  | P = 3TDES\_CBC\_DEC( K, T1, C ) |

1. (N004.100) K\_COS\_G1, Option\_DES   
   Das COS MUSS *P* mittels *K* und *T*1 wie folgt aus *C* berechnen. Dabei sind folgende Aktionen durchzuführen:
   1. Schritt 0: Falls OctetLength( *P* ) mod 8 ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
   2. Schritt 1: Teile *C* auf in Blöcke mit jeweils 64 Bit *C* = *C*1   ||   *C*2   ||   …   ||   *Cn*.
   3. Schritt 2: Setze *C*0 = I2OS( *T*1, 8)
   4. Schritt 3: Berechne *Pi* = 3TDES\_DEC( *K*, *Ci* )   XOR   *Ci*–1 für *i* = 1,…,*n*.
   5. Schritt 4: Berechne *P* = *P*1   ||   *P*2   ||   …   ||   *Pn*–1   ||   *Pn*.

#### Entschlüsselung AES

|  |  |  |
| --- | --- | --- |
| Input: | *C* | Beliebiger Oktettstring, Chiffrat (ciphertext) beliebiger Länge, der entschlüsselt wird |
| *K* | Beliebiger Oktettstring der Länge 16, 24 oder 32 Oktett, der als Schlüssel verwendet wird |
| *T1* | Beliebige nicht-negative Zahl, die als Startwert verwendet wird |
| Output: | *P* | Oktettstring, Klartext (plaintext), der dieselbe Länge wie C besitzt |
| Errors: | *lengthError* | Die Länge von C ist kein ganzzahliges Vielfaches der Blocklänge |
| Notation: |  | P = AES\_CBC\_DEC( K, T1, C ) |

1. (N004.200) K\_COS   
   Das COS MUSS *P* mittels *K* und *T*1 gemäß [NIST sp800-38a#6.2] aus *C* berechnen. Dabei sind folgende Aktionen durchzuführen:
   1. Schritt 0: Falls OctetLength( *P* ) mod 16 ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
   2. Schritt 1: Teile *C* auf in Blöcke mit jeweils 128 Bit *C* = *C*1   ||   *C*2   ||   …   ||   *Cn*.
   3. Schritt 2: Setze *C*0 = I2OS( *T*1, 16)
   4. Schritt 3: Berechne *Pi* = AES\_DEC( *K*, *Ci* )   XOR   *Ci*–1 für *i* = 1,…,*n*.
   5. Schritt 4: Berechne *P* = *P*1   ||   *P*2   ||   …   ||   *Pn*–1   ||   *Pn*.

## Vertraulichkeit von Daten, asymmetrischer Fall

### Asymmetrische Verschlüsselung

Die asymmetrische Verschlüsselung überführt eine Nachricht *M* (Oktettstring beliebigen Inhalts und Länge) in ein Chiffrat *C*. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine asymmetrische Verschlüsselung als Funktion wie folgt verwendet:

#### RSA, ES, PKCS1 V1.5

|  |  |  |
| --- | --- | --- |
| Input: | *PuK* | Öffentlicher RSA-Schlüssel gemäß 8.2.4 |
| *M* | Beliebiger Oktettstring, zu verschlüsselnde Nachricht |
| Output: | *C* | Oktettstring, Chiffrat zur Nachricht *M* |
| Errors: | *ERROR* | „message too long“, falls *M* ist zu lang |
| Notation: |  | *C* = RSAES\_PKCS1\_V1\_5\_ENCRYPT( *PuK*, *M* ) |

1. (N004.300) K\_COS   
   Das COS MUSS *C* mittels *PuK* und *M* gemäß [PKCS#1#7.2.1] berechnen. Es gelten folgende Definitionen: *n* = *PuK*.*n*, *e* = *PuK*.*e*.
   1. Schritt 0: Setze *Plen* = OctetLength( *n* ) – OctetLength( *M* ) – 3.
   2. Falls *Plen* kleiner als acht ist, dann breche diesen Algorithmus mit der Fehlermeldung „*ERROR*“ ab.
   3. Schritt 1: Setze *EM* = ´00´   ||   *M*.
   4. Schritt 2: Setze *PS* = RAND( 1 ).
   5. Schritt 3: Falls *PS* gleich ´00´ ist, fahre mit Schritt 2 fort.
   6. Schritt 4: Setze *EM* = *PS*   ||   *EM*.
   7. Schritt 5: Falls OctetLength( *EM* ) kleiner als OctetLength( *n* ) – 2 ist, dann   
      fahre mit Schritt 2 fort.
   8. Schritt 6: Setze *EM* = ´02´   ||   *EM*.
   9. Schritt 7: Setze *m* = OS2I( *EM* ).
   10. Schritt 8: Setze *c* = *me* mod *n*.
   11. Schritt 9: Setze *C* = I2OS( *c*, OctetLength( *n* ) ).

#### RSA, OAEP, Verschlüsselung

|  |  |  |
| --- | --- | --- |
| Input: | *PuK* | Öffentlicher RSA-Schlüssel gemäß 8.2.4 |
| *M* | Beliebiger Oktettstring, zu verschlüsselnde Nachricht |
| Output: | *C* | Oktettstring, Chiffrat zur Nachricht *M* |
| Errors: | *ERROR* | „message too long“, falls *M* ist zu lang |
| Notation: |  | *C* = RSAES\_OAEP\_ENCRYPT( *PuK*, *M* ) |

1. (N004.400) K\_COS   
   Das COS MUSS *C* mittels *PuK* und *M* gemäß [PKCS#1#7.1.1] berechnen. Es gelten folgende Definitionen: *n* = *PuK*.*n*, *e* = *PuK*.*e*.
   1. Schritt 1: Falls OctetLength( *M* ) größer als OctetLength( *n* ) – 66 ist, dann breche diesen Algorithmus mit dem Fehler „*ERROR*“ ab.
   2. Schritt 2: Setze *L* = ´´,     (Anmerkung: leerer Oktettstring).
   3. Schritt 3: Setze *lHash* = SHA\_256( *L* ).
   4. Schritt 4: Setze *Plen* = OctetLength( *n* ) – OctetLength( *M* ) – 66.
   5. Schritt 5: Setze *PS* = I2OS( 0, *Plen* ).
   6. Schritt 6: Setze *DB* = *lHash*   ||   *PS*   ||   ´01´   ||   *M*.
   7. Schritt 7: Setze *seed* = RAND( 32 ).
   8. Schritt 8: Setze *dbMask* = BS2OS(MGF(   
        OS2BS(*seed*),   
        8 (OctetLength( *n* ) – 33),   
        0   
        )).
   9. Schritt 9: Setze *maskedDB* = *DB*   XOR   *dbMasked*.
   10. Schritt 10: Setze *seedMask* = BS2O2(MGF(   
         OS2BS(*maskedDB*),   
         8 OctetLength( *lHash* ),   
         0,   
         )).
   11. Schritt 11: Setze *maskedSeed* = *seed*   XOR   *seedMask*.
   12. Schritt 12: Setze *EM* = ´00´   ||   *maskedSeed*   ||   *maskedDB*.
   13. Schritt 13: Setze *m* = OS2I( *EM* ).
   14. Schritt 14: Setze *c* = *me* mod *n*.
   15. Schritt 15: Setze *C* = I2OS( *c*, OctetLength( *n* )).

#### Elliptic Curve Key Agreement

Diese Funktionalität ist an der physikalischen Schnittstelle nicht direkt sichtbar, wird aber im Rahmen der asymmetrischen Ver- und Entschlüsselung mittels elliptischer Kurven verwendet.

|  |  |  |
| --- | --- | --- |
| Input: | *d* | natürliche Zahl, die dem privaten Schlüssel *PrK* entspricht |
| *P* | Punkt auf derselben elliptischen Kurve, wie *PrK* |
| *dP* | Domainparameter gemäß (N008.600) |
| Output: | *KAB* | „gemeinsames Geheimnis“ |
| Errors: | *ERROR* | Falls das Ergebnis in Schritt 3 der unendlich ferne Punkt ist |
| Notation: |  | KAB = ECKA( *d*, *P*, *dP* ) |

1. (N004.490) K\_COS   
   Das COS MUSS *KAB* mittels *d*, *P* und *dP* berechnen, wobei die Schritte 1 bis 4 [BSI-TR-03111#4.3.1] entsprechen. Es gelten folgende Definitionen:   
   *h* = *dP*.*h*,   *n* = *dP*.*n*,   *L* = *dP*.*L*
   1. Schritt 1: *l* = *h*–1 mod *n*.
   2. Schritt 2: *Q* = [ *h* ] *P*.
   3. Schritt 3: *S* = [ *d* *l* mod *n* ] *Q*. mit     *S* = ( *xS*, *yS* ).   
      Falls *S* gleich dem unendlich fernen Punkt *O* der Kurve ist, dann gebe den Fehler „*ERROR*“ zurück und beende diesen Algorithmus.
   4. Schritt 4: *KAB* = I2OS( *xS*, *L* ).

#### ELC Verschlüsselung

|  |  |  |
| --- | --- | --- |
| Input: | *M* | Oktettstring, zu verschickende Nachricht mit beliebigem Inhalt und beliebiger Länge, die verschlüsselt wird |
| *POB* | Oktettstring, öffentlicher Punkt *PB* des Empfängers |
| *dP* | Domainparameter gemäß (N008.600) |
| Output: | *POA* | Oktettstring, ephemer Punkt *PEA* des Senders |
| *C* | Oktettstring, Chiffrat der Nachricht *M* |
| *T* | Oktettstring, MAC über das Chiffrat *C* |
| Errors: | *ERROR* | * Falls in Schritt 0 ein Fehler auftritt * Falls in ECKA ein Fehler auftritt |
| Notation: |  | ( *POA*, *C*, *T* ) = ELC\_ENC( *M*, *POB*, *dP* ) |

1. (N004.500) K\_COS   
   Das COS MUSS *POA*, *C* und *T* mittels *M* und *POB* berechnen, wobei die Schritt  3 [BSI-TR-03111#4.3.1] entspricht und folgende Definitionen gelten:   
     *n* = *dP*.*n*, *h* = *dP*.*h*,   
     *G* = *dP*.*G*, *L* = *dP*.*L*
   1. Schritt 0: *PB* = OS2P( *POB*, *dP* ).   
      Falls diese Funktion mit einem Fehler terminiert, dann gebe „*ERROR*“ zurück und beende diesen Algorithmus.
   2. Schritt 1: *d* = RNG({1, 2, …, *n* – 1}), d.h. zufällig erzeugte ganze Zahl aus dem Intervall [1, *n* – 1].
   3. Schritt 2: *PEA* = [ *d* ] *G*.
   4. Schritt 3: *KAB* = ECKA( *d*, *PB*, *dP* ).   
      Falls die Funktion ECKA einen Fehler meldet, dann gebe den Fehler „*ERROR*“ zurück und beende diesen Algorithmus.
   5. Schritt 4: Berechne abgeleitete Schlüssel gemäß (N001.520)
      1. ( *Kenc*, *Kmac*, *T*2 ) = KeyDerivation\_AES256( *KAB* ).
      2. Setze: *T*1 = OS2I(AES\_ENC(*Kenc*, I2OS(*T*2, 16)))
   6. Schritt 5: *POA* = P2OS( *PEA*, *L* ).
   7. Schritt 6: *C* = AES\_CBC\_ENC( *Kenc*, *T*1, PaddingIso(*M*, 16)).
   8. Schritt 7: *T* = CalculateCMAC\_IsoPadding( *Kmac*, *C* ).

### Asymmetrische Entschlüsselung

Die asymmetrische Entschlüsselung überführt ein Chiffrat *C* in eine Nachricht *M*.

#### RSA, ES, PKCS1 V1.5, Decrypt

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Decipher sichtbar (siehe (N090.300)a).

|  |  |  |
| --- | --- | --- |
| Input: | *PrK* | ein privater RSA-Schlüssel gemäß 8.2.3 |
| *C* | beliebiger Oktettstring, Chiffrat der Nachricht *M* |
| Output: | *M* | Oktettstring, Klartextnachricht zum Chiffrat |
| Errors: | *ERROR* | „decryption error“, falls einer der folgenden Fälle eintritt:  *C* ist numerisch größer gleich dem Modulus  *EM* fängt nicht mit ´0002´an  In *EM* gibt es kein ´00´ Oktett, welches *PS* von *M* trennt  *PS* hat weniger als acht Oktett |
| Notation: |  | *M* = RSAES\_PKCS1\_V1\_5\_DECRYPT( *PrK*, *C* ) |

1. (N004.600) K\_COS   
   Das COS MUSS *M* mittels *PrK* und *C* gemäß [PKCS#1#7.2.2] berechnen. Es gelten folgende Definitionen: *n* = *PrK*.*n*, *d* = *PrK*.*d*.
   1. Schritt 1: Falls OctetLength( *C* ) ungleich OctetLength( *n* ) ist, breche diesen Algorithmus mit „decryption error“ ab.
   2. Schritt 2: Setze *c* = OS2I( *C* ).
   3. Schritt 3: Falls *c* größer oder gleich *n* ist, breche diesen Algorithmus mit „decryption error“ ab.
   4. Schritt 4: Setze *EM* = I2OS( *cd* mod *n*, OctetLength( *n* )).
   5. Falls das erste Oktett in *EM* ungleich ´00´ist, oder das zweite Oktett in *EM* ungleich ´02´ ist, breche diesen Algorithmus mit „decryption error“ ab.
   6. Schritt 5: Teile *EM* wie folgt auf: *EM* = ´00 02   ||   *PS*   ||   00   ||   *M*´. Dabei MUSS jedes Oktett in *PS* ungleich ´00´ sein.
   7. Schritt 6: Breche diesen Algorithmus mit „decryption error“ ab, falls
      1. *PS* kürzer als 8 Oktett ist, oder
      2. kein Oktett mit dem Wert ´00´existiert, welches *PS* von *M* trennt.
   8. Schritt 7: Gebe *M* zurück.
2. Die Prüfung in Schritt 1 ist überflüssig, da wegen (N089.000) der in Schritt 1 geprüfte Fall im Rahmen dieser Spezifikation nicht auftritt. Zudem wird in Schritt 3 strenger geprüft.

#### RSA, OAEP, Decrypt

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Decipher sichtbar (siehe (N090.300)b).

|  |  |  |
| --- | --- | --- |
| Input: | *PrK* | Privater RSA-Schlüssel gemäß 8.2.3 |
| *C* | Beliebiger Oktettstring, Chiffrat der Nachricht *M* |
| Output: | *M* | Oktettstring, Klartextnachricht zum Chiffrat |
| Errors: | ERROR | „decryption error“, falls einer der folgenden Fälle eintritt:  *C* ist numerisch größer gleich dem Modulus von *PrK* |
| Notation: |  | *M* = RSAES\_OAEP\_DECRYPT( *PrK*, *C* ) |

1. (N004.700) K\_COS   
   Das COS MUSS *M* mittels *PrK* und *C* gemäß [PKCS#1#7.1.2] berechnen. Es gelten folgende Definitionen: *n* = *PrK*.*n*, *d* = *PrK*.*d*.
   1. Schritt 1: Setze *L* = ´´,     (Anmerkung: leerer Oktettstring).
   2. Schritt 2: Setze *c* = OS2I( *C* ).
   3. Schritt 3: Falls *c* größer gleich *n* ist, dann breche diesen Algorithmus mit dem Fehler „decryption error“ ab.
   4. Schritt 4: Setze *m* = *cd* mod *n*.
   5. Schritt 5: Setze *EM* = I2OS( *m*, OctetLength( *n* ) ).
   6. Schritt 6: Setze *lHash* = SHA\_256( *L* ).
   7. Schritt 7: Teile *EM* wie folgt auf:
      1. Schritt 7.1: *EM* = *Y*   ||   *maskedSeed*   ||   *maskedDB*.
      2. Schritt 7.2: 1 = OctetLength( *Y* ).
      3. Schritt 7.3: 32 = OctetLength( *maskedSeed* ).
   8. Schritt 8: Setze *seedMask* = BS2OS(MGF(   
        OS2BS( *maskedDB* ),   
        8 OctetLength(*lHash*),   
        0   
        )).
   9. Schritt 9: Setze *seed* = *maskedSeed*   XOR   *seedMask*.
   10. Schritt 10: Setze *dbMask* = BS2OS(MGF(   
         OS2BS(*seed*),   
         8 OctetLength(*maskedDB*),   
         0   
         )).
   11. Schritt 11: Setze *DB* = *maskedDB*   XOR   *dbMask*.
   12. Schritt 12: Teile *DB* wie folgt auf:
       1. Schritt 12.1: *DB* = *lHash*’   ||   *PS*   ||   ´01´   ||   *M*.
       2. Schritt 12.2: 32 = OctetLength( *lHash*’ ).
       3. Schritt 12.3: Der möglicherweise leere Oktettstring *PS* darf nur Oktette mit dem Wert ´00´ enthalten.
   13. Schritt 13: Breche diesen Algorithmus mit „decryption error“ ab, wenn
       1. *lHash*’ ungleich *lHash* ist, oder
       2. *Y* nicht den Wert ´00´ besitzt, oder
       3. kein Oktett mit dem Wert ´01´ existiert, welches *PS* von *M* trennt.
   14. Gebe *M* zurück.

#### Asymmetrische Entschlüsselung mittels ELC

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Decipher sichtbar (siehe (N090.300)c).

|  |  |  |
| --- | --- | --- |
| Input: | *PrK* | Privater ELC-Schlüssel gemäß 8.2.3 |
| *PO* | Oktettstring, ephemerer Punkt *PEA* des Senders |
| *C* | Oktettstring, Chiffrat der Nachricht *M* |
| *T´* | Oktettstring, MAC über das Chiffrat *C* |
| Output: | *M* | Oktettstring, Klartextnachricht zum Chiffrat |
| Errors: | ERROR | Falls *PO* nicht im Format „uncompressed encoding“ vorliegt  Falls *PO* einen Punkt bezeichnet, der nicht auf derselben Kurve liegt, wie *PrK*  Falls die Funktion ECKA einen Fehler meldet  Falls die MAC-Prüfung fehlschlägt |
| Notation: |  | *M* = ELC\_DEC( *PO*, *PrK*, *C*, *T´* ) |

1. (N004.800) K\_COS   
   Das COS MUSS *M* mittels *PEA*, *PrK*, *C* und *T´* berechnen, wobei Schritt 1 [BSI-TR-03111#4.3.1] entspricht. Es gelten folgende Definitionen:   
     *d* = *PrK*.*d*, *h* = *PrK*.*domainParameter*.*h*,   
     *n* = *PrK*.*domainParameter*.*n*, *L* = *PrK*.*domainParameter*.*L*
   1. Schritt 0: *PEA* = OS2P( *PO*, *PrK.domainParameter* ).   
      Falls diese Funktion mit einem Fehler terminiert, dann gebe “*ERROR*” zurück und beende diesen Algorithmus.
   2. Schritt 1: *KAB* = ECKA( *d*, *PEA*, *PrK.domainParameter* ).
   3. Schritt 2: Berechne abgeleitete Schlüssel gemäß (N001.520)
      1. ( *Kenc*, *Kmac*, *T*2 ) = KeyDerivation\_AES256( *KAB* )
      2. Setze: *T*1 = OS2I(AES\_ENC (*Kenc*, I2OS(*T*2, 16)))
   4. Schritt 3: *out*  = VerifyCMAC\_IsoPadding( *Kmac, T´, C* ).   
      Falls *out* den Wert *INVALID* besitzt, dann gebe den Fehler „*ERROR*“ zurück und beende diesen Algorithmus.
   5. Schritt 4: *M* = TruncateIso(AES\_CBC\_DEC( *Kenc, T*1*, C* ), 16)

# CV-Zertifikat

In [ISO/IEC 7816-8] werden zwei kompakte Zertifikatsformate definiert, sogenannte Card Verifiable Certificate (CVC), die eine Prüfung und Analyse durch eine Smartcard erleichtern. Typischerweise enthalten CV-Zertifikate nur die Informationen, welche eine Smartcard notwendigerweise für einen bestimmten Use Case benötigt. Sie sind deshalb kompakter als andere Zertifikatsformate, was sich positiv auf die Performanz auswirkt.

## CV-Zertifikat für RSA-Schlüssel, Option\_RSA\_CVC

In diesem Kapitel werden nur so genannte „nicht selbstbeschreibende“ CV-Zertifikate betrachtet, welche eine Signatur mit „Message Recovery“ gemäß [ISO/IEC 9796-2] DS1 ent­halten.

Technisch betrachtet ist ein derartiges Zertifikat eine mittels [ISO/IEC 9796-2] DS1 signierte Nachricht *M*. Die Nachricht *M* enthält unterschiedliche Daten, die ohne Strukturinformationen oder Trennzeichen konkateniert sind, daher die Bezeichnung „nicht selbstbeschreibend“. Damit die enthaltenen Informationen eindeutig extrahierbar sind, wird das höchstwertige Oktett von *M* ausgewertet.

### Bestandteile eines CV-Zertifikats für RSA-Schlüssel

Alle im 7.1 betrachteten signierten Nachrichten *M* enthalten die in den folgenden Unterkapiteln beschriebenen Informationen.

#### Certificate Profile Identifier (CPI)

Der Certificate Profile Identifier (CPI) zeigt die im CV-Zertifikat verwendete Struktur an.

#### Certification Authority Reference (CAR)

Die Certification Authority Reference (CAR) referenziert den Schlüssel der CA, welche das Zertifikat ausstellte. Typischerweise besitzt die CAR eine innere Struktur, welche sicherstellt, dass die CAR weltweit eineindeutig ist.

Typischerweise besitzt die ausstellende CA wiederum ein CV-Zertifikat. Auf diese Weise wird eine baumartige PKI aufgespannt mit der Root-CA an der Wurzel.

Der öffentliche Schlüssel der Root-CA wird oft als Sicherheitsanker bezeichnet und typischerweise in der Vorbereitungsphase (siehe Kapitel 4) in einer Smartcard gespeichert. Deshalb ist es nicht notwendig, den Sicherheitsanker per Zertifikat zu importieren (Abbruch der Rekursion).

#### Certificate Holder Reference (CHR)

Die Certificate Holder Reference (CHR) wird dazu verwendet, dem im Zertifikat enthaltenen öffentlichen Schlüssel einen eindeutigen Identifier zuzuordnen. Typischerweise wird die ebenfalls eindeutige Seriennummer einer Smartcard (ICCSN) als Teil der CHR verwendet. Im Rahmen dieses Dokumentes ist es irrelevant, wie die CHR gebildet wird.

#### Certificate Holder Autorisation (CHA)

Die Certificate Holder Autorisation (CHA) zeigt eine Rolle des Zertifikatsinhabers an. Typischerweise fordern Sicherheitskonzepte, dass zwei verschiedene Rollen unterhalb einer Root-CA nicht dieselbe CHA haben dürfen. Deshalb wird häufiger auch die Forderung erhoben, dass die CHA weltweit eineindeutig ist. Dann wird typischerweise ein Teil des eindeutigen Application Identifiers als Teil der CHA verwendet. Im Rahmen dieses Dokumentes ist es irrelevant, wie konkrete CHA-Werte festgelegt werden.

#### Object Identifier (OID)

Der Object Identifier (OID) in einem CVC beschreibt den Algorithmus, welcher dem Schlüssel im Zertifikat zugeordnet ist. Im Rahmen dieser Spezifikation werden verschiedene Algorithmen für verschiedene Verwendungszwecke genutzt, so dass implizit durch den Algorithmus-OID auch der Verwendungszweck des im Zertifikat enthaltenen öffentlichen Schlüssels festgelegt wird. OIDs sind weltweit eineindeutig.

Im Rahmen von 7.1 werden nur folgende OID-Werte verwendet (siehe Tabelle 271) :   
authS\_ISO9796-2Withrsa\_sha256\_mutual und sigS\_ISO9796-2Withrsa\_sha256.

#### Öffentlicher Schlüssel

Der öffentliche RSA-Schlüssel besteht aus den in den folgenden Unterkapiteln beschriebenen Teilen.

##### Modulus

Der Modulus wird hexadezimal, vorzeichenlos im Big-Endian-Format codiert.

##### Öffentlicher Exponent

Der öffentliche Exponent wird hexadezimal, vorzeichenlos im Big-Endian-Format codiert.

### Zertifikatsprofile für RSA-Schlüssel

Dieses Kapitel listet die zu unterstützenden Zertifikatsprofile auf, die anhand des CPI identifiziert werden.

#### CV-Zertifikat für CA-Schlüssel

In diesem Unterkapitel wird der Aufbau eines CV-Zertifikates beschrieben, welches den öffentlichen Schlüssel einer CA enthält. Dieser öffentliche Schlüssel ist wiederum geeignet, im Rahmen eines PSO Verify Certificate-Kommandos (siehe 14.8.7.1) weitere öffentliche Schlüssel per CV-Zertifikat zu importieren. Für derartige CV-Zertifikate gilt:

1. (N004.900) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der CPI MUSS den Wert ´21´ haben.
2. (N005.000) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die OID MUSS den Wert sigS\_ISO9796-2Withrsa\_sha256 = ´2B24 0304 0202 04´ haben (siehe Tabelle 271).
3. (N005.100) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die Länge des Modulus MUSS 2048 bit betragen.
4. (N005.200) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der öffentliche Exponent MUSS vier Oktett lang sein.
5. (N005.300) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die CHR MUSS acht Oktett lang sein.
6. (N005.400) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die CAR MUSS acht Oktett lang sein.
7. (N005.500) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Für die zu signierende Nachricht *M* gilt:   
       *M* = CPI   ||   Modulus   ||   öffentlicherExponent   ||   OID   ||   CHR   ||   CAR.

Wenn (N002.100)a berücksichtigt wird und der Hash-Wert gemäß 6.1 gebildet wird, dann gehören CHR und CAR stets zum „non recoverable part“ der signierten Nachricht und liegen somit stets im Klartext vor.

#### CV–Zertifikat für Authentisierungsschlüssel

In diesem Unterkapitel wird der Aufbau eines CV-Zertifikats beschrieben, welches den öffentlichen Schlüssel einer Instanz enthält, welche diesen Schlüssel zu Authentisierungszwecken verwendet. Für derartige CV-Zertifikate gilt:

1. (N005.600) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der CPI MUSS den Wert ´22´ haben.
2. (N005.700) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die CHA MUSS sieben Oktett lang sein.
3. (N005.800) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die OID MUSS den Wert authS\_ISO9796-2Withrsa\_sha256\_mutual = ´2B24 0305 0204´ haben (siehe Tabelle 271).
4. (N005.900) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die Länge des Modulus MUSS 2048 bit betragen.
5. (N006.000) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der öffentliche Exponent MUSS vier Oktett lang sein.
6. (N006.100) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die CHR MUSS zwölf Oktett lang sein.
7. (N006.200) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die CAR MUSS acht Oktett lang sein.
8. (N006.300) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Für die zu signierende Nachricht *M* gilt:   
       *M* = CPI || Modulus || öffentlicherExponent || OID || CHA || CHR || CAR.

Wenn (N002.100)a berücksichtigt wird und der Hash-Wert gemäß 7.1.2 gebildet wird, dann gehören CHA, CHR und CAR stets zum „non recoverable part“ der signierten Nachricht und liegen damit stets im Klartext vor.

### Struktur und Inhalt eines CV-Zertifikats für RSA-Schlüssel

Die Inhalte eines Zertifikat-Datenobjektes werden wie folgt berechnet:

1. (N006.400) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Schritt 1: Die Nachricht *M* MUSS gemäß 7.1.2 erzeugt werden.
2. (N006.500) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Schritt 2: Die Nachricht *M* MUSS mit einem privaten RSA-Schlüssel *PrK* signiert werden, dessen Moduluslänge 2048 bit beträgt. Als Signaturverfahren MUSS (N003.100) verwendet werden, so dass gilt:   
     ( *SIG.CA*, *M*2 ) = RSA\_ISO9796\_2\_DS1\_SIGN( *PrK*, *M* ).
3. (N006.600) K\_Anwendungsspezifikation {K\_Karte}, Option\_RSA\_CVC   
   Ein CV-Zertifikats-EF MUSS ein zusammengesetztes Zertifikat-Datenobjekt mit Tag = ´7F21´ (CV-Zertifikat) enthalten. Das Zertifikat-Datenobjekt MUSS genau zwei primitive Datenobjekte in der angegebenen Reihenfolge enthalten:
   1. Datenelement *SIG.CA* als Wertfeld in einem Datenobjekt mit Tag = ´5F37´.
   2. Non-recoverable part *M*2 als Wertfeld in einem Datenobjekt mit Tag = ´5F38´.

CV-Zertifikate für RSA-Schlüssel mit einer Moduluslänge von 2048 bit = 256 Oktett:

Tabelle 5: CV-Zertifikat einer CA mit CPI = ´21´, SHA-256

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **L** | **Wert** | | |
| ´7F21´ | ´820146´ | CV-Zertifikat (´0146´ = 326 Oktett) | | |
|  | | **Tag** | **L** | **Wert** |
| ´5F37´ | ´820100´ | Signatur SIG.CA (´0100´ = 256 Oktett) |
| ´5F38´ | ´3E´ | non recoverable part M2 (´003E´ = 62 Oktett) |

Tabelle 6: CV-Zertifikat zur Authentisierung mit CPI = ´22´, SHA-256

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag** | **L** | **Wert** | | |
| ´7F21´ | ´820150´ | CV-Zertifikat (´0150´ = 336 Oktett) | | |
|  | | **Tag** | **L** | **Wert** |
| ´5F37´ | ´820100´ | Signatur SIG.CA (´0100´ = 256 Oktett) |
| ´5F38´ | ´48´ | non recoverable part M2 (´0048´ = 72 Oktett) |

1. Vorgaben für CV-Zertifikate für ELC-Schüssel finden sich in [gemSpec\_PKI].

## CV-Zertifikate für ELC-Schlüssel (informativ)

Normative Festlegungen zu CV-Zertifikaten für ELC-Schlüssel finden sich in [gemSpec\_PKI].

# Objekte

## Diverse Attribute (normativ)

Dieses Unterkapitel beschreibt einige Attribute, die für mehrere Objekttypen gleichermaßen relevant sind.

### File Identifier

Der Attributstyp *fileIdentifier* wird von den Objekttypen DF und Datei verwendet.

Aus der Norm [ISO/IEC 7816-4] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N006.700) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *fileIdentifier* MUSS eine ganze Zahl im Intervall [´1000´, ´FEFF´] oder Element der Menge {´011C´} sein.
2. (N006.800) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Objekt, das nicht *root* (siehe 9.1) ist, DARF KEINEN *fileIdentifier* mit dem Wert ´3F00´ besitzen.
3. (N006.900) K\_Anwendungsspezifikation {K\_Karte}   
   KEIN Objekt DARF einen *fileIdentifier* mit dem Wert ´3FFF´ besitzen.
4. Die Forderung (N006.700) ist strenger als [ISO/IEC 7816-4]. Dies lässt Raum für herstellerspezifische Ordner (DF) und Dateien (EF).

### Short File Identifier

Es ist möglich, dass der Attributstyp *shortFileIdentifier* von den Objekttypen Datei verwendet wird (siehe 8.3.2).

Aus der Norm [ISO/IEC 7816-4] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N007.000) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *shortFileIdentifier* MUSS eine ganze Zahl im Intervall [1, 30] sein.

### Life Cycle Status

Der Attributstyp *lifeCycleStatus* wird von den Objekttypen Objektsystem, Ordner, Datei, Rekord, Passwort und Schlüssel zur Speicherung eines „physikalischen“ Life Cycle Status (siehe (N020.500)) verwendet. Zusätzlich ist im Rahmen gewisser Operationen der „logische“ Life Cycle Status (siehe (N020.600)) wichtig. Bei der Kommandobeschreibung wird darauf verwiesen, ob der "physikalische" oder der "logische" Life Cycle Status zu verwenden ist.

Aus den Normen [ISO/IEC 7816-4#7.4.10] und [ISO/IEC 7816-9#Abbildung 1] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N007.100) K\_Anwendungsspezifikation {K\_Karte}
   1. Der Wert von *lifeCycleStatus* MUSS ein Element der Menge {   
        „Operational state (active)“,   
        „Operational state (deactivated)“   
        „Termination state"   
      } sein.
   2. Ein COS KANN weitere Werte für *lifeCycleStatus*
      1. unterstützen oder
      2. ablehnen.

### Zugriffsregelliste

Der Attributstyp *interfaceDependentAccessRules* wird von den Objekttypen verwendet, welche die Ausführung von Kommandos von der Erfüllung von Zugriffsregeln abhängig machen. Dabei handelt es sich um eine Ansammlung von Zugriffsregeln. Typischerweise werden einem Objekt mehr als eine Zugriffsregel zugeordnet, weil es dem COS dadurch möglich ist, situationsbezogen zu reagieren. So sind gewisse Leseoperationen typischerweise nur über eine kontaktbehaftete Schnittstelle zulässig, während sie für eine kontaktlose Schnittstelle verboten sind. Andererseits ist der Zugriff auf deaktivierte Objekte typischerweise stark eingeschränkt. Darüber hinaus werden Security Environments genutzt, um Abhängigkeiten der Zugriffsregel von der Einsatzumgebung auszudrücken.

Akademisch betrachtet wäre eine dreidimensionale Matrix mit Zugriffsregeln angebracht, wo je nach Interface (kontaktbehaftet/kontaktlos = erste Dimension) Lebenszyklus (aktiviert/deaktiviert = zweite Dimension) und Einsatzumgebung (SE-Identifier = dritte Dimension) eine Zugriffsregel ausgewählt wird.

In dieser Dokumentenversion beschränkt sich die erste Dimension auf die beiden Fälle kontaktlose Schnittstelle gemäß 11.2.3 einerseits und kontaktbehaftete Schnittstelle gemäß 11.2.2 oder 11.2.1 andererseits und für die zweite Dimension ist es im deaktivierten oder terminierten Fall hinreichend, nur eine Einsatzumgebung vorzusehen. Deshalb werden in dieser Version des Dokumentes statt einer (mehrdimensionalen) Matrix zwei schlichte Listen verwendet.

1. (N007.170) K\_Anwendungsspezifikation {K\_Karte}
   1. Die Liste *interfaceDependentAccessRules* MUSS zwei Elemente umfassen.
   2. Das erste Element der Liste MUSS eine *accessRuleList* gemäß (N007.200) bis (N007.500) sein und die Zugriffsregeln für eine kontaktbehaftete Kommunikation gemäß 11.2.1 und 11.2.2 beinhalten.
   3. Das zweite Element der Liste MUSS eine *accessRuleList* gemäß (N007.200) bis (N007.500) sein und die Zugriffsregeln für eine kontaktlose Kommunikation gemäß 11.2.3 beinhalten.
   4. Ein COS KANN *interfaceDependentAccessRules* Listen mit mehr Elementen
      1. unterstützen oder
      2. ablehnen.
   5. Ein COS, welches die Option\_kontaktlose\_Schnittstelle nicht anbietet, KANN das zweite Listenelement
      1. beachten oder
      2. ignorieren.
2. (N007.200) K\_Anwendungsspezifikation {K\_Karte}
   1. Der Wert von *accessRuleList* MUSS eine Liste mit mindestens drei und maximal sechs Elementen sein.
   2. Ein COS KANN *accessRuleList* mit mehr Elementen
      1. unterstützen oder
      2. ablehnen.
3. (N007.300) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein Listenelement MUSS für den Wert der Variable *lifeCycleStatus* = „Operational state (deactivated)“ (siehe 8.1.3) verwendet werden.
   2. Ein weiteres Listenelement MUSS für den Wert der Variable *lifeCycleStatus* = „Termination state" (siehe 8.1.3) verwendet werden.
4. (N007.400) K\_Anwendungsspezifikation {K\_Karte}   
   Die übrigen Listenelemente MÜSSEN genau einem *seIdentifier* Wert gemäß (N007.900) zugeordnet werden.
5. (N007.500) K\_Anwendungsspezifikation {K\_Karte}   
   Jedes Listenelement MUSS genau eine Zugriffsregel gemäß 10.3 enthalten.

### Rekord

Der Objekttyp *record* wird von den Objekttypen linear variables, linear fixes und zyklisches Elementary File (EF) verwendet.

1. (N007.600) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein *record* MUSS eine Rekordnummer *number* besitzen. Die Rekordnummer MUSS eine ganze Zahl im Intervall [1, 254] sein.
   2. Ein COS KANN Rekordnummern mit weiteren Werten
      1. unterstützen oder
      2. ablehnen.
2. (N007.700) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein *record* MUSS einen Oktettstring *data* besitzen, dessen Anzahl enthaltener Oktette im Intervall [´01´, ´FF´] = [1, 255] liegen MUSS.
   2. Ein COS KANN weitere Längen für *record*
      1. unterstützen oder
      2. ablehnen.
3. (N007.800) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *record* MUSS eine Liste mit Attributen vom Typ *lifeCycleStatus* (siehe 8.1.3) unterstützen. Diese Liste ist entweder
   1. leer (der *record* hat keinen *lifeCycleStatus* und befindet sich implizit im Zustand „Operational state (active)“) oder
   2. enthält genau ein Element (der *record* hat genau einen *lifeCycleStatus*). Dieses Element MUSS einen Wert aus der folgenden Menge besitzen:   
      {„Operational state (active)“, „Operational state (deactivated)“}.
4. Da es aus funktionaler Sicht keine Möglichkeit gibt den lifeCycleStatus eines record in Zustand „Termination state" zu überführen, ist dieser Zustand für einen record irrelevant.

### SE–Identifier

Der Attributstyp *seIdentifier* ist eng mit dem Begriff Security Environment verknüpft (siehe 8.8). Aus dem gemäß [ISO/IEC 7816-4#10.3.3] erlaubten Wertebereich wird hier folgende Untermenge ausgewählt:

1. (N007.900) K\_Anwendungsspezifikation {K\_Karte}
   1. Der Wert von *seIdentifier* MUSS eine ganze Zahl im Intervall [1, 4] sein.
   2. Das COS KANN weitere Werte für *seIdentifier*
      1. akzeptieren oder
      2. ablehnen.

### PIN

Der Attributstyp *pin* wird im Zusammenhang mit der Benutzerverifikation verwendet. Es gilt:

1. (N008.000) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein *pin* MUSS eine Folge von Ziffern sein, die aus dem Wertebereich {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} stammen MÜSSEN. Die Anzahl der Ziffern MUSS im Intervall [4, 12] liegen.
   2. Das COS KANN weitere Zeichen oder weitere Längen für *pin*
      1. unterstützen oder
      2. ablehnen.
2. (N008.100) K\_Anwendungsspezifikation {K\_Karte}   
   Für die Umwandlung des Attributtyps *pin* nach Format-2-PIN-Block gilt:
   1. Der Format-2-PIN-Block ist ein Oktettstring mit acht Oktett = 16 Nibble.
   2. Das erste Nibble MUSS den Wert ´2´ haben.
   3. Das zweite Nibble MUSS hexadezimal die Anzahl Ziffern in *pin* codieren.
   4. Das *i* + 2–te Nibble MUSS hexadezimal die *i*–te Ziffer von *pin* codieren.
   5. Alle anderen Nibble MÜSSEN den Wert ´F´ besitzen.

### Datum

Der Attributstyp *date* wird im Zusammenhang mit dem Gültigkeitszeitraum von Schlüsseln verwendet.

1. (N008.120) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein *date* MUSS ein Datum in der Form YYMMDD in unkomprimierter BCD-Form enthalten. Beispiel: ´010400050203´ = 140523 = 23. Mai 2014. COS intern wird *date* als natürliche Zahl interpretiert.
   2. Das COS KANN weitere Codierungen für *date*
      1. unterstützen oder
      2. ablehnen.

## Schlüsselmaterial (normativ)

### Symmetrische Schlüssel

#### 3TDES-Schlüssel, Option\_DES

Der Attributstyp *3TDES\_Key* dient der Speicherung von Schlüsselmaterial für einen 3TDES-Schlüssel, welcher aus drei Teilschlüsseln *Ka*, *Kb* und *Kc* besteht.

1. (N008.200) K\_Anwendungsspezifikation {K\_Karte}, Option\_DES   
   Der Wert von *3TDES\_Key* MUSS ein Oktettstring mit 24 Oktetten sein.
2. (N008.300) K\_Anwendungsspezifikation {K\_Karte}, Option\_DES   
   Der Wert von *3TDES\_Key* MUSS beliebig wählbar sein, wobei für die Parität jedes Oktetts gilt: Falls die Parität der Oktette nicht korrekt ist, dann
   1. KANN das COS *3TDES\_Key* akzeptieren.
   2. KANN das COS *3TDES\_Key* ablehnen.
3. Aus Sicherheitsgründen sind die Teilschlüssel Ka, Kb und Kc paarweise verschieden zu wählen. Diese Anforderung richtet sich nicht an das COS, sondern an die externe Welt, welche derartige Schlüssel etwa im Rahmen der Personalisierung in die Smartcard einbringt.
4. Aus Sicherheitsgründen darf keiner der Teilschlüssel Ka, Kb oder Kc einen Wert haben, welcher einem schwachen oder halbschwachen DES-Schlüssel entspricht (siehe Glossar). Diese Anforderung richtet sich nicht an das COS, sondern an die externe Welt, welche derartige Schlüssel etwa im Rahmen der Personalisierung in die Smartcard einbringt.

#### AES-128-Schlüssel

Der Attributstyp *aes128Key* dient der Speicherung von Schlüsselmaterial für einen AES-128-Schlüssel.

1. (N008.400) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *aes128Key* MUSS ein Oktettstring mit sechzehn Oktette sein.
2. (N008.500) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *aes128Key* MUSS beliebig wählbar sein.

#### AES-192-Schlüssel

Der Attributstyp *aes192Key* dient der Speicherung von Schlüsselmaterial für einen AES-192-Schlüssel.

1. (N008.520) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *aes192Key* MUSS ein Oktettstring mit 24 Oktette sein.
2. (N008.525) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *aes192Key* MUSS beliebig wählbar sein.

#### AES-256-Schlüssel

Der Attributstyp *aes256Key* dient der Speicherung von Schlüsselmaterial für einen AES-256-Schlüssel.

1. (N008.540) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *aes256Key* MUSS ein Oktettstring mit 32 Oktette sein.
2. (N008.545) K\_Anwendungsspezifikation {K\_Karte}   
   Der Wert von *aes256Key* MUSS beliebig wählbar sein.

### Domainparameter für elliptische Kurven

Der Attributstyp *domainParameter* dient der Speicherung von Parametern, welche eine elliptische Kurve charakterisieren. Konform zu [BSI-TR-03111#Tabelle 2.1] werden die Domainparameter in diesem Dokument wie folgt dargestellt:

Tabelle 7: Liste der Domainparameter einer elliptischen Kurve

|  |  |
| --- | --- |
| **Parameter** | **Bedeutung** |
| *p* | Primzahl, welche die zugrunde liegende Gruppe Fp beschreibt |
| *a* | Erster Koeffizient der Weierstraßschen Gleichung |
| *b* | Zweiter Koeffizient der Weierstraßschen Gleichung |
| *G* | Ein Punkt auf der Kurve E(Fp), Basispunkt |
| *n* | Ordnung des Basispunktes G in E(Fp) |
| *h* | Cofaktor von G in E(Fp);  wegen (N002.500) gilt *h* = 1 für alle Kurven, die ein COS unterstützen MUSS |
| *L* | siehe (N008.600)b |
| τ | siehe (N008.600)c |
| *OID* | Object Identifier, der die elliptische Kurve referenziert, siehe (N008.600)d |

Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten, vergleiche Tabelle 7:

1. (N008.600) K\_Anwendungsspezifikation {K\_Karte}   
   *domainParameter* enthält
   1. alle Parameter aus Tabelle 7 und
   2. eine Zahl *L*, welche die minimale Anzahl Oktette angibt, die nötig sind, um *p* als vorzeichenlose Zahl zu codieren. Dieser Parameter wurde der „offiziellen“ Liste der Domainparameter hinzugefügt, da dieser Parameter in diesem Dokument vielfach verwendet wird. Allgemein gilt *L*=ceiling(log256 *p*). Wegen (N002.500) gilt:
      1. *L* = 32 für brainpoolP256r1   und   ansix9p256r1.
      2. *L* = 48 für brainpoolP384r1   und   ansix9p384r1.
      3. *L* = 64 für brainpoolP512r1.
   3. eine Zahl τ, welche die Bitlänge von *n* angibt (siehe [BSI-TR-03111#Table 1.1]). Dieser Parameter wurde der „offiziellen“ Liste von Domainparametern hinzugefügt, da dieser Parameter in diesem Dokument vielfach verwendet wird. Allgemein gilt τ=ceiling(log2 *n*). Wegen (N002.500) gilt:
      1. τ = 256 für brainpoolP256r1   und   ansix9p256r1.
      2. τ = 384 für brainpoolP384r1   und   ansix9p384r1.
      3. τ = 512 für brainpoolP512r1.
   4. einen Oktettstring *OID*, durch den die Domainparameter (*p*, *a*, *b*, *G*, *n*, *h*) weltweit eindeutig bestimmt werden. Dieser Parameter wurde der „offiziellen“ Liste der Domainparameter hinzugefügt, da in diesem Dokument Domainparameter vielfach per *OID* referenziert werden. *OID* MUSS so aus Tabelle 271 gewählt werden, dass damit eine elliptische Kurve gemäß (N002.500) referenziert wird.

### Privater Schlüssel

Der Attributstyp *privateKey* dient als Oberbegriff für *privateRsaKey* und *privateElcKey*.

#### Privater RSA-Schlüssel

Der Attributstyp *privateRsaKey* dient der Speicherung des privaten Teils eines asymmetrischen RSA-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

1. (N008.700) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *privateRsaKey* MUSS ein Attribut Modulus *n* besitzen, dessen Länge gemäß (N002.100)a gewählt werden MUSS.
2. (N008.710) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *privateRsaKey* MUSS ein Attribut *modulusLength* gemäß (N002.100)a besitzen.
3. (N008.800) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *privateRsaKey* MUSS die im Folgenden genannten Attribute besitzen und es MUSS gelten:
   1. Falls das Attribut *keyAvailable* (siehe (N018.200)) den Wert
      1. False besitzt, dann MÜSSEN alle Attribute den Wert AttributNotSet besitzen.
      2. True besitzt, dann MÜSSEN alle Attribute ganze Zahlen mit den unten genannten Eigenschaften sein.
   2. Attribut Modulus *n* mit   *n* = *p* *q*.
   3. Attribut öffentlicher Exponent *e* mit   gcd( *e*, (*p* – 1)(*q* – 1)) = 1.
   4. Attribut privater Exponent *d* mit   *e* *d* ≡ 1 mod lcm(*p* – 1, *q* – 1).
   5. Attribut Primzahl *p* mit   *p* ist prim und *p* < *q*.
   6. Attribut Primzahl *q* mit   *q* ist prim.
   7. Attrtibut CRT Exponent zu *dP* mit   *dP* = *d* mod (*p* – 1).
   8. Attrtibut CRT Exponent zu *dq* mit   *dq* = *d* mod (*q* – 1).
   9. Attribut *c* mit   *c* = *q*‑1 mod *p*.

#### Privater ELC-Schlüssel

Der Attributstyp *privateElcKey* dient der Speicherung des privaten Teils eines asymmetrischen ELC-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

1. (N008.900) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *privateElcKey* MUSS ein Attribut *domainParameter* gemäß (N008.600) besitzen, die gemäß (N008.600)d per *OID* referenzierbar sind.
2. (N009.000) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *privateElcKey* MUSS ein Attribut *d* besitzen, dessen Wert eine ganze Zahl im Intervall [1, *domainParameter.n* – 1] sein MUSS.

### Öffentlicher Schlüssel

#### Öffentlicher RSA-Schlüssel

Der Attributstyp *publicRsaKey* dient der Speicherung des öffentlichen Teils eines asymmetrischen RSA-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

1. (N009.100) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *publicRsaKey* MUSS ein Attribut Modulus *n* besitzen, dessen Länge gemäß (N002.100)a gewählt werden MUSS.
2. (N009.110) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *publicRsaKey* MUSS ein Attribut *modulusLength* gemäß (N002.100)a besitzen.
3. (N009.200) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *publicRsaKey* MUSS ein Attribut *e* gemäß (N002.100)b besitzen.

#### Öffentlicher ELC-Schlüssel

Der Attributstyp *publicElcKey* dient der Speicherung des öffentlichen Teils eines asymmetrischen ELC-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

1. (N009.300) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *publicElcKey* MUSS ein Attribut *domainParameter* gemäß (N008.600) besitzen, das gemäß (N008.600)d per *OID* referenzierbar ist.
2. (N009.400) K\_Anwendungsspezifikation {K\_Karte}   
   Ein *publicElcKey* MUSS ein Attribut *P* besitzen, das einen vom unendliche fernen Punkt *O* verschiedenen Punkt auf der durch die Domainparameter vorgegebenen Kurve bezeichnet.

### Transportschutz für ein Passwort

Der Attributstyp *transportStatus* gibt an, ob ein Passwort mit einem Transportschutz versehen ist. Der Wert dieses Attributes gibt also an, ob ein Verify Kommando möglich ist (siehe (N082.800)) und welche Variante von Change Reference Data anzuwenden ist. An der externen Schnittstelle des COS wird dieser Attributstyp im Rahmen des Kommandos Get Pin Status verwendet (siehe Tabelle 144).

1. (N009.500) K\_Anwendungsspezifikation {K\_Karte}   
   Für *transportStatus* gilt folgender Wertebereich:
   1. regularPassword
   2. Leer‑PIN
   3. Transport–PIN
2. (N009.600) K\_COS
   1. Das COS MUSS alle Werte aus (N009.500) unterstützen.
   2. Ein COS KANN weitere Werte für *transportStatus*
      1. unterstützen oder
      2. ablehnen.
3. (N009.700) K\_externeWelt {K\_Karte}   
   Wenn zur Aufhebung des Transportschutzes das Kommando Change Reference Data verwendet wird (siehe 14.6.1), dann MUSS in Abhängigkeit des Wertes von *transportStatus* an der physikalischen Schnittstelle eine Variante gemäß Tabelle 8 verwendet werden.
4. (N009.710) K\_COS   
   Wenn *oldSecret* oder *newSecret* in Tabelle 8 für eine Variante nicht erwähnt werden, so MUSS deren Inhalt im Rahmen von (N008.000) und (N008.100) beliebig wählbar sein.
5. Auch das Kommando Reset Retry Counter ist in der Lage, den Transportschutz aufzuheben (siehe Use Cases in 14.6.5.1 und 14.6.5.3, sowie (N081.200)b).

Tabelle 8: Aufheben des Transportschutzes

|  |  |
| --- | --- |
| **Transportschutz** | **Parameter für Change Reference Data** |
| regularPassword | P1=´00´ |
| Leer-PIN | P1=´01´, *oldSecret* = ´´, das heißt leerer String |
| Transport-PIN | P1=´00´ |

## File (normativ)

Dieses Unterkapitel beschreibt file-orientierte Objekttypen. File wird in diesem Zusammenhang als Oberbegriff für Ordner (siehe 8.3.1) und Datei (siehe 8.3.2) verwendet.

### Ordner

Ordner dienen der hierarchischen Anordnung von Objekten in einem Objektsystem. In diesem Dokument wird Ordner als Oberbegriff für

* Applikationen (siehe 8.3.1.1),
* Dedicated Files (siehe 8.3.1.2) und
* Application Dedicated Files (siehe 8.3.1.3)

verwendet. Gemäß der Norm [ISO/IEC 7816-4] gelten für einen Ordner folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N009.800) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Ordner MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
2. (N009.850) K\_Anwendungsspezifikation {K\_Karte}, Option\_logische\_Kanäle   
   Ein Ordner MUSS genau ein Attribut *shareable* vom Typ Boolean besitzen. Im Rahmen dieser Spezifikation MUSS der Wert des Attributes stets True sein, wodurch angezeigt wird, dass dieser Ordner in mehr als einem logischen Kanal als *currentFolder* verwendbar ist (siehe (N029.900)a).
3. (N009.900) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Ordner MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
4. (N010.000) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Ordner MUSS eine (möglicherweise leere) Liste *children* mit Kindobjekten besitzen.
   1. Unter Berücksichtigung der maximalen Schachtelungstiefe (siehe (N020.400)) MÜSSEN Listenelemente der
      1. Objekttypen
         1. Applikation (siehe 8.3.1.1)
         2. Dedicated File (siehe 8.3.1.2)
         3. Application Dedicated File (siehe 8.3.1.3)
         4. Datei (siehe 8.3.2)
         5. Reguläres Passwort (siehe 8.4)
         6. Multireferenz-Passwort (siehe 8.5)
         7. symmetrisches Authentisierungsobjekt (siehe 8.6.1)
         8. Privates Schlüsselobjekt (siehe 8.6.3)

unterstützt werden.

* + 1. Objekttypen Option\_kontaktlose\_Schnittstelle
       1. Symmetrisches Kartenverbindungsobjekt (siehe 8.6.2)

unterstützt werden.

* 1. Das COS KANN weitere Objekttypen als Listenelemente
     1. unterstützen oder
     2. ablehnen.
  2. Für alle Elemente der Liste *children*, welche die Kindobjekte eines Ordners enthält, gilt: In der List *children* eines Ordners DARF es KEINE Kindobjekte geben,
     1. deren Attribut *fileIdentifier* (sofern vorhanden, siehe (N010.600) und (N010.800)) denselben Wert besitzt.
     2. deren Attribut *shortFileIdentifier* (sofern vorhanden, siehe (N010.900)) denselben Wert besitzt.
     3. deren Attribut *pwdIdentifier* (sofern vorhanden, siehe (N015.000)) denselben Wert besitzt.
     4. deren Attribut *keyIdentifier* (sofern vorhanden, siehe (N016.400) und (N017.100)) denselben Wert besitzt.

1. (N010.005) K\_COS   
   Ein Ordner MUSS die folgenden Kommandos unterstützen:
   1. Activate (siehe 14.2.1),
   2. Deactivate (siehe 14.2.3),
   3. Delete (siehe 14.2.4).
   4. Fingerprint (siehe 14.9.2),

* 1. Get Random (siehe 14.9.5, nur Option\_logische\_Kanäle),
  2. Load Application (siehe 14.2.5),
  3. Select (siehe 14.2.6),
  4. Terminate DF (siehe 14.2.8).

Bevor diese Kommandos in der Lage sind mit dem Ordner zu arbeiten, ist er zu selektieren. Dies geschieht mittels Select-Kommando (siehe 14.2.6).

1. In (N010.000)a werden im Vergleich zur Vorgängerversion nun auch Applikationen (siehe 8.3.1.1) genannt, vergleiche auch (N020.200).

Einem Ordner sind weitere, kanalspezifische Attribute zugeordnet, die typischerweise einem flüchtig (etwa im RAM) gespeicherten Kanalkontext (siehe (N030.000)) zugerechnet werden. Sie werden typischerweise im Rahmen einer Selektion des Ordners verändert, aber auch durch Benutzerverifikation oder Komponentenauthentisierung.

#### Applikation

Eine Applikation ist ein Ordner, der nur mittels AID selektierbar ist. Dies ist der Hauptunterschied zu einem DF (siehe 8.3.1.2).

Gemäß der Norm [ISO/IEC 7816-4] gelten für eine Applikation folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N010.100) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Applikation ist eine Erweiterung von Ordner und MUSS deshalb den Anforderungen aus 8.3.1 genügen.
2. (N010.200) K\_Anwendungsspezifikation {K\_Karte}   
   Das Attribut *applicationIdentifier* (AID) MUSS ein Oktettstring sein.
   1. Die Länge von *applicationIdentifier* MUSS im Intervall [5, 16] liegen.
   2. Die Oktette in *applicationIdentifier* MÜSSEN beliebig wählbar sein.
   3. Ein COS KANN *applicationIdentifier*, die gegen diese Regeln verstoßen,
      1. akzeptieren oder
      2. ablehnen.
3. (N010.300) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Applikation MUSS mindestens einen *applicationIdentifier* besitzen.
4. (N010.400) K\_Anwendungsspezifikation {K\_Karte}
   1. Eine Applikation DARF NICHT mehr als zwei *applicationIdentifier* besitzen.
   2. Ein COS KANN für Applikationen mehr als zwei *applicationIdentifier* erlauben.

#### Dedicated File

Ein Dedicated File (DF) ist ein Ordner, der nur mittels File Identifier selektierbar ist. Dies ist der Hauptunterschied zu einer Applikation (siehe 8.3.1.1).

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein DF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N010.500) K\_Anwendungsspezifikation {K\_Karte}   
   Ein DF ist eine Erweiterung von Ordner und MUSS deshalb den Anforderungen aus 8.3.1 genügen.
2. (N010.600) K\_Anwendungsspezifikation {K\_Karte}   
   Ein DF MUSS genau ein Attribut vom Typ *fileIdentifier* (siehe 8.1.1) besitzen.

#### Application Dedicated File

Ein Application Dedicated File (ADF) ist ein Ordner, der sowohl mittels *applicationIdentifier* als auch *fileIdentifier* selektierbar ist. ADF wird in diesem Dokument also als „Vereinigungsmenge“ von Applikation (siehe 8.3.1.1) und DF (siehe 8.3.1.2) gesehen. Dementsprechend gelten für ein ADF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N010.700) K\_Anwendungsspezifikation {K\_Karte}   
   Ein ADF MUSS die Eigenschaften von Applikation und DF in sich vereinen. Deshalb MUSS es sowohl die Anforderungen aus 8.3.1.1 als auch aus 8.3.1.2 erfüllen.

### Datei

Eine Datei dient in diesem Dokument als Oberbegriff für transparente EF (siehe 8.3.2.1) und strukturierte EF (siehe 8.3.2.2).

Gemäß der Norm [ISO/IEC 7816-4] gelten für eine Datei folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N010.800) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Datei MUSS genau ein Attribut vom Typ *fileIdentifier* (siehe 8.1.1) besitzen.
2. (N010.900) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Datei MUSS eine Liste mit Elementen vom Typ *shortFileIdentifier* (siehe 8.1.2) unterstützen. Diese Liste ist entweder
   1. leer (die Datei hat keinen *shortFileIdentifier*) oder
   2. enthält genau ein Element (die Datei hat genau einen *shortFileIdentifier*).
3. (N011.000) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Datei MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
4. (N011.050) K\_Anwendungsspezifikation {K\_Karte}, Option\_logische\_Kanäle   
   Eine Datei MUSS genau ein Attribut *shareable* vom Typ Boolean besitzen. Im Rahmen dieser Spezifikation MUSS der Wert des Attributes stets True sein, wodurch angezeigt wird, dass diese Datei in mehr als einem logischen Kanal als *currentEF* verwendbar ist (siehe (N029.900)m]).
5. (N011.100) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Datei MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
6. (N011.200) K\_Anwendungsspezifikation {K\_Karte}
   1. Eine Datei MUSS ein Attribut *flagTransactionMode* vom Typ Boolean besitzen.
   2. Ein COS KANN pro Datei ein solches Attribut individuell verwalten.
   3. Ein COS KANN dieses Attribut für alle Dateien implizit auf True setzen.
7. (N011.300) K\_Anwendungsspezifikation {K\_Karte}
   1. Eine Datei MUSS ein Attribut *flagChecksum* vom Typ Boolean besitzen.
   2. Ein COS KANN pro Datei ein solches Attribut individuell verwalten.
   3. Ein COS KANN dieses Attribut für alle Dateien implizit auf True setzen.
8. Das Setzen des Flags flagTransactionMode verringert die Performanz bei Schreibzugriffen.
9. Das Setzen des Flags flagChecksum verringert die Performanz bei Schreib- und Lesezugriffen.

#### Transparentes Elementary File

Ein transparentes Elementary File (transparentes EF) dient der Speicherung eines Oktettstrings, wobei beliebige Teile des Oktettstrings zugreifbar sind. Der Inhalt von Oktettstrings in transparenten EF, welche im Rahmen einer Anwendungsspezifikation definiert werden, wird von einem COS lediglich gespeichert, niemals aber interpretiert oder für karteninterne Prozesse verwendet. Der Oktettstring wird in „data units“ unterteilt (siehe [ISO/IEC 7816-4]). Die „data units“ werden über einen Offset referenziert.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein transparentes EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N011.400) K\_Anwendungsspezifikation {K\_Karte}   
   Ein transparentes EF ist eine Erweiterung von Datei und MUSS deshalb den Anforderungen aus 8.3.2 genügen.
2. (N011.500) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein transparentes EF MUSS genau ein Attribut *numberOfOctet* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 32.768] sein MUSS.
   2. Ein COS KANN *numberOfOctet* , falls es außerhalb des Intervalls liegt,
      1. ablehnen oder
      2. akzeptieren.
3. (N011.510) K\_Anwendungsspezifikation {K\_Karte}   
   Ein transparentes EF MUSS genau ein Attribut *positionLogicalEndOfFile* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [0, *numberOfOctet*] sein MUSS.
4. (N011.600) K\_Anwendungsspezifikation {K\_Karte}   
   Ein transparentes EF MUSS ein Attribut *body* vom Typ Oktettstring besitzen. Die Anzahl der Oktette in *body* ist gleich *numberOfOctet*.
5. (N011.700) K\_COS   
   Die Größe einer „data unit“ in *body* MUSS 1 Oktett betragen.
6. (N011.800) K\_COS   
   Der Offset des ersten Oktetts in *body* ist null. Ist *i* der Offset des *i*–ten Oktetts, dann ist (*i* + 1) der Offset des nächsten Oktetts.
7. (N011.900) K\_COS   
   Ein transparentes EF MUSS die folgenden Kommandos unterstützen:
   1. Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das transparente EF zu selektieren. Dies geschieht entweder mittels Select-Kommando (siehe 14.2.6.13 und 14.2.6.14), oder mittels Short File Identifier, der als Parameter dem zugreifenden Kommando mitgegeben wird.
      1. Erase Binary (siehe 14.3.1),
      2. Read Binary (siehe 14.3.2),
      3. Set Logical Eof (siehe 14.3.4),
      4. Update Binary (siehe 14.3.5).
      5. Write Binary (siehe 14.3.6).
   2. Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das transparente EF zu selektieren. Dies geschieht mittels Select-Kommando (siehe 14.2.6.13 und 14.2.6.14).
      1. Activate (siehe 14.2.1),
      2. Deactivate (siehe 14.2.3),
      3. Delete (siehe 14.2.4).
      4. Terminate (siehe 14.2.9).
8. (N012.000) K\_COS   
   Ein transparentes EF KANN weitere Kommandos
   1. unterstützen oder
   2. ablehnen.
9. Das Konzept „logical End Of File“ unterscheidet zwischen einer Dateigröße, die mittels Update Binary beschreibbar ist und einer „logischen“ Dateigröße, die mittels Read Binary auslesbar ist. Dies spiegelt sich in den Attributen numberOfOctet und positionLogicalEndOfFile wieder. Dabei entspricht numberOfOctet einer maximalen Dateigröße, die im Rahmen von Update Binary komplett adressierbar ist, und positionLogicalEndOfFile einer tatsächlichen Dateigröße, die im Rahmen von Read Binary komplett adressierbar ist. Während der Wert von numberOfOctet unveränderlich ist, ist es möglich, dass sich der Wert von positionLogicalEndOfFile im Rahmen der Kommandos Erase Binary, Update Binary oder Write Binary ändert.

#### Strukturiertes Elementary File

Ein strukturiertes Elementary File (strukturiertes EF) dient der Speicherung einer Liste von Rekords (siehe 8.1.5). Ein Zugriff auf beliebige Listenelemente ist möglich. Der Inhalt des Oktettstrings eines Rekords in strukturierten EF, welche im Rahmen einer Anwendungsspezifikation definiert werden, wird von einem COS lediglich gespeichert, niemals aber interpretiert oder für karteninterne Prozesse verwendet.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein strukturiertes EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N012.100) K\_Anwendungsspezifikation {K\_Karte}   
   Ein strukturiertes EF ist eine Erweiterung von Datei und MUSS deshalb den Anforderungen aus 8.3.2 genügen.
2. (N012.200) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein strukturiertes EF MUSS ein Attribut *recordList* vom Typ Liste mit Elementen vom Typ Rekord besitzen. Die Anzahl der Listenelemente MUSS im Intervall [0, 254] liegen.
   2. Ein COS KANN eine größere Anzahl von Listenelementen
      1. zulassen oder
      2. ablehnen.
3. (N012.300) K\_COS   
   Das *i*-te Element in *recordList* MUSS die Rekordnummer *i* besitzen.
4. (N012.400) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein strukturiertes EF MUSS genau ein Attribut *maximumNumberOfRecords* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 254] sein MUSS.
   2. Ein COS KANN *maximumNumberOfRecords*, falls dessen Wert außerhalb dieses Intervalls liegt,
      1. akzeptieren, oder
      2. ablehnen.
5. (N012.500) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein strukturiertes EF MUSS genau ein Attribut *maximumRecordLength* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 255] sein MUSS.
   2. Ein COS KANN *maximumRecordLength*, falls dessen Wert außerhalb dieses Intervalls liegt,
      1. akzeptieren oder
      2. ablehnen.
6. (N012.600) K\_Anwendungsspezifikation {K\_Karte}   
   Ein strukturiertes EF MUSS genau ein Attribut *flagRecordLifeCycleStatus* vom Typ Boolean besitzen. Falls dieses Attribut
   1. True ist, dann MÜSSEN alle Elemente in *recordList* einen *lifeCycleStatus* besitzen.
   2. False ist, dann DARF KEIN Element in *recordList* einen *lifeCycleStatus* besitzen.
7. (N012.700) K\_COS   
   Ein strukturiertes EF MUSS die folgenden Kommandos unterstützen:
   1. Bevor eines der folgenden Kommandos in der Lage ist spezifikationsgemäß zu arbeiten, ist das strukturierte EF zu selektieren. Dies geschieht entweder mittels Select-Kommando (siehe 14.2.6.13 und 14.2.6.14), oder mittels Short File Identifier, der als Parameter dem zugreifenden Kommando mitgegeben wird.
      1. Activate Record (siehe 14.4.1),
      2. Append Record (siehe 14.4.2),
      3. Deactivate Record (siehe 14.4.3),
      4. Delete Record (siehe 14.4.4),
      5. Erase Record (siehe 14.4.5),
      6. Read Record (siehe 14.4.6),
      7. Search Record (siehe 14.4.7),
      8. Update Record (siehe 14.4.8).
   2. Bevor eines der folgenden Kommandos in der Lage ist spezifikationsgemäß zu arbeiten, ist das strukturierte EF zu selektieren. Dies geschieht mittels Select-Kommando (siehe 14.2.6.13 und 14.2.6.14).
      1. Activate (siehe 14.2.1),
      2. Deactivate (siehe 14.2.3),
      3. Delete (siehe 14.2.4).
      4. Terminate (siehe 14.2.9).
8. (N012.800) K\_COS   
   Ein strukturiertes EF KANN weitere Kommandos
   1. unterstützen oder
   2. ablehnen.

##### Linear variables Elementary File

Ein linear variables Elementary File (linear variables EF) dient der Speicherung einer Liste von Elementen des Typs *record* (siehe 8.1.5), wobei es möglich ist, dass der Oktettstring eines jeden Listenelementes eine andere Anzahl von Oktette enthält.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein linear variables EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N012.900) K\_Anwendungsspezifikation {K\_Karte}   
   Ein linear variables EF ist eine Erweiterung des strukturierten EF und MUSS deshalb den Anforderungen aus 8.3.2.2 genügen.
2. (N013.000) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein linear variables EF MUSS genau ein Attribut *numberOfOctet* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 64.770] sein MUSS.
   2. Ein COS KANN *numberOfOctet*, falls es außerhalb des Intervalls liegt,
      1. akzeptieren oder
      2. ablehnen.
3. (N013.100) K\_COS   
   Wird mittels Append Record der Liste ein neuer *record* hinzugefügt, so MUSS der neue *record* am Ende der Liste eingefügt werden.

##### Linear fixes Elementary File

Ein linear fixes Elementary File (linear fixes EF) dient der Speicherung einer Liste von Elementen des Typs *record* (siehe 8.1.5), wobei der Oktettstring eines jeden Listenelementes dieselbe Anzahl von Oktette enthält.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein linear fixes EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N013.200) K\_Anwendungsspezifikation {K\_Karte}   
   Ein linear fixes EF ist eine Erweiterung des strukturierten EF und MUSS deshalb den Anforderungen aus 8.3.2.2 genügen.
2. (N013.300) K\_Anwendungsspezifikation {K\_Karte}   
   Jeder *record* in *recordList* MUSS *maximumRecordLength* Oktette besitzen.
3. (N013.400) K\_COS   
   Wird mittels Append Record der Liste ein neuer *record* hinzugefügt, so MUSS der neue *record* am Ende der Liste eingefügt werden.
4. (N013.410) K\_Anwendungsspezifikation {K\_Karte}   
   Das Produkt aus *maximumNumberOfRecords* und *maximumRecordLength* MUSS kleiner gleich 64.770 sein (vergleiche auch (N013.000)).

##### Zyklisches Elementary File

Ein zyklisches Elementary File (zyklisches EF) dient der Speicherung einer Liste von  
Elementen des Typs *record* (siehe 8.1.5), wobei der Oktettstring eines jeden Listenelementes dieselbe Anzahl von Oktette enthält.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein zyklisches EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N013.500) K\_Anwendungsspezifikation {K\_Karte}   
   Ein zyklisches EF ist eine Erweiterung des strukturierten EF und MUSS deshalb den Anforderungen aus 8.3.2.2 genügen.
2. (N013.600) K\_Anwendungsspezifikation {K\_Karte}   
   Jeder *record* in *recordList* MUSS *maximumRecordLength* Oktette besitzen.
3. (N013.700) K\_COS   
   Wird mittels Append Record der Liste ein neuer *record* hinzugefügt, so MUSS der neue *record* am Anfang der Liste eingefügt werden.
4. (N013.800) K\_Anwendungsspezifikation {K\_Karte}   
   Zyklische Elementary Files MÜSSEN für das Attribut *flagRecordLifeCycleStatus* den Wert False verwenden.
5. (N013.810) K\_Anwendungsspezifikation {K\_Karte}   
   Das Produkt aus *maximumNumberOfRecords* und *maximumRecordLength* MUSS kleiner gleich 64.770 sein (vergleiche auch (N013.000)).

### File Control Parameter

Die File Control Parameter enthalten Attribute eines Files. Im Rahmen dieses Dokumentes werden sie lediglich an der Schnittstelle Interpreter (siehe Abbildung 1) in den Antwortdaten eines Select-Kommandos (siehe (N048.300)a) sichtbar. Für die Codierung der File Control Parameter gilt:

1. (N013.900) K\_COS   
   Die File Control Parameter (FCP) MÜSSEN gemäß DER-TLV in einem DO\_FCP codiert werden, welches ein Tag = ´62´ besitzen MUSS.
2. (N014.000) K\_COS   
   ´80´: Falls das File vom Typ transparentes EF (siehe 8.3.2.1) oder linear variables EF (siehe 8.3.2.2.1) ist, genau dann MUSS DO\_FCP ein DO\_Size enthalten. Es gilt:
   1. DO\_Size MUSS ein Tag = ´80´ besitzen.
   2. Das Wertfeld KANN eine beliebige Länge besitzen.
   3. Wird der Oktettstring des Wertfeldes mittels OS2I(…) konvertiert, dann MUSS diese Zahl gleich *numberOfOctet* (siehe (N011.500), bzw. (N013.000)) sein.
3. (N014.100) K\_COS   
   ´82´: DO\_FCP MUSS ein DO\_FileDescriptor enthalten. Falls File vom Typ
   1. Ordner ist, dann gilt:
      1. DO\_FileDescriptor MUSS ein Tag = ´82´ besitzen
      2. Das Wertfeld MUSS aus einem Oktett bestehen.
      3. Das Wertfeld MUSS einen Wert aus der Menge {´38´, ´78´} enthalten.
   2. Transparentes EF ist, dann gilt:
      1. DO\_FileDescriptor MUSS ein Tag = ´82´ besitzen
      2. Das Wertfeld MUSS aus einem Oktett bestehen.
      3. Das Wertfeld MUSS einen Wert aus der Menge {´01´, ´41´} enthalten.
   3. Strukturiertes EF ist, dann gilt:
      1. DO\_FileDescriptor MUSS ein Tag = ´82´ besitzen.
      2. Die Länge des Wertfeldes von DO\_FileDescriptor MUSS aus der Menge {5, 6} sein.
      3. Das erste Oktett des Wertfeld MUSS einen Wert aus der Menge
         1. {´02´, ´42´} besitzen, falls File vom Typ linear fixes EF ist.
         2. {´04´, ´44´} besitzen, falls File vom Typ linear variables EF ist.
         3. {´06´, ´46´} besitzen, falls File vom Typ zyklisches EF ist.
      4. Das zweite Oktett im Wertfeld MUSS den Wert ´41´ besitzen.
      5. Das 3. und 4. Oktett MUSS gleich I2OS( *maximumRecordLength*, 2) sein.
      6. Das fünfte Oktett MUSS bzw. das fünfte und sechste Oktett MÜSSEN so gewählt werden, dass deren OS2I Konvertierung das Attribut *maximumNumberOfRecords* liefert.
4. (N014.200) K\_COS   
   ´83´: Falls das File ein Attribut vom Typ *fileIdentifier* gemäß 8.1.1 besitzt, genau dann MUSS DO\_FCP ein DO\_FID enthalten. Es gilt:   
   DO\_FID = ´83   02   I2OS( *fileIdentifier*, 2 )´.
5. (N014.300) K\_COS   
   ´84´: Falls das File Attribute vom Typ *applicationIdentifier* gemäß (N010.200) besitzt, dann MUSS DO\_FCP jedes dieser Attribute enthalten. Jedes dieser Attribute MUSS als Wertfeld in einem DO\_AID codiert werden. Es gilt:   
   DO\_AID = ´84 || I2OS(OctetLength( *applicationIdentifier*, 1 ) || *applicationIdentifier*´.
6. (N014.400) K\_COS   
   ´88´: Falls das File vom Typ Datei ist, genau dann MUSS DO\_FCP ein DO\_SFI enthalten. Falls die Datei
   1. ein Attribut *shortFileIdentifier* gemäß 8.1.2 besitzt, gilt:   
      DO\_SFI = ´88   01   I2OS( 8 *shortFileIdentifier*, 1 )´.
   2. kein Attribut *shortFileIdentifier* besitzt, gilt:   
      DO\_SFI = ´88 00´.
7. (N014.500) K\_COS   
   ´8A´: Der physikalische Wert des Attributes *lifeCycleStatus* (siehe 8.1.3) MUSS in DO\_FCP enthalten sein. Es MUSS als Wertfeld in einem DO\_LCS mit Tag = ´8A´ codiert werden. Der Wert von *lifeCycleStatus* MUSS in einem Oktett gemäß [ISO/IEC 7816-4#Tabelle 14] codiert werden. Das bedeutet
   1. „Operational state (active)“ MUSS aus der Menge {´05´, ´07´} stammen.
   2. „Operational state (deactivated)“ MUSS aus der Menge {´04´, ´06´} stammen.

* 1. „Termination state" MUSS aus der Menge {´0C´. ´0D´, ´0E´, ´0F´} stammen.

1. (N014.600) K\_COS   
   ´8F´: Falls das File vom Typ strukturiertes EF ist und das Attribut *flagRecordLifeCycleStatus* hat den Wert
   1. False, dann KANN DO\_FCP
      1. ein DO\_ProfileIdentifier = ´8F 01 00´ enthalten.
      2. kein DO mit Tag = ´8F´ enthalten.
   2. True, genau dann MUSS DO\_FCP ein DO\_ProfileIdentifier = ´8F 01 XX´ enthalten. OS2I(´XX´) MUSS eine ungerade Zahl kleiner als 128 sein.
2. (N014.700) K\_COS   
   ´C5´: Falls das File vom Typ transparentes EF (siehe 8.3.2.1) ist, genau dann MUSS DO\_FCP ein DO\_ReadSize enthalten. Es gilt:
   1. DO\_ReadSize MUSS ein Tag = ´C5´ besitzen.
   2. Das Wertfeld KANN eine beliebige Länge besitzen.
   3. Wird der Oktettstring des Wertfeldes mittels OS2I(…) konvertiert, dann MUSS diese Zahl gleich *positionLogicalEndOfFile* (siehe (N011.510)) sein.
3. (N014.800) K\_COS   
   In DO\_FCP KÖNNEN weitere Datenobjekte enthalten sein. Deren Codierung und Bedeutung KANN herstellerspezifisch sein.
4. (N014.900) K\_COS   
   Für die Reihenfolge der Datenobjekte in DO\_FCP gilt:
   1. Die Reihenfolge aller Datenobjekte ist herstellerspezifisch.
   2. Falls in DO\_FCP vorhanden, dann MÜSSEN die Datenobjekte DO\_Size, DO\_FileDescriptor, DO\_FID, DO\_AID, DO\_SFI, DO\_LCS, DO\_ProfileIdentifier und DO\_ReadSize vollständig in den ersten 256 Oktetten des Oktettstrings DO\_FCP enthalten sein.

## Reguläres-Passwort (normativ)

1. Mit dieser Version des Dokumentes wird der Objekttyp "Multireferenz-Passwort" eingeführt (siehe 8.5). An der Schnittstelle ist es nicht möglich zwischen einem "Regulären-Passwort", welches alle Attribute selbst besitzt und einem "Multireferenz-Passwort", welches gewisse Attribute eines "Regulären-Passworts" nachnutzt, zu unterscheiden. Diesbezüglich ist die Unterscheidung in diesem Dokument willkürlich und einerseits darin begründet die Unterschiede zu älteren Versionen dieses Dokumentes nicht unnötig zu vergrößern und andererseits eine einfache und präzise Darstellung zu ermöglichen. Als Konsequenz werden die Begriffe "Reguläres-Passwort" und "Multireferenz-Passwort" nur verwendet, wenn die Unterscheidung wichtig ist. An Stellen, wo diese Unterscheidung nicht wichtig ist wird der Begriff "Passwort" verwendet.

Ein „Reguläres-Passwort“ dient der Speicherung eines Geheimnisses, das in der Regel nur einem Karteninhaber bekannt ist. Das COS wird bestimmte Dienste erst dann zulassen, wenn dieses Geheimnis im Rahmen einer Benutzerverifikation erfolgreich präsentiert wurde. Die Notwendigkeit der Benutzerverifikation lässt sich einschalten (enable) oder abschalten (disable).

Typischerweise ist das Geheimnis aus Sicherheitsgründen änderbar. Ebenfalls aus Sicherheitsgründen wird die Operation „Benutzerverifikation“ für ein bestimmtes Passwort vom COS gesperrt, wenn die Benutzerverifikation mit diesem Passwort zu oft fehlschlug. Typischerweise ist es möglich, diese Sperrung aufzuheben.

1. In anderen Dokumenten wird in der Regel der Begriff PIN verwendet und bezeichnet dann teilweise das einem Karteninhaber bekannte sechsstellige Geheimnis („Bitte geben Sie Ihre PIN ein.“) und teilweise das Objekt in seiner Gesamtheit („Die PIN ist blockiert.“). In diesem Dokument wird der sprachlichen Klarheit wegen, weitgehend auf den Begriff PIN verzichtet und zur Bezeichnung des Objekttyps stets der Begriff Passwort verwendet.

Bei der Spezifikation von Applikationen sind folgende Regeln einzuhalten:

1. (N015.000) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein Reguläres-Passwort MUSS genau ein Attribut *pwdIdentifier* besitzen, dessen Wert eine ganz Zahl aus dem Intervall [0, 31] sein MUSS.
   2. Ein COS KANN weitere Werte für *pwdIdentifier*
      1. unterstützen oder
      2. ablehnen.
2. (N015.050) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Reguläres-Passwort MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
3. (N015.100) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Reguläres-Passwort MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
4. (N015.200) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Reguläres-Passwort MUSS genau ein Attribut *secret* vom Typ *pin* (siehe 8.1.7) besitzen.
5. (N015.300) K\_Anwendungsspezifikation {K\_Karte}
   1. Das Reguläres-Passwort MUSS genau ein Attribut *minimumLength* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [4, 12] sein MUSS.
   2. Ein COS KANN weitere Werte für die *minimumLength*
      1. unterstützen oder
      2. ablehnen.
6. (N015.310) K\_Anwendungsspezifikation {K\_Karte}
   1. Das Reguläres-Passwort MUSS genau ein Attribut *maximumLength* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [*minimumLength*, 12] sein MUSS.
   2. Ein COS KANN weitere Werte für die *maximumLength*
      1. unterstützen oder
      2. ablehnen.
7. (N015.400) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein Reguläres-Passwort MUSS genau ein Attribut *startRetryCounter* besitzen, dessen Wert eine ganze Zahl im Intervall [1, 15] sein MUSS.
   2. Ein COS KANN weitere Werte für *startRetryCounter*
      1. unterstützen oder
      2. ablehnen.
8. (N015.500) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein Reguläres-Passwort MUSS genau ein Attribut *retryCounter* besitzen, dessen Wert eine ganze Zahl im Intervall [0, 15] sein MUSS.
   2. Ein COS KANN weitere Werte für *retryCounter*
      1. unterstützen oder
      2. ablehnen.
9. (N015.600) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Reguläres-Passwort MUSS genau ein Attribut *transportStatus* gemäß 8.2.5 besitzen.
10. (N015.610) K\_COS {K\_Karte}   
    Falls das Attribut *transportStatus* eines regulären Passwortobjektes vom Zustand "Leer-PIN" in den Zustand "regularPassword" wechselt, dann MUSS auf eine herstellerspezifische Art und Weise sichergestellt sein, dass der Use Case "Setzen eines Benutzergeheimnisses" gemäß 14.6.1.2 für dieses reguläre Passwortobjekt und alle auf dieses reguläre Passwortobjekt verweisenden Multireferenz-Passwortobjekte nicht mehr ausführbar ist.
11. (N015.700) K\_Anwendungsspezifikation {K\_Karte}   
    Ein Reguläres-Passwort MUSS genau ein Attribut *flagEnabled* vom Typ Boolean besitzen. Es wird im Rahmen der Zugriffsregelauswertung verwendet (siehe (N022.200)a.2).
12. (N015.800) K\_Anwendungsspezifikation {K\_Karte}   
    Ein Reguläres-Passwort MUSS genau ein Attribut *startSsecList* besitzen
    1. Die Liste
       1. MUSS mindestens ein Element und
       2. DARF NICHT mehr als vier Elemente enthalten.
    2. Ein COS KANN Listen mit mehr als vier Elementen
       1. unterstützen oder
       2. ablehnen.
    3. Jedes Listenelement MUSS
       1. genau einen *seIdentifier* gemäß (N007.900) und
       2. genau eine ganze Zahl *startSsec* enthalten. *startSsec* MUSS aus dem Intervall [1, 250] sein oder den Wert “unendlich“ repräsentieren.
    4. Ein COS KANN weitere Werte für *startSsec*
       1. unterstützen oder
       2. ablehnen.
13. (N015.900) K\_Anwendungsspezifikation {K\_Karte}   
    Ein Reguläres-Passwort MUSS genau ein Attribut *PUK* vom Typ *pin* (siehe 8.1.7) besitzen, welches ein Geheimnis repräsentiert, welches das Zurücksetzten eines abgelaufenen *retryCounter* ermöglicht.
14. (N016.000) K\_Anwendungsspezifikation {K\_Karte}
    1. Ein Reguläres-Passwort MUSS ein Attribut *pukUsage* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [0, 15] sein MUSS.
    2. Ein COS KANN weitere Werte für *pukUsage*
       1. unterstützen oder
       2. ablehnen.
15. (N016.100) K\_externeWelt {K\_Karte}
    1. Die Ziffernfolgen *secret* und *PUK* MÜSSEN an der Schnittstelle Interpreter (siehe Abbildung 1) stets als Format-2-PIN-Block (siehe (N008.100)) übertragen werden
    2. Ein COS KANN weitere Übertragungsformate
       1. unterstützen oder
       2. ablehnen.
16. (N016.200) K\_COS   
    Ein Passwort MUSS die folgenden Kommandos unterstützen:
    1. Bevor eines der folgenden Kommandos in der Lage ist mit dem Passwort zu arbeiten, ist das Passwort zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.
       1. Change Reference Data (siehe 14.6.1),
       2. Disable Verification Requirement (siehe 14.6.2),
       3. Enable Verification Requirement (siehe 14.6.3),
       4. Get Pin Status (siehe 14.6.4),
       5. Reset Retry Counter (siehe 14.6.5),
       6. Verify (siehe 14.6.6).
    2. Bevor eins der folgenden Kommandos in der Lage ist mit dem Passwort zu arbeiten, ist das Passwort zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.
       1. Activate (siehe 14.2.1),
       2. Deactivate (siehe 14.2.3),
       3. Delete (siehe 14.2.4).
       4. Terminate (siehe 14.2.9).
17. (N016.210) K\_externeWelt {K\_Karte}   
    Die externe Welt DARF ein reguläres Passwort NICHT löschen (Use Case entsprechend 14.2.4.4), solange es Multireferenz-Passwortobjekte gibt, die mittels ihres Attributes *passwordReference* (siehe (N016.320)f) auf dieses reguläre Passwortobjekt verweisen.)
18. (N016.300) K\_COS   
    Ein Reguläres-Passwort KANN weitere Kommandos
    1. unterstützen oder
    2. ablehnen.

Einem Regulären-Passwort ist ein weiteres, kanalspezifisches Attribut *securityStatusEvaluationCounter* (siehe (N029.900)k) zugeordnet, das typischerweise einem flüchtig (etwa im RAM) gespeicherten Sicherheitszustand (siehe 8.9) zugerechnet wird. Es wird im Rahmen von Kommandos zur Benutzerverifikation (siehe 14.6.6) geändert.

1. Absichtlich ist einem Passwort kein Attribut shareable zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein Passwortobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.

## Multireferenz-Passwort (normativ)

Ein Multireferenz-Passwort ermöglicht die Nutzung eines Geheimnisses, welches als Attribut in einem Regulären-Passwort gespeichert ist (siehe (N015.200)), allerdings unter Bedingungen, die von denen des Regulären-Passwortes bewusst abweichen. Wird das Verhalten an der Schnittstelle zur Smartcard betrachtet, so scheinen Reguläres-Passwort und Multireferenz-Passwort gewisse Attribute gemeinsam zu nutzen, während es für die übrigen Attribute möglich ist diese individuell verschieden zu wählen. Aus Sicht eines Karteninhabers scheinen die im bekannten Geheimnisse (*secret* gemäß (N015.200) und *PUK* gemäß (N015.900)) unter mehreren "Referenzen" (genauer *pwdIdentifiern* gemäß (N015.000)) ansprechbar zu sein.

Bei der Spezifikation von Applikationen sind folgende Regeln einzuhalten:

1. (N016.320) K\_Anwendungsspezifikation {K\_Karte}   
   Folgende Attribute lassen sich für ein Multireferenz-Passwort individuell festlegen, das heißt diese Attribute werden nicht gemeinsam mit anderen Regulären-Passwörtern oder anderen Multireferenz-Passwörtern genutzt:
   1. *pwdIdentifier*: Ein Multireferenz-Passwort MUSS genau ein Attribut *pwdIdentifier* besitzen. Für dieses Attribut gelten die in (N015.000) genannten Anforderungen.
   2. *lifeCycleStatus*: Eine Multireferenz-Passwort MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
   3. *accessRules*: Ein Multireferenz-Passwort MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
   4. *startSsecList*: Ein Multireferenz-Passwort MUSS genau ein Attribut *startSsecList* besitzen. Für diese Liste gelten die in (N015.800) genannten Anforderungen.
   5. *flagEnabled*: Ein Multireferenz-Passwort MUSS genau ein Attribut *flagEnabled* besitzen. Für diese Liste gelten die in (N015.700) genannten Anforderungen.
   6. *passwordReference:*
      1. Dieses Attribute MUSS eine Referenz auf ein Passwort enthalten.
      2. Sei *objectFolder* der Ordner, welcher dieses Multireferenz-Passwort enthält, dann MUSS gelten:
         1. Die Funktion *pwd* = SearchPwd(*objectFolder*, *passwordReference*) DARF NICHT den Fehler *pwdNotFound* zurückmelden.
         2. *pwd* MUSS ein Reguläres-Passwort sein (keine Kettenbildung).
         3. Die Attribute, welche dieses Multireferenz-Passwort gemeinsam mit einem Regulären-Passwort nutzt, MÜSSEN dem Passwort *pwd* entnommen werden. Mit anderen Worten: Die Attribute, welches das Multireferenz-Passwort nicht selber speichert, steuert *pwd* bei.
2. (N016.325) K\_Anwendungsspezifikation {K\_Karte}   
   Folgende Attribute werden von einem Mutlireferenz-Passwort und dem mittels *passwordReference* referenzierten Regulären-Passwort *pwd* gemeinsam genutzt:
   1. *secret* MUSS mit *pwd* gemeinsam genutzt werden.
   2. *minimumLength* MUSS mit *pwd* gemeinsam genutzt werden.
   3. *maximumLength* MUSS mit *pwd* gemeinsam genutzt werden.
   4. *startRetryCounter* MUSS mit *pwd* gemeinsam genutzt werden.
   5. *retryCounter* MUSS mit *pwd* gemeinsam genutzt werden.
   6. *transportStatus* MUSS mit *pwd* gemeinsam genutzt werden.
   7. *PUK* MUSS mit *pwd* gemeinsam genutzt werden.
   8. *pukUsage* MUSS mit *pwd* gemeinsam genutzt werden.
3. (N016.330) K\_COS   
   Ein Multireferenz-Passwort MUSS dieselben Kommandos unterstützen wie ein Reguläres-Passwort (siehe (N016.200)).
4. (N016.335) K\_COS   
   Ein Multireferenz-Passwort KANN weitere Kommandos
   1. unterstützen oder
   2. ablehnen.

Einem Multireferenz-Passwort ist ein weiteres, kanalspezifisches Attribut *securityStatusEvaluationCounter* (siehe (N029.900)k) zugeordnet, das typischerweise einem flüchtig (etwa im RAM) gespeicherten Sicherheitszustand (siehe 8.9) zugerechnet wird. Es wird im Rahmen von Kommandos zur Benutzerverifikation (siehe 14.6.6) geändert.

## Schlüsselobjekt (normativ)

Dieses Unterkapitel beschreibt Schlüsselobjekte, die im Rahmen kryptographischer Operationen zum Einsatz kommen. Der Terminus Schlüsselobjekt dient in diesem Dokument als Oberbegriff für symmetrische, private und öffentliche Schlüsselobjekte.

Symmetrische Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

1. Mit persistent gespeichertem Geheimnis (Schlüssel) zur einseitigen Authentisierung (siehe 15.1.1).
2. Mit persistent gespeichertem Geheimnis (Schlüssel) zur gegenseitigen Authentisierung bei gleichzeitiger Aushandlung von Sessionkeys (siehe 15.4.1).
3. Als Sessionkeys zur Sicherstellung einer vertraulichen Kommunikation.
4. Als Sessionkeys zur Sicherstellung einer integren und authentischen Kommunikation.

Private Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

1. Berechnung elektronischer Signaturen (siehe 14.8.2)
2. Entschlüsselung von Daten (siehe 14.8.3)
3. Nachweis der Authentizität dieser Karte (siehe 15.2)
4. Transportsicherung von Sessionkey Material (siehe (N084.400)c)

Öffentliche Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

1. Prüfen elektronischer Signaturen beim Import von Zertifikaten (s. (N095.900))
2. Prüfen von Signaturen im Rahmen von Rollenauthentisierungen (s. (N084.400))
3. Transportsicherung von Sessionkey Material (s. (N086.900)d)
4. Verschlüsseln von Daten (s. 14.8.4)

### Symmetrisches Authentisierungsobjekt

Ein symmetrisches Authentisierungsobjekt wird im Rahmen von Authentisierungen gemäß 15.4.1 und 15.5 eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N016.400) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein symmetrisches Authentisierungsobjekt MUSS ein Attribut *keyIdentifier* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [2, 28] sein MUSS.
   2. Ein COS KANN weitere Werte für *keyIdentifier*
      1. unterstützen oder
      2. ablehnen.
2. (N016.450) K\_Anwendungsspezifikation {K\_Karte}   
   Eine symmetrisches Authentisierungsobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
3. (N016.500) K\_Anwendungsspezifikation {K\_Karte}   
   Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
4. (N016.590) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *keyType* aus der Menge {3TDES (Option\_DES), AES‑128, AES‑192, AES‑256} besitzen.
   2. Ein COS KANN weitere Werte für *keyType*
      1. unterstützen oder
      2. ablehnen.
5. (N016.600) K\_Anwendungsspezifikation {K\_Karte}   
   Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *encKey* gemäß 8.2.1 besitzen.
6. (N016.700) K\_Anwendungsspezifikation {K\_Karte}   
   Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *macKey* gemäß 8.2.1 besitzen.
7. (N016.705) K\_Anwendungsspezifikation {K\_Karte}   
   Die Attribute *encKey* und *macKey* MÜSSEN passend zum Attribut *keyType* gewählt werden.
8. (N016.710) K\_Anwendungsspezifikation {K\_Karte}   
   Falls *algorithmIdentifier* Element der Menge {aesSessionkey4SM} ist, genau dann MUSS ein symmetrisches Authentisierungsobjekt ein Attribut *numberScenario* vom Typ ganze Zahl aus dem Intervall [0, 32.767] besitzen.
9. (N016.800) K\_Anwendungsspezifikation   
   Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut vom Typ *algorithmIdentifier* besitzen, welches angibt, für welchen Zweck es verwendbar ist.
   1. {K\_Karte}   
      Für symmetrische Authentisierungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
      1. aesSessionkey4SM, (siehe, (N084.410)a)
      2. desSessionkey4SM, (siehe, (N084.410)b, Option\_DES)

} sein (siehe Tabelle 267).

* 1. {K\_Karte}, Option\_Kryptobox   
     Für symmetrische Authentisierungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
     1. aesSessionkey4TC, (siehe (N084.402)a, (N086.902)a)
     2. desSessionkey4TC (siehe (N084.402)b, (N086.902)b, Option\_DES)

} sein (siehe Tabelle 267).

* 1. Ein COS KANN weitere Werte für *algorithmIdentifier*
     1. unterstützen oder
     2. ablehnen.

1. (N016.820) K\_Anwendungsspezifiktion {K\_Karte}, Option\_Kryptobox   
   Falls *algorithmIdentifier* Element der Menge {aesSessionkey4TC, desSessionkey4TC} ist, genau dann MUSS ein symmetrisches Authentisierungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
2. (N016.900) K\_COS   
   Ein symmetrisches Authentisierungsobjekt MUSS die folgenden Kommandos unterstützen:
   1. Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels der in 14.9.9 beschriebenen Use Cases.
      1. External Authenticate (siehe 14.7.1.1 und 14.9.9.4),
      2. General Authenticate (siehe 14.7.2.3.1 und 14.9.9.4),
      3. Internal Authenticate (siehe 14.7.4.1 und 14.9.9.2),
      4. Mutual Authenticate (siehe 14.7.1.2 und 14.9.9.6),
   2. Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.
      1. Activate (siehe 14.2.1)
      2. Deactivate (siehe 14.2.3),
      3. Delete (siehe 14.2.4).
      4. Get Security Status Key (siehe 14.7.3.1),
      5. Terminate (siehe 14.2.9).
3. (N017.000) K\_COS   
   Ein symmetrisches Authentisierungsobjekt KANN weitere Kommandos
   1. unterstützen oder
   2. ablehnen.
4. Absichtlich ist einem symmetrischen Authentisierungsobjekt kein Attribut shareable zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein symmetrisches Authentisierungsobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.

### Symmetrisches Kartenverbindungsobjekt

Ein symmetrisches Kartenverbindungsobjekt wird im Rahmen von Authentisierungen gemäß 15.4.2 verwendet. Die normativen Festlegungen in diesem Dokument bilden eine Untermenge der in [BSI-TR-03110-3] für eine CAN (Card Access Number) beschriebenen Funktionalität: Ein symmetrisches Kartenverbindungsobjekt ist nicht änderbar und unterstützt ausschließlich PACE in der Version 2 in der ECDH-Variante.

Für symmetrische Kartenverbindungsobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N017.020) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD
   1. Ein symmetrisches Kartenverbindungsobjekt MUSS ein Attribut *keyIdentifier* besitzen, dessen Wert eine ganz Zahl aus dem Intervall [2, 28] sein MUSS.
   2. Ein COS KANN weitere Werte für *keyIdentifier*
      1. unterstützen oder
      2. ablehnen.
2. (N017.024) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD   
   Eine symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
3. (N017.026) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD   
   Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
4. (N017.028) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD   
   Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut *can* vom Typ *pin* gemäß (N008.000) besitzen.
5. (N017.030) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD   
   Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut vom Typ *algorithmIdentifier* besitzen, welches angibt, für welchen Zweck es verwendbar ist.
   1. Option\_kontaktlose\_Schnittstelle: Für symmetrische Kartenverbindungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
      1. id-PACE-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
      2. id-PACE-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
      3. id-PACE-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)

} sein.

* 1. Option\_PACE\_PCD: Für symmetrische Kartenverbindungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
     1. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
     2. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
     3. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)

} sein.

* 1. Ein COS KANN weitere Werte für *algorithmIdentifier*
     1. unterstützen oder
     2. ablehnen.

1. (N017.031) K\_Anwendungsspezifiktion {K\_Karte},   
   Option\_PACE\_PCD   
   Falls *algorithmIdentifier* Element der Menge {id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256} ist, genau dann MUSS ein symmetrisches Kartenverbindungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
2. (N017.032) K\_COS,   
   Option\_kontaktlose\_Schnittstelle, Option\_PACE\_PCD   
   Ein symmetrisches Kartenverbindungsobjekt
   1. MUSS die folgenden Kommandos unterstützen:
      1. Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Kartenverbindungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.7 beschriebenen Use Cases.
         1. General Authenticate (siehe 14.7.2 und 14.9.9.7).
      2. Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Kartenverbindungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.
         1. Activate (siehe 14.2.1),
         2. Deactivate (siehe 14.2.3),
         3. Delete (siehe 14.2.4).
         4. Terminate (siehe 14.2.9).
   2. KANN weitere Kommandos
      1. unterstützen oder
      2. ablehnen.
3. Absichtlich ist einem symmetrischen Kartenverbindungsobjekt kein Attribut shareable zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein symmetrisches Kartenverbindungsobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.

### Privates Schlüsselobjekt

1. Gegenüber der Generation 1 Spezifikation wird die dort beschriebene Unterteilung in private Authentisierungs-, Entschlüsselungs- und Signierobjekte in der Generation 2 aufgehoben. Mithin ist es möglich, dass ein privates Schlüsselobjekt sowohl für Authentisierungszwecke, als auch zur Entschlüsselung und zur Signaturerstellung nutzbar ist. Aus Sicherheitssicht ist es ratsam auf eine mögliche Vermischung wo irgend möglich zu verzichten. Andererseits wird zum Aufbau einer IPSec-Verbindung gegebenenfalls ein privates Schlüsselobjekt verwendet, welches sowohl entschlüsselt, als auch signiert.

Ein privates Schlüsselobjekt wird

* zur Authentisierung einer Karte gegenüber einer anderen technischen Komponente
* zur Entschlüsselung von Daten und
* zur Berechnung elektronischer Signaturen

verwendet. Für private Schlüsselobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N017.100) K\_Anwendungsspezifikation {K\_Karte}
   1. Ein privates Schlüsselobjekt MUSS ein Attribut *keyIdentifier* besitzen, dessen Wert eine ganz Zahl aus dem Intervall [2, 28] sein MUSS.
   2. Ein COS KANN weitere Werte für *keyIdentifier*
      1. unterstützen oder
      2. ablehnen.
2. (N017.150) K\_Anwendungsspezifikation {K\_Karte}   
   Ein privates Schlüsselobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
3. (N017.200) K\_Anwendungsspezifikation {K\_Karte}   
   Ein privates Schlüsselobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
4. (N017.300) K\_Anwendungsspezifikation {K\_Karte}   
   Ein privates Schlüsselobjekt MUSS genau ein Attribut *privateKey* gemäß 8.2.3 besitzen.
5. (N017.310) Diese Anforderung ist absichtlich leer. Der ursprüngliche Text forderte eine Personalisierung von privaten Schlüsseln, die komplett durch das Kommando Generate Asymmetric Key Pair abgedeckt ist.
6. (N017.400) K\_Anwendungsspezifikation {K\_Karte}   
   Ein privates Schlüsselobjekt MUSS genau ein Attribut *listAlgorithmIdentifier* besitzen, welches angibt, für welche Zwecke das private Schlüsselobjekt verwendbar ist.
   1. Für die Länge der Liste gilt: Die Liste MUSS mindestens ein Element und DARF NICHT mehr als vier Elemente enthalten.
   2. Jedes Listenelement MUSS
      1. genau einen *seIdentifier* gemäß (N007.900) und
      2. genau eine Menge *setAlgorithmIdentifier* mit Elementen des Typs *algorithm­Identifier* enthalten.
      3. Die Werte von *algorithmIdentifier* MÜSSEN aus den in (N017.600), (N017.900) und (N018.300) genannten Mengen gewählt werden.
   3. Die Menge *setAlgorithmIdentifier* DARF NICHT mehr als sechs Elemente enthalten.
   4. Das COS KANN
      1. Listen des Typs *listAlgorithmIdentifier*, welche mehr Elemente enthalten als nach (N017.400)a gefordert,
         1. unterstützen, oder
         2. ablehnen.
      2. Listen des Typs *setAlgorithmIdentifier*, welche andere Elemente enthalten als nach (N017.400)b.3 gefordert,
         1. unterstützen, oder
         2. ablehnen.
      3. Listen des Typs *setAlgorithmIdentifier*, welche mehr Elemente enthalten als nach (N017.400)c gefordert,
         1. unterstützen, oder
         2. ablehnen.
7. (N017.420) K\_Anwendungsspezifiktion {K\_Karte}, Option\_Kryptobox   
   Ist in *listAlgorithmIdentifier* ein *algorithmIdentifier* der Menge {rsaSessionkey4TC, elcSessionkey4TC} enthalten, genau dann MUSS ein privates Authentisierungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
8. (N017.430) K\_Anwendungsspezifiktion {K\_Karte}   
   Ist in *listAlgorithmIdentifier* ein *algorithmIdentifier* der Menge {elcAsynchronAdmin} enthalten, genau dann MUSS ein privates Authentisierungsobjekt ein Attribut *numberScenario* vom Typ ganze Zahl aus dem Intervall [0, 32.767] besitzen.
9. (N017.500) Dieser Punkt ist absichtlich leer.
10. (N017.510) Dieser Punkt ist absichtlich leer.
11. (N017.600) K\_Anwendungsspezifikation {K\_Karte}   
    Wertebereich von *algorithmIdentifier*
    1. Ist das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ
       1. *privateRsaKey* MUSS zum Zwecke der Authentisierung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 268): {
          1. rsaClientAuthentication, (siehe 14.7.4.1 und (N086.900)b)
          2. rsaRoleAuthentication, (siehe 14.7.4.1 und (N086.900)c,   
               Option\_RSA\_CVC)
          3. rsaSessionkey4SM, (siehe 14.7.4.1 und (N086.900)d,   
               Option\_DES)

* + - 1. rsaSessionkey4TC, ((N084.402)c,(N086.902)c,   
           Option\_Kryptobox, Option\_DES)

}.

* + 1. *privateElcKey* MUSS zum Zwecke der Authentisierung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 268): {

* + - 1. elcAsynchronAdmin, (siehe 14.7.2.5 und (N085.068))
      2. elcRoleAuthentication, (siehe 14.7.4.1 und (N086.900)a)
      3. elcSessionkey4SM, (siehe (N085.054))
      4. elcSessionkey4TC, (siehe (N085.056), Option\_Kryptobox)

}.

* 1. Ein COS KANN weitere Werte für *algorithmIdentifier*
     1. unterstützen oder
     2. ablehnen.

1. (N017.610) K\_Anwendungsspezifikation {K\_Karte}, Option\_DES, Option\_RSA\_CVC   
   Zusammenhang zwischen Moduluslänge und Card-to-Card-Algorithmen:
   1. Die *algorithmIdentifier* rsaRoleAuthentication, rsaSessionkey4SM und rsaSessionkey4TC DÜRFEN einem privaten Schlüssel NICHT zugewiesen werden, wenn dessen Moduluslänge ungleich 2048 bit ist.
   2. Das COS KANN die Algorithmen rsaRoleAuthentication, rsaSessionkey4SM und rsaSessionkey4TC für andere Moduluslängen
      1. unterstützen oder
      2. ablehnen.
2. (N017.620) K\_Anwendungsspezifikation {K\_Karte}   
   Zusammenhang zwischen Kurvenparametern und Card-to-Card-Algorithmen:
   1. Die *algorithmIdentifier* elcRoleAuthentication, elcSessionkey4SM und elcSessionkey4TC DÜRFEN einem privaten Schlüssel NICHT zugewiesen werden, wenn dessen *domainParameter* nicht in der Menge {brainpoolP256r1, brainpoolP384r1, brainpoolP512r1} enthalten ist.
   2. Das COS KANN die Algorithmen elcRoleAuthentication, elcSessionkey4SM und elcSessionkey4TC für andere *domainParameter*
      1. unterstützen oder
      2. ablehnen.
3. (N017.700) K\_COS   
   Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels der in 14.9.9 beschriebenen Use Cases. Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N017.600) genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:
   1. External Authenticate (siehe (N084.400)c und 14.9.9.3)
   2. General Authenticate (siehe 15.4.4),
   3. Internal Authenticate (siehe 14.7.4.1 und 14.9.9.3),
   4. PSO Compute Digital Signature (siehe (N088.600)c und 14.9.9.9)
4. (N017.800) Dieser Punkt ist absichtlich leer.
5. (N017.900) K\_Anwendungsspezifikation {K\_Karte}   
   Wertebereich von *algorithmIdentifier*
   1. Ist das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ
      1. *privateRsaKey* MUSS zum Zwecke der Entschlüsselung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 269): {
         1. rsaDecipherPKCS1\_V1\_5, (siehe 14.8.3.1, 14.8.6.1 und 14.8.6.2)
         2. rsaDecipherOaep, (siehe 14.8.3.1, 14.8.6.1 und 14.8.6.2)

}.

* + 1. *privateElcKey* MUSS zum Zwecke der Entschlüsselung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 269): {
       1. elcSharedSecretCalculation, (s. 14.8.3.2, 14.8.6.2, 14.8.6.3, 14.8.6.4)

}.

* 1. Ein COS KANN weitere Werte für *algorithmIdentifier*
     1. unterstützen oder
     2. ablehnen.

1. (N018.000) K\_COS   
   Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.11 beschriebenen Use Cases. Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N017.900) genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:
   1. PSO Decipher (siehe 14.8.3),
   2. PSO Transcipher (siehe 14.8.6).
2. (N018.100) Dieser Punkt ist absichtlich leer.
3. (N018.200) K\_Anwendungsspezifikation {K\_Karte}   
   Ein privates Schlüsselobjekt MUSS ein Attribut *keyAvailable* vom Typ Boolean unterstützen.
   1. Der Wert True zeigt an, dass das Attribut *privateKey* verwendbar ist.
   2. Der Wert False zeigt an, dass das Attribut *privateKey* nicht verwendbar ist.
   3. Auf dieses Attribut wird im Rahmen der folgenden Kommandos zugegriffen:
      1. Internal Authenticate (siehe (N086.810)),
      2. PSO Compute Digital Signature (siehe (N088.500)),
      3. PSO Decipher (siehe (N090.210)),
      4. PSO Transcipher (siehe (N094.010)).
4. (N018.300) K\_Anwendungsspezifikation {K\_Karte}   
   Wertebereich von *algorithmIdentifier*
   1. Ist das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ
      1. *privateRsaKey* MUSS zum Zwecke der Signaturerzeugung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 270): {
         1. sign9796\_2\_DS2, (siehe 14.8.2.2 und (N088.600)a)
         2. signPKCS1\_V1\_5, (siehe 14.8.2.1 und (N088.600)c)
         3. signPSS, (siehe 14.8.2.1 und (N088.600)c)

}.

* + 1. *privateElcKey* MUSS zum Zwecke der Signaturerzeugung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 270): {
       1. signECDSA, (siehe 14.8.2.1 und (N088.600)c)

}.

* 1. Ein COS KANN weitere Werte für *algorithmIdentifier*
     1. unterstützen oder
     2. ablehnen.

1. (N018.400) K\_COS   
   Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.9 beschriebenen Use Cases. Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N018.300) genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:
   1. PSO Compute Digital Signature (siehe 14.8.2).
2. (N018.410) K\_COS   
   Bevor das folgende Kommando in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es selektieren. Dies geschieht entweder mittels des in 14.9.9.9 beschriebenen Use Cases oder mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird. Ein privates Schlüsselobjekt MUSS folgendes Kommandos unterstützen:
   1. Generate Asymmetric Key Pair (siehe 14.9.3).
3. (N018.420) K\_COS   
   Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird. Neben den oben genannten Kommandos MUSS ein privates Schlüsselobjekt zusätzlich die folgenden administrativen Kommandos unterstützen:
   1. Activate (siehe 14.2.1),
   2. Deactivate (siehe 14.2.3),
   3. Delete (siehe 14.2.4).
   4. Terminate (siehe 14.2.9).
4. (N018.422) K\_COS   
   Ein privates Schlüsselobjekt KANN weitere Kommandos
   1. unterstützen oder
   2. ablehnen.
5. Absichtlich ist einem privaten Schlüsselobjekt kein Attribut shareable zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein privates Schlüsselobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.

### Öffentliches Schlüsselobjekt

Ein öffentliches Schlüsselobjekt wird als Oberbegriff für

* öffentliche Signaturprüfobjekte (siehe 8.6.4.1)
* öffentliche Authentisierungsobjekte (siehe 8.6.4.2)
* öffentliches Verschlüsselungsobjekte (siehe 8.6.4.3)

verwendet. Für öffentliche Schlüsselobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N018.500) K\_Anwendungsspezifikation {K\_Karte}   
   Ein öffentliches Schlüsselobjekt MUSS ein Attribut *keyIdentifier* besitzen.
2. (N018.550) K\_Anwendungsspezifikation {K\_Karte}   
   Eine öffentliches Schlüsselobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
3. (N018.600) K\_Anwendungsspezifikation {K\_Karte}   
   Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut *publicKey* gemäß 8.2.4 besitzen.
4. (N018.700) K\_Anwendungsspezifikation {K\_Karte}   
   Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut vom Typ *oid* besitzen, welches angibt, für welche Zwecke es verwendet werden darf.
5. (N018.800) K\_COS   
   Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
6. (N018.820) K\_COS   
   Bevor eines der folgenden Kommandos in der Lage ist mit dem öffentlichen Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird. Neben den in den folgenden Unterkapiteln genannten Kommandos MUSS ein öffentliches Schlüsselobjekt zusätzlich die folgenden administrativen Kommandos unterstützen:
   1. Activate (siehe 14.2.1),
   2. Deactivate (siehe 14.2.3),
   3. Delete (siehe 14.2.4).
   4. Terminate (siehe 14.2.9).
7. (N018.822) K\_COS   
   Ein öffentliches Schlüsselobjekt KANN weitere Kommandos
   1. unterstützen oder
   2. ablehnen.
8. (N018.900) Diese Anforderung ist absichtlich leer.
9. Absichtlich ist einem öffentlichen Schlüsselobjekt kein Attribut shareable zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein öffentliches Schlüsselobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.

#### Öffentliches Signaturprüfobjekt

Ein öffentliches Signaturprüfobjekt wird zur Prüfung von Signaturen in einem CV-Zertifikat eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N019.000) K\_Anwendungsspezifikation {K\_Karte}   
   Ein öffentliches Signaturprüfobjekt ist eine Erweiterung von einem öffentlichen Schlüsselobjekt und MUSS deshalb den Anforderungen aus 8.6.4 genügen.
2. (N019.100) K\_Anwendungsspezifikation {K\_Karte}
   1. Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge acht Oktett möglich sein.
   2. Ein COS KANN weitere Längen für *keyIdentifier*
      1. unterstützen oder
      2. ablehnen.
3. (N019.110) K\_Anwendungsspezifikation {K\_Karte}   
   Ein öffentliches Signaturprüfobjekt MUSS folgende Attribute vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen, die an Schlüsselobjekte weitergereicht werden, die mittels CV-Zertifikat importiert werden (siehe (N095.900)):
   1. Genau ein Attribut *accessRulesPublicSignatureVerificationObject* und
   2. genau ein Attribut *accessRulesPublicAuthenticationObject*.
4. (N019.200) K\_Anwendungsspezifikation {K\_Karte}   
   Wertebereich des Attributes *oid*:
   1. Ist das Attribut *publicKey* des öffentlichen Signaturprüfobjektes vom Typ

* + 1. Option\_RSA\_CVC, *publicRsaKey* MUSS der Wert von *oid* Element der Menge {
       1. sigS\_ISO9796-2Withrsa\_sha256,

} sein (vergleiche Tabelle 271).

* + 1. *publicElcKey* MUSS der Wert von *oid* Element der Menge {
       1. ecdsa-with-SHA256,
       2. ecdsa-with-SHA384,
       3. ecdsa-with-SHA512

} sein (vergleiche Tabelle 271) und zu den Domainparametern von *publicElcKey* passen.

* 1. Ein COS KANN weitere Werte für *oid*
     1. unterstützen oder
     2. ablehnen.

1. (N019.210) K\_Anwendungsspezifikation {K\_Karte}   
   Falls das Attribut *publicKey* vom Typ *publicElcKey* ist, dann gilt: Das öffentliche Signaturprüfobjekt MUSS ein Attribut

* 1. CHAT enthalten, welches Aktionen kennzeichnet, die nach einer erfolgreichen Authentisierung mit einem öffentlichen Authentisierungsobjekt freigeschaltet werden.
  2. *expirationDate* vom Typ *date* (siehe (N008.120)) enthalten, welches angibt, ab welchem Zeitpunkt die Schlüsselverwendung endet.

1. (N019.300) K\_COS   
   Bevor eines der folgenden Kommando in der Lage ist mit dem öffentlichen Signaturprüfobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.10 beschriebenen Use Cases. Ein öffentliches Signaturprüfobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:
   1. PSO Verify Certificate (siehe 14.8.7)
   2. PSO Verify Digital Signature (siehe 14.8.9).

#### Öffentliches Authentisierungsobjekt

Ein öffentliches Authentisierungsobjekt wird zur Authentisierung einer anderen technischen Komponente gegenüber dieser Komponente eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N019.400) K\_Anwendungsspezifikation {K\_Karte}   
   Ein öffentliches Authentisierungsobjekt ist eine Erweiterung von einem öffentlichen Schlüsselobjekt und MUSS deshalb den Anforderungen aus 8.6.4 genügen.
2. (N019.500) K\_Anwendungsspezifikation {K\_Karte}
   1. Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge zwölf möglich sein.
   2. Ein COS KANN weitere Werte für *keyIdentifier*
      1. unterstützen oder
      2. ablehnen.
3. (N019.600) K\_Anwendungsspezifikation {K\_Karte}   
   Wertebereich des Attributes *oid*:
   1. Für öffentliche Authentisierungsobjekte des Typs

* + 1. Option\_RSA\_CVC, *publicRsaKey* MUSS der Wert von *oid* Element der Menge {
       1. authS\_ISO9796-2Withrsa\_sha256\_mutual,

} sein (vergleiche Tabelle 271).

* + 1. *publicElcKey* MUSS der Wert von *oid* Element der Menge {
       1. authS\_gemSpec-COS-G2\_ecc-with-sha256,
       2. authS\_gemSpec-COS-G2\_ecc-with-sha384,
       3. authS\_gemSpec-COS-G2\_ecc-with-sha512

} sein (vergleiche Tabelle 271) und zu den Domainparametern von *publicElcKey* passen.

* 1. Ein COS KANN weitere Werte für *oid*
     1. unterstützen oder
     2. ablehnen.

1. (N019.700) K\_Anwendungsspezifikation {K\_Karte}   
   Falls *publicKey* vom Typ

* 1. Option\_RSA\_CVC*, publicRsaKey* ist, MUSS das öffentliche Authentisierungsobjekt ein Attribut *accessRights* besitzen, welches eine *CHA* gemäß 7.1.1.4 speichert.
  2. *publicElcKey* ist, MUSS das öffentliche Authentisierungsobjekt ein Attribut
     1. *accessRights* besitzen, welches ein CHAT speichert und
     2. *expirationDate* vom Typ *date* (siehe (N008.120)) enthalten, welches angibt, ab welchem Zeitpunkt die Schlüsselverwendung endet.

1. (N019.800) K\_COS   
   Bevor eines der folgenden Kommandos in der Lage ist mit dem öffentlichen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.5 beschriebenen Use Cases. Ein öffentliches Authentisierungsobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:
   1. External Authenticate (siehe 14.7.1),
   2. General Authenticate (siehe 15.4.4),
   3. Internal Authenticate (siehe (N086.900)d).

#### Öffentliches Verschlüsselungsobjekt

Ein öffentliches Verschlüsselungsobjekt wird zum Verschlüsseln von Daten eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

1. (N019.820) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_Kryptobox   
   Ein öffentliches Verschlüsselungsobjekt ist eine Erweiterung von einem öffentlichen Schlüsselobjekt und MUSS deshalb den Anforderungen aus 8.6.4 genügen.
2. (N019.822) K\_Anwendungsspezifikation {K\_ Karte },   
   Option\_Kryptobox
   1. Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge zwölf möglich sein.
   2. Ein COS KANN weitere Werte für *keyIdentifier*
      1. unterstützen oder
      2. ablehnen.
3. (N019.824) K\_Anwendungsspezifikation {K\_Karte},   
   Option\_Kryptobox   
   Wertebereich des Attributes *oid*:
   1. Für öffentliche Verschlüsselungsobjekt des Typs
      1. *publicRsaKey* MUSS der Wert von *oid* Element der Menge {
         1. id-RSAES-OAEP
         2. rsaEncryption

} sein.

* + 1. *publicElcKey* MUSS der Wert von *oid* Element der Menge {
       1. id-ELC-shared-secret-calculation

} sein.

* 1. Ein COS KANN weitere Werte für *oid*
     1. unterstützen oder
     2. ablehnen.

1. (N019.826) K\_COS,   
   Option\_Kryptobox   
   Ein öffentliches Verschlüsselungsobjekt MUSS das folgende Kommando unterstützen:  
   – PSO Encipher (siehe 14.8.4.3, 14.8.4.4 und 14.9.9.12),   
   Bevor dieses Kommando in der Lage ist mit dem öffentlichen Verschlüsselungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.12 beschriebenen Use Cases.

## Datenobjekte (informativ)

Auf Datenobjekte wird im Rahmen der [ISO/IEC 7816-4] Kommandos Get Data und Put Data zugegriffen. Da diese Kommandos für diese Version der Spezifikation nicht verpflichtend sind, werden sie hier nicht weiter behandelt.

## Security Environment (informativ)

Zum Begriff „Security Environment“ gemäß [ISO/IEC 7816-4#10.3.3] existiert eine große Bandbreite an Interpretationen. Deshalb wird versucht, im normativen Bereich dieses Dokumentes auf diesen problematischen Begriff möglichst zu verzichten. Aus demselben Grund wird in diesem informativen (das heißt nicht-normativen) Kapitel nur das weitgehend unstrittige Prinzip der Security Environments kurz skizziert.

Security Environments eröffnen die Möglichkeit, gewisse Attribute von Objekten in Abhängigkeit von der Einsatzumgebung mit konkreten Werten zu versehen. In diesem Dokument wird davon beim Attributstyp *interfaceDependentAccessRules* (siehe 8.1.4) Gebrauch gemacht. Damit ergeben sich beim Design von Anwendungen Gestaltungsmöglichkeiten.

Beispiel: Eine Karte wird in zwei verschiedenen Umgebungen eingesetzt und in beiden sind sowohl vertrauliche Daten (EF.Verordnung) als auch allgemein bekannte Daten (EF.X509\_Zertifikat) auslesbar (Read Binary).

* 1. In der Umgebung\_1 werden Lauschangriffe auf die physikalische Schnittstelle (siehe Abbildung 1) befürchtet. Zur Abwehr derartiger Angriffe wird die Zugriffsregel so gewählt, dass sensitive Daten nur vertraulich und integer übertragbar sind (siehe (N022.600)c). Für allgemein bekannte Daten wird ein derartiger Schutz nicht verlangt.
  2. In der Umgebung\_2 existieren keine Angriffe auf die physikalische Schnittstelle (siehe Abbildung 1). Deshalb wird aus diversen Gründen die Zugriffsregel so gewählt, dass alle Daten ungeschützt übertragen werden.

Zu jeder Einsatzumgebung existiert ein Satz Zugriffsregeln:

* 1. Umgebung\_1:
     1. EF.Verordnung Read Binary an einem Kiosk mit Secure Messaging
     2. EF.X509\_Zertifikat Read Binary immer erlaubt
  2. Umgebung\_2:
     1. EF.Verordnung Read Binary in einer Apotheke ohne Verschlüsselung
     2. EF.X509\_Zertifikat Read Binary immer erlaubt

Welcher Satz von Zugriffsregeln gilt, wird durch die Einsatzumgebung bestimmt. Eine konkrete Einsatzumgebung (ein konkretes Security Environment) wird durch den Use Case in 14.9.9.1 ausgewählt.

Auf der anderen Seite werden Security Environments verwendet, um Schlüssel und andere Parameter für kryptographische Operationen auszuwählen. Die Vorgehensweise ist hier anders als bei Passwörtern. Welches Passwort von einem Kommando aus 14.6 betroffen ist, bestimmt ein Parameter dieses Kommandos.

Die Vorgehensweise ist für Schlüssel eher vergleichbar mit Dateien. Welche Datei von einem Kommando betroffen ist, bestimmt das Attribut *currentEF*, das mittels Select-Kommando vor Ausführung dieses Kommandos passend einzustellen ist. Weil die dateiorientierten Kommandos lediglich in der Lage sind stets nur mit einer Datei pro Kommando zu arbeiten, ist ein einziges Attribut *currentEF* pro Ordner völlig ausreichend (siehe (N029.900)m).

Bei einigen kryptographischen Kommandos ist es hingegen möglich, dass bei der Bearbeitung mehrere kryptographische Objekte beteiligt sind, wie etwa

* Sessionkeys für Vertraulichkeit,
* Sessionkeys für Authentizität,
* Use Case spezifischer Schlüssel, etwa PSO Compute Digital Signature.

Statt nun jedem derartigen Kommando analog zu den Passwortkommandos alle Schlüsselparameter mitzuliefern, ist in [ISO/IEC 7816-4] entschieden worden, diese Parameter zuvor mittels Manage Security Environment Kommando (siehe 14.9.9) auszuwählen. Derartige Selektionen wirken sich im Datenmodell dieses Dokumentes auf das Attribut *keyReferenceList* (siehe (N029.900)c) aus.

1. Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine performante Implementierung zulässig ist, die kanalspezifischen Attribute (siehe 11.8) im RAM zu speichern. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen, wie gezeigt zugeordnet.
2. Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine speicherplatzoptimale Implementierung zulässig ist, die kanalspezifischen Ordnerattribute in (N030.000) nur für currentFolder und allen seinen Vorfahren inklusive root zu speichern, anstatt für alle Ordner innerhalb des Objektsystems. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen, jedem Ordner zugeordnet.

## Sicherheitsstatus (informativ)

Zum Begriff „Sicherheitsstatus“ gemäß [ISO/IEC 7816-4#9] existiert eine große Bandbereite an Interpretationen. Deshalb wird versucht auf diesen problematischen Begriff im normativen Bereich dieses Dokumentes möglichst zu verzichten. Aus demselben Grund wird in diesem informativen (das heißt nicht-normativen) Kapitel nur das weitgehend unstrittige Prinzip des Sicherheitsstatus kurz skizziert.

Gemäß [ISO/IEC 7816-4] zählen zum Sicherheitsstatus

1. Globale Sicherheitsstatus, welche durch Benutzerverifikation oder Komponenten­authentikation mit Objekten im Ordner *root* (siehe (N019.900)a) modifizierbar ist.
2. Applikationsspezifischer Sicherheitsstatus, welche mit Benutzerverifikationen oder Komponentenauthentisierungen mit Objekten in beliebigen Ordnern modifizierbar ist.
3. File-spezifischer Status, welcher in diesem Dokument unberücksichtigt bleibt.
4. Kommando-spezifischer Status, welchem bei der Auswertung von Zugriffsregeln Rechnung getragen wird.

In diesem Dokument wird bezüglich der globalen und applikationsspezifischen Status für alle Ordner ein Sicherheitsstatus angenommen und für die

* 1. Benutzerverifikation durch (N029.900)i bis (N029.900)k Rechnung getragen, wobei dort auch die kleinste Anzahl der zu unterstützenden Sicherheitsstatus für Benutzerverifikation festgelegt wird.
  2. Komponentenauthentisierung durch (N029.900)e bis (N029.900)h Rechnung getragen, wobei dort auch die kleinste Anzahl der zu unterstützenden Sicherheitsstatus für Komponentenauthentisierung festgelegt wird.

1. Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine performante Implementierung zulässig ist, die Attribute globalSecurityList, dfSpecificSecurityList, globalPasswordList und dfSpecificPasswordList im RAM zu speichern. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen wie dargestellt zugeordnet.

# Objektsystem (normativ)

Eine Smartcard wird in diesem Dokument als *sicherer Datenspeicher* betrachtet, wobei die Betonung auf beiden Wörtern liegt.

* Datenspeicher: Eine Smartcard speichert beliebige Informationen ganz analog zur Festplatte eines Computers in Dateien (siehe 8.3.2).
* Sicherheit: Der Zugriff auf die in Dateien gespeicherten Informationen wird durch Regeln festgelegt. Die Einhaltung der Zugriffsregeln wird durch das Betriebs­system gewährleistet. Typischerweise enthalten Regeln Zugriffsbeschrän­kungen dergestalt, dass erst nach erfolgreicher Benutzerverifikation oder Komponentenauthentisierung ein Zugriff gestattet ist. Darüber hinaus wird vielfach zusätzlich ein vertraulicher und authentischer Datenaustausch er­zwungen.

Typischerweise werden Dateien und damit Informationen hierarchisch strukturiert.

Die Regeln in diesem Dokument sind so aufgebaut, dass sich ein hierarchisches System mit mindestens vier Ordnerebenen aufbauen lässt (*root* enthält DF2, DF2 enthält DF3, DF3 enthält DF4, DF4 enthält keinen weiteren Ordner).

Private und symmetrische Schlüssel sowie Passwörter lassen sich per „backtracking“ suchen. Das bedeutet, sie werden zunächst im aktuellen Verzeichnis gesucht. Falls die Suche dort erfolglos ist, wird rekursiv in der nächsthöheren Ebene gesucht.

Eine DF-spezifische Suche bezieht *root* nie mit ein.

Öffentliche Schlüssel werden als zentral gespeicherte Objekte betrachtet. In der Regel gehören solche Schlüssel einer externen Komponente und werden mittels eines Zertifikates importiert (siehe 14.8.7).

## Aufbau und Strukturtiefe

Dieses Kapitel legt die hierarchische Struktur fest, so wie sie an der Schnittstelle gesehen wird. Wie die Information und die Struktur kartenintern gespeichert werden, ist nicht Gegenstand dieses Dokumentes. Zudem wird hier nicht der sonst übliche Terminus „Filesystem“ verwendet, sondern „Objektsystem“, weil neben Files auch andere Objekttypen, wie Passwörter und Schlüssel, als eigenständige Artefakte betrachtet werden.

Bei der Spezifikation von Anwendungen sind folgende Regeln einzuhalten:

1. (N019.900) K\_Anwendungsspezifikation {K\_Karte} und K\_COS   
   Das COS MUSS ein hierarchisches Objektsystem mit mehreren Ebenen unterstützen.
   1. Das Objektsystem MUSS ein Attribut *root* mit Eigenschaften gemäß (N019.910) und (N019.920) besitzen:
   2. Das Objektsystem MUSS zwei Attribute "Answer To Reset" besitzen für die in (N024.100) Anforderungen genannt werden:
      1. Das Attribut *coldAnswerToReset* vom Typ Oktettstring MUSS vom COS im Rahmen eines Cold Reset gemäß (N023.920)b versendet werden.
      2. Das Attribut *warmAnswerToReset* vom Typ Oktettstring MUSS vom COS im Rahmen eines Warm Reset gemäß (N023.920)c versendet werden.
      3. K\_Anwendungsspezifikation MUSS identische Anforderungen an *coldAnswerToReset* und *warmAnswerToReset* enthalten.
      4. Das COS KANN für *coldAnswerToReset* und *warmAnswerToReset* unterschiedliche Werte verwenden.
   3. Das Objektsystem MUSS ein Attribut *iccsn8* vom Typ Oktettstring besitzen. Für *iccsn8* MÜSSEN beliebige Werte mit acht Oktette möglich sein.
   4. Das Objektsystem MUSS ein Attribut *applicationPublicKeyList* unterstützen.
      1. Das COS MUSS eine beliebige Anzahl Listenelemente unterstützen.
      2. Als Listenelemente MUSS der Objekttyp „Öffentliches Schlüsselobjekt" (siehe 8.6.4) unterstützt werden in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).
      3. Elemente der Liste *applicationPublicKeyList* MÜSSEN persistent gespeichert werden.
      4. Elemente der Liste *applicationPublicKeyList* KÖNNEN
         1. zentral an einer Stelle oder
         2. dezentral über mehrere Ordner verteilt gespeichert werden.
   5. Das Objektsystem MUSS ein Attribut *persistentCache* unterstützen.
      1. Das COS MUSS eine beliebige Anzahl Listenelemente unterstützen.
      2. Die Anwendungsspezifikation SOLL eine Begrenzung der Anzahl der Listenelemente vorschreiben.
      3. Als Listenelemente MÜSSEN die Objekttypen „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) und „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) unterstützt werden in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).
      4. Elemente der Liste *persistentCache* MÜSSEN persistent gespeichert werden.
      5. Elemente der Liste *persistentCache* KÖNNEN
         1. zentral an einer Stelle oder
         2. dezentral über mehrere Ordner verteilt gespeichert werden.
   6. Die Anwendungsspezifikation SOLL eine Begrenzung der Anzahl der Listenelemente in *persistentPublicKeyList* vorschreiben.
   7. Das Objektsystem MUSS ein Attribut *volatileCache* unterstützen.
      1. Listenlänge: Das COS KANN Listenlängen mit dem Wert null, eins oder größer unterstützen.
      2. Als Listenelemente MÜSSEN die Objekttypen „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) und „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) unterstützt werden in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).
      3. Elemente der Liste *volatileCache* DÜRFEN NICHT persistent gespeichert werden.
      4. Elemente der Liste *volatileCache* KÖNNEN
         1. zentral an einer Stelle oder
         2. dezentral über mehrere Ordner verteilt gespeichert werden.
   8. Alle Elemente der Liste *allPublicKeyList* MÜSSEN in allen logischen Kanälen zur Verfügung stehen und shareable sein.
   9. Das Objektsystem MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
   10. Das Objektsystem MUSS genau ein Attribut *pointInTime* vom Typ *date* gemäß (N008.120) besitzen. *Hinweis: Der zu unterstützende Wertebereich von pointInTime ergibt sich implizit aus dem Attribut CED, welches in CV-Zertifikaten enthalten ist, siehe [gemSpec\_PKI#6.7.2.6].*
2. (N019.910) K\_Anwendungsspezifikation {K\_Karte}   
   Für das Attribut *root* des Objektsystems gilt:
   1. Die Anwendungsspezifikation MUSS für *root* die Wahl zwischen den Typen Applikation (siehe 8.3.1.1) und Application Dedicated File (siehe 8.3.1.3) zulassen.
   2. Die Anwendungsspezifikation MUSS als Zugriffbedingung für die Zugriffsart Delete (siehe 14.2.4) NEVER (siehe (N022.100)) fordern.
3. (N019.920) K\_COS   
   Für das Attribut *root* des Objektsystems gilt:
   1. Das COS MUSS für *root* den Typ Applikation (siehe 8.3.1.1) unterstützen.
   2. Das COS KANN für *root* weitere Ordnertypen unterstützen.
   3. Das COS KANN für *root* für die Zugriffsart Delete beliebige Zugriffsbedingungen unterstützen.
4. (N019.930) K\_Anwendungsspezifikation {K\_Karte}   
   Im Objektsystem DARF es KEINE zwei verschiedenen Ordner geben, deren Attribut *applicationIdentifier* identisch ist.
5. (N020.000) K\_COS   
   Der Ordner *root* wird der Ebene\_0 zugeordnet. Ebene\_0 wird als „höchste Ebene im Objektsystem“ bezeichnet.
6. (N020.100) K\_COS   
   Wenn ein Ordner der Ebene\_i zugeordnet ist, dann werden alle Elemente der Liste *children* dieses Ordners der Ebene\_(i + 1) zugeordnet.   
   Ebene\_i ist die nächsthöhere Ebene zu Ebene\_(i + 1).   
   Ebene\_(i + 1) ist die nächsttiefere Ebene zu Ebene\_i.
7. (N020.200) K\_COS   
   Dieser Punkt ist absichtlich leer. In einer früheren Dokumentenversion enthielt er die Anforderung Applikationen entweder der Ebene\_0 oder der Ebene\_1 zuzuordnen, siehe dazu auch die Änderung in (N020.300).
8. (N020.300) K\_COS   
   Ordner, die der Ebene\_0 (*root*), Ebene\_1 (DF2) oder Ebene\_2 (DF3) zugeordnet sind, MÜSSEN im Attribut *children* als Listenelement die Objekttypen Applikation (siehe 8.3.1.1) und Dedicated File (siehe 8.3.1.2) und Application Dedicated File (siehe 8.3.1.3) zulassen.
9. (N020.390) K\_Anwendungsspezifikation {K\_Karte}   
   Ordner, die der Ebene\_3 zugeordnet sind DÜRFEN KEIN Objekt des Typs Ordner enthalten.
10. (N020.400) K\_COS   
    Ordner, die der Ebene\_3 zugeordnet sind,
    1. KÖNNEN im Attribut *children* Ordner zulassen.
    2. KÖNNEN im Attribut *children* Ordner ablehnen.

Dem Objektsystem sind weitere, kanalspezifische Attribute zugeordnet, die einem flüchtig (im RAM) gespeicherten Kanalkontext (siehe (N029.900)) zugerechnet werden. Sie werden im Rahmen einer Selektion des Ordners verändert.

Gemäß 8.3, 8.4, 8.5 und 8.6 besitzen die dort genannten Objekttypen ein Attribut *lifeCycleStatus*. Gemäß (N020.000) und (N020.100) lassen sich die Instanzen aller Objekttypen Ebenen zuordnen, wobei Rekords hier als eine nächsttiefere Ebene zu der Datei aufgefasst werden, in der sie enthalten sind. Es wird nun unterschieden zwischen dem physikalischen und dem logischen Wert des *lifeCycleStatus*:

1. (N020.500) K\_COS   
   Als physikalischer Wert des *lifeCycleStatus* wird der Wert bezeichnet, den das entsprechende Attribut eines Objektes besitzt.
2. (N020.600) K\_COS   
   Als logischer Wert des *lifeCycleStatus* wird der Wert bezeichnet, der sich aus der Betrachtung des physikalischen Wertes des *lifeCycleStatus* eines Objektes sowie der logischen Werte des *lifeCycleStatus* in allen höheren Ebenen ergibt. Dabei gilt folgende Rangordnung und Rekursion:
   1. „Operational state (active)“<„Operational state (deactivated)“<„Termination state"
   2. Für das Objekt *root* MUSS als logischer Wert der nächsthöheren Ebene der physikalische Wert des *lifeCycleStatus* des Objektsystems verwendet werden (siehe (N019.900)i).
   3. Für ein Objekt mit einem Attribut *lifeCycleStatus* MUSS gelten: Falls
      1. der logische Wert des *lifeCycleStatus* der nächsthöheren Ebene größer ist als der physikalische Wert des *lifeCycleStatus* des Objektes, dann ist der logische Wert des *lifeCycleStatus* des Objektes gleich dem logischen Wert des *lifeCycleStatus* der nächsthöheren Ebene.
      2. sonst ist der der logische Wert des *lifeCycleStatus* des Objektes gleich seinem physikalischen.
   4. Für ein Objekt, welches selbst kein Attribut *lifeCycleStatus* besitzt, MUSS der logische Wert von *lifeCycleStatus* identisch zum logischen Wert der nächsthöheren Ebene sein.

## Objektsuche

Gemäß 9.1 ist an der Schnittstelle zur Karte ein hierarchisches Objektsystem sichtbar. Dieses Kapitel legt fest, wie ein Objekt gesucht und nach welchen Regeln es gefunden wird.

### Filesuche

Eine Suche nach Files (das heißt Ordnern und Dateien) findet lediglich im Rahmen einer Selektion statt. Deshalb ist die explizite Suche in 14.2.6 beschrieben. Daneben unterstützen die Kommandos, welche sich auf Dateien beziehen, Varianten mit Parameter *shortFileIdentifier*. Bei den jeweiligen Kommandos ist beschrieben, wie in diesem Fall eine Datei gesucht wird.

### Passwortsuche

Passwörter werden im Rahmen der Kommandos in 14.6 verwendet, aber auch bei der Auswertung von Zugriffsregeln (siehe (N022.200)). Bei der Passwortsuche handelt es sich um eine karteninterne Funktionalität, bei der viele Implementierungsdetails eine Rolle spielen. An der Schnittstelle „Interpreter“ (siehe Abbildung 1) wird das im Folgenden festgelegte Verhalten deutlich:

|  |  |  |
| --- | --- | --- |
| Input: | *passwordReference* | Gemäß (N072.800) |
| *startFolder* | Ordner, bei dem die Suche nach einem Passwortobjekt startet |
| Output: | *password* | Enthält das gefundene Passwortobjekt |
| Errors: | *pwdNotFound* | Zu den gegebenen Inputparametern wurde kein passendes Passwortobjekt gefunden |
| Notation: |  | *password* = SearchPwd( *startFolder*, *passwordReference* ) |

1. (N020.700) K\_COS   
   Der Inputparameter *passwordReference* wird gemäß (N072.800) wie folgt zerlegt:
   1. *identifier* = *passwordReference*   mod   128
   2. *location* = *passwordReference* – *identifier*
2. (N020.800) K\_COS   
   Wenn *location* den Wert ´00´ = 0 besitzt und damit ein globales Passwort adressiert, dann wird im Wurzelverzeichnis *root* des Objektsystems nach einem Passwort gesucht, dessen Attribut *pwdIdentifier* identisch zu *identifier* ist. Wenn ein solches Passwortobjekt
   1. existiert, dann wird es als Outputparameter *password* verwendet.
   2. nicht existiert, wird der Fehler „*pwdNotFound*“ zurückgemeldet.
3. (N020.900) K\_COS   
   Wenn *location* den Wert ´80´ = 128 besitzt und damit ein DF-spezifisches Passwort adressiert, dann wird die „lokale Variable“ *folder* auf *startFolder* gesetzt.
4. (N021.000) K\_COS   
   Wenn *folder*
   1. auf das Wurzelverzeichnis *root* zeigt, dann wird der Fehler „*pwdNotFound*“ zurückgemeldet.
   2. ansonsten wird in *folder* nach einem Passwort gesucht, dessen Attribut *pwdIdentifier* identisch zu *identifier* ist. Wenn ein solches Passwortobjekt
      1. existiert, dann wird es als Outputparameter *password* verwendet.
      2. nicht existiert, wird *folder* auf den nächsthöheren Ordner gesetzt und der Algorithmus mit Schritt (N021.000)a fortgesetzt.

### Suche nach einem Schlüsselobjekt

Die hier beschriebene Funktion ist eine Generalisierung der anderen Schlüsselsuchfunktionen und stellt damit den generalisierten Einsprungspunkt für die Schlüsselsuche dar, so wie sie an verschiedenen Stellen in diesem Dokument verwendet wird.

|  |  |  |
| --- | --- | --- |
| Input: | *startFolder* | Ordner, aus dem heraus die Suche startet |
| *identifier* | Für diesen Parameter sind die folgende Wertebereiche möglich:  a) ein Oktett *keyReference* gemäß (N099.600),  b) acht Oktett *keyIdentifier* gemäß (N019.100)a,  c) zwölf Oktett *keyIdentifier* gemäß (N019.500)a,  d) zwölf Oktett *keyIdentifier* gemäß (N019.822)a,  e) herstellerspezifisch zur Auswahl von Sessionkeys |
| *algID* | Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt, oder „Wildcard“ |
| Output: | *key* | Enthält das gefundene Schlüsselobjekt |
| Errors: | *keyNotFound* | Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden |
| *notSupported* | Der Schlüssel unterstützt den von *algID* geforderten Algorithmus nicht |
| Notation: |  | *key* = SearchKey( *startFolder*, *identifier*, *algID* ) |

1. (N021.050) K\_COS   
   Enthält der Parameter *identifier* eine Schlüsselreferenz gemäß
   1. (N099.600) (d. h. symmetrisches Authentisierungsobjekt, symmetrisches Kartenverbindungsobjekt oder privates Schlüsselobjekt), dann gilt:   
        *key* = SearchSecretKey( *startFolder*, *identifier*, *algID* )
   2. (N019.100)a oder (N019.500)a oder (N019.822)a (d. h. ein öffentliches Schlüsselobjekt), dann gilt:   
        *key* = SearchPublicKey( *startFolder*, *identifier*, *algID* )
   3. eines herstellerspezifischen Wertebereiches, dann gilt: Es handelt sich um Sessionkeys, die im Rahmen von PSO-Kommandos zur Unterstützung eines Trusted Channels eingesetzt werden. Solange der Sicherheitszustand der im Rahmen der Sessionkey Aushandlung gesetzt wurde noch besteht (vergleiche (N030.700)), existieren diese Sessionkeys. Solange die Sessionkeys existieren, MÜSSEN sie auch gefunden werden.

#### Suche nach einem geheimen Schlüsselobjekt mittels *keyReference*

In diesem Kapitel werden symmetrische Authentisierungsobjekte gemäß 8.6.1 und private Schlüsselobjekte gemäß 8.6.3 gemeinsam behandelt, da nach ihnen auf gleiche Art und Weise gesucht wird. Sie werden im Rahmen diverser kryptographischer Operationen verwendet. Symmetrische Authentisierungsobjekte werden zudem auch bei der Auswertung von Zugriffsregeln (siehe (N022.300)) verwendet. Bei dieser Art der Schlüsselsuche handelt es sich um eine karteninterne Funktionalität, bei der viele Implementierungsdetails eine Rolle spielen. An der Schnittstelle „Interpreter“ (siehe Abbildung 1) wird das im Folgenden festgelegte Verhalten deutlich:

|  |  |  |
| --- | --- | --- |
| Input: | *keyReference* | Gemäß (N099.600) |
| *algID* | Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt, oder „Wildcard“ |
| *startFolder* | Ordner, bei dem die Suche nach einem Schlüsselobjekt startet |
| Output: | *key* | Enthält das gefundene Schlüsselobjekt |
| Errors: | *keyNotFound* | Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden |
| *notSupported* | Der Schlüssel unterstützt den von *algID* geforderten Algorithmus nicht |
| Notation: |  | *key* = SearchSecretKey( *startFolder*, *keyReference*, *algID* ) |

1. (N021.100) K\_COS   
   Der Inputparameter *keyReference* wird gemäß (N099.600) in die Bestandteile
   1. *identifier* = *keyReference*   mod   128
   2. *location* = *keyReference* – *identifier*
2. (N021.200) K\_COS   
   Wenn *location* den Wert ´00´ = 0 besitzt und damit ein globales Schlüsselobjekt adressiert, dann MUSS im Wurzelverzeichnis *root* des Objektsystems nach einem Schlüsselobjekt gesucht werden, dessen Attribut *keyIdentifier* identisch zu *identifier* ist. Wenn ein solches Schlüsselobjekt
   1. existiert, dann wird es als Outputparameter *key* verwendet.
   2. nicht existiert, wird der Fehler „*keyNotFound*“ zurückgemeldet.
3. (N021.300) K\_COS   
   Wenn *location* den Wert ´80´ = 128 besitzt und damit ein DF-spezifisches Schlüsselobjekt adressiert, dann MUSS die „lokale Variable“ *folder* auf *startFolder* gesetzt.
4. (N021.400) K\_COS   
   Wenn *folder*
   1. auf das Wurzelverzeichnis *root* zeigt, dann wird der Fehler „*keyNotFound*“ zurückgemeldet.
   2. ansonsten wird in *folder* nach einem Schlüsselobjekt gesucht, dessen Attribut *keyIdentifier* identisch zu *identifier* ist. Wenn ein solches Schlüsselobjekt
      1. existiert, dann wird es als Outputparameter *key* verwendet.
      2. nicht existiert, wird *folder* auf den nächsthöheren Ordner gesetzt und der Algorithmus mit Schritt (N021.400)a fortgesetzt.
5. (N021.500) K\_COS   
   Wenn der Outputparameter *key* ein
   1. symmetrischer Schlüssel ist und dessen Attribut *algorithmIdentifier* nicht zum Parameter *algID* passt, dann wird der Fehler „notSupported“ zurückgemeldet.
   2. privater Schlüssel ist, dann wird in den Elementen von *listAlgorithmIdentifier* nach einem Element gesucht, dessen *seIdentifier* (siehe (N017.400)b.1) identisch zum Attribut *seIdentifier* (siehe (N030.000)a) des Ordners ist, in welchem *key* als child enthalten ist. Es wird der Fehler "notSupported" zurückgemeldet, wenn ein solches Element
      1. nicht existiert, oder
      2. existiert, aber die Menge *setAlgorithmIdentifier* (siehe (N017.400)b.2) das Element *algId* nicht enthält.

#### Suche nach einem öffentlichen Schlüsselobjekt

Öffentliche Schlüsselobjekte werden zum Importieren von Schlüsseln mittels Zertifikaten und im Rahmen von Authentisierungsprotokollen oder Verschlüsselung verwendet. Die Unterklasse „Öffentliche Authentisierungsobjekte“ wird zudem auch bei der Auswertung von Zugriffsregeln (siehe (N022.400) und (N022.500)) verwendet. Bei dieser Art der Schlüsselsuche handelt es sich um eine karteninterne Funktionalität, bei der viele Implementierungsdetails eine Rolle spielen. An der Schnittstelle „Interpreter“ (s. Abbildung 1) wird das im Folgenden festgelegte Verhalten deutlich:

|  |  |  |
| --- | --- | --- |
| Input: | *identifier* | Gemäß (N018.500), (N019.100)a, (N019.500)a, (N019.822)a |
| *algID* | Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt |
| *startFolder* | Ordner, aus dem heraus die Suche startet |
| Output: | *key* | Enthält das gefundene Schlüsselobjekt |
| Errors: | *keyNotFound* | Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden |
| *notSupported* | Der Schlüssel unterstützt den von algID geforderten Algorithmus nicht |
| Notation |  | *key* = SearchPublicKey( *startFolder*, *identifier*, *algID* ) |

1. (N021.600) K\_COS   
   Die „lokale Variable“ *folder* wird auf *startFolder* gesetzt. Dann werden folgende Schritte ausgeführt:
   1. Es wird in *allPublicKeyList* nach einem Schlüsselobjekt gesucht,
      1. dessen Attribut *keyIdentifier* identisch zu *identifier* ist und
      2. das dem Ordner *folder* zugeordnet ist.
   2. Wenn ein solches Schlüsselobjekt
      1. existiert, dann wird es als Outputparameter *key* verwendet.
      2. nicht existiert und *folder* ist
         1. gleich *root*, wird der Fehler „*keyNotFound*“ zurückgemeldet.
         2. ungleich *root*, dann wird *folder* auf den nächsthöheren Ordner gesetzt und der Algorithmus mit Schritt (N021.600)a fortgesetzt.
2. (N021.605) K\_COS   
   Wenn das Attribut *oid* des Outputparameters *key* nicht zur vorgegebenen *algID* passt (siehe (N021.610)), genau dann MUSS der Fehler *notSupported* zurückgemeldet werden.
3. (N021.610) K\_COS   
   Zum OID
   1. sigS\_ISO9796-2Withrsa\_sha256 (siehe Tabelle 271) MUSS genau *algID* gleich verifyCertificate (siehe Tabelle 270) passen.
   2. authS\_ISO9796-2Withrsa\_sha256\_mutual (siehe Tabelle 271) MÜSSEN genau die *algID* aus folgender Menge passen (siehe Tabelle 268): {rsaRoleCheck, rsaSessionkey4SM, rsaSessionkey4TC}.
   3. ecdsa-with-SHA256, ecdsa-with-SHA384 und ecdsa-with-SHA512 (siehe Tabelle 271) MUSS genau *algID* gleich verifyCertificate (siehe Tabelle 270) passen.

* 1. authS\_gemSpec-COS-G2\_ecc-with-sha256, authS\_gemSpec-COS-G2\_ecc-with-sha384 und authS\_gemSpec-COS-G2\_ecc-with-sha512 (siehe Tabelle 271) MÜSSEN genau die *algID* aus folgender Menge passen (siehe Tabelle 268): {elcAsynchronAdmin, elcRoleCheck, elcSessionkey4SM, elcSessionkey4TC}.
  2. id-RSAES-OAEP (siehe Tabelle 271) MUSS genau *algID* gleich rsaEncipherOaep (siehe Tabelle 269) passen.
  3. rsaEncryption (siehe Tabelle 271) MUSS genau *algID* gleich rsaEncipherPKCS1\_V1\_5 (siehe Tabelle 269) passen.
  4. id-ELC-shared-secret-calculation (siehe Tabelle 271) MUSS genau *algID* gleich elcSharedSecretCalculation (siehe Tabelle 269) passen.

## Cache für öffentliche Schlüsselobjekte

Die in diesem Unterkapitel beschriebene Funktion speichert ein übergebenes „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) oder „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) im Cache.

|  |  |  |
| --- | --- | --- |
| Input: | *puk* | „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) oder „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) |
| *folder* | Ordner, dem *puk* zuzuordnen ist |
| Output: | *persistent* | True =>   *puk* wurde in *persistentCache* gespeichert  False =>   *puk* wurde in *volatileCache* gespeichert |
| Notation |  | *persistent* = StoreInCache( *folder*, *puk* ) |

1. (N021.630) K\_COS, Option\_RSA\_CVC   
   Wenn *puk.publicKey* vom Typ *publicRsaKey* ist, dann gilt folgendes:
   1. *puk* MUSS in Cache gespeichert werden.
   2. *puk* KANN in *persistentCache* gespeichert.
   3. *puk* KANN in *volatileCache* gespeichert.
2. (N021.633) K\_COS   
   Wenn *puk.publicKey* vom Typ *publicElcKey* ist, dann gilt folgendes:
   1. *puk* MUSS in Cache gespeichert werden.
   2. Wenn in Cache ein Schlüsselobjekt mit identischem Attribut *keyIdentifier* enthalten ist und Schlüssellänge und Schlüsseltyp (RSA oder ELC)
      1. stimmen nicht überein, dann KANN der COS-Hersteller das weitere Verhalten des COS festlegen.
      2. stimmen überein, dann wird das im Cache vorhandene Schlüsselobjekt mit identischem *keyIdentifier* durch *puk* ersetzt.
   3. Wenn in Cache kein Schlüsselobjekt mit identischem Attribut *keyIdentifier* enthalten ist, gilt folgendes:
      1. Wenn in *persistentCache* nicht genügend Platz vorhanden ist, dann MUSS gemäß (N021.636) freier Platz geschaffen werden.
      2. Wenn in *persistentCache*
         1. genügend Platz vorhanden, dann MUSS *puk* in *persistentCache* gespeichert werden.
         2. nicht genügend Platz vorhanden, dann MUSS *puk* in *volatileCache* gespeichert werden.
3. (N021.636) K\_COS   
   Freien Platz in Cache schaffen:
   1. Schritt 1: In *persistentCache* wird nach abgelaufenen EE-Schlüssel(n) und abgelaufenen CA-Schlüssel(n) gesucht, unabhängig davon, ob diese in irgendeinem logischen Kanal in *channelContext*.*keyReferenceList* eingetragen sind oder nicht. Wenn solche Schlüsselobjekte
      1. existieren, dann wird eines oder mehrere aus *persistentCache* entfernt. Wenn dadurch genügend Platz geschaffen wird, dann bricht dieser Algorithmus ab, sonst wird dieser Schritt wiederholt.
      2. nicht existieren, dann wird mit Schritt 2 fortgefahren.
   2. Schritt 2: Dieser Schritt wird erforderlich, wenn durch das Löschen abgelaufener EE-Schlüssel oder abgelaufener CA-Schlüssel nicht genügend Platz geschaffen wurde. Dann wird in *persistentCache* nach EE-Schlüssel(n) oder CA-Schlüssel(n) gesucht, die in keinem logischen Kanal in *channelContext*.*keyReferenceList* eingetragen sind. Falls solche Schlüsselobjekte
      1. existieren, dann wird eines aus *persistentCache* entfernt. Wenn dadurch genügend Platz geschaffen wird, bricht dieser Algorithmus ab, sonst wird dieser Schritt wiederholt.
      2. nicht existieren, dann wird mit Schritt 3 fortgefahren.
   3. Schritt 3: Dieser Schritt wird erforderlich, wenn *persistentCache* voll ist mit Sicherheitsankern und Schlüsseln, die in *channelContext*.*keyReferenceList* selektiert sind. Wenn in *volatileCache*
      1. genügend freier Platz vorhanden ist, dann bricht dieser Algorithmus ab.
      2. nicht genügend Platz vorhanden, dann MUSS ein beliebiges Schlüsselobjekt aus Cache entfernt und dieser Schritt wiederholt werden.
4. (N021.638) K\_COS   
   Wird ein öffentliches Schlüsselobjekt aus Cache entfernt, so
   1. DARF dies in KEINEM logischen Kanal Auswirkungen auf *globalSecurityList* (siehe (N029.900)e) haben.
   2. DARF dies in KEINEM logischen Kanal Auswirkungen auf *dfSpecificSecurityList* (siehe (N029.900)f) haben.
   3. DARF dies in KEINEM logischen Kanal Auswirkungen auf *bitSecurityList* (siehe (N029.900)h) haben.
   4. KÖNNEN in beliebigen logischen Kanälen Einträge in *keyReferenceList*, welche eine Referenz auf das entfernte Schlüsselobjekt enthalten
      1. verändert werden, oder
      2. unverändert bleiben.
5. (N021.639) K\_COS   
   Für den Rückgabewert gilt: Wenn *puk* in *persistentCache* gespeichert wurde, dann ist *persistent* = True, andernfalls False.
6. Die Bedingungen, die zu (N021.633)b.1 führen, sind nur erfüllt, wenn es zwei verschiedene Instanzen mit derselben Schlüsselreferenz gibt. Das wird in [gemSpec\_PKI] ausgeschlossen.
7. In (N021.636) ist eine Strategie beschrieben, die in mehreren Schritten versucht ein „unwichtiges“ Schlüsselobjekt im Cache zu finden. Falls so ein „unwichtiges“ Schlüsselobjekt existiert, wird es aus dem Cache entfernt mit dem Ziel Platz für puk zu schaffen. Der Text wurde nicht mit dem Ziel geschrieben eine bestimmte Implementierung vorzugeben, sondern mit dem Ziel das gewünschte Verhalten zu verdeutlichen. Damit logische Kanäle bezüglich importierter Schlüssel unabhängig voneinander sind, ist persistentCache so zu dimensionieren, dass Schritt 3 in (N021.636)c nicht erforderlich wird. Dem COS ist es nicht möglich dies sicherzustellen. Beispielsweise ist es der externen Welt möglich sehr viele Sicherheitsanker zu importieren. In praxisrelevanten Fällen wären dann alte, überflüssige Sicherheitsanker im Rahmen einer Kartenadministration zu löschen um für freien Platz in persistentCache zu sorgen. Nach Abarbeitung des Algorithmus in (N021.636) ist im Cache genügend Platz für puk frei.
8. Wenn die Bedingung in (N021.636)a auf mehrere Schlüsselobjekte zutrifft, so wird hier weder festgelegt welches gelöscht wird, noch ob nur eines oder alle gelöscht werden.
9. Wenn die Bedingung in (N021.636)a auf mehrere zutrifft aber nur eines gelöscht wird, so erscheint es sinnvoll, das mit dem kleinsten CXD (am längsten abgelaufen) zu entfernen.
10. Wenn die Bedingung (N021.636)b auf mehrere Schlüsselobjekte zutrifft, so wird nicht festgelegt welches gelöscht wird. Es erscheint sinnvoll, aus der Menge der EE-Schlüssel das mit dem kleinsten CXD zu entfernen.

# Zugriffskontrolle (normativ)

Fast alle in Kapitel 14 beschriebenen Kommandos werden durch Zugriffsregeln geschützt. Das bedeutet, dass das Betriebssystem kontrolliert, ob der Sicherheitsstatus für die Ausführung der Operation ausreichend ist. Die in diesem Kapitel aufgeführten Regeln bilden eine Untermenge der in [ISO/IEC 7816-4#9.3.3] Expanded format definierten Regeln.

## Zugriffsart

Die Zugriffsart (access mode) zeigt an, ob die der Zugriffsart zugeordnete Zugriffsbedingung im Rahmen einer Zugriffsregelprüfung auszuwerten ist. Bei der Spezifikation von Anwendungen sind folgende Regeln einzuhalten:

1. (N021.700) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Zugriffsart MUSS eine Liste von Kommandobeschreibungen sein.
2. (N021.800) K\_Anwendungsspezifikation {K\_Karte}   
   Als Untermenge zu [ISO/IEC 7816-4#Tabelle 32] MUSS für jedes Listenelement gelten:
   1. Ein Listenelement MUSS den Namen eines Kommandos enthalten, der gemäß Kapitel 14 äquivalent ist zu genau einer Kombination aus CLA Byte (siehe 11.5.1) und INS Byte (siehe 11.5.2). Dabei gilt:
      1. Das CLA Byte in einer Zugriffsart MUSS die Kanalnummer null enthalten.
      2. Das CLA Byte in einer Zugriffsart DARF KEIN Secure Messaging anzeigen.
   2. Innerhalb eines Listenelementes ist der Parameter P1 (siehe 11.5.3) optional. Das bedeutet: Es MUSS möglich sein, dass ein Listenelement
      1. genau einen Parameter P1 enthält.
      2. keinen Parameter P1 enthält.
   3. Innerhalb eines Listenelementes ist der Parameter P2 (siehe 11.5.4) optional. Das bedeutet: Es MUSS möglich sein, dass ein Listenelement
      1. genau einen Parameter P2 enthält.
      2. keinen Parameter P2 enthält.
   4. Ein COS KANN weitere Kommandobeschreibungen
      1. unterstützen oder
      2. ablehnen.
3. (N021.900) K\_COS   
   Definition: Bei der Auswertung der Zugriffsart MUSS diese genau dann zur aktuellen Kommando-APDU passen, wenn die Bestandteile der Zugriffsart identisch sind zu den Bestandteilen der Kommando-APDU an der Schnittstelle „Interpreter“ (siehe Abbildung 1).

## Zugriffsbedingung

Eine Zugriffsbedingung ist ein boolescher Ausdruck. Bei der Spezifikation einer Anwendung sind folgende Regeln einzuhalten, wobei zur Beschreibung folgende Definitionen eingeführt werden:

|  |  |
| --- | --- |
| *objectFolder* | Falls diese Zugriffsbedingung zu einem Objekt   1. vom Typ Ordner gehört, dann ist *objectFolder* gleich dieser Ordner 2. vom Typ Sessionkey gehört, dann ist *objectFolder* gleich dem Ordner, auf den *SessionkeyContext*.*folderSessionkeys* zeigt. 3. anderen Typs gehört, dann ist *objectFolder* gleich dem Ordner, dessen Attribut *children* dieses Objekt enthält |
| path(*folder*) | Pfad eines Ordners mit Namen *folder*. Zum Pfad gehört nach dieser Definition sowohl der Ordner *folder*, als auch alle seine übergeordneten Ebenen gemäß 9.1 einschließlich *root* |

1. (N022.000) K\_COS   
   Das boolesche Element ALWAYS MUSS stets den Wert True liefern.
2. (N022.100) K\_COS   
   Das boolesche Element NEVER MUSS stets den Wert False liefern.
3. (N022.200) K\_COS   
   Das boolesche Element PWD(*passwordReference*) MUSS
   1. genau dann True liefern, wenn die Passwortsuche gemäß 9.2.2 mit den Inputparametern SearchPwd( *objectFolder,* *passwordReference* ) ein Passwort findet und wenigstens eine der beiden folgenden Bedingungen erfüllt ist:
      1. Dieses Passwortobjekt ist entweder in *globalPasswordList* (siehe (N029.900)i) oder in *dfSpecificPasswordList* (siehe (N029.900)j) enthalten und das Attribut *securityStatusEvaluationCounter* dieses Passwortes ist ungleich null. Dabei MUSS das Attribut *securityStatusEvaluationCounter* dieses Passwortes um eins dekrementiert werden. Falls durch das Dekrementieren der Wert auf Null gefallen ist,
         1. KANN der Eintrag in der Liste verbleiben oder
         2. KANN der Eintrag aus der Liste entfernt werden.
      2. Das Attribut *flagEnabled* dieses Passwortes besitzt den Wert False.
   2. in allen anderen Fällen False liefern.
4. (N022.300) K\_COS   
   Das boolesche Element AUT(*keyReference*) MUSS
   1. genau dann True liefern, wenn die Schlüsselsuche gemäß 9.2.3.1 mit den Inputparametern SearchSecretKey (*objectFolder,* *keyReference*, „Wildcard“) ein Schlüsselobjekt *key* findet und genau dieses Schlüsselobjekt entweder in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist.
   2. in allen anderen Fällen False liefern.
5. (N022.400) K\_COS   
   Das boolesche Element AUT(CHAT) MUSS
   1. genau dann True liefern, wenn es mindestens ein Element in *bitSecurityList* (siehe (N029.900)h) gibt, in welchem die OID übereinstimmt und mindestens dieselben Bits gesetzt sind wie in *accessFlags* (siehe auch Hinweis (50):) und dieses Element ist einem Ordner in path(*objectFolder*) zugeordnet (siehe (N030.400)b).
   2. in allen anderen Fällen False liefern.
6. (N022.500) K\_COS, Option\_RSA\_CVC   
   Das boolesche Element AUT(*CHA*) MUSS
   1. genau dann True liefern, wenn *CHA* in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist und *CHA* einem Ordner in path(*objectFolder*) zugeordnet ist (siehe (N030.300)b.2).
   2. in allen anderen Fällen False liefern.
7. (N022.600) K\_COS   
   Secure Messaging gemäß Kapitel 13 wird wie folgt in Zugriffsbedingungen verwendet:
   1. Das boolesche Element SmMac(*keyInformation*) MUSS genau dann den Wert True liefern, falls *SessionkeyContext.flagSessionEnabled* gleich SK4SM ist, und *SessionkeyContext*.*negotiationKeyInformation*
      1. eine Schlüsselreferenz gemäß (N099.600) enthält und AUT(*keyInformation*) gemäß (N022.300) liefert True, oder
      2. enthält *accessRights* gemäß (N019.700) mit
         1. Option\_RSA\_CVC, *CHA* gemäß 7.1.1.4 und AUT(*keyInformation*) gemäß (N022.500) liefert True, oder
         2. CHAT und AUT(*keyInformation*) gemäß (N022.400) liefert True,
      3. sonst False.
   2. Das boolesche Element SmCmdEnc MUSS den Wert von *SessionkeyContext.flagCmdEnc* liefern.
   3. Das boolesche Element SmRspEnc MUSS stets den Wert True liefern.

* 1. *SessionkeyContext.flagRspEnc* MUSS genau dann auf True gesetzt werden, wenn die Zugriffsbedingung das Element SmRspEnc enthält.

1. (N022.700) K\_Anwendungsspezifikation {K\_Karte}   
   Die zuvor definierten booleschen Elemente MÜSSEN sich zu einem booleschen Ausdruck verbinden lassen.
   1. Der boolesche Ausdruck MUSS den AND-Operator unterstützen.
   2. Der boolesche Ausdruck MUSS den OR-Operator unterstützen.
   3. Der boolesche Ausdruck DARF KEINEN anderen Operator enthalten.
2. (N022.710) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Verschlüsselung ist nur bei gleichzeitiger MAC-Sicherung sinnvoll, das heißt:
   1. Jedes boolsche Element SmCmdEnc MUSS durch den AND-Operator mit SmMac(*keyInformation*) verknüpft werden.
   2. Jedes boolsche Element SmRspEnc MUSS durch den AND-Operator mit SmMac(*keyInformation*) verknüpft werden.
3. (N022.800) K\_COS   
   Der boolesche Wert der Zugriffsbedingung MUSS das Ergebnis des booleschen Ausdrucks sein.
4. (N022.810) K\_COS   
   Für die Kommandos Get Challenge (siehe (N098.900)), Manage Channel (siehe (N099.545)), Manage Security Environment (siehe (N104.100) und Select (siehe (N047.600)) ist nicht festgelegt, ob Zugriffsregeln auszuwerten sind oder nicht. Darum MUSS für die Zugriffsbedingung dieser Kommandos gelten: Falls ein COS für eines der genannten Kommandos
   1. keine Zugriffsregeln auswertet, dann MUSS sich das COS so verhalten, als ob die Zugriffsbedingung ALWAYS lautet.
   2. sonst gilt für die Zugriffsbedingung dieses Kommandos:
      1. Im Element eins von *interfaceDependentAccessRules* (kontaktbehaftete Kommunikation) MUSS die Zugriffsbedingung ALWAYS lauten.
      2. Im Element zwei von *interfaceDependentAccessRules* (kontaktlose Kommunikation) MUSS die Zugriffsbedingung
         1. ALWAYS lauten, wenn beabsichtigt ist, diese Schnittstelle zu nutzen.
         2. NEVER lauten, wenn beabsichtigt ist, diese Schnittstelle nicht zu nutzen.
5. Die normativen Regeln dieses Dokumentes, insbesondere die   
   – zu Attributen eines Passwortes (siehe 8.4 und 8.5)   
   – zu Statusänderungen bei erfolgreicher Benutzerverifikation (siehe (N082.900))   
   – zum booleschen Element PWD (siehe (N022.200))   
   sind so aufgebaut, dass nach erfolgreicher Benutzerverifikation   
   – genau eine Operation (etwa Signatur) möglich ist (startSSEC = 1, s. (N015.800)c.2), oder   
   – genau n Operationen möglich sind (startSSEC = n > 1), oder   
   – beliebig viele Operationen möglich sind (startSSEC = „unendlich“).
6. Obwohl in (N029.900)h.1 festgelgt ist, dass eine Listenlänge von eins für das COS hinreichend ist um spezifikationskonform zu sein, wird in den folgenden Beispielen mit einer Listenlänge von zwei gearbeitet um den Inhalt von (N022.400) zu erläutern. Zur Vereinfachung sind in den Beispielen OID nicht dargestellt.
   1. Für bitSecurityList = {1000, 0111} ist AUT(1100} gleich False, weil es in bitSecurityList kein Element gibt, in welchem die beiden ersten Bits gesetzt sind.
   2. Für bitSecurityList = {1011, 1101} ist AUT(1100} gleich True, weil in bitSecurityList im zweiten Element ebenfalls die beiden ersten Bits gesetzt sind.
7. Gemäß (N022.600)a lässt sich für SmMac festlegen mit welchem Schlüssel Secure Messaging erfolgt. Eine ensprechende Festlegung für SmCmdEnc oder SmRspEnc ist nicht erforderlich, da ausschließlich Sessionkeys betrachtet werden, wo sowohl Kmac als auch Kenc gemeinsam vereinbart werden.

## Zugriffsregel

Eine Zugriffsregel kombiniert Zugriffsart und Zugriffsbedingung, und liefert bei der Auswertung ein boolesches Ergebnis. Bei der Spezifikation einer Anwendung sind folgende Regeln einzuhalten:

1. (N022.900) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Zugriffsregel MUSS eine Liste mit mindestens einem Element sein.
2. (N023.000) K\_Anwendungsspezifikation {K\_Karte}   
   Ein Listenelement MUSS genau eine Zugriffsart gemäß 10.1 und eine Zugriffsbedingung gemäß 10.2 enthalten.
3. (N023.100) K\_COS   
   Ein Listenelement MUSS
   1. genau dann den booleschen Wert True liefern, wenn
      1. die Zugriffsart gemäß (N021.900) passt und
      2. der boolesche Wert der Zugriffsbedingung den Wert True hat.
   2. in allen anderen Fällen False liefern (implizites NEVER).
4. (N023.200) K\_COS   
   Eine Zugriffsregel MUSS
   1. genau dann als erfüllt gelten, wenn mindestens ein Listenelement den Wert True liefert.
   2. in allen anderen Fällen als nicht erfüllt gelten.
5. (N023.300) K\_Anwendungsspezifikation {K\_Karte}   
   Eine Zugriffsregel, welche die übrigen Bedingungen des Kapitels 10 erfüllt, DARF bei Codierung gemäß [ISO/IEC 7816-4#9.3.3] NICHT mehr als 240 Oktette beanspruchen.

## Zugriffsregelauswertung

Dieses Unterkapitel beschreibt wie das COS feststellt, ob ein bestimmtes Kommando erlaubt oder verboten ist.

|  |  |  |
| --- | --- | --- |
| Input: | *obj* | Instanz eines Objektes vom Typ Ordner, Datei, Passwortobjekt, symmetrisches Authentisierungsobjekt oder privates Schlüsselobjekt |
| *CLA INS P1 P2* | CLA Byte  INS Byte  Parameter P1  Parameter P2 |
| Output: | *b* | Boolean, True bedeutet: Kommando erlaubt;   False bedeutet: Kommando verboten |
| Errors: | – | Keine |
| Notation: |  | *b* = AccessRuleEvaluation( *obj*, *CLA*, *INS*, *P1*, *P2* ) |

1. (N023.400) K\_COS   
   Schritt 1: Handelt es sich bei *obj*
   1. um einen Ordner, dann MUSS *se* = *seIdentifier* dieses Ordners sein (siehe (N030.000)a.
   2. um einen Sessionkey, dann MUSS *se* = *seIdentifier* des Ordners sein (siehe (N030.000)a), auf den *SessionkeyContext*.*folderSessionkeys* zeigt.
   3. sonst MUSS *se* = *seIdentifier* des Ordners sein (siehe (N030.000)a), der *obj* in seinem Attribut *children* (siehe (N010.000)) enthält.
2. (N023.500) K\_COS   
   Schritt 2: Unter Berücksichtigung des aktuell aktiven Übertragungsprotokolls (kontaktbehaftet oder kontaktlos) des logischen Wertes von *obj.lifeCycleStatus* (siehe (N020.600)) und *se* MUSS aus *obj*.*accessRules*, welches vom Typ *interfaceDependentAccessRules* ist, die passende Zugriffsregel ausgewählt werden (siehe 8.1.4).
3. (N023.600) K\_COS   
   Schritt 3: Anhand von *CLA*, *INS*, *P1* und *P2* MUSS ermittelt werden, ob die in (N023.500) ausgewählte Zugriffsregel erfüllt ist (siehe 10.3).
4. (N023.700) K\_COS   
   Schritt 4: Ist die ausgewählte Zugriffsregel
   1. erfüllt, dann MUSS *b* = True gelten.
   2. sonst MUSS *b* = False gelten.
5. Falls in folgender Zugriffsbedingung PWD1 OR PWD2 beide Passwortobjekte mit SSEC ungleich unendlich vorkommen, dann ist es herstellerspezifisch, ob beide SSEC oder nur einer und in diesem Falle welcher dekrementiert wird.

## Auslesen von Zugriffsregeln

In 10.1 bis 10.4 wird festgelegt in welchem Rahmen sich eine Anwendungsspezifikation im Bereich Zugriffsregeln zu bewegen hat. Zur COS-internen Codierung von Zugriffsregeln gibt es keine Anforderung. Die COS-interne Codierung von Zugriffsregeln ist und bleibt herstellerspezifisch und liegt damit außerhalb des Regelungsbereiches dieses Dokumentes.

Das Konzept zur Evaluierung und damit zur Zulassung des COS sieht ab der Generation 2 vor, dass Zugriffsregeln aus einer Smartcard auslesbar sind. Dieses Unterkapitel nennt die Anforderungen:

1. Die folgenden Anforderungen beziehen sich ausschließlich auf Zugriffsregeln, die Teil einer Anwendungsspezifikation sind. Da alle Anwendungsspezifikationen auf den Webseiten der gematik allgemein zugänglich veröffentlicht werden, sind sie öffentlich. Aus datenschutzrechtlicher Sicht ist deshalb gegen das Auslesen dieser Zugriffsregeln nichts einzuwenden. Das Auslesen dient dem Nachweis, dass eine Smartcard Zugriffsregeln semantisch spezifikationskonform umsetzt.
2. (N023.740) K\_COS   
   Auslesen von Zugriffsregeln:
   1. Jede Zugriffsregel, die in einer Anwendungsspezifikation definiert wird,
      1. MUSS auslesbar sein.
      2. SOLL frei auslesbar sein, das heißt die Zugriffsregel zum Auslesen von Zugriffsregeln SOLL „Read Always“ sein.
   2. Zugriffsregeln, die nicht in einer Anwendungsspezifikation definiert werden (etwa für herstellerspezifische zusätzliche Objekte EF.Pwd, etc.)
      1. KÖNNEN auslesbar sein,
      2. KÖNNEN nicht auslesbar sein.

# Kommunikation (normativ)

## Request – Response

Für Smartcards entspricht es dem Stand der Technik, dass sie Nachrichten mit einem externen Kommunikationspartner über einen Kanal im Halbduplex-Verfahren austauschen. Zudem arbeiten sie ähnlich wie ein Server. Das bedeutet, dass der externe Kommunikationspartner eine Nachricht (Kommando) über den Kanal schickt. Diese Nachricht (Kommando) wird von der Smartcard verarbeitet. Anschließend sendet die Smartcard über denselben Kanal eine Nachricht (Antwort) zurück. Erst nach dem vollständigen Empfang der Smartcard-Nachricht hat der externe Kommunikationspartner die Möglichkeit, eine weitere Nachricht zu schicken.

## Elektrische Schnittstellen

Dieses Kapitel behandelt elektrische Schnittstellen zum COS. Das Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] ist verpflichtend und wird in 11.2.1 näher spezifiziert. Eine Datenübertragung gemäß [ISO/IEC 7816-12] (siehe 11.2.2) oder kontaklos gemäß ISO/IEC 14443 (siehe 11.2.3) ist optional.

### Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11]

Dieses Kapitel behandelt das kontaktbehaftete Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11]. Die Unterstützung dieses Übertragungsprotokolls ist verpflichtend.

1. (N023.800) Diese Anforderung ist absichtlich leer.
2. (N023.801) K\_COS   
   Das COS MUSS das kontaktbehaftete Übertragungsprotokoll T=1 unterstützen. Das Intervall [(N023.820), (N024.220)] spezifiziert die diesbezüglichen Anforderungen.
3. (N023.820) K\_IC   
   Elektrische Eigenschaften:
   1. Das IC einer Smartcard MUSS elektrische Eigenschaften gemäß [EMV®\_Book-1#5.3] aufweisen.
   2. Das IC einer Smartcard MUSS die Spannungklasse „Class A“ unterstützen.
   3. Das IC einer Smartcard MUSS die Spannungsklasse „Class B“ unterstützen.
   4. Das IC einer Smartcard KANN die Spannungsklasse „Class C“ unterstützen.
   5. Das IC einer Smartcard KANN die Unterstützung der Spannungsklasse „Class C“ ablehnen.
   6. Alle Spannungsklassen, die das IC einer Smartcard im ATR oder in einem EF.ATR anzeigt, MÜSSEN unterstützt werden und konform zu [EMV®\_Book-1#5.3] implementiert sein (siehe auch (N024.100)i).
4. (N023.900) Diese Anforderung ist absichtlich leer.
5. (N023.920) K\_IC, K\_externeWelt {K\_Karte}   
   Eine Kartensession gliedert sich in die Abschnitte Kartenaktivierung, Reset, Nachrichtenaustausch und Deaktivierung. Dabei gilt:
   1. Der externe Kommunikationspartner MUSS die Smartcard gemäß [EMV®\_Book-1#6.1.2] aktivieren.
   2. Im Anschluss an die Aktivierung gemäß (N023.920)a MUSS der externe Kommunikationspartner einen Cold Reset gemäß [EMV®\_Book-1#6.1.3.1] ausführen.
   3. Das IC einer Smartcard MUSS einen Warm Reset gemäß [EMV®\_Book-1#6.1.3.2] unterstützen.
   4. Die „Information Field Size“ IFSD MUSS durch einen IFSD-Request (siehe [ISO/IEC 7816-3#11.4.2]) auf den Wert ´FE´ = 254 gesetzt werden.
   5. Der externe Kommunikationspartner KANN nach Abschluss eines Resets mittels des hier spezifizierten Übertragungsprotokolls beliebige Nachrichten austauschen.
   6. Deaktivierung:
      1. Der externe Kommunikationspartner SOLL das IC einer Smartcard gemäß [EMV®\_Book-1#6.1.5] deaktivieren.
      2. Die Deaktivierung KANN auf andere Art erfolgen.
      3. Geschieht die Deaktivierung zu einem Zeitpunkt, in welchem eine transaktionsgeschützte Schreiboperation durchgeführt wird (siehe 14.1), so MUSS das COS dafür Sorge tragen, dass gemäß den Regeln in 14.1.1 bzw. 14.1.2 verfahren wird, bevor die nächste Kommando-APDU von der Komponente „Cmd Interpreter“ in Abbildung 1 bearbeitet wird.
6. (N023.940) K\_COS   
   Falls das COS keinen IFSD-Request empfängt (siehe (N023.920)d), dann MUSS das COS mit einem Wert IFSD=254 (siehe [EMV®\_Book-1#9.2.4.2.1]) arbeiten.
7. (N024.000) Diese Anforderung ist absichtlich leer.
8. (N024.020) K\_IC, K\_externeWelt {K\_Karte}   
   Im Rahmen des hier spezifizeirten Übertragungsprotkolls MÜSSEN auf der physikalischen Schnittstelle (siehe Abbildung 1) Character gemäß [EMV®\_Book-1#7.2] verwendet werden.
9. (N024.100) K\_COS   
   Ein Cold Reset gemäß (N023.920)b und ein Warm Reset gemäß (N023.920)c lassen sich als spezielle „Nachrichten“ des externen Kommunikationspartners auffassen, die von der Smartcard mit einem Answer to Reset (ATR) gemäß [ISO/IEC 7816-3#8.2] beantwortet wird. Dabei gilt:
   1. Das COS MUSS den ATR gemäß [EMV®\_Book-1#8.1] versenden.
   2. Das COS MUSS als „initial character“ im ATR den Wert ´3B´ verwenden. Daraus ergibt sich, dass die Kommunikation mittels „direct convention” (siehe [ISO/IEC 7816-3#8.1]) erfolgt.
   3. Das COS MUSS im Rahmen eines Cold Reset für das TA1 Byte im ATR einen Wert aus der Menge {´18´, ´95´, ´96´, ´97´} verwenden.
   4. Das COS MUSS im Rahmen eines Warm Reset für das TA1 Byte im ATR einen Wert aus der Menge {´18´, ´95´, ´96´} verwenden.
   5. Für das TC1 Byte im ATR gilt:
      1. Das COS SOLL im ATR ein TC1 Byte enthalten.
      2. Das COS KANN im ATR auf ein TC1 Byte verzichten.
      3. Falls das TC1 Byte im ATR vorhanden ist, dann MUSS es den Wert ´FF´ besitzen.
   6. Das COS MUSS im TD1 Byte des ATR das Protokoll T=1 anzeigen.
   7. Das COS DARF in den ATR kein TA2 Byte einstellen. Daraus folgt, dass das COS den „negotiable mode“ unterstützen MUSS (siehe (N024.220)).
   8. Das COS MUSS im TA3 Byte den Wert 254 = ´FE´ enthalten. Dadurch wird gemäß [ISO/IEC 7816-3#11.4.2] eine „Information Field Size“ IFSC = 254 = ´FE´ angezeigt.
   9. Das COS MUSS im ATR einen „class indicator“ gemäß [ISO/IEC 7816-3#Tabelle 10] anzeigen. Der „class indicator“ MUSS einen Wert aus der Menge {„A und B“, „A, B und C“} = {3, 7} haben.
   10. Falls das COS im ATR in den Historical Bytes die „Third software function table“ gemäß [ISO/IEC 7816-4#Table 119] enthält, dann MUSS die dort angezeigte Anzahl logischer Kanäle kleiner oder gleich der Anzahl maximal unterstützter logischer Kanäle sein.
10. (N024.200) Diese Anforderung ist absichtlich leer.
11. (N024.220) K\_COS   
    Gemäß (N024.100)g zeigt der ATR stets den „negotiable mode“ an. Deshalb MUSS das COS das „Protocol-and-Parameter-Selection“-Verfahren gemäß [ISO/IEC 7816-3#9] unterstützen. Im einzelnen bedeutet das:
    1. Falls TA1 im ATR den Wert ´18´ besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {´12´, ´13´, ´18´} akzeptieren.
    2. Falls TA1 im ATR den Wert ´95´ besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {´92´, ´93´, ´94´, ´95´} akzeptieren.
    3. Falls TA1 im ATR den Wert ´96 besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {´92´, ´93´, ´94´, ´95´, ´96´} akzeptieren.
    4. Falls TA1 im ATR den Wert ´97 besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {´92´, ´93´, ´94´, ´95´, ´96´, ´97´} akzeptieren.
12. (N024.300) Diese Anforderung ist absichtlich leer.
13. (N024.320) K\_COS, K\_externeWelt {K\_Karte}   
    Das COS MUSS bezüglich der Datenübertragung mittels des Blockprotokolls T=1 die Vorgaben aus [EMV®\_Book-1#9.2.4] erfüllen und bezüglich der Fehlerbehandlung die Vorgaben aus [ISO/IEC 7816-3#11.6.3] erfüllen.
14. (N024.400) Diese Anforderung ist absichtlich leer.
15. (N024.500) Diese Anforderung ist absichtlich leer.
16. (N024.600) Diese Anforderung ist absichtlich leer.

### Übertragungsprotokoll gemäß [ISO/IEC 7816-12]

Dieses Kapitel behandelt das kontaktbehaftete Übertragungsprotokoll gemäß [ISO/IEC 7816-12]. Dieses Übertragungsprotokolls wird von der Smartcard optional bereitgestellt.

1. (N024.700) Diese Anforderung ist absichtlich leer.
2. (N024.800) Diese Anforderung ist absichtlich leer.
3. (N024.810) K\_IC, Option\_USB\_Schnittstelle   
   Das IC einer Smartcard MUSS das kontaktbehaftete Übertragungsprotokoll gemäß [ISO/IEC 7816-12] unterstützen, wobei gilt:
   1. Bulk Transfer gemäß [ISO/IEC 7816-12#8.1] MUSS unterstützt werden.
   2. Weitere Übertragungsarten aus [ISO/IEC 7816-12]
      1. KÖNNEN unterstützt werden,
      2. KÖNNEN fehlen.
4. (N024.820) K\_IC, Option\_USB\_Schnittstelle   
   Für die Übertragungsgeschwindigkeit des hier betrachteten Protokolls gilt:
   1. Das IC MUSS die Übertragungsgeschwindigkeit „Low Speed“ unterstützen.
   2. Das IC KANN weitere Übertragungsgeschwindigkeiten
      1. anbieten oder
      2. ablehnen.
5. (N024.900) Diese Anforderung ist absichtlich leer.
6. (N025.000) Diese Anforderung ist absichtlich leer.
7. (N025.100) Diese Anforderung ist absichtlich leer.

### Kontaktlose Datenübertragung gemäß ISO/IEC 14443

Dieses Kapitel behandelt die kontaktlose Datenübertragung gemäß ISO/IEC 14443. Die Unterstützung dieser Datenübertragungsart ist optional.

1. (N025.200) Diese Anforderung ist absichtlich leer.
2. (N025.300) Diese Anforderung ist absichtlich leer.
3. (N025.310) K\_IC, Option\_kontaktlose\_Schnittstelle   
   Das IC einer Smartcard MUSS die kontaktlose Datenübertragung gemäß [ISO/IEC 14443-1], [ISO/IEC 14443-2], [ISO/IEC 14443-3] und [ISO/IEC 14443-4] unterstützen. Dabei gelten folgende Besonderheiten:
   1. Das IC und das COS MÜSSEN einen Typ aus der Menge {„Typ A“, „Typ B“} unterstützen.
   2. Das IC und das COS
      1. MÜSSEN alle Datenraten aus der Menge {*fC* / 128, *fC* / 64, *fC* / 32, *fC* / 16} unterstützen und
      2. KÖNNEN weitere Datenraten unterstützen.
   3. Die gesamte Karte bestehend aus IC, COS und Antenne im Kartenkörper
      1. MUSS alle Datenraten aus der Menge {*fC* / 128} unterstützen.
      2. SOLL alle Datenraten aus der Menge {*fC* / 64, *fC* / 32, *fC* / 16} unterstützen.
      3. KANN weitere Datenraten unterstützen.
4. (N025.320) K\_IC, Option\_kontaktlose\_Schnittstelle   
   Das IC einer Smartcard MUSS bei der kontaktlosen Datenübertragung spätestens mit Beginn der Nutzungsphase (siehe Kapitel 4) dynamisch generierte, zufällig gewählte Kennnummern verwenden, siehe dazu
   1. [ISO/IEC 14443-3#6.5.4] für „Typ A“ Smartcard und
   2. [ISO/IEC 14443-3#7.9.2] für „Typ B“ Smartcard
5. (N025.400) Diese Anforderung ist absichtlich leer.
6. (N025.500) Diese Anforderung ist absichtlich leer.
7. (N025.510) Diese Anforderung ist absichtlich leer.

## OSI-Referenzmodell (informativ)

In Anlehnung an das OSI-Referenzmodell werden hier verschiedene Layer definiert, die an der Bearbeitung einer Nachricht beteiligt sind, welche von einem externen Kommunikationspartner gesendet wurde. Es sei ausdrücklich darauf hingewiesen, dass die Inhalte dieses Abschnittes keine Implementierungsdetails festlegen. Das bedeutet, es ist zulässig, den internen Aufbau eines Smartcard-Betriebssystems oder dessen Kommunikationsaufbau anders zu gestalten. Die normativen Teile dieses Dokumentes lassen sich aber leichter beschreiben und verstehen, wenn man das Folgende zugrunde legt.

In Abbildung 1 wird exemplarisch gezeigt, wie eine Kommando-APDU CmdApdu1 vom externen Kommunikationspartner zunächst entsprechend den Konventionen des Übertragungsprotokolls (siehe 11.2) im physikalischen Layer und im Data Link Layer in einen oder möglicherweise mehrere TPDU zerlegt und im I/O der Smartcard wieder zu einer CmdApdu1 gemäß 11.5 zusammengesetzt wird.

Der nächste Layer verwaltet logische Kanäle (siehe [ISO/IEC 7816-4#5.4.2]) und leitet das empfangene Kommando entsprechend der Kanalnummer im CLA Byte weiter an den entsprechenden logischen Kanal, hier Channel\_x.

Im nächsten Layer „SecMes“ wird die optional per Secure Messaging gesicherte Kommando-APDU ausgepackt und an den Kommandointerpreter weitergeleitet.

Der Kommandointerpreter verarbeitet die Kommando-APDU und erstellt eine entsprechende Antwort-APDU, welche optional per Secure Messaging gesichert wird. Die eventuell gesicherte Antwort wird im I/O in eine oder mehrere TPDU zerlegt, welche dann über die physikalische Schnittstelle an den externen Kommunikationspartner übermittelt werden.



Abbildung 1: Message Sequence Chart für die Kommandobearbeitung

## Kommandobearbeitung

Dieses Kapitel beschreibt die Bearbeitung eines Kommandos, welches von einem externen Kommunikationspartner an das COS gesendet wurde. Es werden die Begriffe aus 11.3 verwendet. Es wird nochmals darauf verwiesen, dass die Inhalte von 11.3 nicht normativ sind. Insofern ist die genaue Zuordnung der normativen Aussagen dieses Kapitels zu Teilen eines COS nicht festgelegt. Wichtig ist aber, dass das Verhalten des COS auf Ebene der physikalischen Schnittstelle den normativen Vorgaben entspricht.

Bei der Bearbeitung einer Kommando-APDU und der Generierung einer Antwort-APDU werden folgende Schritte durchlaufen:

1. (N025.600) K\_externeWelt {K\_Karte}   
   Ein externer Kommunikationspartner sendet über eine physikalische Schnittstelle eine Kommando-APDU CmdApdu1, welche 11.5 entsprechen MUSS. Dazu wird CmdApdu1 auf der physikalischen Schnittstelle entsprechend dem Übertragungsprotokoll (siehe 11.2) als (eine oder mehrere) TPDU transportiert.
2. (N025.650) K\_COS   
   Das I/O der Smartcard MUSS die TPDU zusammensetzen und gewinnt dabei CmdApdu1 zurück.
3. (N025.700) K\_COS, K\_externeWelt {K\_Karte}   
   Die Fehlerbehandlung auf der physikalischen Schnittstelle MUSS [EMV®\_Book-1] entsprechen.
4. (N025.800) K\_COS   
   Anschließend MUSS die Kanalnummer aus dem CLA Byte von CmdApdu1 extrahiert werden.
5. (N025.900) K\_COS   
   Falls der Kanal mit der Kanalnummer aus dem CLA Byte nicht geöffnet ist, MUSS
   1. die Bearbeitung des Kommandos terminieren.
   2. RspApdu3 in diesem Fall keine Antwortdaten enthalten und besteht lediglich aus dem Trailer ChannelClosed = ´68 81´.
6. (N026.000) K\_COS   
   Entsprechend der Kanalnummer MUSS CmdApdu1 unverändert als CmdApdu2 an den entsprechenden logischen Kanal weitergeleitet werden.
7. (N026.100) Dieser Punkt ist absichtlich leer.
8. (N026.200) K\_COS   
   Innerhalb des logischen Kanals MUSS CmdApdu2 mit dem entsprechenden Kanalkontext *channelContext* an den Secure Messaging Layer („SecMes“ in Abbildung 1) weitergeleitet und dort gemäß 13.1.2 bearbeitet werden. Das Ergebnis ist RspApdu2.
9. (N026.300) K\_COS   
   RspApdu2 MUSS unverändert als RspApdu3 an das I/O der Smartcard geschickt werden.
10. (N026.400) K\_COS   
    Das I/O der Smartcard MUSS RspApdu3 gemäß dem verwendeten Übertragungsprotokoll in eine oder mehrere TPDU umwandeln und über die physikalische Schnittstelle senden.

## Kommando-APDU

### Class Byte

Das Class Byte (CLA Byte) enthält die Kommandoklasse. Zusammen mit dem Instruction Byte (siehe 11.5.2) legt es eindeutig das auszuführende Kommando fest.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

1. (N026.500) K\_externeWelt {K\_Karte}   
   Gemäß [ISO/IEC 7816-3] MUSS das CLA Byte in einem Oktett codiert werden.
2. (N026.510) K\_COS   
   Die Bits im CLA Byte MÜSSEN die Bedeutung gemäß Tabelle 9 oder Tabelle 10 haben.

Tabelle 9: Codierung des CLA-Bytes für Kanalnummern kleiner vier

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Beschreibung** |
| x | 0 | 0 | ‑ | ‑ | ‑ | ‑ | ‑ | **Unterscheidung der Kommandoklasse** |
| 0 | 0 | 0 | ‑ | ‑ | ‑ | ‑ | ‑ | Kommando gemäß ISO/IEC 7816 |
| 1 | 0 | 0 | ‑ | ‑ | ‑ | ‑ | ‑ | Kommandoerweiterung |
| ‑ | 0 | 0 | x | ‑ | ‑ | ‑ | ‑ | **Command Chaining Indikator** |
| ‑ | 0 | 0 | 0 | ‑ | ‑ | ‑ | ‑ | Einziges Kommando oder letztes einer Kette |
| ‑ | 0 | 0 | 1 | ‑ | ‑ | ‑ | ‑ | Nicht letztes Kommando einer Kette |
| ‑ | 0 | 0 | ‑ | x | x | ‑ | ‑ | **Secure Messaging Indikator** |
| ‑ | 0 | 0 | ‑ | 1 | 1 | ‑ | ‑ | Secure Messaging gemäß 13.1.2 |
| ‑ | 0 | 0 | ‑ | 0 | 0 | ‑ | ‑ | Kein Secure Messaging |
| ‑ | 0 | 0 | ‑ | ‑ | ‑ | x | x | **Kanalnummer gemäß [ISO/IEC 7816-4#Tab. 2]** |

Tabelle 10: Codierung des CLA-Bytes für Kanalnummern größer gleich vier

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **Beschreibung** |
| x | 1 | ‑ | ‑ | ‑ | ‑ | ‑ | ‑ | **Unterscheidung der Kommandoklasse** |
| 0 | 1 | ‑ | ‑ | ‑ | ‑ | ‑ | ‑ | Kommando gemäß ISO/IEC 7816 |
| 1 | 1 | ‑ | ‑ | ‑ | ‑ | ‑ | ‑ | Kommandoerweiterung |
| ‑ | 1 | x | ‑ | ‑ | ‑ | ‑ | ‑ | **Secure Messaging Indikator** |
| ‑ | 1 | 1 | ‑ | ‑ | ‑ | ‑ | ‑ | Secure Messaging gemäß 13.1.2 |
| ‑ | 1 | 0 | ‑ | ‑ | ‑ | ‑ | ‑ | Kein Secure Messaging |
| ‑ | 1 | ‑ | x | ‑ | ‑ | ‑ | ‑ | **Command Chaining Indikator** |
| ‑ | 1 | ‑ | 0 | ‑ | ‑ | ‑ | ‑ | Einziges Kommando oder letztes einer Kette |
| ‑ | 1 | ‑ | 1 | ‑ | ‑ | ‑ | ‑ | Nicht letztes Kommando einer Kette |
| ‑ | 1 | ‑ | ‑ | x | x | x | x | **Kanalnummer gemäß [ISO/IEC 7816-4#Tab. 3]** |

### Instruction Byte

Das Instruction Byte (INS Byte) enthält die Codierung für den auszuführenden Befehl. Zusammen mit dem Class Byte (siehe 11.5.1) legt es eindeutig das auszuführende Kommando fest.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

1. (N026.600) K\_externeWelt {K\_Karte}   
   Gemäß [ISO/IEC 7816-3] MUSS das INS Byte in einem Oktett codiert werden.

### Parameter P1

Der Parameter P1 enthält in der Regel eine Variable, die bei der Kommandoausführung benötigt wird. Die Bedeutung dieses Parameters hängt gewöhnlich von der Kombination aus CLA, INS und P2 ab.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

1. (N026.700) K\_externeWelt {K\_Karte}   
   Gemäß [ISO/IEC 7816-3] MUSS der Parameter P1 in einem Oktett codiert werden.

### Parameter P2

Der Parameter P2 enthält in der Regel eine Variable, die bei der Kommandoausführung benötigt wird. Die Bedeutung dieses Parameters hängt von der Kombination aus CLA, INS und P1 ab.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

1. (N026.800) K\_externeWelt {K\_Karte}   
   Gemäß [ISO/IEC 7816-3] MUSS der Parameter P2 in einem Oktett codiert werden.

### Datenfeld

Das Datenfeld einer Kommando-APDU ist optional. Es ist möglich, dass es fehlt. Das Datenfeld ist ein Oktettstring der Länge Nc. Gemäß [ISO/IEC 7816-3] ist der Definitionsbereich von Nc gleich {1, …, 65535}.

1. (N026.900) K\_COS   
   An der Schnittstelle „Interpreter“ (siehe Abbildung 1) gilt für Nc:
   1. Das COS MUSS für Nc alle Werte unterstützen, die sich aus (N029.892) ergeben.
   2. Ein COS KANN weitere Werte für Nc
      1. unterstützen oder
      2. ablehnen.
   3. Ein COS DARF eine APDU NICHT wegen eines Längenfehlers abweisen, wenn
      1. die Kommando-APDU den Anforderungen aus 11.7 genügt und
      2. das Datenfeld die Längenbeschränkung aus (N026.900)a oder die gesamte Kommando-APDU die Längenbeschränkung aus EF.ATR erfüllt.
2. (N026.910) K\_externeWelt {K\_Karte}   
   Der Sender einer Kommando-APDU SOLL die Längenbeschränkung durch passende Wahl von Nc einhalten.
3. Die obere Grenze in (N026.900)a gilt für das COS einer Smartcard. Für das IFD (Kartenleser) wird wegen möglicher, zukünftiger Erweiterungen, Entwicklungen und wegen Performanzgewinns empfohlen, mindestens 4096 = ´1000´ zu unterstützen.
4. Anders als in der Generation 1 ist die obere Grenze in (N026.900)a nicht mehr durch ein bestimmtes Kommando motiviert, welches besonders viele Daten überträgt und sich zudem schlecht auf mehrere Kommandos aufteilen lässt. Die in die (N026.900)a genannte Grenze ergibt sich aus Performanzüberlegungen verbunden mit dem aktuellen Stand der Technik.

### LeFeld

Das LeFeld einer Kommando-APDU ist optional. Es ist möglich, dass es fehlt. Das LeFeld enthält eine Zahl Ne, die angibt, wie viele Oktette im Datenfeld der Antwort-APDU erwartet werden. In diesem Dokument gelten folgende Definitionen:

|  |  |
| --- | --- |
| WildCardShort | Dieser Wert wird verwendet, um anzuzeigen, dass innerhalb der Obergrenze von 256 Oktetten alle verfügbaren Oktette in das Datenfeld der Antwortnachricht einzustellen sind. |
| WildCardExtended | Dieser Wert wird verwendet, um anzuzeigen, dass innerhalb der Obergrenze von 65.536 Oktetten alle verfügbaren Oktette in das Datenfeld der Antwortnachricht einzustellen sind. |

Gemäß [ISO/IEC 7816-4] ist der Definitionsbereich für Ne: {1, …, 65.535, WildCardShort, WildCardExtended}

1. (N027.000) K\_COS   
   An der Schnittstelle „Interpreter“ (siehe Abbildung 1) gilt für Ne:
   1. Das COS MUSS für Ne alle Werte aus der Menge {1, 2, …, 65.534, 65.535, WildCardShort, WildCardExtended} unterstützen.
   2. Ein COS DARF eine APDU NICHT wegen eines Längenfehlers abweisen, wenn die Längenbeschränkung aus (N027.000)a oder die gesamte Antwort-APDU die Längenbeschränkung aus EF.ATR erfüllt.
2. (N027.010) K\_externeWelt {K\_Karte}   
   Der Sender einer Kommando-APDU SOLL die Längenbeschränkung durch passende Wahl von Ne einhalten.
3. Die obere Grenze in (N027.000)a gilt für das COS einer Smartcard und (N027.010) gilt für die „externe Welt“. Für das IFD (Kartenleser) wird wegen Performanzgewinns empfohlen, ebenfalls die gemäß [ISO/IEC 7816-3] maximal mögliche Länge einer Antwortnachricht zu unterstützen.
4. Anders als in der Generation 1 ist die obere Grenze in (N027.000)a nicht mehr durch ein bestimmtes Kommando motiviert, welches besonders viele Daten überträgt und sich zudem schlecht auf mehrere Kommandos aufteilen lässt. Im Rahmen dieses Dokumentes ist es das Read Binary Kommando, welches die größtmögliche Antwortdatenmenge überträgt.

## Antwort-APDU

Mit den Definitionen aus den 11.6.1 und 11.6.2 ergibt sich für eine Antwort-APDU:

1. (N027.100) K\_COS   
   Eine Antwort-APDU MUSS ein Oktettstring gemäß ´*rspData*   ||   Trailer´ sein, wobei *rspData* möglicherweise leer ist.

### Datenfeld

Das Datenfeld *rspData* einer Antwort-APDU ist optional. Es ist möglich, dass es fehlt. Das Datenfeld *rspData* ist ein Oktettstring der Länge Nr. Gemäß [ISO/IEC 7816-3] ist der Definitionsbereich von Nr {1, …, 65536}.

1. (N027.200) K\_COS   
   An der Schnittstelle „Interpreter“ (siehe Abbildung 1) gilt für Nr: Nr MUSS kleiner gleich Ne der zugehörigen Kommando-APDU sein. Falls die zugehörige Kommando-APDU
   1. kein LeFeld besitzt, dann MUSS Nr = 0 sein.
   2. ein LeFeld besitzt, gilt für LeFeld gleich
      1. ´00´= WildCardShort: Nr MUSS kleiner gleich ´100´ = 256 sein.
      2. ´0000´= WildCardExtended: Nr MUSS kleiner gleich ´10000´=65.536 sein.
      3. sonst: Nr MUSS kleiner gleich OS2I( LeFeld ) sein.

### Trailer

Der Trailer zeigt den Status nach Bearbeitung einer Kommandonachricht an. Tabelle 272 zeigt die in diesem Dokument definierten Trailer. Der Trailer enthält Statusangaben über die Bearbeitung des Kommandos. Dazu zählen auch Fehlerindikationen.

1. (N027.210) K\_COS   
   Der Trailer (siehe [ISO/IEC 7816-4#Tabelle 1]) besteht aus zwei Oktetten.

## Zulässige Kommando-Antwort-Paare

Wie in 11.5.5 und 11.5.6 dargestellt, sind zwei Bestandteile einer Kommando-APDU optional. Daraus ergeben sich vier Kombinationsmöglichkeiten, die im Folgenden beschrieben werden.

### Case 1 Kommando-Antwort-Paar

Im Fall einer Case 1 Kommando-APDU fehlen Datenfeld und LeFeld. Die Kommando-APDU enthält folgende Angaben:

Tabelle 11: Case 1 Kommando-APDU

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´XX´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 1 Kommando-APDU:

1. (N027.300) K\_externeWelt {K\_Karte}   
   Eine Case 1 Kommando-APDU MUSS aus vier Oktetten bestehen.
2. (N027.400) K\_externeWelt {K\_Karte}   
   Die vier Oktette einer Case 1 Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2.

Die zu einer Case 1 Kommando-APDU gehörende Antwort-APDU besteht nur aus dem Trailer.

Tabelle 12: Case 1 Antwort-APDU

|  |  |
| --- | --- |
| **Inhalt** | **Beschreibung** |
| Trailer | Statusbytes SW1 und SW2 |

### Case 2 Kommando-Antwort-Paar

Im Fall einer Case 2 Kommando-APDU fehlt das Datenfeld. Das LeFeld ist vorhanden. In Abhängigkeit vom Wert Ne, der im LeFeld transportiert wird, werden die Fälle „short“ und „extended“ unterschieden.

#### Case 2 Short Kommando

In diesem Fall enthält das Datenfeld der Antwortnachricht nie mehr als 256 Oktette. Die Kommando-APDU enthält im Fall einer Case 2 Short folgende Angaben:

Tabelle 13: Case 2 Short Kommando-APDU

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´XX´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |
| Le | ´XX´ | Ne aus der Menge {1, …, 255, WildCardShort} |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 2 Short Kommando-APDU:

1. (N027.500) K\_externeWelt {K\_Karte}   
   Eine Case 2 Short Kommando-APDU MUSS aus fünf Oktetten bestehen.
2. (N027.600) K\_externeWelt {K\_Karte}   
   Der Wert von Ne wird wie folgt in einem Oktett codiert, welches als LeFeld bezeichnet wird:
   1. Wenn Ne im Intervall [1, 255] liegt, dann MUSS diese ganze Zahl in einem Oktett codiert werden: LeFeld = I2OS( Ne, 1 ).
   2. Wenn Ne den Wert WildCardShort besitzt, dann MUSS das LeFeld gleich ´00´ sein.
3. (N027.700) K\_externeWelt {K\_Karte}   
   Die fünf Oktette einer Case 2 Short Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2   ||   LeFeld.

#### Case 2 Extended Kommando

In diesem Fall enthält das Datenfeld der Antwortnachricht möglicherweise mehr als 256 Oktette. Die Kommando-APDU enthält im Fall einer Case 2 Extended folgende Angaben:

Tabelle 14: Case 2 Extended Kommando-APDU

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´XX´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |
| Le | ´XXXX´ | Ne aus der Menge {256, …, 65535, WildCardExtended} |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 2 Extended Kommando-APDU:

1. (N027.800) K\_externeWelt {K\_Karte}   
   Eine Case 2 Extended Kommando-APDU MUSS aus sieben Oktetten bestehen.
2. (N027.900) K\_externeWelt {K\_Karte}   
   Der Wert von Ne wird wie folgt in zwei Oktetten codiert, welche das LeFeld bilden:
   1. Wenn Ne im Intervall [256, 65535] liegt, dann MUSS diese ganze Zahl in zwei Oktette codiert werden: LeFeld = I2OS( Ne, 2 ).
   2. Wenn Ne den Wert WildCardExtended besitzt, dann MUSS das LeFeld gleich ´0000´ sein.
3. (N028.000) K\_externeWelt {K\_Karte}   
   Die sieben Oktette einer Case 2 Extended Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2   ||   ´00´  ||   LeFeld.
4. Das Oktett ´00´ nach dem Parameter P2 lässt sich als Indikator für „extended length“ auffassen.

#### Case 2 Response

Die zu einer Case 2 Kommando-APDU gehörende Antwort-APDU besteht aus dem optionalen Datenfeld und aus dem Trailer.

Tabelle 15: Case 2 Antwort-APDU

|  |  |
| --- | --- |
| **Inhalt** | **Beschreibung** |
| Daten | Optionaler Bestandteil der Antwort-APDU. Falls vorhanden, dann gelten die Bestimmungen aus 11.6.1. |
| Trailer | Statusbytes SW1 und SW2 |

### Case 3 Kommando

Im Fall einer Case 3 Kommando-APDU fehlt das LeFeld. Das Datenfeld ist vorhanden. In Abhängigkeit von der Anzahl der Oktette im Datenfeld werden die Fälle „short“ und „extended“ unterschieden.

#### Case 3 Short Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht nie mehr als 255 Oktette. Die Kommando-APDU enthält im Fall einer Case 3 Short folgende Angaben:

Tabelle 16: Case 3 Short Kommando-APDU

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´XX´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |
| Data | ´XX…XX´ | Datenfeld mit beliebigen Oktetten, Anzahl Oktette aus der Menge [1, 255] |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 3 Short Kommando-APDU:

1. (N028.100) K\_externeWelt {K\_Karte}   
   Eine Case 3 Short Kommando-APDU MUSS aus fünf Oktetten plus den Oktetten aus dem Datenfeld bestehen.
2. (N028.200) K\_externeWelt {K\_Karte}   
   Die Anzahl Nc der Oktette des Datenfeldes MUSS in einem Oktett codiert werden, welches als LcFeld bezeichnet wird: LcFeld = I2OS( Nc, 1 ).
3. (N028.300) K\_externeWelt {K\_Karte}   
   Die fünf plus Nc Oktette einer Case 3 Short Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2   ||   LcFeld   ||   Datenfeld.

#### Case 3 Extended Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht mehr als 255 Oktette. Die Kommando-APDU enthält im Fall einer Case 3 Extended folgende Angaben:

Tabelle 17: Case 3 Extended Kommando-APDU

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´XX´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |
| Data | ´XX…XX´ | Datenfeld mit beliebigem Inhalt, Anzahl Oktette aus der Menge [256, 65535] |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 3 Extended Kommando-APDU:

1. (N028.400) K\_externeWelt {K\_Karte}   
   Eine Case 3 Extended Kommando-APDU MUSS aus sieben Oktetten plus den Oktetten aus dem Datenfeld bestehen.
2. (N028.500) K\_externeWelt {K\_Karte}   
   Die Anzahl Nc der Oktette des Datenfeldes MUSS in zwei Oktette codiert werden, welche als LcFeld bezeichnet werden: LcFeld = I2OS( Nc, 2 ).
3. (N028.600) K\_externeWelt {K\_Karte}   
   Die sieben plus Nc Oktette einer Case 3 Extended Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2   ||   ´00´  ||   LcFeld   ||   Datenfeld.
4. Das Oktett ´00´ nach dem Parameter P2 lässt sich als Indikator für „extended length“ auffassen.

#### Case 3 Response

Die zu einer Case 3 Kommando-APDU gehörende Antwort-APDU besteht nur aus dem Trailer.

Tabelle 18: Case 3 Antwort-APDU

|  |  |
| --- | --- |
| **Inhalt** | **Beschreibung** |
| Trailer | Statusbytes SW1 und SW2 |

### Case 4 Kommando

Eine Case 4 Kommando-APDU enthält alle optionalen Bestandteile. In Abhängigkeit von der Anzahl der Oktette im Datenfeld der Kommandonachricht und in Abhängigkeit vom Wert Ne, der im LeFeld transportiert wird, werden die Fälle „short“ und „extended“ unterschieden.

#### Case 4 Short Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht nie mehr als 255 Oktette und das Datenfeld der Antwortnachricht nie mehr als 256 Oktette. Die Kommando-APDU enthält im Fall einer Case 4 Short folgende Angaben:

Tabelle 19: Case 4 Short Kommando-APDU

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´XX´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |
| Data | ´XX…XX´ | Datenfeld mit beliebigen Oktetten, Anzahl Oktette aus der Menge [1, 255] |
| Le | ´XX´ | Ne aus der Menge {1, …, 255, WildCardShort} |
| Notation | | CmdApdu = Case4S( CLA, INS, P1, P2, Data, Ne ) |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 4 Short Kommando-APDU:

1. (N028.700) K\_externeWelt {K\_Karte}   
   Eine Case 4 Short Kommando-APDU MUSS aus sechs Oktetten plus den Oktetten aus dem Datenfeld bestehen.
2. (N028.800) K\_externeWelt {K\_Karte}   
   Die Anzahl Nc der Oktette des Datenfeldes MUSS in einem Oktett codiert werden, welches als LcFeld bezeichnet wird: LcFeld = I2OS( Nc, 1 ).
3. (N028.900) K\_externeWelt {K\_Karte}   
   Der Wert von Ne wird wie folgt in einem Oktett codiert, welches das LeFeld bildet:
   1. Wenn Ne im Intervall [1, 255] liegt, dann MUSS diese ganze Zahl als ein Oktett als LeFeld verwendet werden: LeFeld = I2OS( Ne, 1 ).
   2. Wenn Ne den Wert WildCardShort besitzt, dann MUSS das LeFeld gleich ´00´ sein.
4. (N029.000) K\_externeWelt {K\_Karte}   
   Die sechs plus Nc Oktette einer Case 4 Short Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2   ||   LcFeld   ||   Datenfeld   ||   LeFeld.

#### Case 4 Extended Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht mehr als 255 Oktette oder es wird erwartet, dass das Datenfeld der Antwortnachricht mehr als 256 Oktette enthält. Die Kommando-APDU enthält im Fall einer Case 4 Extended folgende Angaben:

Tabelle 20: Case 4 Extended Kommando-APDU

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´XX´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´XX´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | Erster Parameter |
| P2 | ´XX´ | Zweiter Parameter |
| Data | ´XX…XX´ | Datenfeld mit beliebigen Oktetten, Anzahl Oktette aus der Menge [1, 65535] |
| Le | ´XXXX´ | Ne aus der Menge {1, …, 65535, WildCardExtended} |
| Notation | | CmdApdu = Case4E( CLA, INS, P1, P2, Data, Ne ) |

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 4 Extended Kommando-APDU:

1. (N029.100) K\_externeWelt {K\_Karte}   
   Eine Case 4 Extended Kommando-APDU MUSS aus neun Oktetten plus den Oktetten aus dem Datenfeld bestehen.
2. (N029.200) K\_externeWelt {K\_Karte}   
   Die Anzahl Nc der Oktette des Datenfeldes MUSS in zwei Oktetten codiert werden, welche als LcFeld bezeichnet werden: LcFeld = I2OS( Nc, 2 ).
3. (N029.300) K\_externeWelt {K\_Karte}   
   Der Wert von Ne wird wie folgt in zwei Oktette codiert, welche das LeFeld bilden:
   1. Wenn Ne im Intervall [1, 65535] liegt, dann MUSS diese ganze Zahl in zwei Oktette codiert werden: LeFeld = I2OS( Ne, 2 ).
   2. Wenn Ne den Wert WildCardExtended besitzt, dann MUSS das LeFeld gleich ´0000´ sein.
4. (N029.400) K\_externeWelt {K\_Karte}   
   Die neun plus Nc Oktette einer Case 4 Extended Kommando-APDU MÜSSEN wie folgt konkateniert werden:   
   CLA   ||   INS   ||   P1   ||   P2   ||   ´00´  ||   LcFeld   ||   Datenfeld   ||   LeFeld.
5. Das Oktett ´00´ nach dem Parameter P2 lässt sich als Indikator für „extended length“ auffassen.

#### Case 4 Response

Die zu einer Case 4 Kommando-APDU gehörende Antwort-APDU besteht aus dem optionalen Datenfeld und aus dem Trailer.

Tabelle 21: Case 4 Antwort-APDU

|  |  |
| --- | --- |
| **Inhalt** | **Beschreibung** |
| Daten | Optionaler Bestandteil der Antwort-APDU. Falls vorhanden, dann gelten die Bestimmungen aus 11.6.1. |
| Trailer | Statusbytes SW1 und SW2 |

1. (N029.500) Diese Anforderung ist absichtlich leer.
2. (N029.600) Diese Anforderung ist absichtlich leer.
3. (N029.610) Diese Anforderung ist absichtlich leer.
4. (N029.700) Diese Anforderung ist absichtlich leer.
5. (N029.800) Diese Anforderung ist absichtlich leer.

## Command Chaining

Gemäß [ISO/IEC 7816-4#5.3.3] ist "Command Chaining" die Möglichkeit, auszudrücken, dass konsekutive Kommando-Antwort-Paare verkettet sind, also zusammen gehören. In diesem Sinne wird der Mechanismus genutzt, um die Teilschritte eines mehrschrittigen Ablaufes als zusammengehörig zu kennzeichnen. Während in [ISO/IEC 7816-4#5.3.3] einige Details offen gelassen wurden, gilt für dieses Dokument folgendes Verhalten:

1. (N029.870) K\_externeWelt {K\_Karte}   
   Command Chaining ist eine konsekutive Folge von Kommando-Antwort-Paaren mit folgenden Eigenschaften:
   1. Mit Ausnahme des Bits b5 im CLA-Byte MÜSSEN alle Kommando-APDU einer Chaining-Kette identische Werte für CLA, INS, P1 und P2 haben.
   2. Das Bit b5 im CLA-Byte der letzten Kommando-APDU einer Command-Chaining-Sequenz MUSS den Wert 0 und die Bits b5 aller anderen CLA-Bytes einer Command-Chaining-Sequenz MÜSSEN den Wert 1 haben.
   3. Die Aussagen in (N029.870)a und (N029.870)b MÜSSEN für Kommando-APDU an der Schnittstelle "Interface I/O" aus Abbildung 1 gelten.
2. (N029.874) K\_externeWelt {K\_Karte}   
   Die konsekutive Folge einer Command-Chaining-Sequenz SOLL an der Schnittstelle "Interface I/O" aus Abbildung 1 NICHT
   1. durch Kommandos unterbrochen werden, welche nicht zu dieser Sequenz gehören oder
   2. durch eine Deaktivierung unterbrochen werden.
3. (N029.876) K\_externeWelt {K\_Karte}   
   Die konsekutive Folge einer Command-Chaining-Sequenz DARF an der Schnittstelle "Interface I/O" aus Abbildung 1 NICHT fortgesetz werden, falls ein Kommando der Sequenz mit einem Fehler terminiert, das heißt, falls
   1. Load Application mit einem Trailer aus Tabelle 45 antwortet, oder
   2. General Authenticate mit einem Trailer aus Tabelle 174 antwortet.
4. (N029.878) K\_COS   
   Falls die Anforderung (N029.874)a von der externen Entität nicht eingehalten wird, mithin also die Command-Chaining-Sequenz durch ein Kommando unterbrochen wird, dann
   1. MUSS das COS das unterbrechende Kommando akzeptieren.
   2. KANN das COS ein Fortsetzen der unterbrochenen Sequenz akzeptieren.
   3. SOLL das COS ein Fortsetzen der unterbrochenen Sequenz ablehnen.
5. (N029.880) K\_COS   
   Falls die Anforderung (N029.876) von der externen Entität nicht eingehalten wird, mithin also die Command-Chaining-Sequenz trotz Fehlers im vorherigen Kommando fortgesetzt wird, dann
   1. SOLL das COS ein Fortsetzen der Sequenz ablehnen.
   2. KANN das COS ein Fortsetzen der Sequenz akzeptieren.
6. Es ist gewollt, dass gemäß (N029.870)a alle Kommandos einer Kette dieselbe Kanalnummer aufweisen und die Secure Messaging Bits im CLA-Byte identisch sind. In Verbindung mit (N029.874) folgt daraus, dass Command Chaining nicht durch Kommandos auf anderen logischen Kanälen unterbrechbar ist.

## Längenbeschränkung von APDU

Gemäß [ISO/IEC 7816-4] ist die Länge einer APDU auf rund 64 kByte beschränkt. Zumindest für Kommando-APDU entspricht es dem Stand der Technik, dass diese Grenze wegen der RAM-Speichergrößen heute verfügbarer Halbleiter bei weitem nicht ausgeschöpft wird. Typischerweise zeigt eine Smartcard im EF.ATR an, welche APDU-Längenbe­schränkung für diese Smartcard gilt. Dabei werden Werte getrennt nach Kommando- und Antwortnachricht einerseits und andererseits getrennt nach ungeschützter Übertragung und geschützter Übertragung angegeben.

Im folgenden werden Intervalle für die Länge einer Nachricht angegeben. Dabei bedeutet „ungeschützte Kommandonachricht“, dass im CLA‑Byte kein Secure Messaging angezeigt wird. In diesem Fall wird die korrespondierende Antwortnachricht ebenfalls ungeschützt übertragen. Im Falle einer „geschützten Kommandonachricht“ wird im CLA‑Byte Secure Messaging angezeigt und die korrespondierende Antwortnachricht wird in der Regel ebenfalls geschützt übertragen.

1. (N029.890) K\_COS   
   Für die Längenbeschränkung von Kommando- und Antwortnachrichten gilt: Falls die Option\_logische\_Kanäle
   1. unterstützt wird, dann MUSS das COS folgende Intervalle unterstützen:
      1. ungeschützte Kommandonachricht: [ 4, 1.033] Oktett
      2. ungeschützte Antwortnachricht [ 2, 32.770] Oktett
      3. geschützte Kommandonachricht: [16, 1.033] Oktett
      4. geschützte Antwortnachricht [16, 1.033] Oktett
   2. nicht unterstützt wird, dann MUSS das COS folgende Intervalle unterstützen:
      1. ungeschützte Kommandonachricht: [ 4, 2.057] Oktett
      2. ungeschützte Antwortnachricht [ 2, 32.770] Oktett
      3. geschützte Kommandonachricht: [16, 2.057] Oktett
      4. geschützte Antwortnachricht [16, 2.057] Oktett
   3. Das COS KANN weitere Werte für die Längen von Nachrichten
      1. unterstützen oder
      2. ablehnen.
2. (N029.892) K\_Anwendungsspezifikation {K\_Karte}   
   Längenangaben im EF.ATR:
   1. Die Anwendungsspezifikation MUSS ein EF.ATR so spezifizieren, dass
      1. es als Kindobjekt im Ordner *root* (siehe (N019.900)a) enthalten ist und
      2. Längenbeschränkung sowohl für Kommando- und Antwortnachrichten als auch für geschützte und ungeschützte Nachrichten enthält, dabei gilt:
         1. *limitCmdPlain* ist die maximale Anzahl von Oktetten in einer ungeschützt übertragenen Kommandonachricht.
         2. *limitCmdSecureMessaging* ist die maximale Anzahl von Oktetten in einer geschützt übertragenen Kommandonachricht.
         3. *limitRspPlain* ist die maximale Anzahl von Oktetten in einer ungeschützt übertragenen Antwortnachricht.
         4. *limitRspSecureMessaging* ist die maximale Anzahl von Oktetten in einer geschützt übertragenen Antwortnachricht.
   2. Die Angaben zu Längenbeschränkungen im EF.ATR MÜSSEN größer oder gleich zu den Längenbeschränkungen aus (N029.890) sein.

# Kanalkontext (normativ)

Während das Objektsystem gemäß Kapitel 9 Informationen bündelt, die persistent in der Smartcard zu speichern sind und in allen logischen Kanälen gleichermaßen zur Verfügung stehen, wird in diesem Kapitel auf die Information eingegangen, die lediglich zwischen Öffnen und Schließen eines logischen Kanals zur Verfügung steht und damit kanalspezifisch ist. Die kanalspezifischen Attribute werden im Objekt *channelContext* zusammengefasst.

Die im Folgenden genannten Attribute werden typischerweise einem flüchtig (im RAM) gespeicherten Security Environment (siehe 8.8) oder Sicherheitsstatus (siehe 8.9) zugerechnet, oder durch das in 14.9.9.1 beschriebene Kommando Manage Security Environment verändert.

## Attribute eines logischen Kanals

An der physikalischen Schnittstelle (siehe Abbildung 1) verhält sich das COS so, als gäbe es für jeden logischen Kanal ein Objekt *channelContext* mit den folgenden Attributen:

1. (N029.900) K\_COS   
   Folgende Attribute von *channelContext* sind dem Objektsystem zugeordnet: Der *channelContext* MUSS
   1. genau ein Attribut *currentFolder* enthalten, das auf ein Objekt vom Typ Ordner zeigt.
   2. genau ein Attribut *RND.ICC* enthalten, welches eine vom COS erzeugte Zufallszahl oder den Wert "NoRandom" speichert (siehe (N086.902){a.8, b.8} und (N099.300)).
   3. genau eine Liste *keyReferenceList* besitzen. Jedes Listenelement MUSS einen Wert unterstützen, der anzeigt, dass es leer ist. Die Liste *keyReferenceList* MUSS sich aus folgenden Elementen zusammensetzen:
      1. *externalAuthenticate*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
      2. *internalAuthenticate*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
      3. *verifyCertificate*, mit der Komponente *keyReference*.
      4. *signatureCreation*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
      5. *dataDecipher*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
      6. *dataEncipher* mit den Komponenten *keyReference* und *algorithmIdentifier*.
      7. *macCalculation* mit den Komponenten *keyReference* und *algorithmIdentifier*.
   4. genau ein Attribut *SessionkeyContext* besitzen, welches folgende Elemente enthält:
      1. *flagSessionEnabled* für welches folgender Wertebereich definiert ist:
         1. noSK zeigt an, dass keine Sessionkeys vorhanden sind.
         2. SK4SM zeigt an, dass die übrigen Attribute gebrauchsfertiges kryptographisches Material enthalten, welches im Layer SecMes (siehe Abbildung 1) zum Entsichern einer Kommando-APDU oder zum Sichern einer Response-APDU verwendbar ist.
         3. SK4TC zeigt an, dass die übrigen Attribute gebrauchsfertiges kryptographisches Material enthalten, welches im Layer CmdInterpreter (siehe Abbildung 1) verwendbar ist (beispielsweise im Rahmen von PSO-Kommandos).
      2. *Kenc* ist ein symmetrischer Schlüssel zur Ver- und Entschlüsselung.
      3. *SSCenc* ist eine nicht-negative, ganze Zahl, die als Send Sequence Counter im Zusammenhang mit *Kenc* verwendet wird.
      4. *Kmac* ist ein symmetrischer Schlüssel zur MAC-Berechnung und MAC-Verifikation.
      5. *SSCmac* ist eine nicht-negative, ganze Zahl, die als Send Sequence Counter im Zusammenhang mit *Kmac* verwendet wird.
      6. *flagCmdEnc* ist eine boolesche Variable, welche anzeigt, ob die gesicherte Kommando-APDU ein Datenobjekt mit verschlüsselten Kommandodaten enthält. Dieses Flag wird in (N031.700)a.1 mit einem Wert versehen und in (N022.600)b ausgewertet.
      7. *flagRspEnc* ist eine boolesche Variable, welche anzeigt, ob die Daten einer Antwort-APDU verschlüsselt übertragen werden. Das Flag wird in (N022.600)d mit einem Wert versehen und in (N033.700) bzw. (N033.800) ausgewertet.
      8. *negotiationKeyInformation* ist eine Variable, welche Informationen zum Schlüssel enthält, der an der Etablierung der Sessionkeys beteiligt war. Dabei sind folgende Fälle zu unterscheiden: *negotiationKeyInformation* enthält
         1. eine Schlüsselreferenz gemäß (N099.600) auf das beteiligte Authentisierunsobjekt, wenn dieses folgenden Typ besitzt:
            1. symmetrischem Authentisierungsobjekt (siehe 8.6.1).
            2. symmetrisches Kartenverbindungsobjekt (siehe 8.6.2).
         2. *accessRight* gemäß (N019.700) des beteiligten Schlüsselobjektes, wenn dieses folgenden Typ besitzt: öffentliches Authentisierungsobjekt (siehe 8.6.4.2).
      9. Option\_Kryptobox
         1. *accessRulesSessionkeys* ist eine Variable vom Typ *interfaceDependentAccessRules* (siehe 8.1.4), welche die Zugriffsregeln enthält, die im Rahmen einer Trusted-Channel-Unterstützung ausgewertet werden, siehe (N087.244), (N090.200), (N091.650)b und (N096.364).
         2. *folderSessionkeys* ist eine Variable, die angibt, welchem Ordner die Sessionkeys zugeordnet sind.
   5. genau eine Liste *globalSecurityList* besitzen.
      1. Das COS MUSS alle Werte des Intervalls [0, 3] für die Länge dieser Liste unterstützen.
      2. Ein COS KANN längere Listen unterstützen.
   6. genau eine Liste *dfSpecificSecurityList* besitzen.
      1. Das COS MUSS alle Werte des Intervalls [0, 3] für die Länge dieser Liste unterstützen.
      2. Ein COS KANN längere Listen unterstützen.
   7. Jedes Element der Liste *globalSecurityList* und jedes Element der Liste *dfSpecificSecurityList* MUSS entweder
      1. Option\_RSA\_CVC, ein *CHA* gemäß (N019.700)a sein in Verbindung mit einer Referenz zu einem Ordner, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß (N084.400)b oder (N084.400)c stattgefunden hat mit einem Schlüsselobjekt aus vorgenanntem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d), oder
      2. eine Referenz auf ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) sein, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß (N084.402){a, b} oder (N084.410){a, b} stattgefunden hat.
      3. eine Referenz auf ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) sein, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß 15.4.2 stattgefunden hat.
   8. genau eine Liste *bitSecurityList* besitzen.
      1. Das COS MUSS alle Werte des Intervalls [0, 1] für die Länge dieser Liste unterstützen.
      2. Ein COS KANN längere Listen unterstützen.
      3. Jedes Element der Liste *bitSecurityList* MUSS ein CHAT sein in Verbindung mit einer Referenz zu einem Ordner, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß (N084.400)a, (N085.054) oder (N085.056) stattgefunden hat mit einem Schlüsselobjekt aus vorgenanntem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d.7)
   9. genau eine Liste *globalPasswordList* besitzen.
      1. Das COS MUSS alle Werte des Intervalls [0, 4] für die Länge dieser Liste unterstützen.
      2. Ein COS KANN längere Listen unterstützen.
   10. genau eine Liste *dfSpecificPasswordList* besitzen.
       1. Das COS MUSS alle Werte des Intervalls [0, 4] für die Länge dieser Liste unterstützen.
       2. Ein COS KANN längere Listen unterstützen.
   11. Jedes Element der Liste *globalPasswordList* und jedes Element der Liste *dfSpecificPasswordList* MUSS genau ein Attribut *securityStatusEvaluationCounter* sein in Verbindung mit einer Referenz auf ein Passwortobjekt, wodurch angezeigt wird, dass eine erfolgreiche Benutzerverifikation gemäß 14.6.6.1 mit diesem Passwortobjekt stattgefunden hat.
   12. Der Wertebereich von *securityStatusEvaluationCounter*
       1. MUSS alle Werte von *startSsec* (siehe (N015.800)c.2) umfassen und
       2. KANN den Wert null beinhalten.
   13. genau ein Attribut *currentEF* enthalten, das
       1. entweder unbestimmt ist,
       2. oder auf ein Listenelement von *currentFolder.children* vom Typ Datei zeigt.
2. (N030.000) K\_COS   
   Folgende Attribute von *channelContext* sind einem Ordner zugeordnet: Der *channelContext* MUSS für jeden Ordner im Objektsystem
   1. genau ein Attribut *seIdentifier* gemäß (N007.900) enthalten.
3. (N030.010) K\_COS, Option\_logische\_Kanäle   
   Das COS MUSS mindestens vier logische Kanäle unterstützen. Das heißt neben dem Basiskanal sind mindestens drei weitere logische Kanäle zu unterstützen.

## Reset-Verhalten

Ein „neuer“ Kanalkontext gemäß 12.1 wird dann etabliert, wenn ein Reset durchgeführt wird, oder ein neuer logischer Kanal geöffnet wird. Dann gilt:

1. (N030.100) K\_COS   
   Wenn der logische Kanal mit der Kanalnummer null (Basiskanal) durch
   1. Einschalten des physikalischen Interfaces, d. h.
      1. für das Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] wird eine Aktivierung gemäß (N023.920)a durchgeführt, oder
      2. für das USB-Übertragungsprotokoll gemäß 11.2.2 wird eine Aktivierung gemäß [ISO/IEC 7816-12] durchgeführt, oder
      3. für die kontaktlose Datenübertragung gemäß 11.2.3 wird eine Aktivierung gemäß ISO/IEC 14443 durchgeführt oder
   2. durch einen Warm-Reset gemäß (N023.920)c

geöffnet wird oder ein anderer logischer Kanal durch das Kommando Manage Channel (siehe 14.9.8.1) geöffnet wird, oder ein logischer Kanal zurückgesetzt wird (siehe 14.9.8.3 und 14.9.8.4) dann MUSS für den betroffenen Kanal gelten:

* 1. Dem betroffenen logischen Kanal MUSS exklusiv ein Kanalkontext gemäß 12.1 zugeordnet werden.
  2. Das Attribut *currentFolder* MUSS auf *root* gesetzt werden.
  3. Das Attribut *RND.ICC* MUSS auf den Wert "NoRandom" gesetzt werden.
  4. Das Attribut *keyReferenceList* MUSS so gesetzt werden, dass alle Elemente leer sind.
  5. Das Attribut *SessionkeyContext.flagSessionEnabled* MUSS auf noSK gesetzt werden.
  6. Die Liste *globalSecurityList* MUSS leer sein.
  7. Die Liste *dfSpecificSecurityList* MUSS leer sein.
  8. Das Attribut *bitSecurityList* MUSS leer sein.
  9. Die Liste *globalPasswordList* MUSS leer sein.
  10. Die Liste *dfSpecificPasswordList* MUSS leer sein.
  11. Für alle Ordner gilt:
      1. *seIdentifier* MUSS auf den Wert eins gesetzt werden.
  12. *Das Attribut currentEF* MUSS auf den Wert „unbestimmt“ gesetzt werden.

1. In (N030.100) sind absichtlich keine Zustände oder Zustandsübergänge aus ISO/IEC 14443 aufgeführt. Daraus folgt, dass mit Ausnahme des Zustandsübergangs nach „POWER-OFF“ kein Kanalkontext durch Zustandsübergänge beeinflusst wird.

## Setzen eines Sicherheitsstatus

Die hier beschriebene Routine setzt den Sicherheitsstatus des als Parameter übergebenen Authentisierungsschlüssels.

|  |  |  |
| --- | --- | --- |
| Input: | *obj* | Ein Schlüsselobjekt |
| Output: | – | Kein Rückgabewert |
| Notation: |  | setSecurityStatus( *obj* ) |

1. (N030.200) K\_COS   
   Falls *obj*
   1. im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalSecurityList* gelten (siehe (N029.900)e).
   2. in einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificSecurityList* gelten (siehe (N029.900)f).
2. (N030.300) K\_COS   
   Falls *obj*
   1. ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) und *obj* in der Liste *tmpList*
      1. bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
      2. noch nicht vorhanden ist, dann MUSS *obj* am Anfang von *tmpList* eingetragen werden.
   2. Option\_RSA\_CVC, ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList*
      1. bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
      2. noch nicht vorhanden ist, dann MUSS *obj.CHA* am Anfang von *tmpList* eingetragen werden zusammen mit einem Verweis auf den Ordner, der auch *obj* enthält. Dadurch wird der Sicherheitsstatus von *obj.CHA* demselben Ordner zugeordnet, der auch *obj* enthält.
   3. Falls *tmpList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement (hier als *objLast* bezeichnet) mittels clearSecurityStatusKey( *objLast* ) entfernen (FIFO = first in first out).
3. (N030.400) K\_COS   
   Falls *obj* ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein ELC-Schlüssel ist und *obj*.CHAT (siehe (N019.700)b.1) in der Liste *bitSecurityList* (siehe (N029.900)h)
   1. bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
   2. noch nicht vorhanden ist, dann MUSS *obj*.CHAT am Anfang von *bitSecurityList* eingetragen werden zusammen mit einem Verweis auf den Ordner, der auch *obj* enthält. Dadurch wird der Sicherheitsstatus von *obj.*CHAT demselben Ordner zugeordnet, der auch *obj* enthält.
   3. Falls *bitSecurityList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement (hier als *objLast* bezeichnet) mittels clearSecurityStatusKey( *objLast* ) entfernen (FIFO = first in first out).

## Löschen eines Sicherheitsstatus

### Löschen des Sicherheitszustandes eines Schlüssels

Die hier beschriebene Routine löscht den Sicherheitsstatus des als Parameter übergebenen Authentisierungsschlüssels.

|  |  |  |
| --- | --- | --- |
| Input: | *obj* | Ein Schlüsselobjekt |
| Output: | – | Kein Rückgabewert |
| Notation: |  | clearSecurityStatusKey( *obj* ) |

1. (N030.500) K\_COS   
   Falls *obj*
   1. im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalSecurityList* gelten (siehe (N029.900)e).
   2. in einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificSecurityList* gelten (siehe (N029.900)f).
2. (N030.600) K\_COS   
   Falls *obj*
   1. ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) ist und *obj* in der Liste *tmpList*
      1. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
      2. noch vorhanden ist, dann MUSS *obj* aus *tmpList* entfernt werden.
   2. Option\_RSA\_CVC, ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein
      1. RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList*
         1. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
         2. noch vorhanden ist, dann MUSS *obj.CHA* aus *tmpList* entfernt werden.
      2. ELC-Schlüssel ist und *obj*.CHAT (siehe (N019.700)b.1) in der Liste *bitSecurityList* (siehe (N029.900)h).
         1. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
         2. noch vorhanden ist, dann MUSS *obj*.CHAT aus *bitSecurityList* entfernt werden.
   3. einen anderen Typ besitzt, dann MUSS dieser Algorithmus beendet werden.
3. (N030.700) K\_COS   
   Falls der Sicherheitszustand eines Schlüssels gelöscht wurde, der an der Aushandlung von Sessionkeys beteiligt war, dann MUSS *SessionkeyContext.flagSessionEnabled* (siehe (N029.900)d.1) auf den Wert noSK gesetzt werden.

### Löschen der Sicherheitszustände eines Ordners

Die hier beschriebene Routine löscht die Sicherheitszustände aller Schlüssel, welche dem als Parameter übergebenen Ordner zugeordnet sind.

|  |  |  |
| --- | --- | --- |
| Input: | *obj* | Ein Ordner (siehe 8.3.1) |
| Output: | – | Kein Rückgabewert |
| Notation: |  | clearSecurityStatusFolder( *obj* ) |

1. (N030.720) K\_COS   
   Die dem Ordner *obj* zugeordneten Sicherheitszustände von Schlüsseln sind wie folgt zu löschen: Jedes Element in *globalSecurityList* (siehe (N029.900)e) und in *dfSpecificSecurityList* (siehe (N029.900)f) und in *bitSecurityList* (siehe (N029.900)h) MUSS mittels clearSecurityStatusKey(…) aus den genannten Listen entfernt werden, falls es *obj* zugeordnet ist.

### Löschen von Sessionkeys

Die hier beschriebene Routine löscht den Sicherheitsstatus, welcher bei der Aushandlung von Sessionkeys gesetzt wurde.

|  |  |  |
| --- | --- | --- |
| Input: | – | Kein Parameter |
| Output: | – | Kein Rückgabewert |
| Notation: |  | clearSessionkeys( ) |

1. (N030.740) K\_COS   
   Der Sicherheitszustand, welcher bei der Aushandlung von Sessionkeys gesetzt wurde, ist wie folgt zu löschen:
   1. Das Elementen in *globalSecurityList* (siehe (N029.900)e) oder *dfSpecificSecurityList* (siehe (N029.900)f) oder *bitSecurityList* (siehe (N029.900)h) MUSS aus den genannten Listen entfernt werden, welches an der Aushandlung der Sessionkeys beteiligt war.
   2. *SessionkeyContext.flagSessionEnabled* (siehe (N029.900)d.1) MUSS auf den Wert noSK gesetzt werden.

## Setzen eines Passwortstatus

Die hier beschriebene Routine setzt den Sicherheitsstatus des als Parameter übergebenen Passwortes.

|  |  |  |
| --- | --- | --- |
| Input: | *obj* | Ein Passowrtobjekt |
| Output: | – | Kein Rückgabewert |
| Notation: |  | setPasswordStatus( *obj* ) |

1. (N030.800) K\_COS   
   Falls *obj*
   1. im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalPasswordList* gelten (siehe (N029.900)i).
   2. einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificPasswordList* gelten (siehe (N029.900)j).
2. (N030.900) K\_COS   
   Falls *obj* in der Liste *tmpList*
   1. bereits vorhanden ist, dann MUSS zunächst clearPasswordStatus(*obj*) und dann (N030.900)b ausgeführt werden.
   2. noch nicht vorhanden ist, dann MUSS *obj* am Anfang von *tmpList* eingetragen werden. Dabei MUSS das Attribut *securityStatusEvaluationCounter* des neuen Listenelementes in *tmpList* (siehe (N029.900)k) auf den Wert *startSsec* (siehe (N015.800)c.2) gesetzt werden, der aus *obj.startSsecList* (siehe (N015.800)) unter Berücksichtigung von *seIdentifier* aus (N015.800)c.1 und *seIdentifier* aus (N030.000)a ermittelt wird. Dabei MUSS in (N030.000) der Ordner zu Grunde gelegt werden, der *obj* enthält.
3. (N031.000) K\_COS   
   Falls *tmpList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement entfernen (FIFO = first in first out).
4. (N031.100) K\_COS   
   Im Eintrag zu *obj* in *tmpList* MUSS das Attribut *securityStatusEvaluationCounter* auf den Wert *obj.startSsec* gesetzt werden.

## Löschen eines Passwortstatus

Die hier beschriebene Routine löscht den Sicherheitsstatus des als Parameter übergebenen Passwortes.

|  |  |  |
| --- | --- | --- |
| Input: | *obj* | Ein Passwortobjekt |
| Output: | – | Kein Rückgabewert |
| Notation: |  | clearPasswordStatus( *obj* ) |

1. (N031.200) K\_COS   
   Falls *obj*
   1. im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalPasswordList* gelten (siehe (N029.900)i).
   2. in einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificPasswordList* gelten (siehe (N029.900)j).
2. (N031.300) K\_COS   
   Falls *obj* in der Liste *tmpList*
   1. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
   2. noch vorhanden ist, dann MUSS *obj* aus *tmpList* entfernt werden.

# Gesicherte Kommunikation (normativ)

## Secure Messaging Layer

Dieses Unterkapitel beschreibt die Funktionsweise des Layers „SecMes“ in Abbildung 1. Dieser Layer benutzt neben den Informationen aus (N029.900)d folgende weitere Attribute:

* KD.i ist ein Oktettstring, der eine vom COS generierte Zufallszahl speichert, die im Rahmen der Ableitung von Sessionkeys verwendet wird.
* KD.e ist ein Oktettstring, der eine extern generierte Zufallszahl speichert, die im Rahmen der Ableitung von Sessionkeys verwendet wird.

### Ableitung von Sessionkeys

|  |  |  |
| --- | --- | --- |
| Input: | – | Der benötigte Input wird dem *channelContext* entnommen |
| Output: | – | Kein Output vorhanden |
| Errors: | – | Keine |
| Notation: |  | SessionkeyDerivation() |

Im Folgenden gelten die Definitionen:

|  |  |  |
| --- | --- | --- |
| *algId* | = | *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier*  oder  *channelContext.keyReferenceList.internalAuthenticate.algorithmIdentifier* |
| *PrK* | = | an der Authentisierung beteiligter privater Schlüssel, referenziert in  *channelContext.keyReferenceList.internalAuthenticate.keyReference* |
| *PuK* | = | an der Authentisierung beteiligter öffentlicher Schlüssel |
| *SK* | = | an der Authentisierung beteiligter symmetrischer Schlüssel, referenziert in *channelContext.keyReferenceList.externalAuthenticate.keyReference* |

1. Bei der Aushandlung von Sessionkeys sind zwei Fälle zu unterscheiden: Entweder die Sessionkeyaushandlung erfolgte mittels
   1. symmetrischer Schlüssel, dann ist aus den obigen Definitionen SK relevant und PrK und PuK sind irrelevant, oder
   2. die Sessionkeyaushandlung erfolgte mittels asymmetrischer Schlüssel, dann ist aus den obigen Definitionen SK irrelevant und PrK und PuK sind relevant.
2. (N031.390) K\_COS   
   Das Attribut *SessionkeyContext*.*negotiationKeyInformation* MUSS wie folgt gesetzt werden: Falls die Sessionkeys
   1. asymmetrisch ausgehandelt wurden, dann gilt:   
        *SessionkeyContext*.*negotiationKeyInformation* = *PuK*.*accessRights*.
   2. symmetrisch ausgehandelt wurden, gilt:   
        *SessionkeyContext*.*negotiationKeyInformation* =   
        *channelContext.keyReferenceList.externalAuthenticate.keyReference*.
3. (N031.400) K\_COS\_G1, Option\_DES   
   Falls der an der Aushandlung beteiligte Aushandlungsschlüssel die Aushandlung von 3TDES-Schlüsseln impliziert, dann MUSS für die Attribute aus *SessionkeyContext* gelten:
   1. (*Kenc*, *SSCenc*, *Kmac*, *SSCmac*) = KeyDerivation\_3TDES(KD.i   XOR   KD.e).
   2. Für *algIDSessionkey* in (N031.522) MUSS dann *algIDSessionkey* = desSessionkey gelten.
4. (N031.500) K\_COS   
   Falls der an der Aushandlung beteiligte Aushandlungsschlüssel die Aushandlung von AES-Schlüsseln impliziert, dann MUSS für die Attribute aus *SessionkeyContext* gelten: Falls die Sessionkeys
   1. symmetrisch ausgehandelt wurden und *SK* ein
      1. AES-128 Schlüssel ist:   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES128(KD.i XOR KD.e).
      2. AES-192 Schlüssel ist:   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES192(KD.i XOR KD.e).
      3. AES‑256 Schüssel ist:   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES256(KD.i XOR KD.e).
   2. asymmetrisch ausgehandelt wurden und *PrK.domainParameter* gleich
      1. brainpoolP256r1 ist:   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES128(KD.i XOR KD.e).
      2. brainpoolP384r1 ist:   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES192(KD.i XOR KD.e).
      3. brainpoolP512r1 ist:   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES256(KD.i XOR KD.e).
   3. mittels PACE ausgehandelt wurden und *channelContext.keyReferenceList. externalAuthenticate.algID* ist Element der Menge
      1. {id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128}   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES128(KD.i XOR KD.e).
      2. {id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192}   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES192(KD.i XOR KD.e).
      3. {id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256}   
         (*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation\_AES256(KD.i XOR KD.e).
   4. Für *algIDSessionkey* in (N031.522) MUSS dann *algIDSessionkey* = aesSessionkey gelten.
5. (N031.520) K\_COS   
   Falls *algId* anzeigt, dass die Sessionkeys im Rahmen von Secure Messaging einzusetzen sind, dann MUSS gesetzt werden
   1. *SessionkeyContext.flagSessionEnabled* = SK4SM und
   2. *SessionkeyContext*.*accessRulesSessionkeys* auf einen beliebigen Wert, weil die Zugriffsregeln der Sessionkeys im Rahmen dieser Spezifikation nicht ausgewertet werden (vergleiche auch (N031.522)b).
6. (N031.522) K\_COS, Option\_Kryptobox, Option\_PACE\_PCD   
   Falls *algId* anzeigt, dass die Sessionkeys den Betrieb eines Trusted Channels zu unterstützen haben, dann MÜSSEN mit *algIDSessionkey* aus (N031.400)b bzw. (N031.500)d folgende Änderungen am *channelContext* vorgenommen werden:
   1. setze *SessionkeyContext.flagSessionEnabled* = SK4TC und
   2. setze *SessionkeyContext*.*accessRulesSessionkeys* gleich
      1. *SK*.*accessRulesSessionkeys* falls die Sessionkeys symmetrisch vereinbart wurden, oder
      2. *PrK*.*accessRulesSessionkeys*, falls die Sessionkeys asymmetrisch vereinbart wurden.
   3. setze *SessionkeyContext*.*folderSessionkeys* so, dass dadurch die Sessionkeys demselben Ordner zugeordnet werden wie *SK* bzw. *PrK*.
   4. trage in *channelContext.keyReferenceList.dataDecipher* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein und
   5. trage in *channelContext.keyReferenceList.dataEncipher* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein und
   6. trage in *channelContext.keyReferenceList.macCalculation* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein.

### Bearbeitung einer Kommando-APDU

Dieses Kapitel beschreibt die generische Bearbeitung eines Kommandos. Hier, wie auch in Kapitel 14 werden gelegentlich kanalspezifische Angaben benötigt, wie etwa *currentFolder*. Auf eine spezielle Kennzeichnung, dass damit der Wert aus dem Kanalkontext *channelContext* des logischen Kanals zu nehmen ist, der im CLA-Byte angezeigt wird, wird der Übersichtlichkeit halber verzichtet.

1. (N031.600) K\_COS   
   Falls im CLA-Byte kein Secure Messaging angezeigt wird (siehe [ISO/IEC 7816-4#5.4.1]) und *SessionkeyContext.flagSessionEnabled* den Wert
   1. SK4SM besitzt, dann
      1. MUSS der Sicherheitszustand des zugehörigen Aushandlungsschlüssels mittels clearSessionkeys( ) zurückgesetzt werden (siehe 12.4.3).
      2. MUSS CmdApdu3 = CmdApdu2 gesetzt werden.
   2. sonst, dann MUSS CmdApdu3 = CmdApdu2 gesetzt werden.
2. (N031.700) K\_COS   
   Falls im CLA-Byte Secure Messaging angezeigt wird und *SessionkeyContext.flagSessionEnabled* besitzt den Wert
   1. SK4SM, dann MUSS
      1. die gesicherte CmdApdu2 den normativen Vorgaben aus 13.2 entsprechen. Falls ein DO mit Tag = ´87´ in der CmdApdu2 vorhanden ist, MUSS *flagCmdEnc* auf True gesetzt werden, sonst auf False. Das COS KANN weitere Secure Messaging Formate
         1. akzeptieren oder
         2. ablehnen.
      2. CmdApdu3 mittels der Umkehroperation zu den Regeln in 13.2 aus CmdApdu2 berechnet werden. Falls dabei ein Fehler festgestellt wird, genau dann
         1. MUSS der Sicherheitszustand des zugehörigen Aushandlungsschlüssels mittels clearSessionkeys( ) zurückgesetzt werden (siehe 12.4.3).
         2. MUSS die Bearbeitung des Kommandos terminieren.
         3. RspApdu2 enthält in diesem Fall keine Antwortdaten und besteht lediglich aus dem Trailer IncorrectSmDo = ´69 88´.
   2. andernfalls
      1. MUSS die Bearbeitung des Kommandos terminieren.
      2. RspApdu2 DARF in diesem Fall NICHT mehr als den Trailer IncorrectSmDo = ´69 88´ enthalten. Das bedeutet, dass RspApdu2 in diesem Fall keine Antwortdaten enthält.
3. (N031.800) K\_COS   
   Die vom Secure Messaging Layer („SecMes“ in Abbildung 1) gegebenenfalls weitergeleitete CmdApdu3 MUSS gemäß den Vorgaben aus Kapitel 14 bearbeitet werden, wobei eine Antwortnachricht-RspApdu1 entsteht.
4. In Kapitel 14 wird davon ausgegangen, dass der Secure Messaging Layer CmdApdu3 so aufbereitet, dass im CLA-Byte kein Secure Messaging angezeigt wird und die Kanalnummer auf null gesetzt wird.
5. (N031.900) K\_COS   
   Falls *SessionkeyContext.flagSessionEnabled* den Wert
   1. SK4SM besitzt, dann MUSS die ungesicherte RspApdu1 gemäß 13.3 in die gesicherte RspApdu2 umgewandelt werden.
   2. sonst MUSS RspApdu2 = RspApdu1 gesetzt werden.
6. (N031.920) K\_COS   
   Falls sowohl KD.i, als auch KD.e vorhanden sind (einer oder beide Werte wurden mit dem gerade bearbeiteten Kommando gesetzt), dann
   1. MUSS *SessionkeyContext* mittels SessionkeyDerivation() geändert werden (siehe 13.1.1) und anschließend
   2. MÜSSEN KD.i und KD.e auf den Wert „nicht vorhanden“ gesetzt werden.
7. (N031.940) K\_COS   
   Während der Kommandobearbeitung ist es möglich, dass Fälle eintreten, die eine weitere Bearbeitung von Kommandos verhindern. Dieser Zustand ist wie folgt gekennzeichnet: Die Kommando-APDU CmdApdu1 in Abbildung 1 wird nicht mit einer korrespondierenden Antwort-APDU RspApdu3 beantwortet. Gemäß den in Kapitel 11 beschriebenen Übertragungsprotokollen ist es dann nicht möglich, eine weitere Kommando-APDU zu schicken.
   1. In diesem Zustand KÖNNEN WTX-Requests über „Interface physical“ in Abbildung 1 gesendet werden, oder
   2. In diesem Zustand KANN ein Senden von WTX-Requests über „Interface physical“ in Abbildung 1 unterbleiben.
   3. Das COS MUSS diesen Zustand durch einen Reset verlassen.
8. Falls zum Zeitpunkt des Resets eine Transaktion offen ist (siehe 14.1), dann ist es möglich, dass im Rahmen eines Roll-Forward oder eines Roll-Back Fälle eintreten, die eine weitere Bearbeitung von Kommandos verhindern.

## Sicherung einer Kommando-APDU

Dieses Unterkapitel beschreibt, wie eine gemäß 11.5 strukturierte Kommando-APDU zu sichern ist. Die hier beschriebenen Regeln richten sich an die Instanz, welche Kommando-APDU sendet. Das COS führt die entsprechenden Umkehroperationen durch.

Generell wird hier ein Subset der Regeln aus [ISO/IEC 7816-4#10] beschrieben, wobei

* alle Kommando-APDU mit Integritätsschutz übertragen werden,
* der Kommandoheader stets mit Integritätsschutz übertragen wird,
* Kommandodaten als Klartext oder verschlüsselt übertragen werden.

Hier sei angemerkt, dass der Sender zwar die Wahl hat zwischen der Übertragung von Kommandodaten im Klartext oder als Chiffrat, aber die Zugriffsregel gegebenenfalls eine verschlüsselte Übertragung erzwingt.

Wie in 11.5 beschrieben, besteht eine Kommando-APDU generisch betrachtet aus den Oktetten CLA, INS, P1 und P2 sowie aus dem optionalen Datenfeld Data und dem optionalen LeFeld. Somit gelten folgende Definitionen

|  |  |  |
| --- | --- | --- |
| Input: | CLA | Oktett gemäß (N026.500) |
| INS | Oktett gemäß (N026.600) |
| P1 | Oktett gemäß (N026.700) |
| P2 | Oktett gemäß (N026.800) |
| Data | Optionaler Oktettstring gemäß 11.5.5 |
| LeFeld | Optionaler Oktettstring gemäß 11.5.6, welcher gemäß (N027.600) oder (N027.900) eine Zahl Ne enthält |
| *Kenc* | Symmetrischer Schlüssel für die Verschlüsselung, (N029.900)d.2 |
| *SSCenc* | Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der Verschlüsselung verwendet wird, (N029.900)d.3 |
| *Kmac* | Symmetrischer Schlüssel für die MAC-Berechnung, (N029.900)d.4 |
| *SSCmac* | Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der MAC-Berechnung verwendet wird, (N029.900)d.5 |
| Output: | CmdApdu | Gesicherte Kommando-APDU |
| Errors: | – | Keine |

Zur Sicherung einer generischen Kommando-APDU sind folgende Schritte durchzuführen:

1. (N032.000) K\_externeWelt {K\_Karte}   
   Falls das optionale Datenfeld Data fehlt, MUSS gelten:   
   ProtectedData = ´´ (leerer Oktettstring).
2. (N032.100) K\_externeWelt {K\_Karte}   
   Falls das optionale Datenfeld Data vorhanden ist und im Klartext übertragen wird, MUSS gelten:
   1. Setze lenPDO = OctetLength( Data ), falls lenPDO im Intervall
      1. [1, 127] liegt, gilt: LenP = I2OS(lenPDO, 1)
      2. [128, 255] liegt, gilt: LenP = ´81´   ||   I2OS(lenPDO, 1)
      3. [256, 65535] liegt, gilt: LenP = ´82´   ||   I2OS(lenPDO, 2)
   2. Setze ProtectedData = ´81   ||   LenP   ||   Data´
3. (N032.190) K\_externeWelt {K\_Karte}   
   *SSCmac* MUSS inkrementiert werden, das heißt:   *SSCmac* = *SSCmac* + 1.
4. (N032.200) K\_externeWelt {K\_Karte}, Option\_DES   
   Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird und *Kenc* ein 3TDES-Schlüssel ist, MUSS gelten:
   1. *SSCenc* = *SSCenc* + 1
   2. C = 3TDES\_CBC\_ENC(*Kenc*, *SSCenc*, PaddingIso( Data, 8 ) )
   3. Setze lenCDO = OctetLength( C ) + 1, falls lenCDO im Intervall
      1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)
      2. [128, 255] liegt, gilt: LenC = ´81´   ||   I2OS(lenCDO, 1)
      3. [256, 65535] liegt, gilt LenC = ´82´   ||   I2OS(lenCDO, 2)
   4. Setze ProtectedData = ´87   ||   LenC   ||   01   ||   C´
5. (N032.300) K\_externeWelt {K\_Karte}   
   Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird und *Kenc* ein AES-Schlüssel ist, MUSS gelten:
   1. IVenc   = OS2I(AES\_ENC( *Kenc*, I2OS(SSCmac, 16)))
   2. C   = AES\_CBC\_ENC( *Kenc*, IVenc, PaddingIso( Data, 16 ) )
   3. Setze lenCDO = OctetLength( C ) + 1, falls lenCDO im Intervall
      1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)
      2. [128, 255] liegt, gilt: LenC = ´81´   ||   I2OS(lenCDO, 1)
      3. [256, 65535] liegt, gilt LenC = ´82´   ||   I2OS(lenCDO, 2)
   4. Setze ProtectedData = ´87   ||   LenC   ||   01   ||   C´
6. (N032.400) K\_externeWelt {K\_Karte}   
   Für LeDO MUSS gelten: Falls das optionale LeFeld
   1. fehlt, gilt: LeDO = ´´, (leerer Oktettstring).
   2. vorhanden ist, gilt:: LeDO = ´97 || I2OS(OctetLength( LeFeld ), 1)   ||   LeFeld´
7. (N032.500) K\_externeWelt {K\_Karte}   
   Wenn die Kanalnummer im CLA-Byte im Intervall
   1. [0, 3] liegt, dann MUSS gesetzt werden CLA' = CLA   OR   ´0C´   
      *Hinweis: Dadurch werden die Bits b4 und b3 in CLA gesetzt.*
   2. andernfalls MUSS gesetzt werden CLA' = CLA   OR   ´20´   
      *Hinweis: Dadurch wird das Bit b6 in CLA gesetzt.*
8. (N032.600) K\_externeWelt {K\_Karte}   
   Wenn die Kanalnummer im CLA-Byte im Intervall
   1. [0, 3] liegt, dann MUSS gelten *head* = CLA´ || INS || P1 || P2
   2. andernfalls MUSS gelten *head* =   ´89 04´ || CLA´ || INS || P1 || P2
9. (N032.700) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N032.190) verschoben.
10. (N032.800) K\_externeWelt {K\_Karte}   
    Es MUSS gelten: tmpData = ProtectedData   ||   LeDO   
    Falls *Kmac* ein
    1. Option\_DES, 3TDES-Schlüssel ist, dann MUSS gelten:
       1. MACin = I2OS(*SSCmac*, 8 )
       2. Falls die Kanalnummer im CLA-Byte größer gleich vier ist, oder OctetLength( tmpData ) gleich null ist
          1. dann gilt: MACin = MACin   ||   head   ||   tmpData
          2. sonst: MACin = MACin   ||   PaddingIso( head, 8 )   ||   tmpData
       3. MAC = CALCULATE\_Retail\_MAC(*Kmac*, MACin )
    2. AES-Schlüssel ist, dann MUSS gelten:
       1. MACin = I2OS(*SSCmac*, 16 )
       2. Falls die Kanalnummer im CLA-Byte größer gleich vier ist, oder OctetLength( tmpData ) gleich null ist
          1. dann gilt: MACin = MACin   ||   head   ||   tmpData
          2. sonst: MACin = MACin   ||   PaddingIso( head, 16)   ||   tmpData
       3. MAC = CalculateCMAC\_IsoPadding(*Kmac*, MACin )
11. (N032.900) K\_externeWelt {K\_Karte}   
    Für MDO MUSS gelten:   MDO = ´8E   ||   I2OS(OctetLength(MAC), 1)   ||   MAC´
12. (N033.000) K\_externeWelt {K\_Karte}   
    Für newD MUSS gelten: Wenn die Kanalnummer im CLA-Byte im Intervall
    1. [0, 3] liegt, dann setze: newD   = tmpData   ||   MDO
    2. andernfalls setze: newD   =   head   || tmpData   ||   MDO
13. (N033.100) K\_externeWelt {K\_Karte}   
    Case 1: Falls Data und LeFeld fehlen, dann setze:   
    CmdApdu = Case4S(CLA’, INS, P1, P2, newD, WildCardShort)
14. (N033.200) K\_externeWelt {K\_Karte}   
    Case 2: Falls Data fehlt und LeFeld vorhanden ist, dann MUSS gelten:   
    CmdApdu = Case4E(CLA’, INS, P1, P2, newD, WildCardExtended)
15. (N033.300) K\_externeWelt {K\_Karte}   
    Case 3: Falls Data vorhanden ist und LeFeld fehlt und OctetLength( newD )
    1. kleiner gleich 255 ist, dann MUSS gelten:   
       CmdApdu = Case4S(CLA’, INS, P1, P2, newD, WildCardShort)
    2. andernfalls MUSS gelten:   
       CmdApdu = Case4E(CLA’, INS, P1, P2, newD, WildCardExtended)
16. (N033.400) K\_externeWelt {K\_Karte}   
    Case 4: Falls Data und LeFeld vorhanden sind, dann MUSS gelten:   
    CmdApdu = Case4E(CLA’, INS, P1, P2, newD, WildCardExtended)
17. Zu (N033.100) und (N033.300)a: Gemäß 11.7.1 (11.7.3) hat die korrespondierende Antwort-APDU zu einer Case 1 (Case 3) Kommando-APDU niemals Antwortdaten. Zudem werden in 13.3 ausschließlich symmetrische Verfahren verwendet. Daraus folgt, dass eine gesicherte Antwort-APDU zu einer ungesicherten Case 1 (Case 3) APDU niemals mehr als 256 Oktette Antwortdaten enthält. Deshalb wird hier für die gesicherte Case 1 (Case 3) Kommando-APDU eine Case 4 Short Kommando-APDU verwendet.
18. Zu (N033.200) und (N033.400): Gemäß 11.7.2.1 (11.7.4.1) hat die korrespondierende Antwort-APDU zu einer Case 2 Short (Case 4 Short) Kommando-APDU bis zu 256 Oktette Antwortdaten. Gemäß 13.3 ist es deshalb möglich, dass eine gesicherte Antwort-APDU zu einer ungesicherten Case 2 Short (Case 4 Short) APDU mehr als 256 Oktette Antwortdaten enthält. Deshalb wird hier für gesicherte Case 2 Short (Case 4 Short) Kommando-APDU eine Case 4 Extended Kommando-APDU verwendet.

## Sicherung einer Antwort-APDU

Dieses Unterkapitel beschreibt, wie eine Antwort-APDU gemäß 11.6 zu sichern ist. Die hier beschriebenen Regeln richten sich an das COS. Die Instanz, welche die korrespondierende Kommando-APDU gesendet hat und typischerweise diese gesicherte Antwort-APDU entgegennimmt, führt die entsprechenden Umkehroperationen durch, um in den Besitz der ungesicherten Antwort-APDU zu gelangen.

Generell wird hier ein Subset der Regeln aus [ISO/IEC 7816-4#10] beschrieben, wobei

* alle Antwort-APDU mit Integritätsschutz übertragen werden,
* der Trailer stets mit Integritätsschutz übertragen wird,
* Response-Daten als Klartext oder verschlüsselt übertragen werden.

Hier sei angemerkt, dass das COS anhand der verwendeten Zugriffsbedingung (Vorhandensein oder Fehlen von *flagRspEnc*, siehe (N022.600)d) entscheidet, ob vorhandene Responsedaten im Klartext oder als Chiffrat übertragen werden.

Wie in 11.6 beschrieben, besteht eine Antwort-APDU generisch betrachtet aus dem optionalen Datenfeld Data und dem Trailer. Somit gelten folgende Definitionen:

|  |  |  |
| --- | --- | --- |
| Input: | Data | Optionaler Oktettstring gemäß 11.6.1 |
| Trailer | Oktettstring gemäß 11.6.2 |
| *Kenc* | Symmetrischer Schlüssel für die Verschlüsselung, (N029.900)d.2 |
| *SSCenc* | Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der Verschlüsselung verwendet wird, (N029.900)d.3 |
| *Kmac* | Symmetrischer Schlüssel für die MAC-Berechnung, (N029.900)d.4 |
| *SSCmac* | beliebige nicht-negative Zahl, die als Send Sequence Counter bei der MAC-Berechnung verwendet wird, (N029.900)d.5 |
| Output: | RspApdu | Gesicherte Antwort-APDU |
| Errors: | – | Keine |

Zur Sicherung einer generischen Antwort-APDU sind folgende Schritte durchzuführen:

1. (N033.500) K\_COS   
   Falls das optionale Datenfeld Data fehlt, MUSS gelten:   
   ProtectedData = ´´ (leerer Oktettstring).
2. (N033.600) K\_COS   
   Falls das optionale Datenfeld Data vorhanden ist und im Klartext übertragen wird, MUSS gelten:
   1. Setze lenPDO = OctetLength( Data ), falls lenPDO im Intervall
      1. [1, 127] liegt, gilt: LenP = I2OS(lenPDO, 1)
      2. [128, 255] liegt, gilt: LenP = ´81´   ||   I2OS(lenPDO, 1)
      3. [256, 65535] liegt, gilt: LenP = ´82´   ||   I2OS(lenPDO, 2)
   2. Setze ProtectedData = ´81   ||   LenP   ||   Data´
3. (N033.690) K\_COS   
   *SSCmac* MUSS inkrementiert werden, das heißt:   *SSCmac* = *SSCmac* + 1.
4. (N033.700) K\_COS\_G1, Option\_DES   
   Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird (*flagRspEnc* ist True, siehe (N029.900)d.7 und (N022.600)d) und *Kenc* ein 3TDES-Schlüssel ist, MUSS gelten:
   1. *SSCenc* = *SSCenc* + 1
   2. C = 3TDES\_CBC\_ENC(*Kenc*, *SSCenc*, PaddingIso( Data, 8 ) )
   3. Setze lenCDO = OctetLength( C ) + 1, falls lenCDO im Intervall
      1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)
      2. [128, 255] liegt, gilt: LenC = ´81´   ||   I2OS(lenCDO, 1)
      3. [256, 65535] liegt, gilt LenC = ´82´   ||   I2OS(lenCDO, 2)
   4. Setze ProtectedData = ´87   ||   LenC   ||   01   ||   C´
5. (N033.800) K\_COS   
   Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird (*flagRspEnc* ist True, siehe (N029.900)d.7 und (N022.600)d) und *Kenc* ein AES-Schlüssel ist, MUSS gelten:
   1. IVenc = OS2I(AES\_ENC( Kenc, I2OS(SSCmac, 16)))
   2. C = AES\_CBC\_ENC( *Kenc*, IVenc, PaddingIso( Data, 16 ))
   3. Setze lenCDO = OctetLength( C ) + 1, falls lenCDO im Intervall
      1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)
      2. [128, 255] liegt, gilt: LenC = ´81´   ||   I2OS(lenCDO, 1)
      3. [256, 65535] liegt, gilt LenC = ´82´   ||   I2OS(lenCDO, 2)
   4. Setze ProtectedData = ´87   ||   LenC   ||   01   ||   C´
6. (N033.900) K\_COS   
   Für TDO MUSS gelten: TDO = ´99 02   ||   Trailer´
7. (N034.000) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N033.690) verschoben.
8. (N034.100) K\_COS   
   Falls *Kmac* ein
   1. Option\_DES, 3TDES-Schlüssel ist, MUSS gelten:
      1. MACin = I2OS(*SSCmac*, 8 )
      2. MACin = MACin   || ProtectedData
      3. MACin = MACin   || TDO
      4. MAC = CALCULATE\_Retail\_MAC(*Kmac*, MACin )
   2. AES-Schlüssel ist, MUSS gelten:
      1. MACin = I2OS(*SSCmac*, 16 )
      2. MACin = MACin   || ProtectedData
      3. MACin = MACin   || TDO
      4. MAC = CalculateCMAC\_IsoPadding(*Kmac*, MACin )
9. (N034.200) K\_COS   
   Für MDO MUSS gelten:   MDO = ´8E   ||   I2OS(OctetLength(MAC), 1)   ||   MAC´
10. (N034.300) K\_COS   
    Für die gesicherte Antwort-APDU RspApdu MUSS gelten:
    1. RspApdu.Datenfeld = ProtectedData   ||   TDO   ||   MDO
    2. RspApdu.Trailer = Trailer

# Kommandos (normativ)

Tabelle 22: Kommandos, alphabetisch

|  |  |  |
| --- | --- | --- |
| **Kommando** | **CLA INS** |  |
| Activate | ´00´ ´44´ | 14.2.1 |
| Activate Record | ´00´ ´08´ | 14.4.1 |
| Append Record | ´00´ ´E2´ | 14.4.2 |
| Change Reference Data | ´00´ ´24´ | 14.6.1 |
| Create | ´00´ ´E0´ | 14.2.2 |
| Deactivate | ´00´ ´04´ | 14.2.3 |
| Deactivate Record | ´00´ ´06´ | 14.4.3 |
| Delete | ´00´ ´E4´ | 14.2.4 |
| Delete Record | ´80´ ´0C´ | 14.4.4 |
| Disable Verification Requirement | ´00´ ´26´ | 14.6.2 |
| Enable Verification Requirement | ´00´ ´28´ | 14.6.3 |
| Envelope | ´00´ ´C2´ | 14.9.1 |
| Erase Binary | ´00´ ´0E´ | 14.3.1 |
| Erase Record | ´00´ ´0C´ | 14.4.5 |
| External Authenticate | ´00´ ´82´ | 14.7.1 |
| Fingerprint | ´80´ ´FA´ | 14.9.2 |
| General Authenticate | ´00´ ´86´ | 14.7.2 |
| Generate Asymmetric Key Pair | ´00´ ´46´ | 14.9.3 |
| Get Challenge | ´00´ ´84´ | 14.9.4 |
| Get Data | ´00´ ´CA´ | 14.5.1 |
| Get Pin Status | ´80´ ´20´ | 14.6.4 |
| Get Random | ´80´ ´84´ | 14.9.5 |
| Get Response | ´00´ ´C0´ | 14.9.6 |
| Get Security Status Key | ´80´ ´82´ | 14.7.3 |
| Internal Authenticate | ´00´ ´88´ | 14.7.4 |
| List Public Key | ´80´ ´CA´ | 14.9.7 |
| Load Application | ´00´ ´EA´ | 14.2.5 |
| Manage Channel | ´00´ ´70´ | 14.9.8 |
| Manage Security Environment | ´00´ ´22´ | 14.9.9 |
| Mutual Authenticate | ´00´ ´82´ | 14.7.1 |
| Perform Security Operation | ´00´ ´2A´ | 14.8 |
| Put Data | ´00´ ´DA´ | 14.5.2 |
| Read Binary | ´00´ ´B0´ | 14.3.2 |
| Read Record | ´00´ ´B2´ | 14.4.6 |
| Reset Retry Counter | ´00´ ´2C´ | 14.6.5 |
| Search Binary | ´00´ ´A0´ | 14.3.3 |
| Search Record | ´00´ ´A2´ | 14.4.7 |
| Select | ´00´ ´A4´ | 14.2.6 |
| Set Logical Eof | ´80´ ´0E´ | 14.3.4 |
| Terminate | ´00´ ´E8´ | 14.2.9 |
| Terminate Card Usage | ´00´ ´FE´ | 14.2.7 |
| Terminate DF | ´00´ ´E6´ | 14.2.8 |
| Update Binary | ´00´ ´D6´ | 14.3.5 |
| Update Record | ´00´ ´DC´ | 14.4.8 |
| Verify | ´00´ ´20´ | 14.6.6 |
| Write Binary | ´00´ ´D0´ | 14.3.6 |
| Write Record | ´00´ ´D2´ | 14.4.9 |

Tabelle 23: Kommandos, numerisch

|  |  |  |
| --- | --- | --- |
| **Kommando** | **CLA INS** |  |
| Deactivate | ´00´ ´04´ | 14.2.3 |
| Deactivate Record | ´00´ ´06´ | 14.4.3 |
| Activate Record | ´00´ ´08´ | 14.4.1 |
| Erase Record | ´00´ ´0C´ | 14.4.5 |
| Erase Binary | ´00´ ´0E´ | 14.3.1 |
| Verify | ´00´ ´20´ | 14.6.6 |
| Manage Security Environment | ´00´ ´22´ | 14.9.9 |
| Change Reference Data | ´00´ ´24´ | 14.6.1 |
| Disable Verification Requirement | ´00´ ´26´ | 14.6.2 |
| Enable Verification Requirement | ´00´ ´28´ | 14.6.3 |
| Perform Security Operation | ´00´ ´2A´ | 14.8 |
| Reset Retry Counter | ´00´ ´2C´ | 14.6.5 |
| Activate | ´00´ ´44´ | 14.2.1 |
| Generate Asymmetric Key Pair | ´00´ ´46´ | 14.9.3 |
| Manage Channel | ´00´ ´70´ | 14.9.8 |
| External Authenticate | ´00´ ´82´ | 14.7.1 |
| Mutual Authenticate | ´00´ ´82´ | 14.7.1 |
| Get Challenge | ´00´ ´84´ | 14.9.4 |
| General Authenticate | ´00´ ´86´ | 14.7.2 |
| Internal Authenticate | ´00´ ´88´ | 14.7.4 |
| Search Binary | ´00´ ´A0´ | 14.3.3 |
| Search Record | ´00´ ´A2´ | 14.4.7 |
| Select | ´00´ ´A4´ | 14.2.6 |
| Read Binary | ´00´ ´B0´ | 14.3.2 |
| Read Record | ´00´ ´B2´ | 14.4.6 |
| Get Response | ´00´ ´C0´ | 14.9.6 |
| Envelope | ´00´ ´C2´ | 14.9.1 |
| Get Data | ´00´ ´CA´ | 14.5.1 |
| Write Binary | ´00´ ´D0´ | 14.3.6 |
| Write Record | ´00´ ´D2´ | 14.4.9 |
| Update Binary | ´00´ ´D6´ | 14.3.5 |
| Put Data | ´00´ ´DA´ | 14.5.2 |
| Update Record | ´00´ ´DC´ | 14.4.8 |
| Create | ´00´ ´E0´ | 14.2.2 |
| Append Record | ´00´ ´E2´ | 14.4.2 |
| Delete | ´00´ ´E4´ | 14.2.4 |
| Terminate DF | ´00´ ´E6´ | 14.2.8 |
| Terminate | ´00´ ´E8´ | 14.2.9 |
| Load Application | ´00´ ´EA´ | 14.2.5 |
| Terminate Card Usage | ´00´ ´FE´ | 14.2.7 |
| Delete Record | ´80´ ´0C´ | 14.4.4 |
| Set Logical Eof | ´80´ ´0E´ | 14.3.4 |
| Get Pin Status | ´80´ ´20´ | 14.6.4 |
| Get Security Status Key | ´80´ ´82´ | 14.7.3 |
| Get Random | ´80´ ´84´ | 14.9.5 |
| List Public Key | ´80´ ´CA´ | 14.9.7 |
| Fingerprint | ´80´ ´FA´ | 14.9.2 |

1. (N034.400) K\_COS   
   Das COS KANN Kommando-APDU unterstützen, die in diesem Kapitel nicht aufgeführt sind.

## Roll-Verhalten

In den folgenden Unterkapiteln ist gelegentlich die Rede davon, dass der persistente Speicher mittels „Roll-Forward“ oder „Roll-Back“ zu verändern ist. Als Oberbegriff wird „Transaktionsschutz“ verwendet, um auszudrücken, dass Roll-Forward oder Roll-Back gemeint ist. Kurz gesagt verbirgt sich dahinter Folgendes: Das persistente Speichern von Informationen dauert aus technischen Gründen einige Millisekunden. Da Smartcards aus technischen Gründen nicht in der Lage sind, einen Ausfall der Spannungsversorgung zu puffern, ist es denkbar, dass der Ausfall zu einem Zeitpunkt geschieht, in welchem der Zustand des persistenten Speichers in einem undefinierten Zustand ist. Der Transaktionsschutz legt dann fest, wie mit diesem möglicherweise undefiniertem Zustand umzugehen ist.



Abbildung 2: Zeitlicher Ablauf eines Roll-Back-Kommandos

Für die Bearbeitung eines Kommandos mit Transaktionsschutz werden, wie in Abbildung 2 gezeigt, die folgenden Zeitpunkte definiert:

* Zum Zeitpunkt t0 werde das erste Bit der Kommando-APDU über die physikalische Schnittstelle (siehe Abbildung 1) zur Smartcard gesendet.
* Zum Zeitpunkt t1 sei die Kommandobearbeitung so weit vorgeschritten, dass der Zwischenspeicher bereit ist, befüllt zu werden.
* Zum Zeitpunkt t2 sei der Zwischenspeicher komplett befüllt, aber sein Inhalt noch nicht als gültig gekennzeichnet.
* Zum Zeitpunkt t3 sei der Inhalt als gültig gekennzeichnet und die eigentliche persistente Änderung werde gestartet.
* Zum Zeitpunkt t4 sei die eigentliche persistente Änderung abgeschlossen, aber der Inhalt des Zwischenpuffers sei noch als gültig gekennzeichnet.
* Zum Zeitpunkt t5 sei der Inhalt des Zwischenspeichers als ungültig gekennzeichnet.
* Zum Zeitpunkt te (in Abbildung 2 nicht dargestellt) habe die Smartcard das letzte Bit der Antwort-APDU über die physikalische Schnittstelle versendet.

1. (N034.500) K\_COS   
   Dieses Dokument legt nicht fest, in welchem zeitlichen Zusammenhang te zu den anderen Zeiten steht. Damit KANN te zu einem beliebigen Zeitpunkt nach t0 erfolgen.

### Roll-Back

Roll-Back legt fest, dass die durch den Spannungsausfall unterbrochene Aktion rückgängig zu machen ist, wenn wieder eine Versorgungsspannung anliegt. Typischerweise wird dazu vor Durchführung der Änderung der *ursprüngliche* Inhalt in einen Zwischenspeicher geschrieben, dann die Änderung durchgeführt und anschließend der Inhalt des Zwischenspeichers als ungültig gekennzeichnet.

1. (N034.600) K\_COS   
   Findet der Ausfall der Versorgungsspannung
   1. vor t4 statt, so ist entweder noch keine Änderung erfolgt, oder der Inhalt des Zwischenpuffers definitiv auf gültig gesetzt und der (hier ursprüngliche) Inhalt des Zwischenspeichers MUSS nach Wiederanlegen der Versorgungsspannung wiederhergestellt werden.
   2. nach t5 statt, so ist der Inhalt des Zwischenpuffers definitiv auf ungültig gesetzt und damit eine Wiederherstellung des ursprünglichen Zustandes unmöglich.
   3. zwischen t4 und t5 statt, so hängt es vom Zufall ab, ob der Zustand des Zwischenspeichers als gültig oder ungültig beurteilt wird (physikalische Speicher haben mitnichten ein zeit- oder wertediskretes Verhalten).   
      In diesem Fall MUSS entweder der ursprüngliche Zustand rekonstruiert oder der neue Zustand beibehalten werden.

### Roll-Forward

Roll-Forward legt fest, dass die durch den Spannungsausfall unterbrochene Aktion fortzusetzen ist, wenn wieder eine Versorgungsspannung anliegt. Typischerweise wird dazu vor Durchführung der Änderung der *neue* Inhalt in einen Zwischenspeicher geschrieben, dann die Änderung durchgeführt und anschließend der Inhalt des Zwischenspeichers als ungültig gekennzeichnet.

1. (N034.700) K\_COS   
   Findet der Ausfall der Versorgungsspannung
   1. vor t2 statt, so ist der Inhalt des Zwischenspeichers definitiv nicht auf gültig gesetzt, und damit ist ein Wechsel zum neuen Zustand unmöglich.
   2. nach t3 statt, so ist der Inhalt des Zwischenpuffers definitiv auf gültig gesetzt, und der (hier neue) Inhalt des Zwischenspeichers MUSS nach Wiederanlegen der Versorgungsspannung wiederhergestellt werden.
   3. zwischen t2 und t3 statt, so hängt es vom Zufall ab, ob der Zustand des Zwischenspeichers als gültig oder ungültig beurteilt wird (physikalische Speicher haben mitnichten zeit- oder wertediskretes Verhalten).   
      In diesem Fall MUSS entweder der ursprüngliche Zustand beibehalten oder der neue Zustand gesetzt werden.

## Management des Objektsystems

### Activate

Das Kommando Activate aktiviert reversibel ein Objekt. Ein betroffenes File wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses Activate-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*). Falls ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist.

#### Use Case Aktivieren eines Ordners oder einer Datei

In dieser Variante wird ein Ordner oder eine Datei aktiviert und die APDU des Activate Kommandos enthält einen Parameter:

1. (N034.798) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass das aktuelle File zu aktivieren ist.
2. (N034.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 24 verwendet werden.

Tabelle 24: Activate aktuelles File

|  |  |  |
| --- | --- | --- |
|  |  | **Beschreibung** |
| CLA | ´00´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´44´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | *mode*, der Wert ´00´ zeigt an, dass das aktuelle File zu aktivieren ist |
| P2 | ´00´ | – |

#### Use Case Aktivieren eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt aktiviert und die APDU des Activate-Kommandos enthält zwei Parameter:

1. (N034.810) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu aktivieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.
2. (N034.812) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
3. (N034.814) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 25 verwendet werden.

Tabelle 25: Activate privates oder symmetrisches Schlüsselobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´44´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´20´ | *mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Schlüsselobjekt |

#### Use Case Aktivieren eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt aktiviert und die APDU des Activate-Kommandos enthält zwei Parameter:

1. (N034.820) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu aktivieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.
2. (N034.822) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt.
3. (N034.824) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 26 verwendet werden.

Tabelle 26: Activate öffentliches Schlüsselobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´44´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´21´ | *mode*, hier: Schlüsselreferenz im Datenteil |
| P2 | ´00´ | ‑ |
| Data | ´XX…XX´ | ´83 – I2OS(OctetLength(*reference*), 1) – *reference*´ |

#### Use Case Aktivieren eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt aktiviert und die APDU des Activate-Kommandos enthält zwei Parameter:

1. (N034.830) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu aktivieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.
2. (N034.832) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.
3. (N034.834) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 27 verwendet werden.

Tabelle 27: Activate Passwortobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´44´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´10´ | *mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Passwortobjekt |

#### Antwort der Karte auf Aktivieren eines Files

Tabelle 28: Activate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Aktivierung |

Tabelle 29: Activate Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | ObjectTerminated | Objekt befindet sich im Zustand „Termination state" |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | VolatileKeyWithoutLCS | volatile Schlüssel vom Kommando nicht unterstützt |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´6A 88´ | KeyNotFound | Schlüsselobjekt nicht gefunden |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N034.900) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N035.000) K\_COS
   1. Das COS MUSS die Activate Varianten aus 14.2.1.1, 14.2.1.2, 14.2.1.3 und 14.2.1.4 unterstützen.
   2. Das COS KANN weitere Activate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N035.100) K\_COS   
   Falls der Parameter *mode* in der Kommandonachricht den Wert
   1. ´00´ besitzt und *channelContext*.*currentEF* (siehe (N029.900)m)
      1. auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.
      2. andernfalls MUSS *affectedObject* gleich *currentFolder* gesetzt werden.
   2. ´20´ oder ´21´ besitzt, dann gilt *affectedObject* = SearchKey(   
          *channelContext*.*currentFolder,*   
          *reference*,   
         “WildCard”   
      ). Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
   3. ´10´ besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N035.110) K\_COS   
   Falls *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.
4. (N035.200) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
5. (N035.300) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert
   1. „Termination state" besitzt, dann MUSS das Kommando mit dem Trailer ObjectTerminated terminieren.
   2. „Operational state (active)“ besitzt, dann MUSS als Trailer NoError verwendet werden.
6. (N035.400) K\_COS   
   Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Operational state (active)“ gesetzt werden.
7. (N035.500) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
8. (N035.600) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N035.700) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
10. (N035.800) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 29 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 29 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### Create

1. (N035.900) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-9]
   1. unterstützen oder
   2. ablehnen.
2. Die entsprechende Funktionalität dieses Kommandos wird im Rahmen dieses Dokumentes durch das Kommando Load Application (siehe 14.2.5) bereitgestellt.

### Deactivate

Das Kommando Deactivate deaktiviert reversibel ein Objekt. Ein betroffenes File wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses Deactivate Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*). Falls ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist.

#### Use Case Deaktivieren eines Ordners oder einer Datei

In dieser Variante wird ein Ordner oder eine Datei deaktiviert und die APDU des Deactivate-Kommandos enthält einen Parameter:

1. (N035.998) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass das aktuelle File zu deaktivieren ist.
2. (N036.000) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 30 verwendet werden.

Tabelle 30: Deactivate aktuelles File

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´04´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | *mode*, der Wert ´00´ zeigt an, dass das aktuelle File zu deaktivieren ist |
| P2 | ´00´ | – |

#### Use Case Deaktivieren eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt deaktiviert und die APDU des Deactivate-Kommandos enthält zwei Parameter:

1. (N036.010) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu deaktivieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.
2. (N036.012) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
3. (N036.014) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 31 verwendet werden.

Tabelle 31: Deactivate privates oder symmetrisches Schlüsselobjekt

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´04´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´20´ | *mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Schlüsselobjekt |

#### Use Case Deaktivieren eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt deaktiviert und die APDU des Deactivate-Kommandos enthält zwei Parameter:

1. (N036.020) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu deaktivieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.
2. (N036.022) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.
3. (N036.024) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 32 verwendet werden.

Tabelle 32: Deactivate öffentliches Schlüsselobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´04´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´21´ | *mode*, hier: Schlüsselreferenz im Datenteil |
| P2 | ´00´ | ‑ |
| Data | ´XX…XX´ | ´83 – I2OS(OctetLength(*reference*), 1) – *reference*´ |

#### Use Case Deaktivieren eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt deaktiviert und die APDU des Deactivate-Kommandos enthält zwei Parameter:

1. (N036.030) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu deaktivieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.
2. (N036.032) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.
3. (N036.034) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 33 verwendet werden.

Tabelle 33: Deactivate Passwortobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´04´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´10´ | *mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Passwortobjekt |

#### Antwort der Karte auf Deaktivieren eines Files

Tabelle 34: Deactivate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Deaktivierung |

Tabelle 35: Deactivate Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | ObjectTerminated | Objekt befindet sich im Zustand „Termination state" |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | VolatileKeyWithoutLCS | volatile Schlüssel vom Kommando nicht unterstützt |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´6A 88´ | KeyNotFound | Schlüsselobjekt nicht gefunden |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N036.100) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N036.200) K\_COS
   1. Das COS MUSS die Deactivate-Varianten aus 14.2.3.1, 14.2.3.2, 14.2.3.3 und 14.2.3.4 unterstützen.
   2. Das COS KANN weitere Deactivate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N036.300) K\_COS   
   Falls der Parameter *mode* in der Kommandonachricht den Wert
   1. ´00´ besitzt und *channelContext*.*currentEF* (siehe (N029.900)m)
      1. auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.
      2. andernfalls MUSS *affectedObject* gleich *currentFolder* gesetzt werden.
   2. ´20´ oder ´21´ besitzt, dann gilt *affectedObject* = SearchKey(   
          *channelContext*.*currentFolder,*   
          *reference*,   
         “WildCard”   
      ). Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
   3. ´10´ besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N036.310) K\_COS   
   Falls *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.
4. (N036.400) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
5. (N036.500) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert
   1. „Termination state" besitzt, dann MUSS das Kommando mit dem Trailer ObjectTerminated terminieren.
   2. „Operational state (deactivated)“ besitzt, dann MUSS als Trailer NoError verwendet werden.
6. (N036.600) K\_COS   
   Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Operational state (deactivated)“ gesetzt werden.
7. (N036.700) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
8. (N036.800) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N036.900) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
10. (N037.000) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 35 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 35 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### Delete

Das Kommando Delete löscht Objekte aus dem Objektsystem. Ein zu löschendes File wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses Delete Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*). Welches Schlüsselobjekt oder welches Passwortobjekt vom Kommando betroffen ist, wird durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist. Für dieses Kommando gelten Restriktionen, siehe (N099.500).

#### Use Case Löschen eines Ordners oder einer Datei

In dieser Variante wird ein Ordner oder eine Datei gelöscht und die APDU des Delete-Kommandos enthält einen Parameter:

1. (N037.098) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass das aktuelle File zu löschen ist.
2. (N037.100) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 36 verwendet werden.

Tabelle 36: Delete aktuelles File

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | *mode*, der Wert ´00´ zeigt an, dass das aktuelle File zu löschen ist |
| P2 | ´00´ | – |

#### Use Case Löschen eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt gelöscht und die APDU des Delete-Kommandos enthält zwei Parameter:

1. (N037.110) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu löschen ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.
2. (N037.112) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
3. (N037.114) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 37 verwendet werden.

Tabelle 37: Delete privates oder symmetrisches Schlüsselobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´20´ | *mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Schlüsselobjekt |

#### Use Case Löschen eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt gelöscht und die APDU des Delete-Kommandos enthält zwei Parameter:

1. (N037.120) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu löschen ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.
2. (N037.122) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.
3. (N037.124) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 38 verwendet werden.

Tabelle 38: Delete öffentliches Schlüsselobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´21´ | *mode*, hier: Schlüsselreferenz im Datenteil |
| P2 | ´00´ | ‑ |
| Data | ´XX…XX´ | ´83 –I2OS(OctetLength(*reference*), 1) – *reference*´ |

#### Use Case Löschen eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt gelöscht und die APDU des Delete-Kommandos enthält zwei Parameter:

1. (N037.130) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu löschen ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.
2. (N037.132) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.
3. (N037.134) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 39 verwendet werden.

Tabelle 39: Delete Passwortobjekt

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´10´ | *mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Passwortobjekt |

#### Antwort der Karte auf Löschen eines Files

Tabelle 40: Delete Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Löschoperation |

Tabelle 41: Delete Antwort-APDU im Fehlerfall

|  |  |  |  |
| --- | --- | --- | --- |
| **Trailer** | **Inhalt** | | **Beschreibung** |
| ´65 81´ | MemoryFailure | | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | | Zugriffsregel nicht erfüllt |
| ´6A 88´ | KeyNotFound | Schlüsselobjekt nicht gefunden | |
| ´6A 88´ | PasswordNotFound | | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N037.200) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N037.300) K\_COS
   1. Das COS MUSS die Delete-Varianten aus 14.2.4.1, 14.2.4.2, 14.2.4.3 und 14.2.4.4 unterstützen.
   2. Das COS KANN weitere Delete-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N037.400) K\_COS   
   Falls der Parameter *mode* in der Kommandonachricht den Wert
   1. ´00´ besitzt und *channelContext*.*currentEF* (siehe (N029.900)m)
      1. auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.
      2. andernfalls MUSS *affectedObject* gleich *currentFolder* gesetzt werden.
   2. ´20´ oder ´21´ besitzt, dann gilt *affectedObject* = SearchKey(   
          *channelContext*.*currentFolder,*   
          *reference*,   
         “WildCard”   
      ). Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
   3. ´10´ besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N037.500) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N037.600) K\_COS   
   Die Löschoperation MUSS mit Transaktionsschutz durchgeführt werden.
5. (N037.690) K\_COS   
   Falls die Kommandos-APDU dieses Delete-Kommandos gemäß 13.2 mittels Sessionkeys gesichert war, dann MUSS die korrespondierende Antwort-APDU gemäß 13.3 mit denselben Sessionkeys gesichert sein, auch wenn als Seiteneffekt von (N037.700) Sessionkeys gelöscht werden.
6. (N037.700) K\_COS   
   Wenn *affectedObject* vom Typ
   1. Ordner ist, dann MUSS

* + 1. clearSecurityStatusFolder( *affectedObject* ) ausgeführt werden und
    2. aus *dfSpecificPasswordList* alle Einträge entfernt werden, die zu *affectedObject* gehören und
    3. in *keyReferenceList* alle Einträge auf den Wert „leer" gesetzt werden, die auf Schlüsselobjekte verweisen, die *affectedObject* zugeordner sind und
    4. *currentEF* auf den Wert undefined gesetzt werden und
    5. *currentFolder* auf den Vater von *affectedObject* gesetzt werden und
    6. aus *allPublicKeyList* MÜSSEN alle Einträge entfernt werden, die *affectedObject* zugeordnet sind und
    7. *affectedObject* inklusive Subtree gelöscht werden.
  1. Datei ist, dann MUSS
     1. *currentEF* auf den Wert undefined gesetzt werden und
     2. *affectedObject* gelöscht werden.
  2. Symmetrisches Authentisierungsobjekt oder symmetrisches Kartenverbindungsobjekt ist, dann MUSS
     1. clearSecurityStatusKey( *affectedObject* ) ausgeführt werden und
     2. *affectedObject* gelöscht werden.
  3. Passwortobjekt ist, dann MUSS
     1. clearPasswordStatus( *affectedObject* ) ausgeführt werden und
     2. *affectedObject* gelöscht werden.
  4. sonst MUSS *affectedObject* gelöscht werden.

1. (N037.800) K\_COS   
   Falls *affectedObject* vom Typ
   1. Ordner oder Datei ist, dann MUSS vormals von diesem Objekt allokierter Speicher so freigegeben werden, dass er zum Anlegen anderer Objekte verwendbar ist.
   2. Symmetrisches Authentisierungsobjekt oder Passwortobjekt ist, dann
      1. SOLL vormals von diesem Objekt allokierter Speicher so freigegeben werden, dass er zum Anlegen anderer Objekte verwendbar ist.
      2. KANN das COS das betroffen Objekt löschen, ohne dass allokierter Speicher freigegeben wird.
2. (N037.850) K\_COS   
   Wenn *affectedObject* ein Schlüssel ist, der in einem Element von *keyReferenceList* eingetragen ist, dann KANN der zum gelöschten Schlüssel gehörende Eintrag in *keyReferenceList*
   1. gelöscht werden oder
   2. bestehen bleiben.
3. (N037.900) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
4. (N038.000) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
5. (N038.100) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
6. (N038.200) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 41 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 41 MUSS eine höhere Priorität als UpdateRetryWarning haben.
   3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
7. (N038.300) K\_COS   
   Im Fehlerfall MÜSSEN *currentFolder* und *currentEF* unverändert auf dem Wert vor Ausführung des Kommandos belassen werden.

### Load Application

Das Kommando Load Application wird verwendet, um neue Files im Objektsystem anzulegen. So ist es möglich,

* einen neuen Ordner inklusive Subtree,
* eine neue Datei inklusive Inhalt (transparent oder strukturiert)

anzulegen. Das neu angelegte File wird im currentFolder gespeichert. Typischerweise ist die beim Anlegen neuer Files zur Karte transferierte Datenmenge so groß, dass sie nicht in einer einzigen Kommando-APDU übertragbar ist. Deshalb unterstützt dieses Kommando „Command Chaining“.

#### Use Case Anlegen neues Objekt, nicht Ende der Kommandokette

Diese Variante ist zu wählen, wenn die Datenmenge nicht in einer Kommando-APDU übertragbar ist. Sie kommt zum Einsatz von der ersten bis zur vorletzten Kommando-APDU. Die letzte Kommando-APDU wird gemäß 14.2.5.2 übertragen. In der hier beschriebenen Variante enthält die APDU des Load Application-Kommandos zwei Parameter.

1. (N038.400) K\_externeWelt {K\_Karte}   
   Das CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist. Dies wird durch den Wert CLA = ´10´ codiert.
2. (N038.500) K\_externeWelt {K\_Karte}   
   Der Parameter *cmdData* enthält Daten, welche das neu anzulegende File beschreiben. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem, herstellerspezifischem Inhalt. Die Länge von *cmdData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
3. (N038.600) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 42 verwendet werden.

Tabelle 42: Load Application mit Command Chaining

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´EA´ | Instruction Byte gemäß [ISO/IEC 7816-13] |
| P1 | ´00´ | – |
| P2 | ´00´ | – |
| Data | ´XX…XX´ | *cmdData* |

#### Use Case Anlegen neues Objekt, Ende der Kommandokette

Diese Variante ist zu wählen, wenn die Datenmenge in einer Kommando-APDU transferierbar ist, oder das letzte Kommando einer Chaining-Kette zu übertragen ist. In dieser Variante enthält die APDU des Load Application-Kommandos einen Parameter.

1. (N038.700) K\_externeWelt {K\_Karte}   
   Der Parameter *cmdData* enthält Daten, welche das neu anzulegende File beschreiben. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem, herstellerspezifischem Inhalt. Die Länge von *cmdData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
2. (N038.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 43 verwendet werden.

Tabelle 43: Load Application ohne Command Chaining

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´EA´ | Instruction Byte gemäß [ISO/IEC 7816-13] |
| P1 | ´00´ | – |
| P2 | ´00´ | – |
| Data | ´XX…XX´ | *cmdData* |

#### Antwort der Karte auf Anlegen neues Objekt

Tabelle 44: Load Application Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreicher Ladevorgang |

Tabelle 45: Load Application Antwort-APDU im Fehlerfall

| **Trailer** | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´6A 84´ | OutOfMemory | Zu wenig Speicherplatz für neues Objekt |
| ´6A 89´ | DuplicatedObject | Identifier des neuen Objektes wird bereits verwendet |
| ´6A 8A´ | DfNameExists | AID des neuen Objektes wird bereits verwendet |
| ´6D 00´ | InstructionNotSupported | Die Karte befindet sich im Zustand „Termination state" |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N038.900) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N039.000) K\_COS
   1. Das COS MUSS die Load Application-Varianten aus 14.2.5.1 und 14.2.5.2 unterstützen.
   2. Das COS KANN weitere Load Application-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N039.010) K\_COS   
   Falls das Attribut *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) den Wert „Termination state" besitzt, KANN das Kommando mit dem Trailer InstructionNotSupported terminieren.
3. (N039.100) K\_COS   
   Als *affectedObject* MUSS *channelContext*.*currentFolder* verwendet werden.
4. (N039.200) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
5. (N039.300) K\_COS   
   Das Kommando, und damit eine eventuell aktive Chaining-Kette, KANN akzeptiert oder abgelehnt werden, wenn das neu anzulegende File
   1. ein Ordner ist und ein Attribut *applicationIdentifier* besitzt und dieser *application­Identifier* bereits einem anderen Ordner innerhalb des Objektsystems zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DfNameExists terminieren), oder
   2. ein Ordner ist und ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* bereits einem anderen File innerhalb von *affectedObject* zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren), oder
   3. eine Datei ist, deren Attribut *fileIdentifier* bereits einem anderen File innerhalb von *affectedObject* zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren), oder
   4. eine Datei ist, deren Attribut *shortFileIdentifier* bereits einer anderen Datei innerhalb von *affectedObject* zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren).
6. (N039.400) K\_COS   
   Das Kommando MUSS akzeptiert werden, wenn keine der Bedingungen aus (N039.300) zutrifft und das neu anzulegende Objekt
   1. ein Ordner ist und ein Attribut *applicationIdentifier* besitzt und dieser *application­Identifier* keinem anderen Ordner innerhalb des Objektsystems zugeordnet ist,
   2. ein Ordner ist und ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* keinem anderen File innerhalb von *affectedObject* zugeordnet ist, oder
   3. eine Datei ist, deren Attribut *fileIdentifier* keinem anderen File innerhalb von *affectedObject* zugeordnet ist, oder
   4. eine Datei ist, deren Attribut *shortFileIdentifier* keiner anderen Datei innerhalb von *affectedObject* zugeordnet ist.
7. (N039.500) K\_COS   
   Wenn insgesamt ausreichender, aber nicht genügend zusammenhängender Speicherplatz vorhanden ist, dann MUSS das COS intern dafür sorgen, dass diese Operation trotzdem erfolgreich durchführbar ist. Typischerweise wird diese Operation als „Defragmentieren“ bezeichnet.
8. (N039.600) K\_COS   
   Wenn nicht genügend Speicherplatz zum Anlegen des neuen Objektes vorhanden ist, genau dann MUSS das Kommando mit dem Trailer OutOfMemory terminieren.
9. (N039.700) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
10. (N039.800) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
11. (N039.900) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
12. (N040.000) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 45 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 45 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
13. (N040.100) K\_COS   
    Wenn dies das einzige oder letzte Kommando einer Command-Chaining-Kette ist und das neu angelegte Objekt ist
    1. ein Ordner, dann MUSS im Erfolgsfall *channelContext*.*currentFolder* auf den neu angelegten Ordner gesetzt werden. Dabei sind die Regeln zum Setzen des Kanalkontextes für den neuen Ordner zu berücksichtigen (siehe (N048.200)b).
    2. eine Datei, dann MUSS im Erfolgsfall *currentEF* auf die neu angelegte Datei gesetzt werden.
14. (N040.200) K\_COS   
    Wenn das Kommando nicht erfolgreich verlief, dann DÜRFEN *currentFolder* und *currentEF* NICHT verändert werden, auch wenn
    1. ein Load Application-Kommando mit einem Trailer aus Tabelle 45 terminierte, sondern MÜSSEN denselben Wert besitzen, wie vor der Ausführung dieses Load Application-Kommandos.
    2. eine Command-Chaining-Kette abgebrochen wird (siehe (N029.874)), sondern MÜSSEN denselben Wert besitzen, wie vor dem Start dieser Command-Chaining-Kette.
15. (N040.300) K\_COS   
    Wenn dieses Load Application-Kommando mit einem Trailer aus Tabelle 45 terminierte, dann MUSS ein eventuell aktives Command Chaining abgebrochen werden.
16. (N040.400) K\_COS   
    Das neu angelegte bzw. neu anzulegende Objekt MUSS inklusive Freigabe eines eventuell von ihm allokierten Speichers aus dem Objektsystem gelöscht werden (komplettes Roll-Back der Chaining-Kette), wenn
    1. das Load Application-Kommando mit einem Trailer aus Tabelle 45 terminierte, oder
    2. das Load Application-Kommando während der Kommandobearbeitung durch einen Reset abgebrochen wird, oder
    3. eine Command-Chaining-Kette abgebrochen wird (siehe (N029.874)).
17. Es ist denkbar, dass Anforderung (N039.500) wie folgt getestet wird:
    1. Ausgangspunkt sei eine Smartcard, deren Objektsystem nur eine sehr geringe (minimale) Anzahl von Objekten enthält.
    2. Im ersten Schritt werde per Load Application-Kommando eine Datei (transparent oder strukturiert wird zufällig bestimmt) mit 200 Oktett Nutzdaten angelegt.
    3. Schritt b wird so lange wiederholt, bis das Load Application-Kommando mit dem Trailer OutOfMemory terminiert.
    4. Im zweiten Schritt werden zwei der zuvor angelegten Dateien zufällig ausgewählt und gelöscht (Delete). Anschließend werde per Load Application eine neue Datei (transparent oder strukturiert wird zufällig bestimmt) angelegt. Wenn die Summe der Nutzdaten der in diesem Schritt gelöschten Dateien x ist, dann werde als Größe der Nutzdaten der in diesem Schritt neu angelegten Datei ebenfalls x gewählt. Es wird erwartet, dass dieses Load Application-Kommando nicht mit OutOfMemory terminiert.
    5. Schritt d werde so lange wiederholt, bis nur noch eine Datei übrig ist, welche durch diesen Algorithmus angelegt wurde, oder nur noch Dateien übrig sind, welche die maximal mögliche Dateigröße gemäß (N011.500) bzw. (N013.000) besitzen. Zwar bezieht sich (N013.000) nur auf linear variable EF, ist aber analog übertragbar auf andere lineare EF.

### Select

Das Kommando Select sucht im Objektsystem nach einem File (Ordner oder Datei) und wählt dieses aus. Das Auswählen ist vielfach Voraussetzung, damit andere Use Cases (Lesen, Schreiben, …) erfolgreich durchführbar sind. Optional ist es möglich, in den Antwortdaten die wesentlichen Attribute des Files zurückzumelden. Welches File selektiert wird, bestimmen Parameter in der Kommandonachricht.

#### Use Case Selektieren ohne AID, first, keine Antwortdaten

Diese Variante selektiert das Wurzelverzeichnis des Objektsystems. In dieser Variante enthält die APDU des Select-Kommandos drei Parameter:

1. (N040.500) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N040.600) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N040.700) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
4. (N040.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 46 verwendet werden.

Tabelle 46: Select, kein AID, first occurrence, keine Antwortdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer) |
| P2 | ´0C´ | *fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten |

#### Use Case Selektieren ohne AID, first, Antwortdaten mit FCP

Diese Variante selektiert das Wurzelverzeichnis des Objektsystems. In dieser Variante enthält die APDU des Select-Kommandos vier Parameter:

1. (N040.900) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N041.000) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N041.100) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
4. (N041.200) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
5. (N041.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 47 verwendet werden.

Tabelle 47: Select, kein AID, first occurrence, Antwortdaten mit FCP

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer) |
| P2 | ´04´ | *fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Selektieren ohne AID, next, keine Antwortdaten

Der wiederholte Aufruf dieser Variante selektiert nacheinander alle Ordner, die das Attribut *applicationIdentifier* besitzen (Applikationen gemäß 8.3.1.1 und ADF gemäß 8.3.1.3). In dieser Variante enthält die APDU des Select-Kommandos drei Parameter:

1. (N041.400) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N041.500) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´2´ gewählt werden.
3. (N041.600) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
4. (N041.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 48 verwendet werden.

Tabelle 48: Select, kein AID, next occurrence, keine Antwortdaten

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer) |
| P2 | ´0E´ | *fileOccurrence* + *responseType* = next occurrence, keine Antwortdaten |

#### Use Case Selektieren ohne AID, next, Antwortdaten mit FCP

Der wiederholte Aufruf dieser Variante selektiert nacheinander alle Ordner, die das Attribut *applicationIdentifier* besitzen (Applikationen gemäß 8.3.1.1 und ADF gemäß 8.3.1.3). In dieser Variante enthält die APDU des Select-Kommandos vier Parameter:

1. (N041.800) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N041.900) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´2´ gewählt werden.
3. (N042.000) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
4. (N042.100) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
5. (N042.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 49 verwendet werden.

Tabelle 49: Select, kein AID, next occurrence, Antwortdaten mit FCP

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer) |
| P2 | ´06´ | *fileOccurrence* + *responseType* = next occurrence, FCP-Antwortdaten |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Selektieren per AID, first, keine Antwortdaten

In dieser Variante enthält die APDU des Select-Kommandos vier Parameter:

1. (N042.300) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N042.400) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N042.500) K\_externeWelt {K\_Karte}   
   Der Parameter *aid* enthält einen Oktettstring gemäß (N010.200) oder dessen Anfang. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.
4. (N042.600) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
5. (N042.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 50 verwendet werden.

Tabelle 50: Select, AID, first occurrence, keine Antwortdaten

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit applicationIdentifier |
| P2 | ´0C´ | *fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten |
| Data | ´XX…XX´ | *aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16] |

#### Use Case Selektieren per AID, first, Antwortdaten mit FCP

In dieser Variante enthält die APDU des Select-Kommandos fünf Parameter:

1. (N042.800) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N042.900) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N043.000) K\_externeWelt {K\_Karte}   
   Der Parameter *aid* enthält einen Oktettstring gemäß (N010.200) oder dessen Anfang. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.
4. (N043.100) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
5. (N043.200) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
6. (N043.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 51 verwendet werden.

Tabelle 51: Select, AID, first occurrence, Antwortdaten mit FCP

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit applicationIdentifier |
| P2 | ´04´ | *fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP |
| Data | ´XX…XX´ | *aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16] |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Selektieren per AID, next, keine Antwortdaten

In dieser Variante enthält die APDU des Select-Kommandos vier Parameter:

1. (N043.400) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N043.500) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´2´ gewählt werden.
3. (N043.600) K\_externeWelt {K\_Karte}   
   Der Parameter *aid* enthält einen Oktettstring gemäß (N010.200) oder dessen Anfang. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.
4. (N043.700) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
5. (N043.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 52 verwendet werden.

Tabelle 52: Select, AID, next occurrence, keine Antwortdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit applicationIdentifier |
| P2 | ´0E´ | *fileOccurrence* + *responseType* = next occurrence, keine Antwortdaten |
| Data | ´XX…XX´ | *aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16] |

#### Use Case Selektieren per AID, next, Antwortdaten mit FCP

In dieser Variante enthält die APDU des Select-Kommandos fünf Parameter:

1. (N043.900) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´04´ gewählt werden.
2. (N044.000) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´2´ gewählt werden.
3. (N044.100) K\_externeWelt {K\_Karte}   
   Der Parameter *aid* enthält einen Oktettstring gemäß (N010.200) oder dessen Anfang. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.
4. (N044.200) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
5. (N044.300) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
6. (N044.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 53 verwendet werden.

Tabelle 53: Select, AID, next occurrence, Antwortdaten mit FCP

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´04´ | *selectionMode* = Ordnerselektion mit applicationIdentifier |
| P2 | ´06´ | *fileOccurrence* + *responseType* = next occurrence,Antwortdaten mit FCP |
| Data | ´XX…XX´ | *aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16] |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Selektieren, DF oder ADF, keine Antwortdaten

In dieser Variante enthält die APDU des Select-Kommandos vier Parameter:

1. (N044.500) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´01´ gewählt werden.
2. (N044.600) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N044.700) K\_externeWelt {K\_Karte}   
   Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einem Ordner mit dazu passendem Attribut *fileIdentifier* gesucht.
4. (N044.800) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
5. (N044.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 54 verwendet werden.

Tabelle 54: Select, DF oder ADF mit FID, keine Antwortdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´ | *selectionMode* = Selektion eines Ordners mit fileIdentifier |
| P2 | ´0C´ | *fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten |
| Data | ´XXXX´ | *fid* |

#### Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP

In dieser Variante enthält die APDU des Select-Kommandos fünf Parameter:

1. (N045.000) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´01´ gewählt werden.
2. (N045.100) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N045.200) K\_externeWelt {K\_Karte}   
   Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einem Ordner mit dazu passendem Attribut *fileIdentifier* gesucht.
4. (N045.300) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
5. (N045.400) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
6. (N045.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 55 verwendet werden.

Tabelle 55: Select, DF oder ADF mit FID, Antwortdaten mit FCP

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´ | *selectionMode* = Ordnerselektion mit fileIdentifier |
| P2 | ´04´ | *fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP |
| Data | ´XXXX´ | *fid* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Selektieren übergeordnetes Verzeichnis ohne FCP

Diese Variante selektiert den eine Ebene höher liegenden Ordner (siehe (N020.000), (N020.100)). In dieser Variante enthält die APDU des Select-Kommandos zwei Parameter:

1. (N045.600) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´03´ gewählt werden.
2. (N045.700) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
3. (N045.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 56 verwendet werden.

Tabelle 56: Select, höhere Ebene keine Antwortdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´03´ | *selectionMode* = Selektion des eine Ebene höheren Ordners |
| P2 | ´0C´ | *responseType* = keine Antwortdaten |

#### Use Case Selektieren übergeordnetes Verzeichnis mit FCP

Diese Variante selektiert den eine Ebene höher liegenden Ordner (siehe (N020.000), (N020.100)). In dieser Variante enthält die APDU des Select-Kommandos drei Parameter:

1. (N045.900) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´03´ gewählt werden.
2. (N046.000) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
3. (N046.100) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
4. (N046.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 57 verwendet werden.

Tabelle 57: Select, höhere Ebene, Antwortdaten mit FCP

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´03´ | *selectionMode* = Selektion des eine Ebene höheren Ordners |
| P2 | ´04´ | *responseTyp*e = Antwortdaten mit FCP |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Selektieren einer Datei, keine Antwortdaten

In dieser Variante enthält die APDU des Select-Kommandos vier Parameter:

1. (N046.300) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´02´ gewählt werden.
2. (N046.400) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N046.500) K\_externeWelt {K\_Karte}   
   Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einer Datei mit dazu passendem Attribut *fileIdentifier* gesucht.
4. (N046.600) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´0C´ gewählt werden.
5. (N046.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 58 verwendet werden.

Tabelle 58: Select, EF mit FID, keine Antwortdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´02´ | *selectionMode* = Selektion einer Datei mit fileIdentifier |
| P2 | ´0C´ | *fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten |
| Data | ´XXXX´ | *fid* |

#### Use Case Selektieren einer Datei, Antwortdaten mit FCP

In dieser Variante enthält die APDU des Select-Kommandos fünf Parameter:

1. (N046.800) K\_externeWelt {K\_Karte}   
   Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ´02´ gewählt werden.
2. (N046.900) K\_externeWelt {K\_Karte}   
   Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = ´0´ gewählt werden.
3. (N047.000) K\_externeWelt {K\_Karte}   
   Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einer Datei mit dazu passendem Attribut *fileIdentifier* gesucht.
4. (N047.100) K\_externeWelt {K\_Karte}   
   Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ´04´ gewählt werden.
5. (N047.200) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
6. (N047.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 59 verwendet werden.

Tabelle 59: Select, EF mit FID, Antwortdaten mit FCP

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´02´ | *selectionMode* = Selektion einer Datei mit fileIdentifier |
| P2 | ´04´ | *fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP |
| Data | ´XXXX´ | *fid* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Zusammenfassung der Select-Kommando-Varianten

Wegen der Vielzahl an Varianten für dieses Kommando werden hier alle auf einen Blick dargestellt. Es sei darauf hingewiesen, dass nicht alle Kombinationen der folgenden Tabelle in den vorangegangenen Kapiteln enthalten sind. Deshalb sind solche nicht zwingend zu unterstützen.

Tabelle 60: Select, Kommandoparameter im Überblick

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A4´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´  ´02´  ´03´  ´04´ | Selektion eines Ordners mit *fileIdentifier*  Selektion einer Datei mit *fileIdentifier*  Selektion des eine Ebene höher liegenden Ordners  Selektion eines Ordners mit (möglicherweise leerem) *applicationIdentifier* |
| P2 | ´04´  ´06´  ´0C´  ´0E´ | first occurrence, Antwortdaten mit FCP  next occurrence, Antwortdaten mit FCP  first occurrence, keine Antwortdaten  next occurrence, keine Antwortdaten |
| Data | ´XX…XX´ | P1 = ´01´: zwei Byte *fid*  P1 = ´02´: zwei Byte *fid*  P1 = ´03´: abwesend  P1 = ´04´: abwesend, oder bis zu 16 Oktette *aid* |
| Le | *length* | P2 = ´04´: Anzahl der erwarteten Oktette in den Antwortdaten  P2 = ´06´: Anzahl der erwarteten Oktette in den Antwortdaten  P2 = ´0C´: abwesend  P2 = ´0E´: abwesend |

#### Antwort der Karte auf Selektieren eines Files

Tabelle 61: Select Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | FCP | Abwesend, oder File Control Parameter |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 83´ | FileDeactivated | Selektiertes File ist logisch oder physikalisch deaktiviert |
| ´62 85´ | FileTerminated | Selektiertes File ist logisch oder physikalisch terminiert |
| ´90 00´ | NoError | Erfolgreiche Selektion eines Files |

Tabelle 62: Select Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalte** | **Beschreibung** |
| ´6A 82´ | FileNotFound | Zu selektierendes File wurde nicht gefunden |
| ´6D 00´ | InstructionNotSupported | Die Karte befindet sich im Zustand „Termination state" |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N047.400) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

Zur Beschreibung der Kommandobearbeitung werden folgende Definitionen eingeführt:

|  |  |
| --- | --- |
| oldFolder | Dies ist die Bezeichnung von *currentFolder* vor Ausführung dieses  Select-Kommandos |
| newFile | Dies ist die Bezeichnung für das File (Ordner oder Datei), welches  im Rahmen der Selektion ausgewählt wird |
| path(*folder*) | Pfad eines Ordners mit Namen *folder*. Zum Pfad gehören nach dieser Definition sowohl der Ordner *folder* als auch alle seine übergeordneten Verzeichnisse einschließlich *root*. |

1. (N047.500) K\_COS
   1. Das COS MUSS die Select-Varianten aus 14.2.6.1, 14.2.6.2, 14.2.6.5, 14.2.6.6, 14.2.6.9, 14.2.6.10, 14.2.6.11, 14.2.6.12, 14.2.6.13 und 14.2.6.14 unterstützen.
   2. Das COS KANN weitere Select-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N047.590) K\_COS   
   Falls das Attribut *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) den Wert „Termination state" besitzt, genau dann MUSS das Kommando mit dem Trailer InstructionNotSupported terminieren.
3. (N047.600) K\_COS   
   Zugriffsregeln für das Select Kommando:
   1. Das COS KANN die Auswertung von Zugriffsregeln für das Select Kommando unterstützen.
   2. Das COS KANN als Zugriffsbedingung für das Select Kommando stets ALWAYS verwenden.
4. (N047.700) K\_COS   
   Wenn der Parameter P1 den Wert ´01´ besitzt, dann MUSS in *oldFolder.children* nach einem Ordner gesucht werden, der ein Attribut *fileIdentifier* besitzt, dessen Wert mit dem Parameter *fid* aus den Kommandodaten übereinstimmt. Wenn ein solcher Ordner existiert, dann MUSS *newFile* auf den gefundenen Ordner gesetzt werden. Andernfalls ist die Suche erfolglos.
5. (N047.800) K\_COS   
   Wenn der Parameter P1 den Wert ´02´ besitzt, dann MUSS in *oldFolder.children* nach einer Datei gesucht werden, deren Attribut *fileIdentifier* mit dem Parameter *fid* aus den Kommandodaten übereinstimmt. Wenn eine solche Datei existiert, dann MUSS *newFile* auf die gefundene Datei gesetzt werden. Andernfalls ist die Suche erfolglos.
6. (N047.900) K\_COS   
   Wenn der Parameter P1 den Wert ´03´ besitzt, und *currentFolder*
   1. ist gleich *root*, dann MUSS das Kommando mit dem Trailer FileNotFound terminieren.
   2. ist nicht *root*, dann ist *newFile* der im Vergleich zu *currentFolder* eine Ebene höher liegende Ordner (siehe (N020.000), (N020.100)).
7. (N048.000) K\_COS   
   Wenn der Parameter P1 den Wert ´04´ besitzt, dann enthält die Kommandonachricht einen (möglicherweise leeren) Parameter *aid*.
   1. Im gesamten Objektsystem wird nach Ordnern gesucht, die ein Attribut *applicationIdentifier* besitzen, welches zu *aid* passt. Ein Attribut *applicationIdentifier*
      1. MUSS als passend betrachtet werden, wenn *applicationIdentifier* identisch zu *aid* ist.
      2. MUSS als passend betrachtet werden, wenn *aid* leer ist.
      3. dessen Most Significant Bytes identisch sind zu *aid*
         1. KANN als passend betrachtet werden.
         2. KANN als unpassend betrachtet werden.
   2. Alle passenden Ordner werden zu einer Liste zusammengestellt.
      1. Ist *root* in dieser Liste enthalten, dann MUSS es das erste Listenelement sein.
      2. Solange das Objektsystem nicht verändert wird (Delete oder Load Application) MUSS bei identischem *aid* stets dieselbe Liste erstellt werden.
   3. Hat P2 einen Wert aus der Menge {´04´, ´0C´}, dann MUSS *newFile* auf das erste Listenelement gesetzt werden.
   4. Hat P2 einen Wert aus der Menge {´06´, ´0E´} und
      1. *oldFolder* ist ebenfalls in der Liste und *oldFolder* ist
         1. nicht das letzte Listenelement, dann MUSS *newFile* auf das nächste Listenelement gesetzt werden.
         2. das letzte Listenelement, dann MUSS die Suche erfolglos sein.
      2. *oldFolder* ist nicht in der Liste enthalten, dann
         1. KANN *newFile* auf irgendein Listenelement gesetzt werden, oder
         2. die Suche KANN erfolglos sein.
8. (N048.100) K\_COS   
   Wenn die Suche erfolglos war, genau dann MUSS das Kommando mit dem Trailer FileNotFound terminieren. Dabei DÜRFEN *currentFolder*, *currentEF* sowie der Kanalkontext (siehe Kapitel 12) NICHT verändert werden.
9. (N048.200) K\_COS   
   Wenn
   1. *newFile* eine Datei ist, dann MUSS *currentEF* auf *newFile* gesetzt werden.
   2. *newFile* ein Ordner ist, dann MUSS
      1. *currentFolder* gleich *newFile* gesetzt werden.
      2. *currentEF* auf den Wert „undefiniert“ gesetzt werden.
      3. in allen Ordnern, die sowohl zu path(*oldFolder*) als auch zu path(*newFile*) gehören, *seIdentifier* unverändert bleiben.
      4. in allen anderen Ordnern *seIdentifier* auf den Wert 1 gesetzt werden.
      5. die Funktion clearSecurityStatusFolder( *folder* ) für alle Ordner ausgeführt werden, die nicht zu path(*newFile*) gehören.
      6. Option\_Kryptobox: die Funktion clearSessionkeys( ) ausgeführt werden, falls *SessionkeyContext*.*folderSessionkeys* nicht zu path(*newFile*) gehört.
      7. aus *dfSpecificPasswordList* MÜSSEN mittels clearPasswordStatus(…) alle Einträge entfernt werden, die nicht zu path(*newFile*) gehören.
      8. jedes Element von *keyReferenceList*, welches ein Schlüsselobjekt referenziert, das zu einem Ordner außerhalb von path(*newFile*) gehört, auf den Wert „leer“ gesetzt werden.
10. (N048.300) K\_COS   
    Für das Datenfeld *rspData* der Antwortnachricht gilt:
    1. Wenn P2 einen Wert aus der Menge {´04´, ´06´} hat, genau dann MUSS das Datenfeld *rspData* der Antwortnachricht die File Control Parameter gemäß 8.3.3 wie folgt enthalten: Sei FCP ein Oktettstring, der die File Control Parameter gemäß 8.3.3 enthält, dann gilt: Falls OctetLength( FCP )
       1. kleiner Nr gemäß (N027.200): *rspData* = FCP.
       2. sonst *rspData* = Extract\_MSByte( FCP, Nr ).
    2. Andernfalls fehlt das Datenfeld der Antwortnachricht.
11. (N048.400) K\_COS   
    Wenn der logische Wert von *newFile.lifeCycleStatus* (siehe (N020.600)) den Wert
    1. „Operational state (deactivated)“ hat, genau dann MUSS als Trailer FileDeactivated gewählt werden.
    2. „Termination state" hat, genau dann MUSS als Trailer FileTerminated gewählt werden.
12. (N048.500) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
13. (N048.600) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 62 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 62 MUSS eine höhere Priorität als FileDeactivated haben.
    3. FileDeactivated MUSS eine höhere Priorität als NoError haben.
14. Gemäß den Regeln dieses Dokumentes ist es zulässig, in deaktivierten oder terminierten Ordnern Unterordner oder Dateien mittels Select-Kommando zu selektieren.

### Terminate Card Usage

Das Kommando Terminate Card Usage überführt eine Karte irreversibel in den Zustand „Termination state". Das Kommando ist unabhängig vom aktuellen Wert von *currentFolder* und *currentEF* ausführbar. Während der Kommandoausführung wird im Objektsystem das Attribut *lifeCycleStatus* auf den Wert „Termination state" gesetzt und, mit Ausnahme des Basiskanals, werden alle weiteren logischen Kanäle geschlossen. Anschließend ist die Funktionalität des Select-Kommandos nicht mehr verfügbar (siehe (N047.590)).

#### Use Case Terminieren der Karte

In dieser Variante enthält die APDU des Terminate Card Usage-Kommandos keinen Parameter:

1. (N048.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 63 verwendet werden.

Tabelle 63: Terminate Card Usage

|  |  |  |
| --- | --- | --- |
|  |  | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´FE´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | – |
| P2 | ´00´ | – |

#### Antwort der Karte auf terminieren der Karte

Tabelle 64: Terminate Card Usage Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Terminierung |

Tabelle 65: Terminate Card Usage Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N048.738) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N048.740) K\_COS
   1. Das COS MUSS die Terminate Card Usage-Variante aus 14.2.7.1 unterstützen.
   2. Das COS KANN weitere Terminate Card Usage-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N048.742) K\_COS   
   *affectedObject* MUSS gleich *root* gesetzt werden.
3. (N048.744) K\_COS   
   Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N048.748) K\_COS   
   Falls die Kommandos-APDU dieses Terminate Card Usage-Kommandos gemäß 13.2 mittels Sessionkeys gesichert war, dann MUSS die korrespondierende Antwort-APDU gemäß 13.3 mit denselben Sessionkeys gesichert sein, auch wenn als Seiteneffekt von (N048.756)b oder (N048.756)c Sessionkeys gelöscht werden.
5. (N048.752) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* des Objektsystems den Wert „Termination state" besitzt, dann MUSS als Trailer NoError verwendet werden.
6. (N048.756) K\_COS   
   Es werden folgende Aktionen ausgeführt:
   1. Der physikalische Wert von *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) MUSS mittels Transaktionsschutz auf den Wert „Termination state" gesetzt werden.
   2. Bis auf den Basiskanal MÜSSEN alle anderen logischen Kanäle geschlossen werden.
   3. Im Basiskanal MUSS *channelContext* gemäß (N030.100) gesetzt werden.
7. (N048.760) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
8. (N048.762) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N048.764) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
10. (N048.766) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 65 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 65 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### Terminate DF

Das Kommando Terminate DF überführt einen Ordner irreversibel in den Zustand „Termination state". Der betroffene Ordner wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses Terminate DF-Kommandos durch eine Select-Operation (Select-Kommandos).

#### Use Case Terminieren eines Ordners

In dieser Variante enthält die APDU des Terminate DF-Kommandos keinen Parameter:

1. (N048.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 66 verwendet werden.

Tabelle 66: Terminate DF aktueller Ordner

|  |  | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E6´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | – |
| P2 | ´00´ | – |

#### Antwort der Karte auf terminieren der Karte

Tabelle 67: Terminate DF Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Terminierung |

Tabelle 68: Terminate DF Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N048.838) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N048.840) K\_COS
   1. Das COS MUSS die Terminate DF-Variante aus 14.2.8.1 unterstützen.
   2. Das COS KANN weitere Terminate DF-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N048.842) K\_COS   
   *affectedObject* MUSS gleich *currentFolder* gesetzt werden.
3. (N048.844) K\_COS   
   Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N048.852) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert „Termination state" besitzt, dann MUSS als Trailer NoError verwendet werden.
5. (N048.856) K\_COS   
   Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Termination state" gesetzt werden.
6. (N048.860) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
7. (N048.862) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
8. (N048.864) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
9. (N048.866) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 68 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 68 MUSS eine höhere Priorität als UpdateRetryWarning haben.
   3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### Terminate

Das Kommando Terminate überführt eine Datei, ein Schlüsselobjekt oder ein Passwortobjekt irreversibel in den Zustand „Termination state". Eine betroffene Datei wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses Terminate-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*). Falls ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist.

#### Use Case Terminieren einer Datei

In dieser Variante wird eine Datei terminiert und die APDU des Terminate-Kommandos enthält einen Parameter.

1. (N048.900) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass *currentEF* zu terminieren ist.
2. (N048.903) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 69 verwendet werden.

Tabelle 69: Terminate aktuelle Datei

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E8´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | *mode*, der Wert ´00´ zeigt an, dass *currentEF* terminiert wird |
| P2 | ´00´ | – |

#### Use Case Terminieren eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt terminiert und die APDU des Terminate-Kommandos enthält zwei Parameter:

1. (N048.910) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu terminieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.
2. (N048.912) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
3. (N048.914) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 70 verwendet werden.

Tabelle 70: Terminate privates oder symmetrisches Schlüsselobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E8´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´20´ | *mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Schlüsselobjekt |

#### Use Case Terminieren eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt terminiert und die APDU des Terminate-Kommandos enthält zwei Parameter:

1. (N048.920) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu terminieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.
2. (N048.922) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt.
3. (N048.924) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 71 verwendet werden.

Tabelle 71: Terminate öffentliches Schlüsselobjekt

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E8´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´21´ | *mode*, hier: Schlüsselreferenz im Datenteil |
| P2 | ´00´ | ‑ |
| Data | ´XX…XX´ | ´83 –I2OS(OctetLength(*reference*), 1) – *reference*´ |

#### Use Case Terminieren eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt terminiert und die APDU des Terminate-Kommandos enthält zwei Parameter:

1. (N048.930) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu terminieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.
2. (N048.932) K\_externeWelt {K\_Karte}   
   Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.
3. (N048.934) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 72 verwendet werden.

Tabelle 72: Terminate Passwortobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E8´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´10´ | *mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2 |
| P2 | ´XX´ | *reference* auf ein Passwortobjekt |

#### Antwort der Karte auf terminieren von Datei, Schlüssel- oder Passwortobjekt

Tabelle 73: Terminate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Terminierung |

Tabelle 74: Terminate Antwort-APDU im Fehlerfall

|  |  |  |  |
| --- | --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** | |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich | |
| ´69 81´ | VolatileKeyWithoutLCS | volatile Schlüssel vom Kommando nicht unterstützt | |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt | |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt | |
| ´6A 88´ | KeyNotFound | Schlüsselobjekt nicht gefunden | |
| ´6A 88´ | PasswordNotFound | | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N048.948) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N048.950) K\_COS
   1. Das COS MUSS die Terminate-Varianten aus 14.2.9.1, 14.2.9.2, 14.2.9.3 und 14.2.9.4 unterstützen.
   2. Das COS KANN weitere Terminate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N048.954) K\_COS   
   Falls der Parameter *mode* in der Kommandonachricht den Wert
   1. ´00´ besitzt und *channelContext*.*currentEF* (siehe (N029.900)m)
      1. unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
   2. ´20´ oder ´21´ besitzt, dann gilt *affectedObject* = SearchKey(   
          *channelContext*.*currentFolder,*   
          *reference*,   
         “WildCard”   
      ). Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
   3. ´10´ besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N048.955) K\_COS   
   Falls *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.
4. (N048.957) K\_COS   
   Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
5. (N048.960) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert „Termination state" besitzt, dann MUSS als Trailer NoError verwendet werden.
6. (N048.963) K\_COS   
   Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Termination state" gesetzt werden.
7. (N048.966) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
8. (N048.969) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N048.972) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
10. (N048.975) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 74 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 74 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

## Zugriff auf Daten in transparenten EF

Es ist möglich auf Daten im *body* eines transparenten EF lesend (Read Binary) oder schreibend (Update Binary) zuzugreifen.

Die Kommandos in diesem Unterkapitel unterstützen zwei Varianten:

1. Variante ohne *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass für eine erfolgreiche Kommandoabarbeitung *currentEF* notwendigerweise ein transparentes EF ist. Die Variable *currentEF* lässt sich unter anderem durch gewisse Varianten des Select-Kommandos setzen.
2. Variante mit *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass das vom Kommando betroffene EF erst während der Kommandoabarbeitung gesetzt wird. Ein vorausgehendes Select-Kommando ist also nicht notwendig. In der Variante mit *shortFileIdentifier* lassen sich keine Dateien addressieren, die nicht *currentFolder* zugeordnet sind.

### Erase Binary

Das Kommando Erase Binary ersetzt bereits vorhandene Daten im *body* eines transparenten EF durch Oktette mit dem Wert ´00´. Das betroffene transparente EF wird vor der Löschoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Erase Binary-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Erase Binary-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Informationen im *body* gelöscht werden, bestimmt der Offset, der als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Löschen ohne *shortFileIdentifier* in transparenten EF

In dieser Variante werden Daten in einem transparenten EF gelöscht, ohne das Attribut *positionLogicalEndOfFile* zu verändern. Die APDU des Erase Binary-Kommandos enthält einen Parameter:

1. (N049.000) K\_externeWelt {K\_Karte}   
   Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [`0000´, ´7FFF´] sein (vergleiche (N011.500)).
2. (N049.100) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 75 verwendet werden.

Tabelle 75: Erase Binary, logical EOF unverändert, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4], *positionLogicalEndOfFile* konstant |
| INS | ´0E´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | (*offset* – P2) / 256, MSByte von *offset* |
| P2 | ´XX´ | *offset* mod 256, LSByte von *offset* |

#### Use Case Löschen mit *shortFileIdentifier* in transparenten EF

In dieser Variante werden Daten in einem transparenten EF gelöscht, ohne das Attribut *positionLogicalEndOfFile* zu verändern. Die APDU des Erase Binary-Kommandos enthält zwei Parameter:

1. (N049.200) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N049.300) K\_externeWelt {K\_Karte}   
   Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = [´00´, ´FF´] sein.
3. (N049.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 76 verwendet werden.

Tabelle 76: Erase Binary, logical EOF unverändert, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4], *positionLogicalEndOfFile* konstant |
| INS | ´0E´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | 128 + *shortFileIdentifier*, das heißt ´80´ + *shortFileIdentifier* |
| P2 | ´XX´ | *offset* |

#### Antwort der Karte auf Löschen in transparenten EF

Tabelle 77: Erase Binary Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreicher Löschvorgang |

Tabelle 78: Erase Binary Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht transparent |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6B 00´ | OffsetTooBig | Parameter *offset* in Kommando–APDU ist zu groß |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N049.500) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N049.600) K\_COS
   1. Das COS MUSS die Erase Binary-Varianten aus 14.3.1.1 und 14.3.1.2 unterstützen.
   2. Das COS KANN weitere Erase Binary-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N049.700) K\_COS   
   Falls die APDU des Erase Binary-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N049.800) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N049.900) K\_COS   
   Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N050.000) K\_COS   
   Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.
6. (N050.100) K\_COS   
   Falls *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MUSS *affectedObject.body* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen, sofern diese vorhanden ist.
   2. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
7. (N050.190) K\_COS   
   Falls *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.body*
   1. DARF NICHT zum Abbruch des Kommandos führen.
   2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
8. (N050.200) K\_COS   
   Falls *offset* kleiner als *affectedObject*.*positionLogicalEndOfFile* ist dann
   1. MÜSSEN die durch *offset* gekennzeichnete Stelle und alle weiteren Oktette in *affectedObject.body* auf den Wert ´00´ gesetzt werden.
   2. MUSS, falls *affectedObject.body* durch eine Checksumme geschützt ist, diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.
9. (N050.300) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
10. (N050.400) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
11. (N050.500) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
12. (N050.600) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 78 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 78 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
13. (N050.700) Diese Anforderung ist absichtlich leer.

(N050.800) Diese Anforderung ist absichtlich leer.

1. Es ist aus funktionaler Sicht irrelevant, ob die in (N050.200) beschriebene erase-Funktion an der Stelle positionLogicalEndOfFile stoppt, oder bis zum Ende von affectedObject.body weiterarbeitet, weil sich diese beiden Fälle an der Kartenschnittstelle nicht unterscheiden.

### Read Binary

Das Kommando Read Binary dient dem Auslesen von Informationen aus dem *body* eines transparenten EF. Deshalb enthält das Datenfeld der Antwortnachricht (Teile von) *body*. Das betroffene transparente EF wird vor der Leseoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Read Binary-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Read Binary-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Informationen aus *body* ausgelesen werden, bestimmen Offset und Länge, die als Parameter in der Kommandonachricht enthalten sind.

#### Use Case Lesen ohne *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des Read Binary-Kommandos zwei Parameter:

1. (N050.900) K\_externeWelt {K\_Karte}   
   Der Parameter *offset* bestimmt, ab welcher Position gelesen wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [´0000´, ´7FFF´] sein (vergleiche (N011.500)).
2. (N051.000) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
3. (N051.100) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 79 verwendet werden.

Tabelle 79: Read Binary ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´B0´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | (*offset* – P2) / 256, MSByte von *offset* |
| P2 | ´XX´ | *offset* mod 256, LSByte von *offset* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Lesen mit *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des Read Binary-Kommandos drei Parameter:

1. (N051.200) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N051.300) K\_externeWelt {K\_Karte}   
   Der Parameter *offset* bestimmt, ab welcher Position gelesen wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = [´00´, ´FF´] sein.
3. (N051.400) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
4. (N051.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 80 verwendet werden.

Tabelle 80: Read Binary mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´B0´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | 128 + *shortFileIdentifier*, das heißt ´80´ + *shortFileIdentifier* |
| P2 | ´XX´ | *offset* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Lesen in transparenten EF

Tabelle 81: Read Binary Antwort-APDU im Erfolgsfall

| **Daten** | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| ´xx…xx´ |  | Ausgelesene Daten |
| Trailer | Inhalt | Beschreibung |
| ´62 81´ | CorruptDataWarning | Möglicherweise sind die Antwortdaten korrupt |
| ´62 82´ | EndOfFileWarning | Weniger Daten vorhanden, als mittels Ne angefordert |
| ´90 00´ | NoError | Erfolgreiche Leseoperation |

Tabelle 82: Read Binary Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht transparent |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6B 00´ | OffsetTooBig | Parameter *offset* in Kommando-APDU ist zu groß |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N051.600) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N051.700) K\_COS
   1. Das COS MUSS die Read Binary-Varianten aus 14.3.2.1 und 14.3.2.2 unterstützen.
   2. Das COS KANN weitere Read Binary-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N051.800) K\_COS   
   Falls die APDU des Read Binary-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N051.900) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N052.000) K\_COS   
   Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N052.100) K\_COS   
   Wenn *offset* größer oder gleich *affectedObject.positionLogicalEndOfFile* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.
6. (N052.200) K\_COS   
   Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten von *affectedObject.body* inkonsistent zur Checksumme sind, genau dann MUSS
   1. entweder als Trailer CorruptDataWarning gewählt werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
7. (N052.300) K\_COS   
   Wenn das LeFeld der Kommando-APDU keine WildCard enthält und (*offset* + *length*) größer als *affectedObject.positionLogicalEndOfFile* ist, genau dann MUSS als Trailer EndOfFileWarning gewählt werden.
8. (N052.400) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
9. (N052.500) K\_COS   
   Für das Datenfeld der Antwortnachricht gilt:
   1. Aus dem Oktettstring von *affectedObject.body* MÜSSEN die durch *offset* gekennzeichnete Position und die nachfolgenden Oktette übernommen werden.
   2. Es DÜRFEN NICHT mehr Oktette übernommen werden, als durch Ne angegeben.
   3. Die Übernahme der Oktette MUSS so gestoppt werden, dass weder das Oktett an der Position *positionLogicalEndOfFile* noch nachfolgende Oktette übernommen werden.
10. (N052.600) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 82 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 82 MUSS eine höhere Priorität als CorruptDataWarning haben.
    3. CorruptDataWarning MUSS eine höhere Priorität als EndOfFileWarning haben.
    4. EndOfFileWarning MUSS eine höhere Priorität als NoError haben.
11. (N052.700) Diese Anforderung ist absichtlich leer.

(N052.800) Diese Anforderung ist absichtlich leer.

### Search Binary

1. (N052.900) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]
   1. unterstützen oder
   2. ablehnen.

### Set Logical Eof

1. Dieses Kommando ist nicht in der Normenreihe ISO/IEC 7816 enthalten. Wegen ähnlicher Funktionalität teilt sich dieses Kommando das INS Byte mit Erase Binary.

Das Kommando Set Logical Eof verändert den Wert des Attributes *positionLogicalEndOfFile* eines transparenten EF, wodurch Daten gelöscht werden. Das betroffene transparente EF wird vor der Löschoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Set Logical Eof-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Set Logical Eof-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Auf welchen Wert das Attribut *positionLogicalEndOfFile* gesetzt wird und damit welche Daten in *body* gelöscht werden, bestimmt der Offset, der als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Setzen logical EOF ohne *shortFileIdentifier*

In dieser Variante werden Daten in einem transparenten EF gelöscht und das Attribut *positionLogicalEndOfFile* wird verändert.

1. (N052.930) K\_externeWelt {K\_Karte}   
   Die APDU des Set Logical Eof-Kommandos enthält einen Parameter:
   1. Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [`0000´, ´7FFF´] sein (vergleiche (N011.500)).
2. (N052.932) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 83 verwendet werden.

Tabelle 83: Set Logical Eof ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´0E´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | (*offset* – P2) / 256, MSByte von *offset* |
| P2 | ´XX´ | *offset* mod 256, LSByte von *offset* |

#### Use Case Setzen logical EOF mit *shortFileIdentifier*

In dieser Variante werden Daten in einem transparenten EF gelöscht und das Attribut *positionLogicalEndOfFile* wird verändert.

1. (N052.934) K\_externeWelt {K\_Karte}   
   Die APDU des Set Logical Eof-Kommandos enthält zwei Parameter:
   1. Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
   2. Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = [´00´, ´FF´] sein.
2. (N052.936) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 84 verwendet werden.

Tabelle 84: Set Logical Eof mit *shortFileIdentifier*

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´0E´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | 128 + *shortFileIdentifier*, das heißt ´80´ + *shortFileIdentifier* |
| P2 | ´XX´ | *offset* |

#### Antwort der Karte auf Setzen logical EOF in transparenten EF

Tabelle 85: Set Logical Eof Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreicher Löschvorgang |

Tabelle 86: Set Logical Eof Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht transparent |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6B 00´ | OffsetTooBig | Parameter *offset* in Kommando–APDU ist zu groß |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N052.938) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N052.940) K\_COS
   1. Das COS MUSS die Set Logical Eof-Varianten aus 14.3.4.1 und 14.3.4.2 unterstützen.
   2. Das COS KANN weitere Set Logical Eof-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N052.942) K\_COS   
   Falls die APDU des Set Logical Eof-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N052.944) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N052.946) K\_COS   
   Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N052.948) K\_COS   
   Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.
6. (N052.950) K\_COS   
   Wenn *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MUSS *affectedObject.body* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen, sofern diese vorhanden ist.
   2. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
7. (N052.952) K\_COS   
   Falls *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.body*
   1. DARF NICHT zum Abbruch des Kommandos führen.
   2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
8. (N052.954) K\_COS   
   Falls *offset*
   1. kleiner als *affectedObject*.*positionLogicalEndOfFile* ist dann
      1. MUSS, falls *affectedObject.body* durch eine Checksumme geschützt ist, diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.
      2. MUSS *positionLogicalEndOfFile* auf den Wert von *offset* geändert werden.   
         *Hinweis: Aus Sicherheitsgründen erscheint es sinnvoll, die durch* offset *gekennzeichnete Stelle und alle weiteren Oktette in* affectedObject.body *auf den Wert ´00´ zu setzten. Ob so eine Aktion erfolgt, ist mit den normativen Mitteln aus diesem Dokument nicht testbar, vergleiche auch (N054.500)a und (N055.258).*
   2. größer gleich *positionLogicalEndOfFile* ist, dann DARF *positionLogicalEndOfFile* NICHT geändert werden.
9. (N052.956) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
10. (N052.958) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
11. (N052.960) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
12. (N052.962) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 86 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 86 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
13. Es ist aus funktionaler Sicht irrelevant, ob die in (N052.954) beschriebene erase-Funktion an der Stelle positionLogicalEndOfFile stoppt, oder bis zum Ende von affectedObject.body weiterarbeitet, weil sich diese beiden Fälle an der Kartenschnittstelle nicht unterscheiden.

### Update Binary

Das Kommando Update Binary ersetzt Daten im *body* eines transparenten EF durch Daten, die im Datenfeld der Kommandonachricht enthalten sind. Das betroffene transparente EF wird vor der Schreiboperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Update Binary-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Update Binary-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Informationen im *body* geändert werden, bestimmen Offset und Schreibdaten, die als Parameter in der Kommandonachricht enthalten sind.

#### Use Case Schreiben ohne *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des Update Binary-Kommandos zwei Parameter:

1. (N053.000) K\_externeWelt {K\_Karte}   
   Der Parameter *offset* bestimmt, ab welcher Position geschrieben wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [`0000´, ´7FFF´] sein (vergleiche (N011.500)).
2. (N053.100) K\_externeWelt {K\_Karte}   
   Der Parameter *newData* enthält die neuen Daten, welche die, ab der durch *offset* gekennzeichneten Stelle, enthaltenen Daten in *body* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
3. (N053.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 87 verwendet werden.

Tabelle 87: Update Binary ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´D6´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | (*offset* – P2) / 256, MSByte von *offset* |
| P2 | ´XX´ | *offset* mod 256, LSByte von *offset* |
| Data | ´XX…XX´ | *newData* |

#### Use Case Schreiben mit *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des Update Binary-Kommandos drei Parameter:

1. (N053.300) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N053.400) K\_externeWelt {K\_Karte}   
   Der Parameter *offset* bestimmt, ab welcher Position geschrieben wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = [´00´, ´FF´] sein.
3. (N053.500) K\_externeWelt {K\_Karte}   
   Der Parameter *newData* enthält die neuen Daten, welche die ab der durch *offset* gekennzeichneten Stelle enthaltenen Daten in *body* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
4. (N053.600) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 88 verwendet werden.

Tabelle 88: Update Binary mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´D6´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | 128 + *shortFileIdentifier*, das heißt ´80´ + *shortFileIdentifier* |
| P2 | ´XX´ | *offset* |
| Data | ´XX…XX´ | *newData* |

#### Antwort der Karte auf Schreiben in transparenten EF

Tabelle 89: Update Binary Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreicher Schreibvorgang |

Tabelle 90: Update Binary Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht transparent |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per shortFileIdentifier adressiertes EF nicht gefunden |
| ´6A 84´ | DataTooBig | Parameter newData ragt über das Dateiende hinaus |
| ´6B 00´ | OffsetTooBig | Parameter offset in Kommando–APDU ist zu groß |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N053.700) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N053.800) K\_COS
   1. Das COS MUSS die Update Binary-Varianten aus 14.3.5.1 und 14.3.5.2 unterstützen.
   2. Das COS KANN weitere Update Binary-Varianten
      1. unterstützen
      2. ablehnen.
2. (N053.900) K\_COS   
   Falls die APDU des Update Binary-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N054.000) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N054.100) K\_COS   
   Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N054.200) K\_COS   
   Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.
6. (N054.300) K\_COS   
   Wenn (*offset* + OctetLength(*newData*)) größer als *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer DataTooBig terminieren.
7. (N054.400) K\_COS   
   Wenn *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MÜSSEN *affectedObject.body* und gegebenenfalls *affectedObject.positionLogicalEndOfFile* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen.
   2. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
8. (N054.490) K\_COS   
   Falls *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
   1. DARF NICHT zum Kommandoabbruch führen.
   2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N054.500) K\_COS
   1. Falls *affectedObject.positionLogicalEndOfFile* kleiner als *offset* ist, dann MUSS das Oktett an der Position *positionLogicalEndOfFile* und alle nachfolgenden Oktette bis zur Position (*offset* – 1) auf den Wert ´00´ gesetzt werden.
   2. Die durch *offset* gekennzeichnete Stelle in *affectedObject.body* und die folgenden Oktette MÜSSEN durch *newData* ersetzt werden.
   3. Falls *positionLogicalEndOfFile* kleiner als (*offset* + OctetLength(*newData*)) ist, dann MUSS *positionLogicalEndOfFile* = (*offset* + OctetLength(*newData*)) gesetzt werden.
   4. Falls *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.
10. (N054.600) K\_COS   
    Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
11. (N054.700) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
12. (N054.800) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
13. (N054.900) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 90 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 90 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
14. (N055.000) Diese Anforderung ist absichtlich leer.

(N055.100) Diese Anforderung ist absichtlich leer.

### Write Binary

Das Kommando Write Binary fügt den vorhandenen Daten im *body* eines transparenten EF Daten hinzu, die im Datenfeld der Kommandonachricht enthalten sind. Das betroffene transparente EF wird vor der Schreiboperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Write Binary-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Write Binary-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Wieviele Daten *body* hinzugefügt werden, bestimmen Schreibdaten, die als Parameter in der Kommandonachricht enthalten sind.

#### Use Case Anfügen ohne *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des Write Binary-Kommandos einen Parameter:

1. (N055.200) K\_externeWelt {K\_Karte}   
   Der Parameter *newData* enthält die neuen Daten, welche die, ab der durch *positionLogicalEndOfFile* gekennzeichneten Stelle, enthaltenen Daten in *body* ergänzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
2. (N055.205) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 91 verwendet werden.

Tabelle 91: Write Binary ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´D0´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Bit b8 = 0 🡺 die Kommando APDU enthält keinen *shortFileIdentifier* |
| P2 | ´00´ | fester Wert |
| Data | ´XX…XX´ | *newData* |

#### Use Case Anfügen mit *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des Write Binary-Kommandos zwei Parameter:

1. (N055.220) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N055.223) K\_externeWelt {K\_Karte}   
   Der Parameter *newData* enthält die neuen Daten, welche die, ab der durch *positionLogicalEndOfFile* gekennzeichneten Stelle, enthaltenen Daten in *body* ergänzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
3. (N055.226) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 92 verwendet werden.

Tabelle 92: Write Binary mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´D0´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | 128 + *shortFileIdentifier*, das heißt ´80´ + *shortFileIdentifier* |
| P2 | ´00´ | fester Wert |
| Data | ´XX…XX´ | *newData* |

#### Antwort der Karte auf Anfügen in transparenten EF

Tabelle 93: Write Binary Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreicher Schreibvorgang |

Tabelle 94: Write Binary Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht transparent |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per shortFileIdentifier adressiertes EF nicht gefunden |
| ´6A 84´ | DataTooBig | Parameter newData ragt über das Dateiende hinaus |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N055.240) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N055.244) K\_COS
   1. Das COS MUSS die Write Binary-Varianten aus 14.3.6.1 und 14.3.6.2 unterstützen.
   2. Das COS KANN weitere Write Binary-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N055.246) K\_COS   
   Falls die APDU des Write Binary-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N055.248) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N055.250) K\_COS   
   Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N055.252) K\_COS   
   Wenn (*positionLogicalEndOfFile* + OctetLength(*newData*)) größer als *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer DataTooBig terminieren.
6. (N055.254) K\_COS   
   Wenn *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MÜSSEN *affectedObject.body* und *affectedObject.positionLogicalEndOfFile* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen.
   2. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
7. (N055.256) K\_COS   
   Falls *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
   1. DARF NICHT zum Kommandoabbruch führen.
   2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
8. (N055.258) K\_COS
   1. Die durch *positionLogicalEndOfFile* gekennzeichnete Stelle in *affectedObject.body* und die folgenden Oktette MÜSSEN durch *newData* ersetzt werden.
   2. Das Attribut *positionLogicalEndOfFile* MUSS um OctetLength(*newData*) inkrementiert werden.
   3. Falls *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.
9. (N055.260) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
10. (N055.262) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
11. (N055.264) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
12. (N055.266) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 94 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 94 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
13. (N055.268) Diese Anforderung ist absichtlich leer.

(N055.270) Diese Anforderung ist absichtlich leer.

## Zugriff auf strukturierte Daten

Es ist möglich auf Listenelemente von *recordList* in strukturierten EF lesend (Read Record) oder schreibend (Update Record) zuzugreifen. Zudem lassen sich neue Listenelemente anlegen (Append Record). Es ist möglich, ein Listenelement oder dessen **Inhalt** zu löschen (Erase Record). Listenelemente lassen sich aktivieren (Activate Record) und deaktivieren (Deactivate Record), was sich auf die Nutzung des Rekordinhaltes auswirkt. Des Weiteren ist es möglich, in der Liste nach Elementen zu suchen, deren Inhalt zu einem frei wählbaren Suchmuster passt (Search Record).

Die Kommandos in diesem Unterkapitel unterstützen zwei Varianten:

1. Variante ohne *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass für eine erfolgreiche Kommandoabarbeitung *currentEF* notwendigerweise ein strukturiertes EF ist. Die Variable *currentEF* lässt sich unter anderem durch gewisse Varianten des Select-Kommandos setzen.
2. Variante mit *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass das vom Kommando betroffene EF erst während der Kommandoabarbeitung gesetzt wird. Ein vorausgehendes Select-Kommando ist also nicht notwendig. In der Variante mit *shortFileIdentifier* lassen sich keine Dateien addressieren, die nicht *currentFolder* zugeordnet sind.

### Activate Record

Das Kommando Activate Record aktiviert ein oder mehrere Listenelemente aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses Activate Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Activate Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Listenelemente aktiviert werden, bestimmen Rekordnummer und Modus, welche als Parameter in der Kommandonachricht enthalten sind.

#### Use Case Aktivieren eines Rekords ohne *shortFileIdentifier*

In dieser Variante enthält die APDU des Activate Record-Kommandos zwei Parameter:

1. (N055.300) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N055.400) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´04´ gewählt werden.
3. (N055.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 95 verwendet werden.

Tabelle 95: Activate Record, ein Rekord, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´08´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´04´ | *mode*, Codierung ´04´ bedeutet „nutze Listenelement P1“ |

#### Use Case Aktivieren eines Rekords mit *shortFileIdentifier*

In dieser Variante enthält die APDU des Activate Record-Kommandos drei Parameter:

1. (N055.600) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N055.700) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N055.800) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´04´ gewählt werden.
4. (N055.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 96 verwendet werden.

Tabelle 96: Activate Record, ein Rekord, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´08´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´  Codierung ´04´ bedeutet „nutze Listenelement P1“ |

#### Use Case Aktivieren aller Rekords ab P1 ohne *shortFileIdentifier*

In dieser Variante enthält die APDU des Activate Record-Kommandos zwei Parameter:

1. (N056.000) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N056.100) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´05´ gewählt werden.
3. (N056.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 97 verwendet werden.

Tabelle 97: Activate Record, alle Rekords ab P1, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´08´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´05´ | *mode*, Codierung ´05´ bedeutet „nutze Listenelemente ab P1“ |

#### Use Case Aktivieren aller Rekords ab P1 mit *shortFileIdentifier*

In dieser Variante enthält die APDU des Activate Record-Kommandos drei Parameter:

1. (N056.300) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N056.400) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N056.500) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´05´ gewählt werden.
4. (N056.600) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 98 verwendet werden.

Tabelle 98: Activate Record, alle Rekords ab P1, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´08´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´05´  Codierung ´05´ bedeutet „nutze Listenelemente ab P1“ |

#### Antwort der Karte auf Aktivieren eines Rekords

Tabelle 99: Activate Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Aktivierung |

Tabelle 100: Activate Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoRecordLifeCycleStatus | Rekords in ausgewähltem EF besitzen keinen LCS |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement *recordNumber* existiert nicht |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N056.700) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N056.800) K\_COS
   1. Das COS MUSS die Activate Record-Varianten aus 14.4.1.1, 14.4.1.2, 14.4.1.3 und 14.4.1.4 unterstützen.
   2. Das COS KANN weitere Activate Record-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N056.900) K\_COS   
   Falls die APDU des Activate Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N057.000) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N057.100) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N057.200) K\_COS   
   Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer NoRecordLifeCycleStatus terminieren.
6. (N057.300) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
7. (N057.400) K\_COS   
   Wenn *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MUSS der *lifeCycleStatus* mit Transaktionsschutz geändert werden.
   2. False hat, dann MUSS das COS entscheiden, ob der *lifeCycleStatus* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
8. (N057.500) K\_COS   
   Wenn *mode* = ´04´ ist und der physikalische Wert von *lifeCycleStatus* des durch *recordNumber* adressierten *record* in *affectedObject.recordList* bereits den Wert „Operational state (active)“ besitzt, dann MUSS als Trailer NoError verwendet werden.
9. (N057.600) K\_COS   
   Der physikalische Wert von *lifeCycleStatus* der durch *recordNumber* und *mode* adressierten *record* in *affectedObject.recordList* MÜSSEN auf den Wert „Operational state (active)“ gesetzt werden. Dabei gilt: Wenn *mode* den Wert
   1. ´04´ besitzt, dann ist nur das durch *recordNumber* adressierte Listenelement betroffen.
   2. ´05´ besitzt, dann ist das durch *recordNumber* adressierte Listenelement und alle folgenden betroffen.
10. (N057.900) K\_COS   
    Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
11. (N058.000) K\_COS   
    Für den Trailer der Antwort-APDU gilt:
    1. Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
       1. entweder als Trailer MemoryFailure verwendet werden,
       2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
    2. Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
    3. Für die Priorität der Trailer gilt:
       1. Die Priorität der Trailer in Tabelle 100 ist herstellerspezifisch.
       2. Jeder Trailer in Tabelle 100 MUSS eine höhere Priorität als UpdateRetryWarning haben.
       3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
12. (N058.100) Diese Anforderung ist absichtlich leer.

(N058.200) Diese Anforderung ist absichtlich leer.

### Append Record

Das Kommando Append Record fügt ein neues Listenelement an *recordList* eines strukturierten EF an, wobei die Daten für den Oktettstring des neuen Listenelementes im Datenfeld der Kommandonachricht enthalten sind. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses Append Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Append Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde.

#### Use Case Anlegen neuer Rekord, ohne *shortFileIdentifier*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des Append Record-Kommandos enthält einen Parameter:

1. (N058.300) K\_externeWelt {K\_Karte}   
   Der Parameter *recordData* enthält die Daten des neuen Rekords. Der Parameter *recordData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *recordData* MUSS aus dem in (N007.700) definierten Bereich gewählt werden.
2. (N058.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 101 verwendet werden.

Tabelle 101: Append Record, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E2´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Parameter ohne Bedeutung |
| P2 | ´00´ | Parameter ohne Bedeutung |
| Data | ´XX…XX´ | *recordData* |

#### Use Case Anlegen neuer Rekords, mit *shortFileIdentifier*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des Append Record-Kommandos enthält zwei Parameter:

1. (N058.500) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N058.600) K\_externeWelt {K\_Karte}   
   Der Parameter *recordData* enthält die Daten des neuen Rekords. Der Parameter *recordData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *recordData* MUSS aus dem in (N007.700) definierten Bereich gewählt werden.
3. (N058.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 102 verwendet werden.

Tabelle 102: Append Record, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´E2´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Parameter ohne Bedeutung |
| P2 | ´XX´ | 8 *shortFileIdentifier*, das heißt *shortFileIdentifier* << 3 |
| Data | ´XX…XX´ | *recordData* |

#### Antwort der Karte auf Anlegen eines neuen Rekords

Tabelle 103: Append Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiches Hinzufügen eines Rekords |

Tabelle 104: Append Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´67 00´ | WrongRecordLength | *recordData* hat nicht die richtige Länge |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6A 84´ | FullRecordList | Rekordliste lässt keine weiteren Elemente zu |
| ´6A 84´ | OutOfMemory | Zu viele Oktette in *recordData* |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N058.800) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N058.900) K\_COS
   1. Das COS MUSS die Append Record-Varianten aus 14.4.2.1 und 14.4.2.2 unterstützen.
   2. Das COS KANN weitere Append Record-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N059.000) K\_COS   
   Falls die APDU des Append Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N059.100) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N059.200) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N059.300) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)a verschoben.
6. (N059.400) K\_COS   
   Wenn die Anzahl der Listenelemente in *affectedObject.recordList* gleich *affectedObject.maximumNumberOfRecords* ist und *affectedObject* vom Typ linear fixes EF oder linear variables EF ist, genau dann MUSS das Kommando mit dem Trailer FullRecordList terminieren.
7. (N059.500) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)b verschoben.
8. (N059.600) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)d verschoben.
9. (N059.650) K\_COS   
   Es werden folgende Schritte ausgeführt:
   1. Wenn *affectedObject* vom Typ
      1. linear fixes EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.
      2. zyklisches EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.
      3. linear variables EF ist und
         1. die Anzahl Oktette in *recordData* größer als *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.
         2. die Anzahl Oktette in den Oktettstrings aller *record* von *recordList* nach durchgeführter Listenerweiterung größer als *affectedObject.numberOfOctet* wäre, genau dann MUSS das Kommando mit dem Trailer OutOfMemory terminieren.
   2. Wenn *affectedObject.flagTransactionMode* den Wert
      1. True hat, genau dann MUSS *affectedObject.recordList* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.recordList* oder der Checksumme des neu angelegten Rekords umfassen, sofern diese vorhanden ist.
      2. False hat, dann MUSS das COS entscheiden, ob *affectedObject.recordList* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
   3. Falls *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.recordList*
      1. DARF NICHT zum Abbruch des Kommandos führen.
      2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
   4. Wenn *affectedObject* vom Typ
      1. linear fixes EF oder linear variables EF ist, dann wird ein neuer Rekord an das Ende von *affectedObject.recordList* angehängt.
      2. zyklische EF ist, dann wird ein neuer Rekord am Anfang von *affectedObject.recordList* eingefügt. Falls dadurch die Anzahl der Listenelemente größer als *affectedObject.maximumNumberOfRecords* wird, genau dann MUSS das letzte Element in *affectedObject.recordList* gelöscht werden.
   5. Falls *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.
   6. Falls der neu angelegte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS dessen Checksumme auf einen Wert gesetzt werden, der konsistent zum Inhalt des neuen Rekords ist.
   7. Als Oktettstring des neuen Rekords MUSS *recordData* verwendet werden.
   8. Falls *affectedObject.flagRecordLifeCycleStatus* den Wert True besitzt, dann MUSS der physikalische Wert des *lifeCycleStatus* des neuen Rekords auf „Operational state (active)“ gesetzt werden.
10. (N059.700) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)g verschoben.
11. (N059.800) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)h verschoben.
12. (N059.900) K\_COS   
    Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
13. (N060.000) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
14. (N060.100) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
15. (N060.200) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 104 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 104 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
16. (N060.300) Diese Anforderung ist absichtlich leer

(N060.400) Diese Anforderung ist absichtlich leer

### Deactivate Record

Das Kommando Deactivate Record deaktiviert ein oder mehrere Listenelemente aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses Deactivate Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Deactivate Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Listenelemente deaktiviert werden, bestimmen Rekordnummer und Modus, welche als Parameter in der Kommandonachricht enthalten sind.

#### Use Case Deaktivieren eines Rekords ohne *shortFileIdentifier*

In dieser Variante enthält die APDU des Deactivate Record-Kommandos Parameter:

1. (N060.500) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N060.600) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´04´ gewählt werden.
3. (N060.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 105 verwendet werden.

Tabelle 105: Deactivate Record, ein Rekord, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´06´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´04´ | *mode*, Codierung ´04´ bedeutet „nutze Listenelement P1“ |

#### Use Case Deaktivieren eines Rekords mit *shortFileIdentifier*

In dieser Variante enthält die APDU des Deactivate Record-Kommandos drei Parameter:

1. (N060.800) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N060.900) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N061.000) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´04´ gewählt werden.
4. (N061.100) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 106 verwendet werden.

Tabelle 106: Deactivate Record, ein Rekord, mit *shortFileIdentifier*

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´06´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´  Codierung ´04´ bedeutet „nutze Listenelement P1“ |

#### Use Case Deaktivieren aller Rekords ab P1 ohne *shortFileIdentifier*

In dieser Variante enthält die APDU des Deactivate Record-Kommandos zwei Parameter:

1. (N061.200) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N061.300) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´05´ gewählt werden.
3. (N061.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 107 verwendet werden.

Tabelle 107: Deactivate Record, alle Rekords ab P1, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´06´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´05´ | *mode*, Codierung ´05´ bedeutet „nutze Listenelemente ab P1“ |

#### Use Case Deaktivieren aller Rekords ab P1 mit *shortFileIdentifier*

In dieser Variante enthält die APDU des Deactivate Record-Kommandos drei Parameter:

1. (N061.500) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N061.600) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N061.700) K\_externeWelt {K\_Karte}   
   Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = ´05´ gewählt werden.
4. (N061.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 108 verwendet werden.

Tabelle 108: Deactivate Record, alle Rekords ab P1, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´06´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´05´  Codierung ´05´ bedeutet „nutze Listenelemente ab P1“ |

#### Antwort der Karte auf Deaktivieren eines Rekords

Tabelle 109: Deactivate Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiches Deaktivieren |

Tabelle 110: Deactivate Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoRecordLifeCycleStatus | Rekords in ausgewähltem EF besitzen keinen LCS |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement *recordNumber* existiert nicht |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N061.900) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N062.000) K\_COS
   1. Das COS MUSS die Deactivate Record-Varianten aus 14.4.3.1, 14.4.3.2, 14.4.3.3 und 14.4.3.4 unterstützen.
   2. Das COS KANN weitere Deactivate Record-Varianten
      1. unterstützen
      2. ablehnen.
2. (N062.100) K\_COS   
   Falls die APDU des Deactivate Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N062.200) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N062.300) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N062.400) K\_COS   
   Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer NoRecordLifeCycleStatus terminieren.
6. (N062.500) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
7. (N062.600) K\_COS   
   Wenn *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MUSS der *lifeCycleStatus* mit Transaktionsschutz geändert werden.
   2. False hat, dann MUSS das COS entscheiden, ob der *lifeCycleStatus* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
8. (N062.700) K\_COS   
   Wenn *mode* = ´04´ ist und der physikalische Wert von *lifeCycleStatus* des durch *recordNumber* adressierten *record* in *affectedObject.recordList* bereits den Wert „Operational state (deactivated)“ besitzt, dann MUSS als Trailer NoError verwendet werden.
9. (N062.800) K\_COS   
   Der physikalische Wert von *lifeCycleStatus* der durch *recordNumber* adressierten *record* in *affectedObject.recordList* MUSS auf den Wert „Operational state (deactivated)“ gesetzt werden. Dabei gilt: Wenn *mode* den Wert
   1. ´04´ besitzt, dann ist nur das durch *recordNumber* adressierte Listenelement betroffen.
   2. ´05´ besitzt, dann ist das durch *recordNumber* adressierte Listenelement und alle folgenden betroffen.
10. (N062.900) K\_COS   
    Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
11. (N063.000) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
12. (N063.100) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
13. (N063.200) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 110 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 110 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
14. (N063.300) Diese Anforderung ist absichtlich leer.

(N063.400) Diese Anforderung ist absichtlich leer.

### Delete Record

Das Kommando Delete Record entfernt ein bereits vorhandenes Listenelemente aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses Delete Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Delete Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement betroffen ist, bestimmt die Rekordnummer, welche als Parameter in der Kommandonachricht enthalten ist.

1. Durch das Entfernen eines Elementes aus einer Liste ändert sich die Adressierung nachfolgender Listenelemente. Enthält beispielsweise eine strukturierte Datei eine Liste mit drei Rekords gemäß ´01´🡪´02´🡪´03´ wird ein Read Record Kommando für Rekord zwei (Kommando APDU = ´00B2020400´) Listenelement 2 liefern (Response APDU = ´029000´). Nach einem Delete Record (Kommando APDU = ´800C0204´) enthält die Datei folgende Liste: ´01´🡪´03´. Ein Read Record Kommando für Rekord zwei liefert dann: ´039000´)

#### Use Case Löschen eines Rekords ohne *shortFileIdentifier*

In dieser Variante wird ein Rekord aus der Liste *recordList* gelöscht.

1. (N063.420) K\_externeWelt {K\_Karte}   
   Die APDU des Delete Record-Kommandos enthält einen Parameter:
   1. Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N063.422) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 111 verwendet werden.

Tabelle 111: Delete Record, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´0C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´04´ | Codierung für „nutze Listenelement P1“ |

#### Use Case Löschen eines Rekords mit *shortFileIdentifier*

In dieser Variante wird ein Rekord aus der Liste *recordList* gelöscht.

1. (N063.424) K\_externeWelt {K\_Karte}   
   Die APDU des Delete Record-Kommandos enthält zwei Parameter:
   1. Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
   2. Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N063.426) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 112 verwendet werden.

Tabelle 112: Delete Record, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´0C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´ Codierung ´04´ bedeutet „nutze Listenelement P1“ |

#### Antwort der Karte auf Entfernen eines Rekords

Tabelle 113: Delete Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiches Entfernen eines Rekords |

Tabelle 114: Delete Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 87´ | RecordDeactivated | Adressierter Rekord ist deaktiviert |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement *recordNumber* existiert nicht |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N063.428) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N063.430) K\_COS
   1. Das COS MUSS die Delete Record-Varianten aus 14.4.4.1 und 14.4.4.2 unterstützen.
   2. Das COS KANN weitere Delete Record-Varianten
      1. unterstützen
      2. ablehnen.
2. (N063.432) K\_COS   
   Falls die APDU des Delete Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N063.434) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N063.436) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N063.438) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
6. (N063.440) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand „Operational state (deactivated)“ hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.
7. (N063.442) K\_COS   
   Falls *affectedObject.recordList* oder der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
   1. DARF NICHT zum Kommandoabbruch führen.
   2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
8. (N063.446) K\_COS   
   Der Rekord wird wie folgt aus der der Liste *recordList* entfernt:
   1. Der durch *recordNumber* adressierte *record* MUSS aus *affectedObject*.*recordList* entfernt werden und
   2. falls *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.
9. (N063.448) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
10. (N063.450) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
11. (N063.452) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
12. (N063.454) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 114 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 114 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### Erase Record

Das Kommando Erase Record ersetzt den Oktettstring eines bereits vorhandenen Listenelementes in *recordList* eines strukturierten EF durch einen Oktettstring, der nur Oktette mit dem Wert ´00´ besitzt. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses Erase Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Erase Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement betroffen ist, bestimmt die Rekordnummer, welche als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Löschen eines Rekordinhaltes ohne *shortFileIdentifier*

In dieser Variante werden Daten in einem Rekord durch Oktette mit dem Wert ´00´ ersetzt. Der Rekord selbst verbleibt in der Liste *recordList*. Die APDU des Erase Record Kommandos enthält einen Parameter:

1. (N063.500) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N063.600) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 115 verwendet werden.

Tabelle 115: Erase Record, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4], adressierter Rekord bleibt erhalten |
| INS | ´0C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´04´ | Codierung für „nutze Listenelement P1“ |

#### Use Case Löschen eines Rekordinhaltes mit *shortFileIdentifier*

In dieser Variante werden Daten in einem Rekord durch Oktette mit dem Wert ´00´ ersetzt. Der Rekord selbst verbleibt in der Liste *recordList*. Die APDU des Erase Record-Kommandos enthält zwei Parameter:

1. (N063.700) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N063.800) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N063.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 116 verwendet werden.

Tabelle 116: Erase Record, mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4], adressierter Rekord bleibt erhalten |
| INS | ´0C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´  Codierung ´04´ bedeutet „nutze Listenelement P1“ |

#### Antwort der Karte auf Löschen des Inhaltes eines Rekords

Tabelle 117: Erase Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiches Löschen eines Rekordinhaltes |

Tabelle 118:Erase Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 87´ | RecordDeactivated | Adressierter Rekord ist deaktiviert |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement *recordNumber* existiert nicht |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N064.000) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N064.100) K\_COS
   1. Das COS MUSS die Erase Record-Varianten aus 14.4.5.1 und 14.4.5.2 unterstützen.
   2. Das COS KANN weitere Erase Record-Varianten
      1. unterstützen
      2. ablehnen.
2. (N064.200) K\_COS   
   Falls die APDU des Erase Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N064.300) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N064.400) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N064.500) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
6. (N064.600) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand „Operational state (deactivated)“ hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.
7. (N064.700) K\_COS   
   Wenn *affectedObject.flagTransactionMode* den Wert
   1. True hat, genau dann MUSS der Inhalt des adressierten Rekords mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.recordList* oder der Checksumme des durch *recordNumber* adressierten *record* in *affectedObject.recordList* umfassen, sofern diese vorhanden ist.
   2. False hat, dann MUSS das COS entscheiden, ob der Rekordinhalt mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
8. (N064.710) K\_COS   
   Falls *affectedObject.recordList* oder der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
   1. DARF NICHT zum Kommandoabbruch führen.
   2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N064.800) K\_COS   
   Der Rekordinhalt wird wie folgt gelöscht
   1. Es MÜSSEN alle Oktette im Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* auf den Wert ´00´ gesetzt werden und
   2. falls *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist und
   3. falls der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS dessen Checksumme auf einen Wert gesetzt werden, der konsistent zum neuen Inhalt von diesem Rekord ist.
10. (N064.900) K\_COS   
    Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
11. (N065.000) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
12. (N065.100) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
13. (N065.200) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 118 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 118 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
14. (N065.300) Diese Anforderung ist absichtlich leer.

(N065.400) Diese Anforderung ist absichtlich leer.

### Read Record

Das Kommando Read Record dient dem Auslesen (des Anfangs) eines Listenelementes aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Leseoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Read Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Read Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement (oder dessen Anfang) ausgelesen wird, bestimmen Rekordnummer und Länge, die als Parameter in der Kommandonachricht enthalten sind.

#### Use Case Lesen ohne *shortFileIdentifier* in strukturierten EF

In dieser Variante enthält die APDU des Read Record-Kommandos zwei Parameter:

1. (N065.500) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N065.600) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
3. (N065.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 119 verwendet werden.

Tabelle 119: Read Record ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´B2´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´04´ | Codierung für „nutze Listenelement P1“ |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Lesen mit shortFileIdentifier in strukturierten EF

In dieser Variante enthält die APDU des Read Record-Kommandos drei Parameter:

1. (N065.800) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N065.900) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N066.000) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
4. (N066.100) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 120 verwendet werden.

Tabelle 120: Read Record mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´B2´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´  Codierung ´04´ bedeutet „nutze Listenelement P1“ |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Lesen in strukturierten EF

Tabelle 121: Read Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ |  | Ausgelesene Daten |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 81´ | CorruptDataWarning | Möglicherweise sind die Antwortdaten korrupt |
| ´62 82´ | EndOfRecordWarning | Mittels Ne mehr Daten angefordert, als vorhanden sind |
| ´90 00´ | NoError | Erfolgreiche Leseoperation |

Tabelle 122: Read Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 87´ | RecordDeactivated | Adressierter Rekord ist deaktiviert |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* adressiertes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement *recordNumber* existiert nicht |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N066.200) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N066.300) K\_COS
   1. Das COS MUSS die Read Record-Varianten aus 14.4.6.1 und 14.4.6.2 unterstützen.
   2. Das COS KANN weitere Read Record-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N066.400) K\_COS   
   Falls die APDU des Read Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N066.500) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N066.600) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N066.700) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
6. (N066.800) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand „Operational state (deactivated)“ hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.
7. (N066.900) K\_COS   
   Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten des adressierten Listenelementes inkonsistent zur Checksumme sind, genau dann MUSS
   1. entweder als Trailer CorruptDataWarning gewählt werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
8. (N067.000) K\_COS   
   Wenn das LeFeld der Kommando-APDU keine WildCard enthält und *length* größer als die Länge des adressierten Listenelementes *affectedObject.numberOfOctet* ist, genau dann MUSS als Trailer EndOfRecordWarning gewählt werden.
9. (N067.100) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
10. (N067.200) K\_COS   
    Für das Datenfeld der Antwortnachricht gilt:
    1. Aus dem Oktettstring des adressierten *record* MÜSSEN vom ersten Oktett an die nachfolgenden Oktette übernommen werden.
    2. Es DÜRFEN NICHT mehr Oktette übernommen werden, als durch *length* angegeben.
    3. Die Übernahme MUSS am Ende von *record* stoppen.
11. (N067.300) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 122 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 122 MUSS eine höhere Priorität als CorruptDataWarning haben.
    3. CorruptDataWarning MUSS eine höhere Priorität als EndOfRecordWarning haben.
    4. EndOfRecordWarning MUSS eine höhere Priorität als NoError haben.
12. (N067.400) Diese Anforderung ist absichtlich leer.

(N067.500) Diese Anforderung ist absichtlich leer.

### Search Record

Das Kommando Search Record sucht in den Listenelementen von *recordList* eines strukturierten EF nach einem Muster, welches im Datenfeld der Kommandonachricht übergeben wird. Die Antwortdaten enthalten die Nummern der Rekords, welche das Muster enthalten. Das betroffene, strukturierte EF wird vor der Suchoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Search Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Search Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Bei welchem Listenelement in *recordList* die Suche startet, wird durch die Rekordnummer bestimmt, die als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Suchen ohne *shortFileIdentifier* in strukturierten EF

In dieser Variante enthält die APDU des Search Record-Kommandos drei Parameter:

1. (N067.600) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das Listenelement, welches als erstes von der Suche betroffen ist. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N067.700) K\_externeWelt {K\_Karte}   
   Der Parameter *searchString* enthält das Muster, nach welchem in den Oktettstrings der Listenelemente gesucht wird. Der Parameter *searchString* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *searchString* MUSS kleiner oder gleich 255 Oktett sein.
3. (N067.800) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
4. (N067.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 123 verwendet werden.

Tabelle 123: Search Record ohne *shortFileIdentifier*

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A2´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | recordNumber |
| P2 | ´04´ | Codierung für „suche in Listenelement P1 und allen folgenden“ |
| Data | ´XX…XX´ | *searchString* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Suchen mit *shortFileIdentifier* in strukturierten EF

In dieser Variante enthält die APDU des Search Record-Kommandos vier Parameter:

1. (N068.000) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N068.100) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das Listenelement, welches als erstes von der Suche betroffen ist. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N068.200) K\_externeWelt {K\_Karte}   
   Der Parameter *searchString* enthält das Muster, nach welchem in den Oktettstrings der Listenelemente gesucht wird. Der Parameter *searchString* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *searchString* MUSS kleiner oder gleich 255 Oktett sein.
4. (N068.300) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.
5. (N068.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 124 verwendet werden.

Tabelle 124: Search Record mit *shortFileIdentifier*

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´A2´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´  ´04´ bedeutet: „suche in Listenelement P1 und allen folgenden“ |
| Data | ´XX…XX´ | *searchString* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Suchen in strukturierten EF

Tabelle 125: Search Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *rspData* | Nummern der Listenelemente, in denen das Muster gefunden wurde |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 81´ | CorruptDataWarning | Möglicherweise sind Antwortdaten korrupt |
| ´62 82´ | UnsuccessfulSearch | Erfolglose Suche in adressierten Rekords |
| ´90 00´ | NoError | Erfolgreiche Suchoperation |

Tabelle 126: Search Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per *shortFileIdentifier* ausgewähltes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement *recordNumber* existiert nicht |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N068.500) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N068.600) K\_COS
   1. Das COS MUSS die Search Record-Varianten aus 14.4.7.1 und 14.4.7.2 unterstützen.
   2. Das COS KANN weitere Search Record-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N068.700) K\_COS   
   Falls die APDU des Search Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N068.800) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N068.900) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N069.000) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
6. (N069.100) K\_COS   
   Im Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* und allen folgenden Elementen der Liste MUSS nach dem Muster *searchString* gesucht werden.
7. (N069.200) K\_COS   
   Die Suche in einem Listenelement MUSS genau dann erfolgreich sein, wenn
   1. der physikalische Wert des *lifeCycleStatus* des Listenelementes den Wert „Operational state (active)“ hat UND
   2. *searchString* im Oktettstring des Listenelementes vollständig enthalten ist.
8. (N069.300) K\_COS   
   Wenn die Suche in einem Listenelement erfolgreich war, dann MUSS die Nummer des Rekords (siehe (N007.600)) in einem Oktett (gemäß I2OS(*recordNumber*, 1)) codiert zum Datenfeld *rspData* der Antwortnachricht hinzugefügt werden.
9. (N069.400) K\_COS   
   Die Oktette im Datenfeld *rspData* MÜSSEN aufsteigend sortiert sein.
10. (N069.500) K\_COS   
    Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten wenigstens eines adressierten Listenelementes inkonsistent zur Checksumme sind, genau dann MUSS
    1. entweder als Trailer CorruptDataWarning gewählt werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
11. (N069.600) K\_COS   
    Wenn das Datenfeld *rspData* leer ist, das heißt die Suche war in keinem der adressierten Listenelemente erfolgreich, genau dann MUSS als Trailer UnsuccessfulSearch verwendet werden.
12. (N069.700) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
13. (N069.710) K\_COS   
    Für das Datenfeld *rspData* der Antwortnachricht gilt: Falls OctetLength( *rspData* )
    1. kleiner Nr gemäß (N027.200): *rspData* = *rspData*.
    2. sonst *rspData* = Extract\_MSByte( *rspData*, Nr ).
14. (N069.800) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 126 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 126 MUSS eine höhere Priorität als CorruptDataWarning haben.
    3. CorruptDataWarning MUSS eine höhere Priorität als UnsuccessfulSearch haben.
    4. UnsuccessfulSearch MUSS eine höhere Priorität als NoError haben.
15. (N069.900) Diese Anforderung ist absichtlich leer.

(N070.000) Diese Anforderung ist absichtlich leer.

### Update Record

Das Kommando Update Record ersetzt den Oktettstring eines bereits vorhandenen Listenelementes in *recordList* eines strukturierten EF durch Daten, die im Datenfeld der Kommandonachricht enthalten sind. Das betroffene strukturierte EF wird vor der Schreiboperation ausgewählt. Dies geschieht entweder vor dem Senden dieses Update Record-Kommandos durch eine Select-Operation (Select-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses Update Record-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement in *recordList* ersetzt wird, bestimmt die Rekordnummer, die als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Rekordinhalt schreiben, ohne *shortFileIdentifier*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des Update Record-Kommandos enthält zwei Parameter:

1. (N070.100) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
2. (N070.200) K\_externeWelt {K\_Karte}   
   Der Parameter *newData* enthält die neuen Daten, welche den Oktettstring des adressierten *record* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N007.700) definierten Bereich gewählt werden.
3. (N070.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 127 verwendet werden.

Tabelle 127: Update Record, ohne *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´DC´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´04´ | Codierung für „nutze Listenelement P1“ |
| Data | ´XX…XX´ | *newData* |

#### Use Case Rekordinhalt schreiben, mit *shortFileIdentifier*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des Update Record-Kommandos enthält drei Parameter:

1. (N070.400) K\_externeWelt {K\_Karte}   
   Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
2. (N070.500) K\_externeWelt {K\_Karte}   
   Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.
3. (N070.600) K\_externeWelt {K\_Karte}   
   Der Parameter *newData* enthält die neuen Daten, welche den Oktettstring des adressierten *record* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N007.700) definierten Bereich gewählt werden.
4. (N070.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 128 verwendet werden.

Tabelle 128: Update Record mit *shortFileIdentifier*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´DC´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´XX´ | *recordNumber* |
| P2 | ´XX´ | 8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + ´04´  Codierung ´04´ bedeutet „nutze Listenelement P1“ |
| Data | ´XX…XX´ | *newData* |

#### Antwort der Karte auf Schreiben in strukturierten EF

Tabelle 129: Update Record Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreicher Schreibvorgang |

Tabelle 130: Update Record Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 87´ | RecordDeactivated | Adressierter Rekord ist deaktiviert |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´67 00´ | WrongRecordLength | newData hat nicht die richtige Länge |
| ´69 81´ | WrongFileType | Ausgewähltes EF ist nicht strukturiert |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 86´ | NoCurrentEF | Es ist kein EF ausgewählt |
| ´6A 82´ | FileNotFound | Per shortFileIdentifier ausgewähltes EF nicht gefunden |
| ´6A 83´ | RecordNotFound | Listenelement recordNumber existiert nicht |
| ´6A 84´ | OutOfMemory | Zu viele Oktette in newData |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N070.800) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N070.900) K\_COS
   1. Das COS MUSS die Update Record-Varianten aus 14.4.8.1 und 14.4.8.2 unterstützen.
   2. Das COS KANN weitere Update Record-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N071.000) K\_COS   
   Falls die APDU des Update Record-Kommandos
   1. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
      1. erfolgreich verlief, dann MUSS
         1. *affectedObject* auf dieses EF gesetzt werden und
         2. *currentEF* auf dieses EF gesetzt werden.
      2. nicht erfolgreich verlief, genau dann MUSS
         1. *currentEF* unverändert bleiben und
         2. das Kommando mit dem Trailer FileNotFound terminieren.
   2. keinen *shortFileIdentifier* enthält
      1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
      2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
3. (N071.100) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N071.200) K\_COS   
   Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.
5. (N071.300) K\_COS   
   Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.
6. (N071.400) K\_COS   
   Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand „Operational state (deactivated)“ hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.
7. (N071.500) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N071.750)a verschoben.
8. (N071.600) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N071.750)b verschoben.
9. (N071.700) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N071.750)d verschoben.
10. (N071.750) K\_COS   
    Es werden folgende Schritte ausgeführt:
    1. Wenn *affectedObject* vom Typ
       1. linear fixes EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.
       2. zyklisches EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.
       3. linear variables EF ist und
          1. die Anzahl Oktette in *recordData* größer als *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.
          2. die Anzahl Oktette in den Oktettstrings aller *record* von *affectedObject.recordList* nach durchgeführter Ersetzung größer als *affectedObject.numberOfOctet* wäre, genau dann MUSS das Kommando mit dem Trailer OutOfMemory terminieren.
    2. Wenn *affectedObject.flagTransactionMode* den Wert
       1. True hat, genau dann MUSS der Rekordinhalt mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.recordList* oder der Checksumme des durch *recordNumber* adressierten *record* in *affectedObject.recordList* umfassen, sofern diese vorhanden ist.
       2. False hat, dann MUSS das COS entscheiden, ob der Rekordinhalt mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
    3. Falls *affectedObject.recordList* oder der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
       1. DARF NICHT zum Kommandoabbruch führen.
       2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
    4. K\_COS
       1. Der Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* wird durch *recordData* ersetzt.
       2. Falls *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.
       3. Falls der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS dessen Checksumme auf einen Wert gesetzt werden, der konsistent zum neuen Inhalt von diesem Rekord ist.
11. (N071.800) K\_COS   
    Falls *affectedObject* vom Typ linear variables EF ist, dann MÜSSEN alle folgenden Fälle unterstützt werden: *recordData* enthält im Vergleich zum *record*, der durch *recordNumber* adressiert wird
    1. weniger Oktette,
    2. gleich viele Oktette, oder
    3. mehr Oktette.
12. (N071.900) K\_COS   
    Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
13. (N072.000) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
14. (N072.100) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
15. (N072.200) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 130 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 130 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.
16. (N072.300) Diese Anforderung ist absichtlich leer.

(N072.400) Diese Anforderung ist absichtlich leer

### Write Record

1. (N072.500) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]
   1. unterstützen oder
   2. ablehnen.

## Zugriff auf Datenobjekte

1. Gemäß 8.7 sind die Kommandos dieses Kapitels nicht verpflichtend.

### Get Data

1. (N072.600) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]
   1. unterstützen oder
   2. ablehnen.

### Put Data

1. (N072.700) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]
   1. unterstützen oder
   2. ablehnen.

## Benutzerverifikation

Alle Kommandos dieses Kapitels benutzen bei der Kommandobearbeitung Passwortobjekte gemäß 8.4 oder 8.5. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in den Kommandodaten enthalten ist. Für diese Passwortreferenz gilt:

1. (N072.800) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* besteht aus den zwei Teilen *location* und *identifier*. *location* zeigt an, ob ein globales oder DF-spezifisches Passwort von der Aktion betroffen ist. Als Wert für *location* MUSS ein Element der Menge {0, 128} = {´00´, ´80´} verwendet werden. Dabei gilt:
   1. Der Wert *location* = ´00´ MUSS verwendet werden, wenn ein globales Passwortobjekt betroffen ist (siehe (N020.800)).
   2. Der Wert *location* = ´80´ MUSS verwendet werden, wenn ein DF-spezifisches Passwortobjekt betroffen ist (siehe (N020.900)).
   3. Der Parameter *identifier* bestimmt das betroffene Passwortobjekt. Der Wert von *identifier* MUSS konform zu (N015.000) gewählt werden.
   4. Der Parameter *passwordReference* MUSS in einem Oktett mit folgendem Wert codiert werden: *passwordReference = location + identifier*.

### Change Reference Data

Das Kommando Change Reference Data ersetzt das Attribut *secret* eines Passwortobjektes durch Daten, die im Datenfeld der Kommandonachricht enthalten sind. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Dokument spezifiziert die folgenden Varianten:

* Das Kommandodatenfeld enthält das alte und neue Benutzergeheimnis.
* Das Kommandodatenfeld enthält nur das neue Benutzergeheimnis.

#### Use Case Ändern eines Benutzergeheimnisses

In dieser Variante enthält die APDU des Change Reference Data-Kommandos drei Parameter:

1. (N072.900) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N073.000) K\_externeWelt {K\_Karte}   
   Der Parameter *oldSecret* enthält das alte Benutzergeheimnis.
3. (N073.100) K\_externeWelt {K\_Karte}   
   Der Parameter *newSecret* enthält das neue Benutzergeheimnis.
4. (N073.200) K\_externeWelt {K\_Karte}   
   Die Parameter *oldSecret* und *newSecret* MÜSSEN gemäß (N008.100) codiert sein.
5. (N073.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 131 verwendet werden.

Tabelle 131: Change Reference Data mit altem und neuem Benutzergeheimnis

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´24´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Data enthält altes und neues Benutzergeheimnis |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *oldSecret*   ||   *newSecret* |

#### Use Case Setzen eines Benutzergeheimnisses

In dieser Variante enthält die APDU des Change Reference Data-Kommandos zwei Parameter:

1. (N073.400) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N073.500) K\_externeWelt {K\_Karte}   
   Der Parameter *newSecret* enthält das neue Benutzergeheimnis.
3. (N073.600) K\_externeWelt {K\_Karte}   
   Der Parameter *newSecret* MUSS gemäß (N008.100) codiert sein.
4. (N073.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 132 verwendet werden.

Tabelle 132: Change Reference Data, nur neues Benutzergeheimnis

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´24´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´ | Data enthält neues Benutzergeheimnis |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *newSecret* |

#### Antwort der Karte auf Ändern eines Benutzergeheimnisses

Tabelle 133: Change Reference Data Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | WrongSecretWarning | *oldSecret* ist falsch |
| ´90 00´ | NoError | Erfolgreiches Ändern des Benutzergeheimnisses |

Tabelle 134: Change Reference Data Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | PasswordBlocked | Abgelaufener Fehlbedienungszähler |
| ´69 85´ | LongPassword | *newData* enthält ein zu langes Passwort |
| ´69 85´ | ShortPassword | *newData* enthält ein zu kurzes Passwort |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N073.800) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N073.900) K\_COS   
   Das COS MUSS die Change Reference Data-Varianten aus 14.6.1.1 und 14.6.1.2 unterstützen.
2. (N074.000) Diese Anforderung ist absichtlich leer.
3. (N074.100) K\_COS   
   Das COS KANN weitere Change Reference Data-Varianten
   1. unterstützen oder
   2. ablehnen.
4. (N074.200) K\_COS   
   Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
5. (N074.250) Diese Anforderung ist absichtlich leer.
6. (N074.300) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
7. (N074.400) K\_COS   
   Wenn *affectedObject.retryCounter* den Wert null hat, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.
8. (N074.500) K\_COS   
   Wenn die in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die
   1. kleiner als *affectedObject.minimumLength* ist, genau dann MUSS das Kommando mit dem Trailer ShortPassword terminieren.
   2. größer als *affectedObject.maximumLength* ist, genau dann MUSS das Kommando mit dem Trailer LongPassword terminieren.
9. (N074.600) Diese Anforderung ist absichtlich leer.
10. Die Aussage in (N074.700)a.1 bezieht sich bewusst nur auf den momentan aktiven logischen Kanal. Daraus folgt, dass Sicherheitszustände in anderen logischen Kanälen von der dort beschriebenen Aktion unberührt bleiben.
11. (N074.700) K\_COS   
    Wenn das Datenfeld der Kommandonachricht *oldSecret* enthält, genau dann MUSS das Attribut *affectedObject.secret* mit *oldSecret* verglichen werden.
    1. Wenn der Vergleich fehlschlägt, genau dann MUSS
       1. der Sicherheitszustand im *channelContext* des momentan aktiven logischen Kanals mittels clearPasswordStatus( *affectedObject* ) zurückgesetzt werden,
       2. *affectedObject.retryCounter* um eins dekrementiert werden und
       3. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert ´F´ gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.
    2. Wenn der Vergleich erfolgreich ist, genau dann MUSS das Attribut *affectedObject.retryCounter* auf den Wert *affectedObject.startRetryCounter* gesetzt werden.
    3. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
12. (N074.710) K\_COS   
    Das Attribut *affectedObject.secret* MUSS auf den in *newSecret* codierten Wert gesetzt werden.
13. (N074.720) K\_COS   
    Das Attribut *affectedObject.transportStatus* MUSS auf den Wert *regularPassword* geändert werden (siehe (N009.500)).
14. (N074.800) K\_COS   
    Alle persistenten Änderungen in (N074.700)b, (N074.710) und (N074.720) MÜSSEN mit Transaktionsschutz ausgeführt werden.
15. (N074.900) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
16. (N075.000) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
17. (N075.100) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
18. (N075.200) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 134 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 134 MUSS eine höhere Priorität als WrongSecretWarning haben.
    3. WrongSecretWarning MUSS eine höhere Priorität als NoError haben.
19. (N075.300) K\_COS   
    Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für konditionale Änderung von *affectedObject.retryCounter* und die Änderung von *affectedObject.transportStatus* und *affectedObject.secret*:
    1. Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.
    2. *retryCounter* KANN in einer eigenen Transaktion zeitlich vor anderen Attributen geändert werden.
    3. *secret* KANN in einer eigenen Transaktion zeitlich vor *transportStatus* geändert werden.
    4. *transportStatus* DARF NICHT persistent geändert sein, wenn nicht auch *secret* persistent geändert ist.

### Disable Verification Requirement

Das Kommando Disable Verification Requirement ändert das Attribut *flagEnabled* eines Passwortobjektes (siehe 8.4 und 8.5) so, dass das COS sich so verhält, als sei der Sicherheitszustand des Passwortes ständig gesetzt. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Kommando gibt es in den Varianten mit und ohne Verifikationsdaten.

#### Use Case Abschalten der Benutzerverifikation mit Benutzergeheimnis

In dieser Variante wird der Zwang zur Benutzerverifiaktion nicht abgeschaltet, falls das Benutzergeheimnis nicht korrekt ist. Die APDU des Disable Verification Requirement-Kommandos enthält zwei Parameter:

1. (N075.380) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N075.382) K\_externeWelt {K\_Karte}   
   Der Parameter *verificationData* enthält das Benutzergeheimnis.
3. (N075.384) K\_externeWelt {K\_Karte}   
   Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.
4. (N075.386) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 135 verwendet werden.

Tabelle 135: Disable Verification Requirement mit Verifikationsdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´26´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Verifikationsdaten |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *verificationData* |

#### Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis

In dieser Variante enthält die APDU des Disable Verification Requirement-Kommandos einen Parameter:

1. (N075.400) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N075.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 136 verwendet werden.

Tabelle 136: Disable Verification Requirement ohne Verifikationsdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´26´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´ | Keine Verifikationsdaten |
| P2 | ´XX´ | *passwordReference* |

#### Antwort der Karte auf Abschalten der Benutzerverifikation

Tabelle 137: Disable Verification Requirement Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | WrongSecretWarning | *verificationData* ist falsch |
| ´90 00´ | NoError | Erfolgreiches Abschalten des Passwortobjektes |

Tabelle 138: Disable Verification Requirement Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | PasswordBlocked | Abgelaufener Fehlbedienungszähler |
| ´69 85´ | PasswordNotUsable | Passwort mit Transportschutz versehen |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N075.600) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N075.700) K\_COS
   1. Das COS MUSS die Disable Verification Requirement-Variante aus 14.6.2.1 und 14.6.2.2 unterstützen.
   2. Das COS KANN weitere Disable Verification Requirement-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N075.800) K\_COS   
   Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N075.900) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N076.000) Diese Anforderung ist absichtlich leer.
5. (N076.100) K\_COS   
   Falls der Parameter P1 den Wert
   1. ´00´besitzt, dann MÜSSEN folgende Schritte abgearbeitet werden:
      1. Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.
      2. Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.
      3. Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.
         1. Wenn der Vergleich fehlschlägt, genau dann MUSS
            1. *affectedObject.retryCounter* um eins dekrementiert werden und
            2. clearPasswordStatus( *affectedObject* ) ausgeführt werden und
            3. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert ´F´ gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.
         2. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
         3. Wenn der Vergleich erfolgreich ist, genau dann MUSS
            1. das Attribut *affectedObject.flagEnabled* auf den Wert False gesetzt werden und
            2. das Attribut *affectedObject*.*retryCounter* auf den Wert *affectedObject*.*startRetryCounter* gesetzt werden.
   2. ´01´besitzt, dann MUSS das Attribut *affectedObject.flagEnabled* auf den Wert False gesetzt werden.
6. (N076.110) K\_COS   
   Alle persistenten Änderungen in (N076.100) MÜSSEN mit Transaktionsschutz ausgeführt werden.
7. (N076.120) K\_COS   
   Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für konditionale Änderung von *affectedObject*.*retryCounter* und die Änderung von *affectedObject*.*flagEnabled*:
   1. Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.
   2. *retryCounter* KANN in einer eigenen Transaktion zeitlich vor anderen Attributen geändert werden.
8. (N076.200) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
9. (N076.300) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
10. (N076.400) Diese Anforderung ist absichtlich leer.
11. (N076.500) K\_COS   
    Für die Wahl des Trailers gilt: Falls *affectedObject.flagEnabled* den Wert
    1. False besitzt, genau dann MUSS als Trailer NoError verwendet werden
    2. True besitzt, dann ist die Priorität der Trailer in Tabelle 138 herstellerspezifisch.
    3. Jeder Trailer in Tabelle 138 MUSS eine höhere Priorität als WrongSecretWarning haben.

### Enable Verification Requirement

Das Kommando Enable Verification Requirement ändert das Attribut *flagEnabled* eines Passwortobjektes (siehe 8.4 und 8.5) so, dass der Sicherheitszustand des Passwortes nur durch eine erfolgreiche Benutzerverifikation gesetzt wird. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Kommando gibt es in den Varianten mit und ohne Verifikationsdaten.

#### Use Case Einschalten der Benutzerverifikation mit Benutzergeheimnis

In dieser Variante wird der Zwang zur Benutzerverifiaktion nicht eingeschaltet, falls das Benutzergeheimnis nicht korrekt ist. Die APDU des Enable Verification Requirement-Kommandos enthält zwei Parameter:

1. (N076.580) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N076.582) K\_externeWelt {K\_Karte}   
   Der Parameter *verificationData* enthält das Benutzergeheimnis.
3. (N076.584) K\_externeWelt {K\_Karte}   
   Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.
4. (N076.586) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 139 verwendet werden.

Tabelle 139: Enable Verification Requirement mit Verifikationsdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´28´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Verifikationsdaten |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *verificationData* |

#### Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis

In dieser Variante enthält die APDU des Enable Verification Requirement-Kommandos einen Parameter:

1. (N076.600) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N076.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 140 verwendet werden.

Tabelle 140: Enable Verification Requirement ohne Verifikationsdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´28´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´ | Keine Verifikationsdaten |
| P2 | ´XX´ | *passwordReference* |

#### Antwort der Karte auf Einschalten der Benutzerverifikation

Tabelle 141: Enable Verification Requirement Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | WrongSecretWarning | *verificationData* ist falsch |
| ´90 00´ | NoError | Erfolgreiches Einschalten des Passwortobjektes |

Tabelle 142: Enable Verification Requirement Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | PasswordBlocked | Abgelaufener Fehlbedienungszähler |
| ´69 85´ | PasswordNotUsable | Passwort mit Transportschutz versehen |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N076.800) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N076.900) K\_COS
   1. Das COS MUSS die Enable Verification Requirement-Variante aus 14.6.3.1 und 14.6.3.2 unterstützen.
   2. Das COS KANN weitere Enable Verification Requirement-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N077.000) K\_COS   
   Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N077.100) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N077.200) Diese Anforderung ist absichtlich leer.
5. (N077.300) K\_COS   
   Falls der Parameter P1 den Wert
   1. ´00´besitzt, dann MÜSSEN folgende Schritte abgearbeitet werden:
      1. Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.
      2. Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.
      3. Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.
         1. Wenn der Vergleich fehlschlägt, genau dann MUSS
            1. *affectedObject.retryCounter* um eins dekrementiert werden und
            2. clearPasswordStatus( *affectedObject* ) ausgeführt werden und
            3. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert ´F´ gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.
         2. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
         3. Wenn der Vergleich erfolgreich ist, genau dann MUSS
            1. das Attribut *affectedObject.flagEnabled* auf den Wert ´True gesetzt werden und
            2. das Attribut *affectedObject*.*retryCounter* auf den Wert *affectedObject*.*startRetryCounter* gesetzt werden.
   2. ´01´ besitzt, dann MUSS das Attribut *affectedObject.flagEnabled* auf den Wert True gesetzt werden.
6. (N077.310) K\_COS   
   Alle persistenten Änderungen in (N077.300) MÜSSEN mit Transaktionsschutz ausgeführt werden.
7. (N077.320) K\_COS   
   Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für konditionale Änderung von *affectedObject*.*retryCounter* und die Änderung von *affectedObject*.*flagEnabled*:
   1. Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.
   2. *retryCounter* KANN in einer eigenen Transaktion zeitlich vor anderen Attributen geändert werden.
8. (N077.400) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
9. (N077.500) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
10. (N077.600) Diese Anforderung ist absichtlich leer.
11. (N077.700) K\_COS   
    Für die Wahl des Trailers gilt: Falls *affectedObject.flagEnabled* den Wert
    1. True besitzt, genau dann MUSS als Trailer NoError verwendet werden.
    2. False besitzt, dann ist die Priorität der Trailer in Tabelle 142 herstellerspezifisch.
    3. Jeder Trailer in Tabelle 142 MUSS eine höhere Priorität als WrongSecretWarning haben.

### Get Pin Status

1. Dieses Kommando ist nicht in der Normenreihe ISO/IEC 7816 enthalten. Es ließe sich kombinieren mit dem Kommando Verify (leere Kommandodaten). Einem DIN NIA17.4 Votum gemäß wurde auf eine derartige Kombination verzichtet.

Das Kommando Get Pin Status zeigt in den Antwortdaten an,

* ob der Sicherheitszustand des Passwortobjektes gesetzt ist.
* welchen Wert das Attribut *retryCounter* besitzt.
* ob ein Passwortobjekt mit einem Transportschutz versehen ist und falls ja, welches Transportschutzverfahren vom Passwortobjekt verwendet wird.

Welches Passwortobjekt von diesem Kommando betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Auslesen des Status eines Passwortobjektes

In dieser Variante enthält die APDU des Get Pin Status-Kommandos einen Parameter:

1. (N077.800) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N077.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 143 verwendet werden.

Tabelle 143: Get Pin Status

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´20´ | Instruction Byte (dies ist derselbe Wert wie beim Kommando Verify) |
| P1 | ´00´ | – |
| P2 | ´XX´ | *passwordReference* |

#### Antwort der Karte auf Auslesen des PIN Status

Tabelle 144: Get Pin Status Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 Cx´: ´62 C1´ ´62 C7´ | TransportStatus Transport–PIN Leer–PIN | Passwortobjekt eingeschaltet, Passwortobjekt ist mit Transportschutz versehen, Transportschutzverfahren enthalten im Least Significant Nibble (siehe auch 8.2.5) |
| ´62 D0´ | PasswordDisabled | Passwortobjekt ausgeschaltet, Verifikation nicht erforderlich |
| ´63 Cx´ | RetryCounter | Passwortobjekt eingeschaltet, Passwortobjekt ohne Transportschutz (das bedeutet regularPassword), Wert des Fehlbedienungszählers enthalten im Least Significant Nibble |
| ´90 00´ | NoError | Passwortobjekt eingeschaltet, Sicherheitszustand gesetzt |

Tabelle 145: Get Pin Status Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N078.000) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N078.100) K\_COS
   1. Das COS MUSS die Get Pin Status-Variante aus 14.6.4.1 unterstützen.
   2. Das COS KANN weitere Get Pin Status-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N078.200) K\_COS   
   Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N078.300) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N078.400) K\_COS   
   Falls das Attribut *affectedObject.flagEnabled* (siehe (N015.700)) den Wert False besitzt, dann MUSS als Trailer PasswordDisabled verwendet werden.
5. (N078.500) K\_COS   
   Falls *affectedObject.flagEnabled* (siehe (N015.700)) den Wert True besitzt und *affectedObject* in *globalPasswordList* (siehe (N029.900)i) oder in *dfSpecificPasswordList* (siehe (N029.900)j) enthalten ist und dort das Attribut *securityStatusEvaluationCounter* (siehe (N029.900)k) einen Wert ungleich null besitzt, dann MUSS als Trailer NoError verwendet werden.
6. (N078.600) K\_COS   
   Falls das Attribut *affectedObject.transportStatus* (siehe (N015.600)) einen Wert ungleich regularPassword besitzt, genau dann MUSS als Trailer TransportStatus verwendet werden mit TransportStatus = ´62 Cx´. Dabei ist ´x´ durch die Codierung von *affectedObject.transportStatus* gemäß Tabelle 144 zu ersetzen.
7. (N078.700) K\_COS   
   Falls das Attribut *affectedObject.transportStatus* (siehe (N015.600)) den Wert regularPassword besitzt, genau dann MUSS als Trailer RetryCounter verwendet werden mit RetryCounter = ´63 Cx´. Dabei ist ´x´ zu ersetzen durch das Minimum der beiden Zahlen 15 = ´F´ und *affectedObject.retryCounter*.
8. (N078.800) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 145 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 145 MUSS eine höhere Priorität als PasswordDisabled haben.
   3. PasswordDisabled MUSS eine höhere Priorität als NoError haben.
   4. NoError MUSS eine höhere Priorität als TransportStatus haben.
   5. TransportStatus MUSS eine höhere Priorität als RetryCounter haben.
9. Gemäß (N078.600), (N078.700) und (N078.800) ist es nicht möglich den Fehlbedienungszähler auszulesen, wenn Transportschutz besteht.

### Reset Retry Counter

Das Kommando Reset Retry Counter setzt das Attribut *retryCounter* eines Passwortobjektes auf den Startwert *startRetryCounter*. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Kommando gibt es in den Varianten

* mit und ohne PUK
* mit und ohne neuen Wert für das Attribut *secret* des Passwortobjektes.

Die Variante „mit PUK“ wird typischerweise verwendet, wenn eine „nicht technische“ Instanz (etwa der Kartenbesitzer) das Recht zur Durchführung dieser Aktion besitzt.

Die Variante „ohne PUK“ wird typischerweise verwendet, wenn eine technische Instanz (etwa das CMS) diese Aktion durchzuführen hat. In den Zugriffsbedingungen wird dann typischerweise zumindest eine Rollenauthentisierung gefordert.

Die Variante „mit neuem Geheimnis“ wird typischerweise gewählt, wenn der Karteninhaber sein Passwort definitiv vergessen hat.

Die Variante „ohne neues Geheimnis“ wird typischerweise gewählt, wenn es der Instanz, welche zum Rücksetzen des Fehlbedienungszählers berechtigt ist, verboten ist, durch das Passwort geschützte Aktionen auszuführen. Dies ist häufig im Umfeld qualifizierter Signaturanwendungen und dem Signaturpasswort anzutreffen.

#### Use Case Entsperren mit PUK, mit neuem Geheimnis

In dieser Variante enthält die APDU des Reset Retry Counter-Kommandos drei Parameter:

1. (N078.900) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N079.000) K\_externeWelt {K\_Karte}   
   Der Parameter *PUK* enthält ein Geheimnis, welches diese Aktion autorisiert.
3. (N079.100) K\_externeWelt {K\_Karte}   
   Der Parameter *newSecret* enthält das neue Benutzergeheimnis.
4. (N079.200) K\_externeWelt {K\_Karte}   
   Die Parameter *PUK* und *newSecret* MÜSSEN gemäß (N008.100) codiert sein.
5. (N079.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 146 verwendet werden.

Tabelle 146: Reset Retry Counter, mit PUK, mit *newSecret*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Data enthält *PUK* und neues Benutzergeheimnis |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *PUK*   ||   *newSecret* |

#### Use Case Entsperren mit PUK, ohne neues Geheimnis

In dieser Variante enthält die APDU des Reset Retry Counter-Kommandos zwei Parameter:

1. (N079.400) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N079.500) K\_externeWelt {K\_Karte}   
   Der Parameter *PUK* enthält ein Geheimnis, welches diese Aktion autorisiert.
3. (N079.600) K\_externeWelt {K\_Karte}   
   Der Parameter *PUK* MUSS gemäß (N008.100) codiert sein.
4. (N079.700) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 147 verwendet werden.

Tabelle 147: Reset Retry Counter, mit PUK, ohne *newSecret*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´01´ | Data enthält nur *PUK* |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *PUK* |

#### Use Case Entsperren ohne PUK, mit neuem Geheimnis

In dieser Variante enthält die APDU des Reset Retry Counter-Kommandos zwei Parameter:

1. (N079.800) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N079.900) K\_externeWelt {K\_Karte}   
   Der Parameter *newSecret* enthält das neue Benutzergeheimnis.
3. (N080.000) K\_externeWelt {K\_Karte}   
   Der Parameter *newSecret* MUSS gemäß (N008.100) codiert sein.
4. (N080.100) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 148 verwendet werden.

Tabelle 148: Reset Retry Counter, ohne PUK, mit *newSecret*

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´02´ | Data enthält nur neues Benutzergeheimnis |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *newSecret* |

#### Use Case Entsperren ohne PUK, ohne neues Geheimnis

In dieser Variante enthält die APDU des Reset Retry Counter-Kommandos einen Parameter:

1. (N080.200) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N080.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 149 verwendet werden.

Tabelle 149: Reset Retry Counter, ohne PUK, ohne *newSecret*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2C´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´03´ | Kommandodatenfeld fehlt |
| P2 | ´XX´ | *passwordReference* |

#### Antwort der Karte auf Entsperren eines Benutzergeheimnisses

Tabelle 150: Reset Retry Counter Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | WrongSecretWarning | *PUK* ist falsch |
| ´90 00´ | NoError | Erfolgreiches Rücksetzen des Fehlbedienungszählers |

Tabelle 151: Reset Retry Counter Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | CommandBlocked | Bedienungszähler der PUK abgelaufen |
| ´69 85´ | LongPassword | newData enthält ein zu langes Passwort |
| ´69 85´ | ShortPassword | newData enthält ein zu kurzes Passwort |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N080.400) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N080.500) K\_COS
   1. Das COS MUSS die Reset Retry Counter-Varianten aus 14.6.5.1, 14.6.5.2, 14.6.5.3 und 14.6.5.4 unterstützen.
   2. Das COS KANN weitere Reset Retry Counter-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N080.600) K\_COS   
   Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N080.700) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N080.800) K\_COS   
   Wenn die konditional vorhandene und in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die
   1. kleiner als *affectedObject.minimumLength* ist, genau dann MUSS das Kommando mit dem Trailer ShortPassword terminieren.
   2. größer als *affectedObject.maximumLength* ist, genau dann MUSS das Kommando mit dem Trailer LongPassword terminieren.
5. (N080.900) K\_COS   
   Mittels clearPasswordStatus( *affectedObject* ) MUSS der Sicherheitsstatus zurückgesetzt werden.
6. (N081.000) K\_COS   
   Wenn das Kommandodatenfeld einen Parameter *PUK* enthält, genau dann MUSS dieser mit dem Attribut *affectedObject.PUK* verglichen werden.
   1. Wenn *affectedObject.pukUsage* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer CommandBlocked terminieren.
   2. *affectedObject.pukUsage* MUSS mit Transaktionsschutz um eins dekrementiert werden.
   3. Wenn der Vergleich fehlschlägt, genau dann MUSS das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert ´F´ gesetzt werden, wenn *affectedObject.pukUsage* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.pukUsage*.
7. Die Bearbeitungsreihenfolge der Punkte (N080.700), (N080.800), (N080.900) und (N081.000) ist COS spezifisch. Deshalb ist es COS spezifisch, ob in gewissen Fehlerfällen der Sicherheitsstatus von affectedObject zurückgesetzt wird oder nicht.
8. (N081.100) K\_COS   
   Das Attribut *affectedObject.retryCounter* MUSS auf den Wert *affectedObject.startRetryCounter* gesetzt werden.
9. (N081.200) K\_COS   
   Wenn das Kommandodatenfeld einen Parameter *newSecret* enthält, genau dann MUSS das Attribut
   1. *affectedObject.secret* auf den in *newSecret* codierten Wert gesetzt werden.
   2. *affectedObject.transportStatus* auf den Wert „regularPassword“ (siehe (N009.500)) gesetzt werden, falls dieses Attribut einen anderen Wert besitzt.
10. (N081.300) K\_COS   
    Die persistenten Änderungen in (N081.000), (N081.100) und (N081.200) MÜSSEN mit Transaktionsschutz ausgeführt werden.
11. (N081.400) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
12. (N081.500) K\_COS   
    Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
    1. entweder als Trailer MemoryFailure verwendet werden,
    2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
13. (N081.600) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
14. (N081.700) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 151 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 151 MUSS eine höhere Priorität als WrongSecretWarning haben.
    3. WrongSecretWarning MUSS eine höhere Priorität als NoError haben.
15. (N081.800) K\_COS   
    Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für die möglichen Änderungen von *secret*, *transportStatus* und *retryCounter*:
    1. Alle Änderungen SOLLEN gemeinsam in einer Transaktion geändert werden.
    2. Jede Änderung KANN in einer eigenen Transaktion durchgeführt werden. In diesem Fall MUSS *retryCounter* als letztes geändert werden.

### Verify

Das Kommando Verify vergleicht das Attribut *secret* eines Passwortobjektes mit Daten, die im Datenfeld der Kommandonachricht enthalten sind. Falls der Vergleich erfolgreich ist, wird der Sicherheitszustand der Karte geändert. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist.

#### Use Case Vergleich eines Benutzergeheimnisses

In dieser Variante enthält die APDU des Verify Kommandos zwei Parameter:

1. (N081.900) K\_externeWelt {K\_Karte}   
   Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.
2. (N082.000) K\_externeWelt {K\_Karte}   
   Der Parameter *verificationData* enthält das Benutzergeheimnis.
3. (N082.100) K\_externeWelt {K\_Karte}   
   Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.
4. (N082.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 152 verwendet werden.

Tabelle 152: Verify, Vergleich eines Benutzergeheimnisses

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´20´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Data enthält Verifikationsdaten |
| P2 | ´XX´ | *passwordReference* |
| Data | ´XX…XX´ | *verificationData* |

#### Antwort der Karte auf Vergleich eines Benutzergeheimnisses

Tabelle 153: Verify Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | WrongSecretWarning | *verificationData* ist falsch |
| ´90 00´ | NoError | Erfolgreicher Vergleich |

Tabelle 154: Verify Antwort-APDU im Fehlerfall

| **Trailer** | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | PasswordBlocked | Abgelaufener Fehlbedienungszähler |
| ´69 85´ | PasswordNotUsable | Passwort mit Transportschutz versehen |
| ´6A 88´ | PasswordNotFound | Adressiertes Passwort wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N082.300) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N082.400) K\_COS
   1. Das COS MUSS die Verify-Variante aus 14.6.6.1 unterstützen.
   2. Das COS KANN weitere Verify-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N082.500) K\_COS   
   Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference* ). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
3. (N082.600) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N082.700) K\_COS   
   Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.
5. (N082.800) K\_COS   
   Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.
6. (N082.900) K\_COS   
   Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.
   1. Wenn der Vergleich fehlschlägt, genau dann MUSS
      1. *affectedObject.retryCounter* um eins dekrementiert werden und

* + 1. clearPasswordStatus( *affectedObject* ) ausgeführt werden und
    2. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert ´F´ gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.
  1. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
  2. Wenn der Vergleich erfolgreich ist, genau dann MUSS

* + 1. setPasswordStatus( *affectedObject* ) ausgeführt werden und
    2. das Attribut *affectedObject.retryCounter* auf den Wert *affectedObject.startRetryCounter* gesetzt werden und
    3. die Änderung von *affectedObject.retryCounter* mit Transaktionsschutz ausgeführt werden.

1. (N083.000) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
2. (N083.100) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
3. (N083.200) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
4. (N083.300) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 154 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 154 MUSS eine höhere Priorität als WrongSecretWarning haben.
   3. WrongSecretWarning MUSS eine höhere Priorität als NoError haben.

## Komponentenauthentisierung

### External Authenticate / Mutual Authenticate

Die Kommandos External Authenticate und Mutual Authenticate prüfen die Authentizität einer externen Instanz anhand der Antwort auf ein von der Karte generiertes Token, mittels eines symmetrischen oder öffentlichen Schlüssels. Der Schlüssel wird vor der Authentisierungsoperation ausgewählt. Dies geschieht vor dem Senden dieses Kommandos durch ein MSE-Set-Kommando (siehe 14.9.9.4, 14.9.9.5, 14.9.9.6). Die Antwort der externen Instanz auf das von der Karte generierte Token ist als Parameter in der Kommandonachricht enthalten.

1. In [gemSpec\_eGK\_P1] wurden die Varianten EXTERNAL versus MUTUAL anhand eines Algorithm Identifiers unterschieden. Hier werden die Varianten, wie schon in [HBA\_P1], anhand des LeFeldes unterschieden
2. Der Wert ´82´ für das INS-Byte wird für die Kommandovarianten External Authenticate und Mutual Authenticate verwendet. Die Varianten lassen sich anhand der Existenz des LeFeldes unterscheiden. Davon wird in (N084.400) und (N084.410) Gebrauch gemacht. Dies schließt die Verwendung des Protokolls T=0 für eGK, HBA und SMC aus. Dies erscheint vertretbar, da es derzeit für die Verwendung des Protokolls T=0 keine Anforderung gibt.

#### Use Case externe Authentisierung ohne Antwortdaten

In dieser Variante enthält die APDU des External Authenticate-Kommandos einen Parameter:

1. (N083.400) K\_externeWelt {K\_Karte}   
   Der Parameter *cmdData* enthält die Antwort der externen Instanz. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *cmdData* ist abhängig von der mittels (N101.300) oder (N101.800) ausgewählten *algId*. Wenn *algId* gleich
   1. elcRoleCheck ist, dann MUSS unter Hinzuziehung der Domainparameter des öffentlichen Schlüssels gelten: 8 OctetLength(*cmdData*) = 2 *domainParameter*.τ.
   2. Option\_RSA\_CVC, rsaRoleCheck ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.
   3. Option\_DES, rsaSessionkey4SM ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.
2. (N083.402) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Der Parameter *cmdData* enthält die Antwort der externen Instanz. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *cmdData* ist abhängig von der mittels (N101.300) oder (N101.800) ausgewählten *algId*. Wenn *algId* gleich
   1. aesSessionkey4TC ist, dann MUSS *cmdData* gleich 120 Oktett lang sein.
   2. Option\_DES, desSessionkey4TC ist, dann MUSS *cmdData* gleich 104 Oktett lang sein.
   3. Option\_DES, rsaSessionkey4TC ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.
3. (N083.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 155 verwendet werden.

Tabelle 155: External Authenticate ohne Antwortdaten

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´82´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Information zum Algorithmus bereits in der Karte vorhanden |
| P2 | ´00´ | Schlüsselreferenz bereits in der Karte vorhanden |
| Data | ´XX…XX´ | *cmdData* |

#### Use Case externe Authentisierung mit Antwortdaten

In dieser Variante enthält die APDU des Mutual Authenticate-Kommandos zwei Parameter:

1. (N083.600) K\_externeWelt {K\_Karte}   
   Der Parameter *cmdData* enthält die Antwort der externen Instanz. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *cmdData* ist in abhängig von der mittels (N102.300) ausgewählten *algId*. Wenn *algId* gleich
   1. aesSessionkey4SM ist, dann MUSS *cmdData* 120 Oktett lang sein.
   2. Option\_DES, desSessionkey4SM ist, dann MUSS *cmdData* 104 Oktett lang sein.
2. (N083.700) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
3. (N083.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 156 verwendet werden.

Tabelle 156: Mutual Authenticate mit Antwortdaten

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´82´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Information zum Algorithmus bereits in der Karte vorhanden |
| P2 | ´00´ | Schlüsselreferenz bereits in der Karte vorhanden |
| Data | ´XX…XX´ | *cmdData* |
| Le | ´00´ | length, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf externe Authentisierung

Tabelle 157: External Authenticate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *authData* | Authentisierungsdaten (konditional, abhängig von *algId*) |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 00´ | AuthenticationFailure | Authentisierung fehlgeschlagen |
| ´90 00´ | NoError | Erfolgreiche Authentisierung |

Tabelle 158: External Authenticate Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | KeyExpired | Gültigkeitszeitraum des Schlüssels ist abgelaufen |
| ´69 85´ | NoKeyReference | Kein Authentisierungsschlüssel ausgewählt |
| ´69 85´ | NoPrkReference | Kein Entschlüsselungsschlüssel ausgewählt |
| ´69 85´ | NoRandom | Keine Zufallszahl vorhanden (siehe auch 15.1) |
| ´69 85´ | WrongRandomLength | Zufallszahl hat die falsche Länge |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | KeyNotFound | Authentisierungsschlüssel nicht gefunden |
| ´6A 88´ | PrKNotFound | Entschlüsselungsschlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N083.900) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N084.000) Kommandovarianten
   1. K\_COS   
      Das COS MUSS die External Authenticate-Variante aus 14.7.1.1 und 14.7.1.2 unterstützen.
   2. Das COS KANN weitere External Authenticate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N084.100) K\_COS   
   Wenn *RND.ICC* den Wert "NoRandom" besitzt (siehe (N099.300), (N029.900)b und Kapiel 15), genau dann MUSS das Kommando mit dem Trailer NoRandom terminieren.
3. (N084.200) K\_COS   
   Wenn *channelContext.keyReferenceList.externalAuthenticate*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchKey(   
          *channelContext*.*currentFolder,*   
          *keyReferenceList.externalAuthenticate.keyReference*,   
          *keyReferenceList.externalAuthenticate.algID*   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
4. (N084.220) K\_COS   
   Das Kommando MUSS mit dem Trailer WrongRandomLength terminieren, falls *RND.ICC* eine Länge von
   1. acht Oktetten besitzt und *keyReferenceList.externalAuthenticate.algID* Element der folgenden Menge ist: {aesSessionkey4\*, elcRoleCheck}.
   2. 16 Oktetten besitzt und *keyReferenceList.externalAuthenticate.algID* Element der folgenden Menge ist: {desSessionkey4\* (Option\_DES), rsa\* (Option\_RSA\_CVC)}.
5. (N084.280) K\_COS   
   Das COS KANN den Sicherheitszustand mittels clearSecurityStatusKey(*affectedObject*) vor der Auswertung der Zugriffsregel (siehe (N084.300)) oder vor der Auswertung der Kommandodaten (siehe (N084.400)) zurücksetzen. In diesem Fall ist das Löschen des Sicherheitszustandes gemäß (N084.500) nicht erforderlich.
6. (N084.300) K\_COS   
   Falls AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
7. (N084.400) K\_COS   
   Die Antwort der externen Instanz ist in *cmdData* enthalten. Falls die Kommando APDU kein LeFeld enthält, mithin also ein Case 3 gemäß 11.7.3 vorliegt, dann handelt es sich um die External Authenticate Variante des Kommandos und es gilt *authData* = ´´ (leerer Oktettstring). In diesem Fall wird *cmdData* wie folgt verarbeitet: Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert

* 1. elcRoleCheck besitzt, dann gilt:
     1. Falls *affectedObject*.*expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
     2. *cmdData* MUSS wie folgt auf *R* und *S* aufgeteilt werden:
        1. *cmdData* = *R*   ||   *S*.
        2. OctetLength( *R* ) = OctetLength( *S* ).
     3. Setze *PuK* = *affectedObject.publicKey*.
     4. Setze *hash* = I2OS(   
          OS2I( *RND.ICC*   ||   *iccsn8*   ||   elcRoleCheck),   
          *PuK*.*domainParameter*.τ / 8   
          ).  
        *Hinweis:* iccsn8 *ist in (N019.900)c definiert.*
     5. Setze *out* = ELC\_VER\_SIG(*PuK*, *R*, *S*, *hash*).   
        Falls *out* den Wert *False* besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.

* 1. Option\_RSA\_CVC, rsaRoleCheck besitzt, dann gilt mit   *PuK* = *affectedObject.publicKey*
     1. Setze  *M2* = *RND.ICC*   ||   *iccsn8*. *Hinweis:* iccsn8 *ist in (N019.900)c definiert.*
     2. ( *out*, *M* ) = RSA\_ISO9796\_2\_DS1\_VERIFY( *PuK*, *cmdData*, *M2* )   
        Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.

* 1. Option\_DES, rsaSessionkey4SM besitzt, dann gilt *authData* = ´´ (leerer Oktettstring) und es werden folgende Schritte ausgeführt:
     1. Wenn *channelContext.keyReferenceList.internalAuthenticate*
        1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoPrkReference terminieren.
        2. nicht leer ist, dann wird *tmpObject* = SearchKey(   
               *channelContext*.*currentFolder,*   
               *keyReferenceList.internalAuthenticate.keyReference*,   
               *keyReferenceList.externalAuthenticate.algID*   
           ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
           1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.
           2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
     2. Das COS KANN eine Zugriffsregelprüfung für den privaten Schlüssel in *tmpObject* gemäß AccessRuleEvaluation( *tmpObject*, *CLA*, *INS*, *P1*, *P2* ) durchführen oder unterlassen (siehe Hinweis (110):).
     3. Setze *tmpData* = *cmdDataPrK.d* mod *PrK.n*,   
        mit *PrK* = *tmpObject.privateKey*
     4. Setze *M2* = *RND.ICC*   ||   *iccsn8*
     5. ( *out*, *M* ) = RSA\_ISO9796\_2\_DS1\_VERIFY( *PuK*, *tmpData*, *M2* )   
        Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
     6. *M* wird aufgeteilt in *M* = *M1* || *M2*. Die 64 LSByte in *M1* MÜSSEN als KD.e an den Secure Messaging Layer übergeben werden (siehe 13.1).

1. (N084.402) K\_COS, Option\_Kryptobox   
   Die Antwort der externen Instanz ist in *cmdData* enthalten. Falls die Kommando APDU kein LeFeld enthält, mithin also ein Case 3 gemäß 11.7.3 vorliegt, dann handelt es sich um die External Authenticate Variante des Kommandos und es gilt *authData* = ´´ (leerer Oktettstring). In diesem Fall wird *cmdData* wird wie folgt weiterverarbeitet: Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert

* 1. aesSessionkey4TC besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
     1. *cmdData* wird wie folgt aufgeteilt:
        1. *cmdData* = *C1*   ||   *MAC1*
        2. OctetLength( *MAC1* ) = 8.
     2. *out* = VerifyCMAC\_IsoPadding( *affectedObject.macKey*, *MAC1*, *C1* )
     3. Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
     4. *R* = AES\_CBC\_DEC( *affectedObject.encKey*, 0, *C1* )
     5. *R* wird wie folgt aufgeteilt:
        1. *R* = RND.ext   ||   ICCSN8.ext   ||   RND.int   ||   ICCSN8.int   ||   KD.e
        2. Es sei 16 = OctetLength( RND.ext )
        3. Es sei 8 = OctetLength( ICCSN8.ext )
        4. Es sei 16 = OctetLength( RND.int )
        5. Es sei 8 = OctetLength( ICCSN8.int )
        6. Es sei 64 = OctetLength( KD.e )
     6. Das Kommando MUSS mit AuthenticationFailure terminieren, falls
        1. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
        2. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist, oder
        3. ICCSN8.int identisch zu ICCSN8.ext ist.
     7. KD.e MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).

* 1. Option\_DES, desSessionkey4TC besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
     1. *cmdData* wird wie folgt aufgeteilt:
        1. *cmdData* = *C1*   ||   *MAC1*
        2. OctetLength( *MAC1* ) = 8.
     2. *out* = VERIFY\_Retail\_MAC( *affectedObject.macKey*, *MAC1*, *C1* )
     3. Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren, sonst MÜSSEN die folgenden Schritte durchgeführt werden:
     4. *R* = 3TDES\_CBC\_DEC( *affectedObject.encKey*, 0, *C1* )
     5. *R* wird wie folgt aufgeteilt:
        1. *R* = RND.ext   ||   ICCSN8.ext   ||   RND.int   ||   ICCSN8.int   ||   KD.e
        2. Es sei 8 = OctetLength( RND.ext )
        3. Es sei 8 = OctetLength( ICCSN8.ext )
        4. Es sei 8 = OctetLength( RND.int )
        5. Es sei 8 = OctetLength( ICCSN8.int )
        6. Es sei 64 = OctetLength( KD.e )
     6. Das Kommando MUSS mit AuthenticationFailure terminieren, falls
        1. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
        2. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist, oder
        3. ICCSN8.int identisch zu ICCSN8.ext ist.
     7. KD.e MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).

* 1. Option\_DES, rsaSessionkey4TC besitzt, dann MUSS im Rahmen der Kommandobearbeitung identisch zu rsaSessionkey4SM verfahren werden.

1. (N084.410) K\_COS   
   Die Antwort der externen Instanz ist in *cmdData* enthalten. Falls die Kommando APDU ein LeFeld enthält, mithin also ein Case 4 gemäß vorliegt, dann handelt es sich um die Mutual Authenticate Variante des Kommandos. In diesem Fall wird *cmdData* wie folgt weiterverarbeitet, wobei *authData* berechnet wird:   
   Wenn *channelContext.keyReferenceList.externalAuthenticate.algID* den Wert

* 1. aesSessionkey4SM besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
     1. *cmdData* wird wie folgt aufgeteilt:
     2. *cmdData* = *C1* || *MAC1*

* + 1. OctetLength( *MAC1* ) = 8.
    2. *out* = VerifyCMAC\_IsoPadding( *affectedObject.macKey*, *MAC1*, *C1* )
    3. Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren, sonst MÜSSEN die folgendene Schritte durchgeführt werden:
    4. *S* = AES\_CBC\_DEC( *affectedObject.encKey*, 0, *C1* )
       1. *S* wird wie folgt aufgeteilt:
       2. *S* = RND.ext || ICCSN8.ext || RND.int || ICCSN8.int || KD.e
       3. Es sei 16 = OctetLength( RND.ext )
       4. Es sei 8 = OctetLength( ICCSN8.ext )
       5. Es sei 16 = OctetLength( RND.int )
       6. Es sei 8 = OctetLength( ICCSN8.int )
       7. Es sei 64 = OctetLength( KD.e )
    5. Das Kommando MUSS mit AuthenticationFailure terminieren, falls
       1. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
       2. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist.
       3. ICCSN8.int identisch zu ICCSN8.ext ist.
    6. Setze KD.i = RAND( 64 )
    7. Setze *R* = RND.int || ICCSN8.int || RND.ext || ICCSN8.ext || KD.i
    8. Setze *C2* = AES\_CBC\_ENC( *affectedObject.encKey*, 0, *R* )
    9. Setze *MAC2* = CalculateCMAC\_IsoPadding( *affectedObject.macKey*, *C2* )
    10. Setze *authData* = *C2*   ||   *MAC2*
    11. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).

* 1. Option\_DES, desSessionkey4SM besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
     1. *cmdData* wird wie folgt aufgeteilt:
     2. *cmdData* = *C1* || *MAC1*
     3. OctetLength( *MAC1* ) = 8.
     4. *out* = VERIFY\_Retail\_MAC( *affectedObject.macKey*, *MAC1*, *C1* ). Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
     5. Der Oktettstring *C1* in den Kommandodaten wird wie folgt behandelt:
        1. *S* = 3TDES\_CBC\_DEC( *affectedObject.encKey*, 0, *C1* )
        2. *S* wird wie folgt aufgeteilt:
        3. *S* = RND.ext || ICCSN8.ext || RND.int || ICCSN8.int || KD.e
        4. Es sei 8 = OctetLength( RND.ext )
        5. Es sei 8 = OctetLength( ICCSN8.ext )
        6. Es sei 8 = OctetLength( RND.int )
        7. Es sei 8 = OctetLength( ICCSN8.int )
        8. Es sei 64 = OctetLength( KD.e )
     6. Das Kommando MUSS mit AuthenticationFailure terminieren, falls
        1. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
        2. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist.
        3. ICCSN8.int identisch zu ICCSN8.ext ist.
     7. Setze KD.i = RAND( 64 )
     8. Setze *R* = RND.int || ICCSN8.int || RND.ext || ICCSN8.ext || KD.i
     9. Setze *C2* = 3TDES\_CBC\_ENC( *affectedObject.encKey*, 0, *R* )
     10. Setze *MAC2* = CALCULATE\_Retail\_MAC( *affectedObject.macKey*, *C2* )
     11. Setze *authData* = *C2*   ||   *MAC2*
     12. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).

1. (N084.500) K\_COS   
   Falls das Kommando mit dem Trailer AuthenticationFailure terminiert, genau dann MUSS clearSecurityStatusKey( *affectedObject* ) ausgeführt werden.
2. (N084.600) K\_COS   
   Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus( *affectedObject* ) ausgeführt werden.
3. (N084.700) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS der (möglicherweise leere) Oktettstring *authData* verwendet werden.
4. (N084.800) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
5. (N084.900) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 158 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 158 MUSS eine höhere Priorität als NoError haben.
6. (N084.910) K\_COS   
   Das COS MUSS spätestens nach dem Versenden der Antwort-APDU und vor dem Empfang der nächsten Kommando-APDU *RND.ICC* auf den Wert "NoRandom" setzen (siehe (N029.900)b).
7. Im Rahmen dieser Spezifikation sind Authentisierungssequenzen nicht unterbrechbar (siehe (N104.600)). Das Authentisierungsprotokoll rsaSessionkey4SM gemäß 15.4.3 führt zunächst ein interne Authentisierung durch, wobei eine Zugriffsregelprüfung für den privaten Schlüssel stattfindet (siehe (N086.800)). Deshalb ergäbe eine Zugriffsregelprüfung gemäß (N084.400)c.2 in allen praxisrelevanten Fällen stets True. Funktional ist eine Zugriffsregelprüfung gemäß (N084.400)c.2 deshalb nicht erforderlich. Möglicherweise ist aber im Rahmen einer Evaluierung nachzuweisen, dass ein Unterlassen der Zugriffsregelprüfung gemäß (N084.400)c.2 die Sicherheit nicht unzulässig mindert.
8. Bei der Schlüsselsuche in (N084.400)c.1.ii wird nach einem privaten Schlüsselobjekt gesucht. Trotzdem wird algID aus keyReferenceList.externalAuthenticate entnommen, damit sicher gestellt ist, dass auch das private Schlüsselobjekt den hier zum Tragen kommenden Algorithmus unterstützt. Aus der übrigen Spezifikation ergibt sich, dass dieser Algorithmus Element von {rsaSessionkey4SM, rsaSessionkey4TC} ist.

### General Authenticate

Das Kommando General Authenticate wird in komplexen Authentisierungsprotokollen eingesetzt um die Authentizität einer Smartcard nachzuweisen oder um die Authentizität der „externen Welt“ zu prüfen. Allen hier behandelten Authentisierungsprotokollen ist gemeinsam, dass sie aus mehr als einem Schritt bestehen und für jeden Schritt ein General Authenticate verwendet wird. Alle Kommandos des Authentisierungsprotokolls sind mittels Command Chaining (siehe 11.8) miteinander verknüpft. Anhand den Einträgen in *channelContext*.*keyReferenceList* und der Schrittnummer erkennt das COS, welche Aktion ansteht, wie die Daten in der Kommandonachricht zu verarbeiten sind und welche Antwortdaten zu erstellen sind. Von der externen Welt wird gefordert, dass sie die Abfolge des Authentisierungsprotokolls einhält.

#### Gegenseitige Authentisierung mittels PACE für Endnutzerkarten

Dieser Abschnitt behandelt die Kommandonachrichten für die in Abbildung 7 mit "COSa *PICC*" bezeichnete Seite, die in [BSI-TR-03110-2#3.2] "MRTD Chip (PICC)“ genannt wird. Die Kommandonachrichten für die andere Seite werden in 14.7.2.4 beschrieben. Nach erfolgreichem Abschluss dieses Authentisierungsprotokolls liegen in der Karte Sessionkeys vor, die im Rahmen von Secure Messaging verwendbar sind.

##### Use Case PACE für Endnutzerkarten, Schritt 1a

In dieser Variante wird der erste Schritt des PACE Authentisierungsprotokolls für eine Endnutzerkarte durchgeführt. Der komplette Ablauf ist in 15.4.2 beschrieben. Der erste Schritt entspricht [BSI-TR-03110-3#B.1, Step 1, B.1.1] encrypted nonce.

1. (N085.000) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS zwei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.001) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 159 verwendet werden.

Tabelle 159: General Authenticate PACE Endnutzerkarte, Schritt 1a

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ 00´ |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case PACE für Endnutzerkarten, Schritt 2a

In dieser Variante wird der zweite Schritt von PACE für eine Endnutzerkarte durchgeführt, der [BSI-TR-03110-3#B.1, Step 2, B.1.2.1] entspricht, map nonce.

1. (N085.002) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS drei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *~PK1PCD* ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.064)b.2 kein Fehler auftritt.
   3. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.003) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 160 verwendet werden.

Tabelle 160: General Authenticate PACE Endnutzerkarte, Schritt 2a

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ L7C (81 – L81 ‑ P2OS(*~PK1PCD*))´, DER codiertes Datenfeld |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case PACE für Endnutzerkarten, Schritt 3a

In dieser Variante wird der dritte Schritt von PACE für eine Endnutzerkarte durchgeführt, der [BSI-TR-03110-3#B.1, Step 3] entspricht, perform key agreement.

1. (N085.004) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS drei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *~PK2PCD* ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.064)c.1 kein Fehler auftritt.
   3. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.005) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 161 verwendet werden.

Tabelle 161: General Authenticate PACE Endnutzerkarte, Schritt 3a

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ L7C (83 – L83 ‑ P2OS(*~PK2PCD*))´, DER codiertes Datenfeld |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case PACE für Endnutzerkarten, Schritt 4a

In dieser Variante wird der vierte und letzte Schritt von PACE für eine Endnutzerkarte durchgeführt, der [BSI-TR-03110-3#B.1, Step 4] entspricht, mutual authentication.

1. (N085.006) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS zwei Parameter enthalten:
   1. Der Parameter *TPCD* ist ein Oktettstring der Länge acht mit beliebigem Inhalt.
   2. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.007) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 162 verwendet werden.

Tabelle 162: General Authenticate PACE Endnutzerkarte, Schritt 4a

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C – 0A (85 – 08 ‑ *TPCD*)´ |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Gegenseitige Authentisierung mittels ELC Schlüsseln

##### Use Case gegenseitige ELC-Authentisierung, Schritt 1

In dieser Variante wird der erste Schritt einer gegenseitigen Authentisierung mittels ELC-Schlüssel durchgeführt, bei der auch Sessionkeys ausgehandelt werden. Der komplette Ablauf ist in 15.4.4 beschrieben.

1. (N085.010) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS drei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *keyRef* MUSS zwölf Oktett lang sein. Er enthält eine Schlüsselreferenz für einen öffentlichen Schlüssel, der mittels CV-Zertifikat importiert wurde.
   3. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.012) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 163 verwendet werden.

Tabelle 163: General Authenticate gegenseitige ELC-Authentisierung, Schritt 1

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ 0E ‑( C3 ‑ 0C ‑ *keyRef* )´ |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case gegenseitige ELC-Authentisierung, Schritt 2

In dieser Variante wird der zweite und letzte Schritt einer gegenseitigen Authentisierung mittels ELC-Schlüssel durchgeführt, bei der auch Sessionkeys ausgehandelt werden.

1. (N085.014) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS einen Parameter enthalten:
   1. Der Parameter *ephemeralPK\_oponent* MUSS einen Punkt auf derselben elliptischen Kurve enthalten, wie der öffentliche Schlüssel, der in Schritt 1 selektiert wurde.
2. (N085.016) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 164 verwendet werden.

Tabelle 164: General Authenticate gegenseitige ELC-Authentisierung, Schritt 2

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ L7C ‑ ( 85 ‑ L85 ‑ *ephemeralPK\_oponent* )´, DER codiertes Datenfeld |

#### Authentisierung für asynchrone, symmetrische Kartenadministration

##### Use Case Authentisierung für asynchrone, sym. Administration, Schritt 1

In dieser Variante wird der erste Schritt durchgeführt, bei der Sessionkeys symmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben.

1. (N085.020) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS zwei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.022) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 165 verwendet werden.

Tabelle 165: General Authenticate, asynchrone, sym. Admin., Schritt 1

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ 02 ‑ ( 81 ‑ 00 )´ |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case Authentisierung für asynchrone, sym. Administration, Schritt 2

In dieser Variante wird der zweite und letzte Schritt durchgeführt, bei der Sessionkeys symmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben.

1. (N085.024) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS einen Parameter enthalten:
   1. Der Parameter *cmdData* MUSS 76 Oktett lang sein.
2. (N085.026) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 166 verwendet werden.

Tabelle 166: General Authenticate, asynchrone, sym. Admin., Schritt 2

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ 4E ‑ ( 82 ‑ 4C ‑ *cmdData* )´ |

#### Gegenseitige Authentisierung mittels PACE für Sicherheitsmodule

Dieser Abchnitt behandelt die Kommandonachrichten für die in Abbildung 7 mit "COSb *PCD*" bezeichnete Seite, die in [BSI-TR-03110-2#3.2] "Terminal (PCD) genannt wird. Die Kommandonachrichten für die andere Seite werden in 14.7.2.1 beschrieben. Nach erfolgreichem Abschluss dieses Authentisierungsprotokolls liegen in der Karte Sessionkeys vor, die im Rahmen von PSO-Kommandos einen Trusted Channel unterstützen.

##### Use Case PACE für Sicherheitsmodule, Schritt 1b

In dieser Variante wird der erste Schritt des PACE Authentisierungsprotokolls für ein Sicherheitsmodul durchgeführt. Der komplette Ablauf ist in 15.4.2 beschrieben. Die Karte wird veranlasst ein ephemeres Schlüsselpaar zu generieren.

1. (N085.030) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS zwei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.031) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 167 verwendet werden.

Tabelle 167: General Authenticate PACE Sicherheitsmodul, Schritt 1b

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ 00´ |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case PACE für Sicherheitsmodule, Schritt 2b

In dieser Variante wird der zweite Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte wird veranlasst die *nonce* der Gegenseite zu rekonstruieren.

1. (N085.032) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS drei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *z* ist ein Oktettstring der Länge 16 mit beliebigem Inhalt.
   3. Der Parameter *CAN* ist ein OktettString mit einer Länge aus dem Intervall [1, 16] und beliebigem Inhalt.
2. (N085.033) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 168 verwendet werden.

Tabelle 168: General Authenticate PACE Sicherheitsmodul, Schritt 2b

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ L7C [(80 – 10 ‑ *z*) (C0 – LC0 – *CAN*)]´, DER codiertes Datenfeld |

##### Use Case PACE für Sicherheitsmodule, Schritt 3b

In dieser Variante wird der dritte Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte wird veranlasst ein ephemeres Schlüsselpaar für ephemere Domainparameter zu generieren.

1. (N085.034) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS drei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *~PK1PICC* ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.066)c.1 kein Fehler auftritt.
   3. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.035) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 169 verwendet werden.

Tabelle 169: General Authenticate PACE Sicherheitsmodul, Schritt 3b

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ L7C (82 – L82 ‑ P2OS(*~PK1PICC*))´, DER codiertes Datenfeld |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case PACE für Sicherheitsmodule, Schritt 4b

In dieser Variante wird der vierte Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte wird veranlasst Sessionkeys abzuleiten.

1. (N085.036) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS drei Parameter enthalten:
   1. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
   2. Der Parameter *~PK2PICC* ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.066)d.1 kein Fehler auftritt.
   3. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.
2. (N085.037) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 170 verwendet werden.

Tabelle 170: General Authenticate PACE Sicherheitsmodul, Schritt 4b

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´10´ | CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C ‑ L7C (84 – L84 ‑ P2OS(*~PK2PICC*))´, DER codiertes Datenfeld |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

##### Use Case PACE für Sicherheitsmodule, Schritt 5b

In dieser Variante wird der fünfte und letzte Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte überprüft den MAC der Gegenseite.

1. (N085.038) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS einen Parameter enthalten:
   1. Der Parameter *TPICC* ist ein Oktettstring der Länge acht mit beliebigem Inhalt.
2. (N085.039) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 168 verwendet werden.

Tabelle 171: General Authenticate PACE Sicherheitsmodul, Schritt 5b

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C – 0A (86 – 08 ‑ *TPICC*)´ |

#### Authentisierung für asynchrone, asymmetrische Kartenadministration

##### Use Case Authentisierung für asynchrone, asym. Administration, Schritt 1

In dieser Variante wird der erste Schritt durchgeführt, bei der Sessionkeys asymmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben. Die Kommando-APDU, deren Parameter und deren Bedeutung sind identisch zum Use Case in 14.7.2.2.1.

##### Use Case Authentisierung für asynchrone, asym. Administration, Schritt 2

In dieser Variante wird der zweite und letzte Schritt durchgeführt, bei der Sessionkeys asymmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben.

1. (N085.040) K\_externeWelt {K\_Karte}   
   Die APDU des General Authenticate-Kommandos MUSS einen Parameter enthalten:
   1. Der Parameter *cmdData* MUSS gemäß (N085.068)b.7 gewählt werden, wobei die Länge von KD.e in (N085.068)b.7.viii 64 Oktett betragen MUSS.
2. (N085.041) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 172 verwendet werden.

Tabelle 172: General Authenticate, asynchrone, asym. Admin., Schritt 2

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´86´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | no information given |
| P2 | ´00´ | no information given |
| Data | ´XX…XX´ | ´7C – 81F8 ‑ ( 82 ‑ 81F5 ‑ *cmdData* )´ falls brainpoolP256r1 ´7C – 820139 ‑ ( 82 ‑ 820135 ‑ *cmdData* )´ falls brainpoolP384r1 ´7C – 82013B ‑ ( 82 ‑ 820177 ‑ *cmdData* )´ falls brainpoolP512r1 |

#### Antwort der Karte auf generelle Authentisierung

Tabelle 173: General Authenticate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *responseData* | Antwortdaten vorhanden |
| ‑ | ‑ | keine Antwortdaten |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 00´ | AuthenticationFailure | Authentisierung fehlgeschlagen |
| ´90 00´ | NoError | Erfolgreiche Authentisierung |

Tabelle 174: General Authenticate Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | ParameterMismatch | Domainparameter passen nicht zusammen |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | KeyExpired | Gültigkeitszeitraum des Schlüssels ist abgelaufen |
| ´69 85´ | NoKeyReference | Keinen symmetrischen Schlüssel ausgewählt |
| ´69 85´ | NoPrkReference | Keinen privaten Schlüssel ausgewählt |
| ´6A 80´ | NumberPreconditionWrong | Vorbedingung zum Laden des Scenarios nicht erfüllt |
| ´6A 80´ | NumberScenarioWrong | Scenario wurde bereits geladen |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | PrKNotFound | privaten Schlüssel nicht gefunden |
| ´6A 88´ | KeyNotFound | Authentisierungsschlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N085.048) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N085.050) Kommandounterstützung
   1. K\_COS   
      Das COS MUSS die General Authenticate-Varianten aus 14.7.2.2, 14.7.2.3 und 14.7.2.5 unterstützen.
   2. K\_COS, Option\_kontaktlose\_Schnittstelle   
      Das COS MUSS die General Authenticate-Variante aus 14.7.2.1 unterstützen.
   3. K\_COS, Option\_PACE\_PCD   
      Das COS MUSS die General Authenticate-Variante aus 14.7.2.4 unterstützen.
   4. Das COS KANN weitere General Authenticate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N085.051) K\_externeWelt {K\_Karte}   
   Schlüsselauswahl in *channelContext.keyReferenceList*: Die externe Welt MUSS sicherstellen, dass bei Aufruf des Kommandos General Authenticate einer der folgenden Fälle vorliegt:
   1. (Fall gegenseitige ELC-Authentisierung, Schlüsselauswahl gemäß 14.9.9.3)
      1. in *channelContext.keyReferenceList.internalAuthenticate*
         1. eine Schlüsselreferenz eingetragen ist, die auf ein privates ELC-Schlüsselobjekt verweist und
         2. dort *algID* aus der Menge {elcSessionkey4SM, elcSessionkey4TC} ist und
      2. *channelContext.keyReferenceList.externalAuthenticate* leer ist.
   2. (Fall asynchrone, sym. Administration, Schlüsselauswahl gemäß 14.9.9.6)
      1. in *channelContext.keyReferenceList.externalAuthenticate* eine Schlüsselreferenz eingetragen ist und dort *algID* gleich aesSessionkey4SM ist und
      2. *channelContext.keyReferenceList.internalAuthenticate* leer ist.
   3. (Fall PACE, Schlüsselauswahl gemäß 14.9.9.7)   
      sowohl in *channelContext.keyReferenceList.internalAuthenticate*, als auch in *channelContext.keyReferenceList.externalAuthenticate*
      1. identische Schlüsselreferenzen und
      2. identische *algID* aus den in (N102.440) genannten Mengen eingetragen ist.
   4. (Fall asynchrone, asym. Administration, Schlüsselauswahl gemäß 14.9.9.3)
      1. in *channelContext.keyReferenceList.internalAuthenticate*
         1. eine Schlüsselreferenz eingetragen ist, die auf ein privates ELC-Schlüsselobjekt verweist und
         2. dort *algID* aus der Menge {elcAsynchronAdmin} ist und
      2. *channelContext.keyReferenceList.externalAuthenticate* leer ist.
3. Gemäß (N085.051) findet das COS in channelContext.keyReferenceList hinreichend Information anhand derer die diversen Authentisierungsprotokolle unterscheidbar sind. Deshalb ist eine Unterscheidung der hier behandelten Authentisierungsprotokolle anhand von weiteren Kommandoparametern (etwa P1 und/oder P2) nicht erforderlich.
4. (N085.052) K\_COS   
   Falls dies der erste Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann werden folgende Schritte ausgeführt:
   1. Falls die acht LSByte von *keyRef* identisch sind zu *iccsn8* (siehe (N019.900)c), genau dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
   2. Falls *channelContext.keyReferenceList.internalAuthenticate* nicht leer ist, dann wird *affectedObject\_PrK* = SearchKey(   
           *channelContext*.*currentFolder,*   
           *channelContext*.*keyReferenceList.internalAuthenticate.keyReference*,   
           *channelContext*.*keyReferenceList.internalAuthenticate.algID*   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
   3. Es wird *affectedObject\_PuK* = SearchKey(   
          *channelContext*.*currentFolder,*   
          *keyRef*,   
          *channelContext*.*keyReferenceList.internalAuthenticate.algID*   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
   4. Falls *affectedObject\_PuK*.*expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
   5. Falls *affectedObject\_PrK*.*domainParameter* verschieden ist von *affectedObject\_PuK*.*domainParameter*, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren.
   6. Zu den Domainparametern von *affectedObject\_PrK* bzw. *affectedObject\_PuK* wird ein ephemeres Schlüsselpaar (*ephemeralSK\_self*, *ephemeralPK\_self*) erzeugt.
   7. Der private Schlüssel *ephemeralSK\_self* MUSS mindestens bis zum nächsten Schritt dieses Authentisierungsprotokolls gespeichert werden.
   8. Der öffentliche Schlüssel *ephemeralPK\_self* MUSS DER codiert wie folgt in die Antwortdaten eingestellt werden:   
      *responseData* = ´ 7C‑L7C‑[ 85‑L85‑P2OS(   
        *ephemeralPK\_self*,   
        *affectedObject\_PrK.domainParameter.L*   
        ) ]´.
5. (N085.054) K\_COS   
   Falls dies der zweite Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann werden die in (N085.052) ausgewählten Objekte *affectedObject\_PrK* und *affectedObject\_PuK* in den folgende Schritten verwendet:
   1. Der ephemere öffentliche Schlüssel *PK\_oponent* des Protokollpartners wird wie folgt aus den Kommandodaten extrahiert:   
      *PK\_oponent* = OS2P(   
        *ephemeralPK\_oponent*,   
        *affectedObject\_PrK.domainParameter*   
        ).   
      *Hinweis: Falls (N085.014)a eingehalten wird, ist diese Operation stets fehlerfrei.*
   2. Falls AccessRuleEvaluation( *affectedObject\_PrK*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
   3. Das COS MUSS die Werte zur Ableitung der Sessionkeys wie folgt berechnen:
      1. K1 = ECKA( *affectedObject\_PrK.d*, *PK\_oponent*, *affectedObject\_PrK.domainParameter* ).
      2. K2 = ECKA( *ephemeralSK\_self*, *affectedObject\_PuK*, *affectedObject\_PrK.domainParameter* ).
      3. KD.i = K1   ||   K2
      4. KD.e = I2OS(0, OctetLength(KD.i)) = ´00…00´.
      5. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
   4. Es MUSS setSecurityStatus( *affectedObject\_PuK* ) ausgeführt werden.
   5. Die Antwortdaten MÜSSEN leer sein: *responseData* = ´´.
6. (N085.056) K\_COS, Option\_Kryptobox   
   Falls dies der zweite Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann werden die in (N085.052) ausgewählten Objekte *affectedObject\_PrK* und *affectedObject\_PuK* in den folgende Schritten verwendet:
   1. Der ephemere öffentliche Schlüssel *PK\_oponent* des Protokollpartners wird wie folgt aus den Kommandodaten extrahiert:   
      *PK\_oponent* = OS2P(   
        *ephemeralPK\_oponent*,   
        *affectedObject\_PrK.domainParameter*   
        ).   
      *Hinweis: Falls (N085.014)a eingehalten wird, ist diese Operation stets fehlerfrei.*
   2. Falls AccessRuleEvaluation( *affectedObject\_PrK*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
   3. Das COS MUSS die Werte zur Ableitung der Sessionkeys wie folgt berechnen:
      1. K1 = ECKA( *ephemeralSK\_self*, *affectedObject\_PuK*, *affectedObject\_PrK.domainParameter* ).
      2. K2 = ECKA( *affectedObject\_PrK.d*, *PK\_oponent*, *affectedObject\_PrK.domainParameter* ).
      3. KD.i = K1   ||   K2
      4. KD.e = I2OS(0, OctetLength(KD.i)) = ´00…00´.
      5. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
   4. Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus( *affectedObject\_PuK* ) ausgeführt werden.
   5. Die Antwortdaten MÜSSEN leer sein: *responseData* = ´´.
7. (N085.060) K\_COS   
   Falls dies der erste Schritt des Authentisierungsprotokolls für eine asynchrone, symmetrische Kartenadministration ist, dann werden folgende Schritte ausgeführt:
   1. Falls *channelContext.keyReferenceList.externalAuthenticate*
      1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
      2. nicht leer ist, dann wird *affectedObject* = SearchKey(   
             *channelContext*.*currentFolder,*   
             *keyReferenceList externalAuthenticate.keyReference*,   
             *keyReferenceList.externalAuthenticate.algID*   
         ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
         1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
         2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
   2. Es gilt: *responseData* = ´7C– 04‑( 81‑02‑I2OS(*affectedObject*.*numberScenario*))´.
8. (N085.062) K\_COS   
   Falls dies der zweite Schritt des Authentisierungsprotokolls für eine asynchrone, symmetrische Kartenadministration ist, dann wird das in (N085.060) ausgewählten Objekte *affectedObject* in den folgenden Schritten verwendet:
   1. Falls AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
   2. *cmdData* wird wie folgt aufgeteilt:
      1. *cmdData* = *M*   ||   *MAC*.
      2. OctetLength( *MAC* ) = 8.
   3. *out* = VerifyCMAC\_IsoPadding( *affectedObject.macKey*, *MAC*, *M* )
   4. Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
   5. Die Nachricht *M* wird wie folgt aufgeteilt:
      1. *M* = *NumberPrecondition*   ||   *NumberScenario*   ||   *C*.
      2. OctetLength( *NumberPrecondition* ) = 2.
      3. OctetLength( *NumberScenario* ) = 2.
      4. OctetLength( *C* ) = 64.
   6. Falls *affectedObject*.*numberScenario*
      1. kleiner als OS2I(*NumberPrecondition*) ist, genau dann MUSS das Kommando mit dem Trailer NumberPreconditionWrong terminieren.
      2. größer gleich OS2I(*NumberScenario*) ist, genau dann MUSS das Kommando mit dem Trailer NumberScenarioWrong terminieren.
   7. *affectedObject*.*numberScenario* MUSS transaktionsgesichert auf den Wert OS2I(*NumberScenario*) gesetzt werden.
   8. Es gilt:
      1. KD.e = AES\_CBC\_DEC( *affectedObject.encKey*, 0, *C* ).
      2. KD.i = ´00…00´ = I2OS(0, OctetLength(KD.e)).
      3. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
   9. Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus( *affectedObject* ) ausgeführt werden.
   10. Die Antwortdaten MÜSSEN leer sein: *responseData* = ´´.
9. (N085.064) K\_COS, Option\_kontaktlose\_Schnittstelle   
   PACE Authentisierungsprotokoll für eine Endnutzerkarte:   
   Falls in *channelContext.keyReferenceList.internalAuthenticate.algID* ein Algorithmus aus der in (N102.440)a genannten Menge eingetragen ist und dies ist
   1. der erste Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
      1. Schlüsselsuche *affectedObject* = SearchKey(   
              *channelContext*.*currentFolder,*   
              *channelContext*.*keyReferenceList.internalAuthenticate.keyReference*,   
              *channelContext*.*keyReferenceList.internalAuthenticate.algID*   
         ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
         1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
         2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
      2. Falls AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
      3. Das Attribut *affectedObject.can* (siehe (N017.028)) wird in einen Oktettstring gewandelt (vergleiche [BSI-TR-03110-3#D.2.1.4]): Die Ziffernfolge *can* besteht aus den Ziffern *z1*, …, *zn* und MUSS in einen Oktettstring *CAN* = ´*o1*, …, *on*´ umgewandelt werden, indem jede Ziffer *zi* in ein Oktett *oi* transformiert wird mit
         1. *zi* = 0 🡪 *oi* = ´30´,
         2. *zi* = 1 🡪 *oi* = ´31´,
         3. *zi* = 2 🡪 *oi* = ´32´,
         4. *…*
         5. *zi* = 9 🡪 *oi* = ´39´,
      4. *s* = RAND( 16 ),
      5. *z* = AES\_ENC(   
             KDF(*CAN*, 3, *channelContext.keyReferenceList.internalAuthenticate.algID*),  
             *s*   
         ),
      6. *responseData* = ´7C 12 (80 10 *z*)´.
   2. der zweite Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
      1. Zuordnung *channelContext*.*keyReferenceList.internalAuthenticate.algID* zu *dP* (diese Zuordnung gilt auch für die übrigen Schritte des Protokolls):
         1. id-PACE-ECDH-GM-AES-CBC-CMAC-128 🡪 *dP* = brainpoolP256r1
         2. id-PACE-ECDH-GM-AES-CBC-CMAC-192 🡪 *dP* = brainpoolP384r1
         3. id-PACE-ECDH-GM-AES-CBC-CMAC-256 🡪 *dP* = brainpoolP512r1
      2. Aus dem Wertfeld *value81* des DO81 im Kommandodatenfeld wird ein Punkt extrahiert: *~PK1PCD* = OS2P(*value81*, *dP*),
      3. Es wird ein ephemeres Schlüsselpaar generiert:
         1. ~*SK1PICC* = zufällige Zahl aus dem Intervall [1, *dP*.*n* – 1],
         2. ~*PK1PICC* = *~SK1PICC* \* *dP*.*G*,
      4. Berechne den ephemeren Basispunkt ~*G* mit *s* aus Schritt 1:   
         ~*G* = *s* \* *dP*.*G*   +   *~SK1PICC* \* *~PK1PCD*,
      5. Berechne ephemere Domainparameter ~*D* aus *dP* und ~*G*:   
         ~*D* = (*dP*.*p*, *dP*.*a*, *dP*.*b*, ~*G*, *dP*.*n*, *dP*.*h*, *dP*.*L*, *dP*.τ, *dP*.*OID*),
      6. *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:   
         *responseData* = ´7C – L7C – (82 – L82 ‑ P2OS(*~PK1PICC*, *dP*.τ / 8 ))´
   3. der dritte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
      1. Aus dem Wertfeld *value83* des DO83 im Kommandodatenfeld wird ein Punkt extrahiert: *~PK2PCD* = OS2P(*value83*, ~*D*),
      2. Es wird ein weiteres ephemeres Schlüsselpaar generiert:
         1. ~*SK2PICC* = zufällige Zahl aus dem Intervall [1, ~*D*.*n* – 1],
         2. ~*PK2PICC* = *~SK2PICC* \* ~*D*.*G*,
      3. *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:   
         *responseData* = ´7C – L7C – (84 – L84 ‑ P2OS(*~PK2PICC*, ~*D*.τ / 8 ))´
   4. der vierte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
      1. Berechne gemeinsames Geheimnis zur Ableitung von Sessionkeys:
         1. KD.i   = KAB = ECKA(~*SK2PICC*, *~PK2PCD*, ~*D* ),
         2. KD.e = I2OS(0, OctetLength(KD.i)) = ´00…00´,
      2. Die *algID* aus *channelContext.keyReferenceList.internalAuthenticate* bestimmt, wie der Schlüssel *kMAC* berechnet wird: *kMAC* = KDF(KD.i, 2, *algID* ),
      3. Für den Zusammenhang zwischen *algID* aus *externalAuthenticate* oder *internalAuthenticate* in *channelContext.keyReferenceList* und *OID* gilt hier *algID* = *OID*.
      4. Berechne den DER codierten Oktettstring *MPCD* gemäß:   
         *MPCD* = ´7F49 – L7F49 – [(06 0A *OID*) || (86 – L86 ‑ P2OS(*~PK2PICC*, ~*D* ))]´,
      5. Berechne den DER codierten Oktettstring *MPICC* gemäß:   
         *MPICC* = ´7F49 – L7F49 – [(06 0A *OID*) || (86 – L86 ‑ P2OS(*~PK2PCD*, ~*D* ))]´,
      6. Überprüfe den MAC aus der Kommandonachricht: Falls   
         *result* = VerifyCMAC\_NoPadding( *kMAC*, *TPCD*, *MPCD* ) den Wert
         1. INVALID besitzt, dann MUSS
            1. clearSecurityStatusKey(*affectedObject*) ausgeführt werden und
            2. das Kommando mit dem Trailer AuthenticationFailure terminieren.
         2. VALID besitzt, dann
            1. MUSS der MAC für die Antwortnachricht wie folgt berechnet werden: *TPICC* = CalculateCMAC\_NoPadding( *kMAC*, *MPICC* ), und
            2. MUSS setSecurityStatus( *affectedObject* ) ausgeführt werden und
            3. MÜSSEN die Oktettstrings KD.e und KD.i an den Secure Messaging Layer übergeben werden (siehe 13.1), und
            4. MUSS das Datenfeld der Antwortnachricht wie folgt berechnet werden: *responseData* = ´7C 0A (86 08 *TPICC*)´.
10. (N085.066) K\_COS, Option\_PACE\_PCD   
    PACE Authentisierungsprotokoll für ein Sicherheitsmodul:   
    Falls in *channelContext.keyReferenceList.internalAuthenticate.algID* ein Algorithmus aus der in (N102.440)b genannten Menge eingetragen ist und dies ist
    1. der erste Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
       1. Schlüsselsuche *affectedObject* = SearchKey(   
              *channelContext*.*currentFolder,*   
              *channelContext*.*keyReferenceList.internalAuthenticate.keyReference*,   
              *channelContext*.*keyReferenceList.internalAuthenticate.algID*   
          ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
          1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
          2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
       2. Falls AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
       3. Zuordnung *channelContext*.*keyReferenceList.internalAuthenticate.algID* zu *dP* (diese Zuordnung gilt auch für die übrigen Schritte des Protokolls).
          1. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128 🡪 *dP* = brainpoolP256r1
          2. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192 🡪 *dP* = brainpoolP384r1
          3. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256 🡪 *dP* = brainpoolP512r1
       4. Es wird ein ephemeres Schlüsselpaar generiert:
          1. ~*SK1PCD* = zufällige Zahl aus dem Intervall [1, *dP*.*n* – 1],
          2. ~*PK1PCD* = *~SK1PCD* \* *dP*.*G*,
       5. *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:   
          *responseData* = ´7C – L7C – (81 – L81 ‑ P2OS(*~PK1PCD*, *dP*.τ / 8 ))´
    2. der zweite Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
       1. Aus dem Datenfeld der Kommandonachricht werden *z* und *CAN* extrahiert.
       2. *s* = OS2I( AES\_DEC(   
              KDF(*CAN*, 3, *channelContext.keyReferenceList.internalAuthenticate.algID*),  
              *z*   
          ) ),
       3. *responseData* = ´´ (leerer Oktettstring).
    3. der dritte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
       1. Aus dem Wertfeld *value82* des DO82 im Kommandodatenfeld wird ein Punkt extrahiert: *~PK1PICC* = OS2P(*value82*, *dP*),
       2. Berechne den ephemeren Basispunkt ~*G* mit *s* aus Schritt 2:   
          ~*G* = *s* \* *dP*.*G*   +   *~SK1PCD* \* *~PK1PICC*,
       3. Berechne ephemere Domainparameter ~*D* aus *dP* und ~*G*:   
          ~*D* = (*dP*.*p*, *dP*.*a*, *dP*.*b*, ~*G*, *dP*.*n*, *dP*.*h*, *dP*.*L*, *dP*.τ, *dP*.*OID*),
       4. Es wird ein weiteres ephemeres Schlüsselpaar generiert:
          1. ~*SK2PCD* = zufällige Zahl aus dem Intervall [1, ~*D*.*n* – 1],
          2. ~*PK2PCD* = *~SK2PCD* \* ~*D*.*G*,
       5. *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:   
          *responseData* = ´7C – L7C – (83 – L83 ‑ P2OS(*~PK2PCD*, ~*D*.τ / 8 ))´
    4. der vierte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
       1. Aus dem Wertfeld *value84* des DO84 im Kommandodatenfeld wird ein Punkt extrahiert: *~PK2PICC* = OS2P(*value84*, ~*D*),
       2. Berechne gemeinsames Geheimnis zur Ableitung von Sessionkeys:
          1. KD.i = KAB = ECKA(~*SK2PCD*, *~PK2PICC*, ~*D* ),
          2. KD.e = I2OS(0, OctetLength(KD.i)) = ´00…00´,
       3. Die *algID* aus *channelContext.keyReferenceList.internalAuthenticate* bestimmt, wie der Schlüssel *kMAC* berechnet wird: *kMAC* = KDF(KD.i, 2, *algID* ),
       4. Für den Zusammenhang zwischen *algID* aus *externalAuthenticate* oder *internalAuthenticate* in *channelContext.keyReferenceList* und *OID* gilt hier*:*
          1. *algID* =   id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128   
             => *OID* =   id-PACE-ECDH-GM-AES-CBC-CMAC-128
          2. *algID* =   id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192   
             => *OID* =   id-PACE-ECDH-GM-AES-CBC-CMAC-192
          3. *algID* =   id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256   
             => *OID* =   id-PACE-ECDH-GM-AES-CBC-CMAC-128
       5. Berechne den Oktettstring *MPCD* identisch zu (N085.064)d.4 und den zugehörigen MAC: *TPCD* = CalculateCMAC\_NoPadding( *kMAC*, *MPCD* ).
       6. Berechne den Oktettstring *MPICC* identisch zu (N085.064)d.5.
       7. Setze *responseData* = ´7C 0A (85 08 *TPCD*)´
    5. der fünfte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
       1. Überprüfe den MAC aus der Kommandonachricht: Falls   
          *result* = VerifyCMAC\_NoPadding( *kMAC*, *TPICC*, *MPICC* ) den Wert
          1. INVALID besitzt, dann MUSS
             1. clearSecurityStatusKey(*affectedObject*) ausgeführt werden und
             2. das Kommando mit dem Trailer AuthenticationFailure terminieren.
          2. VALID besitzt, dann
             1. MUSS setSecurityStatus( *affectedObject* ) ausgeführt werden und
             2. MÜSSEN die Oktettstrings KD.e und KD.i an den Secure Messaging Layer übergeben werden (siehe 13.1), und
             3. MUSS gelten *responseData* = ´´ (leerer Oktettstring).
11. (N085.068) K\_COS   
    Asynchrone, asymmetrische Kartenadministration:   
    Falls *channelContext.keyReferenceList.internalAuthenticate.algID* der Algorithmus elcAsynchronAdmin eingetragen ist und dies ist
    1. der erste Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
       1. Schlüsselsuche *affectedObject\_PrK* = SearchKey(   
               *channelContext*.*currentFolder,*   
               *channelContext*.*keyReferenceList.internalAuthenticate.keyReference*,   
               *channelContext*.*keyReferenceList.internalAuthenticate.algID*   
          ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
          1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.
          2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
       2. Es wird *affectedObject\_PuK* = SearchKey(   
               *channelContext*.*currentFolder,*   
               *keyRef*,   
               *channelContext*.*keyReferenceList.internalAuthenticate.algID*   
          ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
          1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
          2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
       3. Falls *affectedObject\_PuK*.*expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
       4. Falls *affectedObject\_PrK*.*domainParameter* verschieden ist von *affectedObject\_PuK*.*domainParameter*, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren.
       5. Es gilt:   
          *responseData* = ´7C– 04‑(81‑02‑I2OS(*affectedObject\_PrK*.*numberScenario*))´.
    2. der zweite Schritt des Authentisierungsprotokolls ist, dann werden folgende Schritte ausgeführt:
       1. Falls AccessRuleEvaluation( *affectedObject\_PrK*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
       2. *cmdData* wird wie folgt aufgeteilt:
          1. *cmdData* = *M*   ||   *SIG*.
          2. In Abhängigkeit von den *affectecObject\_PrK*.*domainParameter* gilt:
             1. brainpoolP256r1: OctetLength(*SIG*) = 64, *hash* = SHA\_256( *M* )
             2. brainpoolP384r1: OctetLength(*SIG*) = 96, *hash* = SHA\_384( *M* )
             3. brainpoolP512r1: OctetLength(*SIG*) = 128, *hash* = SHA\_512( *M* )
          3. *SIG* = *R*   ||   *S*
          4. OctetLength( *R* ) = OctetLength( *S* ).
       3. Falls ELC\_VER\_SIG( *affectedObject\_PuK*, *R*, *S*, *hash* ) den Wert False besitzt, MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
       4. Die Nachricht *M* wird wie folgt aufgeteilt:
          1. *M* = *NumberPrecondition*   ||   *NumberScenario*   ||   *cipher*.
          2. OctetLength( *NumberPrecondition* ) = 2.
          3. OctetLength( *NumberScenario* ) = 2.
       5. Falls *affectedObject\_PrK*.*numberScenario*
          1. kleiner als OS2I(*NumberPrecondition*) ist, genau dann MUSS das Kommando mit dem Trailer NumberPreconditionWrong terminieren.
          2. größer gleich OS2I(*NumberScenario*) ist, genau dann MUSS das Kommando mit dem Trailer NumberScenarioWrong terminieren.
       6. *affectedObject\_PrK*.*numberScenario* MUSS transaktionsgesichert auf den Wert OS2I(*NumberScenario*) gesetzt werden.
       7. Es gilt (*Hinweis:* cipher *ist hier identisch zu (N090.300)c, (N091.700)d und (N094.400)c definiert*):
          1. *cipher* MUSS ein DER codiertes DOA6 sein.
          2. *cipher* = ´A6–LA6–( *oidDO*   ||   *keyDO*   ||   *cipherDO*   ||   *macDO* ).
          3. *oidDO* = ´06–L06–*oid* ´.
          4. *keyDO* = ´7F49–L7F49–( 86 – L86 – *POA* )´.
          5. *cipherDO* = ´86–L86–( 02   ||   *C* )´.
          6. *macDO* = ´8E–L8E–*T*´.
          7. Falls *oid* verschieden ist zur OID, die gemäß (N008.600)d zu *affectedObject\_PrK*.*privateElcKey*.*domainParameter* gehört, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren. Diese Domainparameter werden im Folgenden mit *dP* bezeichnet.
          8. KD.e = ELC\_DEC(*POA*, *affectedObject\_PrK.privateElcKey*, *C*, *T*).
          9. KD.i = ´00…00´ = I2OS(0, OctetLength(KD.e)).
          10. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
       8. Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus( *affectedObject\_PuK* ) ausgeführt werden.
       9. Die Antwortdaten MÜSSEN leer sein: *responseData* = ´´.
12. Wegen (N008.600)d und Tabelle 271 ist L06 in (N085.068)b.7.iii stets 9 und damit oidDO stets 11 Oktett lang.
13. In Abhängigkeit von den Domainparametern von affectedObject\_PrK, (N000.300)a und (N008.600)b gilt für L86 in (N085.068)b.7.iv:
    1. brainpoolP256r1: L86 = 65 => keyDO ist 70 Oktett lang.
    2. brainpoolP384r1: L86 = 97 => keyDO ist 102 Oktett lang.
    3. brainpoolP512r1: L86 = 129 => keyDO ist 136 Oktett lang.
14. Wegen (N085.040) und (N004.800)e ist L86 in (N085.068)b.7.v gleich 1 + 64 + 16 = 81 und damit cipherDO 83 Oktett lang.
15. Wegen (N002.810)h ist L8E in (N085.068)b.7.vi gleich 8 und damit macDO 10 Oktett lang.
16. (N085.070) K\_COS   
    Als Datenfeld der Antwortnachricht MUSS der (möglicherweise leere) Oktettstring *responseData* verwendet werden.
17. (N085.072) K\_COS   
    Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
18. (N085.074) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 174 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 174 MUSS eine höhere Priorität als NoError haben.
19. Im Rahmen dieser Spezifikation sind Authentisierungssequenzen nicht unterbrechbar (siehe (N104.600)).
20. Bei der Schlüsselsuche in (N085.052)c wird nach einem öffentlichen Schlüsselobjekt gesucht. Trotzdem wird algID aus keyReferenceList.internalAuthenticate entnommen, damit sichergestellt ist, dass öffentliches wie privates Schlüsselobjekt denselben Algorithmus verwenden. Aus der übrigen Spezifikation ergibt sich, dass dieser Algorithmus Element der Menge { elcSessionkey4SM, elcSessionkey4TC } ist.

### Get Security Status Key

1. Dieses Kommando ist nicht in der Normenreihe ISO/IEC 7816 enthalten. Es ließe sich kombinieren mit dem Kommando External Authenticate.
2. Bei der Spezifikation der Kommando-APDU wurde das MSE-Set-Kommando zugrunde gelegt.
3. Motivation für das Kommando: Ausgehend von der Annahme, dass ein bestimmtes Kommando mit dem Trailer SecurityStatusNotSatisfied terminiert, ist für die steuernde Software gelegentlich nicht offensichtlich, was zu tun ist, um Zugriff zu erhalten. Selbst bei Kenntnis der Zugriffsregel des Objektes lässt sich lediglich beurteilen, ob Secure Messaging Vorgaben eingehalten wurden. Falls eine PIN-Verifikation erforderlich ist, so lässt sich der zugehörige Sicherheitszustand mittels Get Pin Status erfragen. Falls aber mehrere Sicherheitszustände von Schlüsseln ins Spiel kommen, dann hat die steuernde Software nur zwei Möglichkeiten: Entweder „Try and error“, was schlecht für die Performanz ist, oder sie baut die COS spezifische Zustandsmaschine für den Sicherheitsstatus von Schlüsseln nach (und hofft, dass interner und externer Zustand synchron bleiben).

Das Kommando Get Security Status Key wird verwendet, um den Sicherheitszustand von Schlüsselobjekten zu erfragen. Welcher Sicherheitszustand ausgelesen wird, bestimmt eine Referenz, die als Parameter in den Kommandodaten enthalten ist.

#### Use Case Auslesen Sicherheitsstatus symmetrischer Schlüssels, Option\_DES

In dieser Variante enthält die APDU des MSE Kommandos einen Parameter:

1. (N085.100) K\_externeWelt {K\_Karte}, Option\_DES   
   Der Parameter *cmdData* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
2. (N085.200) K\_externeWelt {K\_Karte}, Option\_DES   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 175 verwendet werden.

Tabelle 175: Get Security Status Key, symmetrischer Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´82´ | Instruction Byte (derselbe Wert wie bei External Authenticate) |
| P1 | ´80´ | Auslesen Sicherheitszustand |
| P2 | ´00´ | – |
| Data | ´XX…XX´ | ´83 01 || *cmdData* ´ |

#### Use Case Auslesen des Sicherheitsstatus einer Rolle, Option\_RSA\_CVC

In dieser Variante enthält die APDU des MSE Kommandos einen Parameter:

1. (N085.300) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der Parameter *cmdData* enthält eine Rollenkennung. Wert und Codierung MÜSSEN gemäß 7.1.1.4 und (N005.700) gewählt werden.
2. (N085.400) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 176 verwendet werden.

Tabelle 176: Get Security Status Key, Rollenkennung

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´82´ | Instruction Byte (derselbe Wert wie bei External Authenticate) |
| P1 | ´80´ | Auslesen Sicherheitszustand |
| P2 | ´00´ | – |
| Data | ´XX…XX´ | ´5F4C ‑ 07 ‑ *cmdData*´ |

#### Use Case Auslesen des Sicherheitsstatus einer Bitliste

In dieser Variante enthält die APDU des MSE Kommandos zwei Parameter:

1. (N085.440) K\_externeWelt {K\_Karte}   
   Der Parameter *oid* MUSS eine OID aus der Menge {oid\_cvc\_fl\_ti, oid\_cvc\_fl\_cms} enthalten.
2. (N085.442) K\_externeWelt {K\_Karte}   
   Der Parameter *cmdData* MUSS eine sieben Oktett lange Flagliste enthalten. Die beiden höchstwertigen Bit in *cmdData* MÜSSEN den Wert 0 besitzen. Für die Werte übrigen Bit gibt es keine Beschränkung.
3. (N085.444) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 177 verwendet werden.

Tabelle 177: Get Security Status Key, *bitSecurityList*

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´82´ | Instruction Byte (derselbe Wert wie bei External Authenticate) |
| P1 | ´80´ | Auslesen Sicherheitszustand |
| P2 | ´00´ | – |
| Data | ´XX…XX´ | ´7F4C – 13 ‑ (06 – 08 – *oid* || 53 ‑ 07 ‑ *cmdData*)´ |

#### Antwort der Karte auf Auslesen Sicherheitsstatus eines Schlüssels

Tabelle 178: Get Security Status Key Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 CF´ | NoAuthentication | Authentisierungsstatus ist nicht gesetzt |
| ´90 00´ | NoError | Authentisierungsstatus ist gesetzt |

Tabelle 179: Get Security Status Key Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´6A 88´ | KeyNotFound | Adressiertes Schlüsselobjekt wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N085.500) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N085.600) K\_COS
   1. Das COS MUSS die Get Security Status Key-Varianten aus
      1. Option\_DES, 14.7.3.1,
      2. Option\_RSA\_CVC, 14.7.3.2 und
      3. 14.7.3.3 unterstützen.
   2. Das COS KANN weitere Get Security Status Key-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N085.700) K\_COS, Option\_DES   
   Falls *cmdData* die Referenz eines symmetrischen Schlüssels gemäß 14.7.3.1 ist, dann
   1. wird *affectedObject* = SearchSecretKey(   
          *channelContext*.*currentFolder,*   
          *cmdData*,   
          Wildcard   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren. *Hinweis: Der Fehler notSupported ist wegen der Wildcard-Suche nicht möglich.*
   2. Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
   3. Falls *affectedObject* in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist, dann MUSS als Trailer NoError verwendet werden.
3. (N085.800) K\_COS, Option\_RSA\_CVC   
   Falls *cmdData* eine Rolle gemäß 14.7.3.2. ist und diese Rolle in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist, dann MUSS als Trailer NoError verwendet werden.
4. (N085.900) K\_COS   
   Falls *cmdData* ein CHAT enthält und es gibt mindestens ein Element in *bitSecurityList* (siehe (N029.900)h), in welchem dieselbe OID und mindestens dieselben Bits gesetzt sind wie in *flagList* (siehe auch Hinweis (50):), dann MUSS als Trailer NoError verwendet werden.
5. (N086.000) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoAuthentication gewählt werden.
6. (N086.100) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 179 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 179 MUSS eine höhere Priorität als NoError haben.
   3. NoError MUSS eine höhere Priorität als NoAuthentication haben.

### Internal Authenticate

Das Kommando Internal Authenticate berechnet Authentisierungsdaten zu einem Token mittels eines symmetrischen oder privaten Schlüssels. Der Schlüssel wird vor der Authentisierungsoperation ausgewählt. Dies geschieht vor dem Senden dieses Internal Authenticate-Kommandos durch ein MSE-Set-Kommando (siehe 14.9.9.3). Das Token ist als Parameter in der Kommandonachricht enthalten.

#### Use Case interne Authentisierung

In dieser Variante enthält die APDU des Internal Authenticate-Kommandos zwei Parameter:

1. (N086.200) K\_externeWelt {K\_Karte}   
   Der Parameter *token* enthält die zu authentisierenden Daten. Der Parameter *token* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *token* ist abhängig von der mittels (N100.800) ausgewählten *algId*. Wenn *algId* gleich

* 1. elcRoleAuthentication ist, dann MUSS *token* gleich 24 Oktett lang sein.
  2. rsaClientAuthentication ist, dann DARF *token* NICHT länger als 64 Oktette sein.
  3. Option\_RSA\_CVC, rsaRoleAuthentication ist, dann MUSS *token* gleich 16 Oktette lang sein.

* 1. Option\_DES, rsaSessionkey4SM ist, dann MUSS *token* gemäß (N106.700) gewählt werden.

* 1. signPKCS1\_V1\_5 ist, dann MUSS die Anzahl Oktette in *token* kleiner als 0,4 OctetLength( *n* ) sein, mit *n* als Modulus des ausgewählten Authentisierungsschlüssels.

1. (N086.202) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Der Parameter *token* enthält die zu authentisierenden Daten. Der Parameter *token* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *token* ist abhängig von der mittels (N100.800) ausgewählten *algId*. Wenn *algId* gleich
   1. aesSessionkey4TC ist, dann MUSS *token* gleich 24 Oktett lang sein.
   2. Option\_DES, desSessionkey4TC ist, dann MUSS *token* gleich 16 Oktett lang sein.
   3. Option\_DES, rsaSessionkey4TC ist, dann MUSS *token* gleich 16 Oktett lang sein.
2. (N086.300) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *response* in der Antwortnachricht enthalten ist.
3. (N086.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 180 verwendet werden.

Tabelle 180: Internal Authenticate

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´88´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Information zum Algorithmus bereits in der Karte vorhanden |
| P2 | ´00´ | Schlüsselreferenz bereits in der Karte vorhanden |
| Data | ´XX…XX´ | *token* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf interne Authentisierung

Tabelle 181: Internal Authenticate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *response* | Authentisierende Daten |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Authentisierungsoperation |

Tabelle 182: Internal Authenticate Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | KeyInvalid | Schlüsseldaten fehlen |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoKeyReference | Kein Authentisierungsschlüssel ausgewählt |
| ´69 85´ | NoPukReference | Kein Verschlüsselungsschlüssel ausgewählt |
| `6A 80´ | WrongToken | fehlerhaftes Token |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | KeyNotFound | Authentisierungsschlüssel nicht gefunden |
| ´6A 88´ | PukNotFound | Verschlüsselungsschlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N086.500) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N086.600) K\_COS
   1. Das COS MUSS die Internal Authenticate-Variante aus 14.7.4.1 unterstützen.
   2. Das COS KANN weitere Internal Authenticate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N086.700) K\_COS   
   Wenn *channelContext.keyReferenceList.internalAuthenticate*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchKey(   
        *currentFolder,   
          keyReferenceList.internalAuthenticate.keyReference,   
          keyReferenceList.internalAuthenticate.algID*) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
3. (N086.800) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N086.810) K\_COS   
   Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.
5. (N086.820) K\_COS   
   Wenn *channelContext.keyReferenceList.internalAuthenticate.algID* einen Wert aus der Menge {aesSessionkey4TC, desSessionkey4TC, elcRoleAuthentication, rsaRoleAuthentication, rsaSessionkey4SM, rsaSessionkey4TC} besitzt und die acht LSByte von *token* identisch sind zu *iccsn8* (siehe (N019.900)c), genau dann MUSS das Kommando mit dem Trailer WrongToken terminieren.
6. (N086.900) K\_COS   
   Die Antwort *response* wird wie folgt berechnet:   
   Wenn *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert

* 1. elcRoleAuthentication besitzt, dann gilt mit *PrK* = *affectedObject.privateKey*   
     *response* = ELC\_SIG(   
       *PrK*,   
       I2OS(   
       OS2I( *token*   ||   elcRoleAuthentication),   
       *PrK*.*domainParameter*.τ / 8   
       )   
       ).

* 1. rsaClientAuthentication besitzt, dann gilt   
     *response* = RSASSA\_PSS\_SIGN( *affectedObject.privateKey*, *token* )

* 1. Option\_RSA\_CVC, rsaRoleAuthentication besitzt, dann gilt mit *PrK* = *affectedObject.privateKey*
     1. PRND = RAND(OctetLength( *PrK.n* ) – 34 )
     2. *M* = PRND   ||   *token*
     3. ( *response*, *M2* ) = RSA\_ISO9796\_2\_DS1\_SIGN( *PrK*, *M* )   
        *Hinweis: Die Länge von PRND ist so gewählt, dass* M2 = token *ist. Darum ist es nicht notwendig,* M2 *in die Antwortnachricht einzustellen.*

* 1. Option\_DES, rsaSessionkey4SM besitzt, dann gilt mit *PrK* = *affectedObject.privateKey*
     1. KD.i = RAND( 64 )   
        Der Oktettstring KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).
     2. PRND = RAND(OctetLength( *PrK.n* ) – 34 – OctetLength(KD.i) )
     3. *M* = PRND   ||   KD.i   ||   *token*
     4. ( *sig*, *M2* ) = RSA\_ISO9796\_2\_DS1\_SIGN( *PrK*, *M* )   
        *Hinweis: Die Länge von PRND ist so gewählt, dass M2 = token ist. Darum ist es nicht notwendig, M2 in die Antwortnachricht einzustellen.*
     5. Wenn *channelContext.keyReferenceList.externalAuthenticate*
        1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoPukReference terminieren.
        2. nicht leer ist, dann wird *tmpObject* = SearchKey(   
               *currentFolder,*   
               *keyReferenceList.externalAuthenticate.keyReference*,   
           *keyReferenceList.internalAuthenticate.algID*) gesetzt. Gemäß 9.2.3.2 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
           1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PukNotFound terminieren.
           2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
     6. *response* = *sigPuK.e* mod *PuK.n*, mit *PuK* = *tmpObject.publicKey*.

* 1. signPKCS1\_V1\_5 besitzt, dann gilt:   
     *response* = RSASSA\_PKCS1\_V1\_5\_SIGN( *PrK*, *token* )

1. (N086.902) K\_COS, Option\_Kryptobox   
   Die Antwort *response* wird wie folgt berechnet:   
   Wenn *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert

* 1. aesSessionkey4TC besitzt, dann werden folgende Schritte ausgeführt:
     1. Setze RND.int = RAND( 16 )
     2. Setze ICCSN8.int = *iccsn8* (siehe (N019.900)c)
     3. Setze KD.i = RAND( 64 )
     4. Setze S = RND.int || ICCSN8.int || *token*  || KD.i
     5. Setze *C*1 = AES\_CBC\_ENC( *affectedObject.encKey*, 0, *S* )
     6. Setze *MAC*1 = CalculateCMAC\_IsoPadding( *affectedObject.macKey*, *C*1 )
     7. Setze response = *C*1   ||   *MAC*1
     8. RND.int MUSS als *RND.ICC* gespeichert werden (siehe (N029.900)).
     9. KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).

* 1. Option\_DES, desSessionkey4TC besitzt, dann werden folgende Schritte ausgeführt:
     1. Setze RND.int = RAND( 8 )
     2. Setze ICCSN8.int = *iccsn8* (siehe (N019.900)c)
     3. Setze KD.i = RAND( 64 )
     4. Setze S = RND.int || ICCSN8.int || *token* || KD.i
     5. Setze *C*1 = 3TDES\_CBC\_ENC( *affectedObject.encKey*, 0, *S* )
     6. Setze *MAC*1 = CALCULATE\_Retail\_MAC( *affectedObject.macKey*, *C*1 )
     7. Setze response = *C*1   ||   *MAC*1
     8. RND.int MUSS als *RND.ICC* gespeichert werden (siehe (N029.900)).
     9. KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).

* 1. Option\_DES, rsaSessionkey4TC besitzt, dann MUSS im Rahmen der Kommandobearbeitung identisch zu rsaSessionkey4SM verfahren werden.

1. (N087.000) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS *response* verwendet werden.
2. (N087.100) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
3. (N087.200) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 182 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 182 MUSS eine höhere Priorität als NoError haben.
4. Bei der Schlüsselsuche in (N086.900)d.5.ii wird nach einem öffentlichen Schlüsselobjekt gesucht. Trotzdem wird algID aus keyReferenceList.internalAuthenticate entnommen, damit sicher gestellt ist, dass auch das öffentliche Schlüsselobjekt den hier zum Tragen kommenden Algorithmus unterstützt. Aus der übrigen Spezifikation ergibt sich, dass dieser Algorithmus Element von {rsaSessionkey4SM, rsaSessionkey4TC} ist.

## Kryptoboxkommandos

Dieses Unterkapitel behandelt Funktionalitäten, die in [ISO/IEC 7816-8] mit dem Kommando Perform Security Operation verknüpft und alle über den INS Code ´2A´ erreichbar sind. Die folgende Tabelle gibt einen informativen Überblick über die hier behandelten Funktionalitäten. Einzelheiten finden sich in den nachfolgenden Kapiteln.

Tabelle 183: Tabelle aller PSO Kommando Header

| **CLA** | **INS** | **P1** | **P2** | **Kommando** | **Referenz** |
| --- | --- | --- | --- | --- | --- |
| ´00´ | ´2A´ | ´8E´ | ´80´ | PSO Compute Cryptographic Checksum | 14.8.1 |
| ´9E´ | ´9A´ | PSO Compute Digital Signature, ohne "recovery" | 14.8.2.1 |
| ´9E´ | ´AC´ | PSO Compute Digital Signature, mit "recovery" | 14.8.2.2 |
| ´80´ | ´86´ | PSO Decipher | 14.8.3 |
| ´86´ | ´80´ | PSO Encipher | 14.8.4 |
| ´90´ | ´XX´ | PSO Hash, siehe [ISO/IEC 7816-8] | ‑ |
| ´86´ | ´B8´ | PSO Transcipher mittels RSA | 14.8.6.1 |
| ´86´ | ´B8´ | PSO Transcipher mittels ELC | 14.8.6.3 |
| ´00´ | ´AE´ | PSO Verify Certificate | 14.8.7 |
| ´00´ | ´A2´ | PSO Verify Cryptographic Checksum | 14.8.8 |
| ´00´ | ´A8´ | PSO Verify Digital Signature | 14.8.9 |

### PSO Compute Cryptographic Checksum

Das Kommando PSO Compute Cryptographic Checksum berechnet zu gegebenen Daten eine kryptographische Checksumme mittels eines symmetrischen Schlüssels. Der symmetrische Schlüssel wird im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, 15.4.3, 15.4.4) ausgehandelt. Die durch eine Checksumme zu schützenden Daten sind als Parameter in der Kommandonachricht enthalten.

#### Use Case Berechnen einer kryptographischen Checksumme

Diese Variante gilt für Algorithmen aus der folgenden Menge: {aesSessionkey, desSessionkey (Option\_DES)}.

1. (N087.220) K\_externeWelt {K\_Karte}   
   In dieser Variante enthält die APDU des PSO Compute Cryptographic Checksum Kommandos drei Parameter:
   1. Der Parameter *flagSSCmacIncrement* MUSS wie folgt gewählt werden: Falls es sich um Sessionkeys für den Algorithmus
      1. Option\_DES, desSessionkey handelt, dann ist *flagSSCmacIncrement* ein leerer Oktettstring.
      2. aesSessionkey handelt und *SSCmac*
         1. zu inkrementieren ist, dann ist *flagSSCmacIncrement* = ´01´.
         2. unverändert zu verwenden ist, dann ist *flagSSCmacIncrement* = ´00´.
   2. Der Parameter *data* enthält die zu schützenden Daten. Der Parameter *data* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *data* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.
   3. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.
2. (N087.228) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 184 verwendet werden.

Tabelle 184: PSO Compute Cryptographic Checksum, Berechnen eines MAC

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´8E´ | Beschreibung der Antwortdaten, hier kryptographische Checksumme |
| P2 | ´80´ | Beschreibung der Kommandodaten, hier Klartext |
| Data | ´XX…XX´ | *flagSSCmacIncrement*   ||   *data* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Berechnen einer kryptographischen Checksumme

Tabelle 185: PSO Compute Cryptographic Checksum Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | mac | kryptographische Checksumme |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | erfolgreiche Berechnung eines MAC |

Tabelle 186: PSO Compute Cryptographic Checksum Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoKeyReference | kein Schlüssel für MAC Berechnung ausgewählt |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | KeyNotFound | kein Schlüssel für MAC Berechnung vorhanden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N087.232) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N087.236) K\_COS, Option\_Kryptobox, Option\_PACE\_PCD
   1. Das COS MUSS die PSO Compute Cryptographic Checksum Variante aus 14.8.1.1 unterstützen.
   2. Das COS KANN weitere PSO Compute Cryptographic Checksum Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N087.240) K\_COS   
   Wenn *channelContext.keyReferenceList.macCalculation*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchKey(   
      *currentFolder,   
          keyReferenceList.macCalculation.keyReference,   
          keyReferenceList.macCalculation.algID*) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
3. (N087.244) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N087.248) K\_COS   
   Mit den Attributen *SSCmac* und *Kmac* aus dem Attribut *SessionkeyContext* gilt: Falls *keyReferenceList.macCalculation.algID* den Wert

* 1. aesSessionkey besitzt, MUSS gelten:
     1. Falls *flagSSCmacIncrement* = ´01´ ist, dann wird *SSCmac* inkrementiert:   
        *SSCmac* = *SSCmac* + 1, sonst bleibt *SSCmac* unverändert.
     2. *mac* = CalculateCMAC\_IsoPadding(*Kmac*, I2OS(*SSCmac*, 16) || *data* ).
  2. Option\_DES, desSessionkey besitzt, MUSS gelten:
     1. *SSCmac* = *SSCmac* + 1.
     2. *mac* = CALCULATE\_Retail\_MAC (*Kmac*, I2OS(*SSCmac*, 8) || *data* ).

1. (N087.252) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS *mac* verwendet werden.
2. (N087.256) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
3. (N087.260) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 186 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 186 MUSS eine höhere Priorität als NoError haben.

### PSO Compute Digital Signature

Das Kommando PSO Compute Digital Signature signiert Daten mittels eines privaten Schlüssels. Der private Schlüssel und der zu verwendende Algorithmus werden vor der Signieroperation ausgewählt. Dies geschieht vor dem Senden dieses PSO Compute Digital Signature-Kommandos durch ein MSE-Set-Kommando (siehe 14.9.9.9). Die zu signierenden Daten sind als Parameter in der Kommandonachricht enthalten.

#### Use Case Signieren des Datenfeldes, ohne „message recovery“

Diese Variante gilt für folgende Algorithmen (siehe (N017.600) und (N018.300)): rsaClientAuthentication, signECDSA, signPKCS1\_V1\_5, signPSS.

In dieser Variante enthält die APDU des PSO Compute Digital Signature-Kommandos zwei Parameter:

1. (N087.300) K\_externeWelt {K\_Karte}   
   Der Parameter *dataToBeSigned* enthält die zu signierenden Daten. Der Parameter *dataToBeSigned* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *dataToBeSigned* ist abhängig von der mittels (N102.800) ausgewählten *algId*. Wenn *algId* gleich

* 1. rsaClientAuthentication ist, dann DARF *dataToBeSigned* nicht länger als 64 Oktette sein.

* 1. signECDSA ist, dann MUSS die Länge von *dataToBeSigned* gleich dem Parameter *domainParameter*.τ des Signierschlüssels sein.
  2. signPKCS1\_V1\_5 ist, dann MUSS die Anzahl Oktette in *dataToBeSigned* kleiner als 0,4 OctetLength( *n* ) sein, mit *n* als Modulus des ausgewählten Signaturschlüssels.
  3. signPSS ist, dann DARF *dataToBeSigned* nicht länger als 64 Oktette sein.

1. (N087.400) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *signature* in der Antwortnachricht enthalten ist.
2. (N087.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 187 verwendet werden.

Tabelle 187: PSO Compute Digital Signature, Signieren ohne „message recovery“

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´9E´ | Beschreibung der Antwortdaten, hier digitale Signature |
| P2 | ´9A´ | Beschreibung der Kommandodaten, hier „zu signierende Daten“ |
| Data | ´XX…XX´ | *dataToBeSigned* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Signieren des Datenfeldes, mit „message recovery“

Diese Variante gilt für folgenden Algorithmus (siehe (N018.300)): sign9796\_2\_DS2.

In dieser Variante enthält die APDU des PSO Compute Digital Signature-Kommandos drei Parameter:

1. (N087.600) K\_externeWelt {K\_Karte}   
   Der Parameter *M1* enthält den „recoverable part“ der zu signierenden Nachricht. Der Parameter *M1* ist ein Oktettstring mit beliebigem Inhalt. Die Bitlänge von *M1* DARF die Kapazität des Schemas gemäß [ISO/IEC 9796-2#9.2.4] NICHT überschreiten.
2. Die Kapazität eines RSA-Schlüssels (siehe (N002.100)) beträgt für die Moduluslänge
   1. 2048 bit: c = k – Lh – Ls – 8t – 2 = (2048 – 256 – 256 – 8\*1 – 2) bit = 1526 bit.
   2. 3072 bit: c = k – Lh – Ls – 8t – 2 = (3072 – 256 – 256 – 8\*1 – 2) bit = 2550 bit.
3. (N087.700) K\_externeWelt {K\_Karte}   
   Der Parameter *hashM2* enthält den Hash-Wert des „non recoverable part“ der zu signierenden Nachricht. Der Parameter *hashM2* ist ein Oktettstring mit beliebigem Inhalt, dessen Länge kleiner gleich 64 sein MUSS.
4. (N087.800) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *signature* in der Antwortnachricht enthalten ist.
5. (N087.900) K\_externeWelt {K\_Karte}   
   Die Parameter *M1* und *hashM2* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N088.600)a spezifiziert. Die Anzahl der Oktette in *signInput9796\_2\_DS2* wird durch (N087.600) und (N087.700) beschränkt.
6. (N088.000) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 188 verwendet werden.

Tabelle 188: PSO Compute Digital Signature, Signieren „message recovery“

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´9E´ | Beschreibung der Antwortdaten, hier digitale Signature |
| P2 | ´AC´ | Beschreibung der Kommandodaten, hier zu signierende Daten als DE |
| Data | ´XX…XX´ | *signInput9796\_2\_DS2*, DER codiertes Datenfeld |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Signieren von Daten

Tabelle 189: PSO Compute Digital Signature Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *signature* | Signatur |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Signaturoperation |

Tabelle 190: PSO Compute Digital Signature Antwort-APDU im Fehlerfall

| **Trailer** | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| ´64 00´ | KeyInvalid | Schlüsseldaten fehlen |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoKeyReference | Kein Signierschlüssel ausgewählt |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den geforderten Algorithmus nicht |
| ´6A 88´ | KeyNotFound | Schlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N088.100) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N088.200) K\_COS
   1. Das COS MUSS die PSO Compute Digital Signature-Varianten aus 14.8.2.1 und 14.8.2.2 unterstützen.
   2. Das COS KANN weitere PSO Compute Digital Signature-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N088.300) K\_COS   
   Wenn *channelContext.keyReferenceList.signatureCreation*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchSecretKey(   
          *currentFolder,*   
          *keyReferenceList.signatureCreation.keyReference*,   
          *keyReferenceList.signatureCreation.algID*   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
3. (N088.400) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N088.500) K\_COS   
   Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.
5. (N088.600) K\_COS   
   Die Signatur *signature* MUSS wie folgt berechnet werden:   
   Wenn *keyReferenceList.signatureCreation.algID* den Wert

* 1. rsaClientAuthentication oder signPSS besitzt, dann gilt   
       *signature* = RSASSA\_PSS\_SIGN(   
       *affectedObject.privateRsaKey*,   
       *dataToBeSigned*   
       )

* 1. sign9796\_2\_DS2 besitzt, dann gilt
     1. *signInput9796\_2\_DS2* = *plainDO*   ||   *hashDO*.
     2. *plainDO* = ´80 – L80 – *M1*´.
     3. *hashDO* = ´90 – L90 – *hashM2*´.
     4. *signature* = RSA\_ISO9796\_2\_DS2\_SIGN(   
          *affectedObject.privateRsaKey*,   
          *M1*,   
          *hashM2*,  
         )

* 1. signECDSA besitzt, dann gilt   
       *signature* = *R*   ||   *S*,   mit   
       ( *R*, *S* ) = ELC\_SIG( *affectedObject.privateElcKey*, *dataToBeSigned* )

* 1. signPKCS1\_V1\_5 besitzt, dann gilt:   
       *signature* = RSASSA\_PKCS1\_V1\_5\_SIGN(   
       *affectedObject.privateRsaKey*,   
       *dataToBeSigned*   
       )

1. (N088.700) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS *signature* verwendet werden.
2. (N088.800) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
3. (N088.900) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 190 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 190 MUSS eine höhere Priorität als NoError haben.

### PSO Decipher

Das Kommando PSO Decipher entschlüsselt Daten, die als Parameter in der Kommandonachricht enthalten sind. Für den Entschlüsselungsschlüssel und den zu verwendenden Algorithmus gibt es mehrere Optionen

* Wird ein symmetrischer Schlüssel verwendet, so wird dieser im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, 15.4.3, 15.4.4) ausgehandelt und zusammen mit einem Algorithmus implizit ausgewählt.
* Wird ein asymmetrischer Schlüssel verwendet, der persistent in der Smartcard gespeichert ist, so wird dieser Schlüssel und der zu verwendende Algorithmus vor dem Senden dieses PSO Decipher-Kommandos durch ein explizites MSE-Set-Kommando ausgewählt (siehe 14.9.9.11).

#### Use Case Entschlüsseln mittels RSA

Diese Variante gilt für Algorithmen (siehe (N017.900)) aus folgender Menge: {rsaDecipherPKCS1\_V1\_5, rsaDecipherOaep}.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos zwei Parameter:

1. (N089.000) K\_externeWelt {K\_Karte}   
   Der Parameter *C* enthält die zu entschlüsselnden Daten. Der Parameter *C* ist ein Oktettstring mit beliebigem Inhalt. Die Anzahl Oktette in *C* MUSS identisch sein zu OctetLength( *n* ) mit *n* als Modulus des Schlüssels, der zum Entschlüsseln verwendet wird.
2. (N089.100) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
3. (N089.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 191 verwendet werden.

Tabelle 191: PSO Decipher, Entschlüsseln mittels RSA

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´80´ | Beschreibung der Antwortdaten, hier Klartext |
| P2 | ´86´ | Beschreibung der Kommandodaten, hier Chiffrat |
| Data | ´XX…XX´ | ´00´ || *C*, dies bedeutet: PaddingIndicator || Cryptogram |
| Le | ´0000´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Entschlüsseln mittels ELC

Diese Variante gilt für folgenden Algorithmus (siehe (N017.900)): elcSharedSecretCalculation.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos vier Parameter:

1. (N089.300) K\_externeWelt {K\_Karte}   
   Der Parameter *POA* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt wurde vom Sender der Nachricht gewählt. Der Parameter *POA* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).
2. (N089.400) K\_externeWelt {K\_Karte}   
   Der Parameter *C* enthält die zu entschlüsselnden Daten. Der Parameter *C* ist ein Oktettstring mit beliebigem Inhalt.
3. (N089.500) K\_externeWelt {K\_Karte}   
   Der Parameter *T* enthält einen MAC, der die Integrität von *C* schützt. Der Parameter *T* ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.
4. (N089.600) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.
5. (N089.700) K\_externeWelt {K\_Karte}   
   Die Parameter *POA*, *C* und *T* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N090.300)c spezifiziert.
6. (N089.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 192 verwendet werden.

Tabelle 192: PSO Decipher, Entschlüsseln mittels elliptischer Kurven

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´80´ | Beschreibung der Antwortdaten, hier Klartext |
| P2 | ´86´ | Beschreibung der Kommandodaten, hier Chiffrat |
| Data | ´XX…XX´ | *cipher*, DER codiertes Datenfeld |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

1. Möglicherweise wird P2 in einer späteren Dokumentenversion geändert.

#### Use Case Entschlüsseln mittels symmetrischer Schlüssel

Diese Variante gilt für Algorithmen aus der folgenden Menge: {aesSessionkey, desSessionkey (Option\_DES)}.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos zwei Parameter:

1. (N089.840) K\_externeWelt {K\_Karte}   
   Der Parameter *C* enthält die zu entschlüsselnden Daten. Der Parameter *C* ist ein Oktettstring mit beliebigem Inhalt. Die Anzahl Oktette in *C* MUSS ein ganzzahliges Vielfaches der Blocklänge des verwendeten Kryptoalgorithmus sein.
2. (N089.843) K\_externeWelt {K\_ Karte }   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.
3. (N089.845) K\_externeWelt {K\_ Karte }   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 193 verwendet werden.

Tabelle 193: PSO Decipher, Entschlüsseln mittels symmetrischem Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´80´ | Beschreibung der Antwortdaten, hier Klartext |
| P2 | ´86´ | Beschreibung der Kommandodaten, hier Chiffrat |
| Data | ´XX…XX´ | ´01´ || *C* = Paddingindikator || Kryptogramm |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Entschlüsseln von Daten

Tabelle 194: PSO Decipher Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *plain* | Klartext |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Entschlüsselungsoperation |

Tabelle 195: PSO Decipher Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | KeyInvalid | Schlüsseldaten fehlen |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoKeyReference | Kein Schlüssel für Entschlüsselung ausgewählt |
| ´6A 80´ | WrongCiphertext | Fehler beim Entschlüsseln des Chiffrats |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | KeyNotFound | Schlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N089.900) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N090.000) Kommandovarianten
   1. K\_COS   
      Das COS MUSS die PSO Decipher-Varianten aus 14.8.3.1 und 14.8.3.2 unterstützen.
   2. K\_COS, Option\_Kryptobox, Option\_PACE\_PCD   
      Das COS MUSS die PSO Decipher-Variante aus 14.8.3.3 unterstützen.
   3. Das COS KANN weitere PSO Decipher-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N090.100) K\_COS   
   Wenn *channelContext.keyReferenceList.dataDecipher*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchKey(   
          *currentFolder,*   
          *keyReferenceList.dataDecipher.keyReference*,   
          *keyReferenceList.dataDecipher.algID*   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
3. (N090.200) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N090.210) K\_COS   
   Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.
5. (N090.300) K\_COS   
   Die Klartextnachricht *plain* MUSS wie folgt berechnet werden:   
   Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert

* 1. rsaDecipherPKCS1\_V1\_5 besitzt, dann gilt   
     *plain* = RSAES\_PKCS1\_V1\_5\_DECRYPT( *affectedObject.privateRsaKey*, *C* )

* 1. rsaDecipherOaep besitzt, dann gilt   
     *plain* = RSAES\_OAEP\_DECRYPT( *affectedObject.privateRsaKey*, *C* )

* 1. elcSharedSecretCalculation besitzt, dann gilt (*Hinweis:* cipher *ist hier identisch zu (N085.068)b.7, (N091.700)d und (N094.400)c definiert*)
     1. *cipher* = ´A6–LA6–( *oidDO*   ||   *keyDO*   ||   *cipherDO*   ||   *macDO* ).
     2. *oidDO* = ´06–L06–*oid* ´.
     3. *keyDO* = ´7F49–L7F49–( 86 – L86 – *POA* )´.
     4. *cipherDO* = ´86–L86–( 02   ||   *C* )´.
     5. *macDO* = ´8E–L8E–*T*´.
     6. Falls *oid* verschieden ist zur OID, die gemäß (N008.600)d zu *affectedObject*.*privateElcKey*.*domainParameter* gehört, dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren. Diese Domainparameter werden im Folgenden mit *dP* bezeichnet.
     7. *plain* = ELC\_DEC( OS2P( *POA*, *dP* ), *affectedObject.privateElcKey*, *C*, *T* ).

1. (N090.302) K\_COS, Option\_Kryptobox   
   Die Klartextnachricht *plain* MUSS wie folgt berechnet werden:   
   Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert

* 1. aesSessionkey besitzt, dann gilt mit den Attributen *SSCmac* und *Kenc* aus dem Attribut *SessionkeyContext*
     1. Schritt 1: *SSCenc* = OS2I(AES\_ENC(*Kenc*, I2OS(*SSCmac*, 16))).
     2. Schritt 2: *P* = AES\_CBC\_DEC(*Kenc*, *SSCenc*, *C* ).
     3. Schritt 3: *plain* = TruncateIso( *P*, 16 ).
     4. Falls die Funktion TruncateIso mit dem Fehler paddingError terminiert, dann
        1. MUSS das Kommando mit dem Trailer WrongCiphertext terminieren und
        2. die Sessionkeys MÜSSEN mittels clearSessionkeys( ) gelöscht werden.
  2. Option\_DES, desSessionkey besitzt, dann gilt mit den Attributen *SSCenc* und *Kenc* aus dem Attribut *SessionkeyContext*
     1. Schritt 1: *SSCenc* = *SSCenc* + 1.
     2. Schritt 2: *P* = 3TDES\_CBC\_DEC(*Kenc*, *SSCenc*, *C* ).
     3. Schritt 3: *plain* = TruncateIso( *P*, 8 ).
     4. Falls die Funktion TruncateIso mit dem Fehler paddingError terminiert, dann
        1. MUSS das Kommando mit dem Trailer WrongCiphertext terminieren und
        2. die Sessionkeys MÜSSEN mittels clearSessionkeys( ) gelöscht werden.

1. (N090.400) K\_COS   
   Wenn die Entschlüsselung gemäß (N090.300) mit dem Fehler „*ERROR*“ terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.
2. (N090.500) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS *plain* verwendet werden.
3. (N090.600) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
4. (N090.700) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 195 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 195 MUSS eine höhere Priorität als NoError haben.

### PSO Encipher

Das Kommando PSO Encipher verschlüsselt Daten, die als Parameter in der Kommandonachricht enthalten sind. Für den Verschlüsselungsschlüssel und den zu verwendenden Verschlüsselungsalgorithmus gibt es mehrere Optionen:

* Wird ein symmetrischer Schlüssel verwendet, so wird dieser im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, 15.4.3, 15.4.4) ausgehandelt und zusammen mit einem Verschlüsselungsalgorithmus implizit ausgewählt.
* Wird ein asymmetrischer Schlüssel verwendet, der persistent in der Smartcard gespeichert ist, so wird dieser Schlüssel und der zu verwendende Algorithmus vor dem Senden dieses PSO Encipher-Kommandos durch ein explizites MSE-Set-Kommando ausgewählt (siehe 14.9.9.12).
* Wird ein asymmetrischer Schlüssel verwendet, der als Parameter in der Kommandonachricht enthalten ist, so ist auch der zu verwendende Algorithmus als Parameter in der Kommandonachricht enthalten.

#### Use Case Verschlüsseln von Daten mittels übergebenem RSA-Schlüssel

In dieser Variante wird der Schlüssel als Parameter in der Kommandonachricht übergeben. Folgende Algorithmen sind zulässig: {rsaEncipherOaep, rsaEncipherPKCS1\_V1\_5}.

1. Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher leer ist.

In dieser Variante enthält die APDU des PSO Encipher-Kommandos vier Parameter:

1. (N090.780) K\_externeWelt {K\_Karte}   
   Der Parameter *algID* enthält Informationen zum Algorithmus, der für die Verschlüsselung verwendet wird. Der Parameter *algID* MUSS aus der Menge {rsaEncipherOaep, rsaEncipherPKCS1\_V1\_5} gewählt werden.
2. (N090.782) K\_externeWelt {K\_Karte}   
   Der Parameter *PuK* enthält den öffentlichen Schlüssel des Empfängers gemäß 8.2.4.1. Der Parameter *PuK* ist ein Oktettstring, dessen Inhalt so gewählt werden MUSS, dass bei der Decodierung kein Fehler auftritt (siehe (N091.700){b.3, c.3}).
3. (N090.784) K\_externeWelt {K\_Karte}   
   Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die maximal mögliche Länge von *M* ist abhängig von der gemäß (N090.780) übergebenen *algId* und der Länge des Modulus *n*. Wenn *algId* gleich
   1. rsaEncipherOaep ist, dann DARF OctetLength( *M* ) NICHT größer als OctetLength( *n* ) – 66 sein.
   2. rsaEncipherPKCS1\_V1\_5 ist, dann DARF OctetLength( *M* ) NICHT größer als OctetLength( *n* ) – 11 sein.
4. (N090.786) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
5. (N090.788) K\_externeWelt {K\_Karte}   
   Die Parameter *PuK*, *algID* und *M* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N091.700){b, c} spezifiziert.
6. (N090.790) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 196 verwendet werden.

Tabelle 196: PSO Encipher, Verschlüsseln mittels übergebenem RSA-Schlüssel

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´80´ | Beschreibung der Kommandodaten, hier Klartext |
| Data | ´XX…XX´ | *plainDO*, DER codiertes Datenfeld |
| Le | ´0000´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Verschlüsseln von Daten mittels übergebenem ELC-Schlüssel

In dieser Variante wird der Schlüssel als Parameter in der Kommandonachricht übergeben. Folgende Algorithmen sind zulässig: {elcSharedSecretCalculation}.

1. Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher leer ist.

In dieser Variante enthält die APDU des PSO Encipher Kommandos fünf Parameter:

1. (N090.800) K\_externeWelt {K\_Karte}   
   Der Parameter *algID* enthält Informationen zum Algorithmus, der für die Verschlüsselung verwendet wird. Der Parameter *algID* MUSS aus der Menge {elcSharedSecretCalculation} gewählt werden.
2. (N090.900) K\_externeWelt {K\_Karte}   
   Der Parameter *oid* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve referenziert. Der Parameter *oid* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.
3. (N091.000) K\_externeWelt {K\_Karte}   
   Der Parameter *POB* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Empfängers. Der Parameter *POB* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.500)a).
4. (N091.100) K\_externeWelt {K\_Karte}   
   Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *M* MUSS so gewählt werden, dass für die Kommandonachricht die Vorgaben aus (N026.900) eingehalten werden und die Antwortnachricht nicht länger ist als *limitRspSecureMessaging*.
5. (N091.200) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipher* in der Antwortnachricht enthalten ist.
6. (N091.300) K\_externeWelt {K\_Karte}   
   Die Parameter *algID*, *oid*, *POB* und *M* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N091.700)d spezifiziert.
7. (N091.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 197 verwendet werden.

Tabelle 197: PSO Encipher, Verschlüsseln mittels übergebenem ELC-Schlüssel

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´80´ | Beschreibung der Kommandodaten, hier Klartext |
| Data | ´XX…XX´ | *plainDO* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Verschlüsseln mittels gespeichertem RSA-Schlüssel

In dieser Variante wird ein in der Smartcard gespeicherter Schlüssel verwendet. Folgende Algorithmen sind zulässig: {rsaEncipherOaep, rsaEncipherPKCS1\_V1\_5}.

1. Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher Daten enthält.

In dieser Variante enthält die APDU des PSO Encipher-Kommandos zwei Parameter:

1. (N091.420) K\_externeWelt {K\_COS}   
   Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die maximal mögliche Länge von *M* ist abhängig von der mittels (N103.845) ausgewählten *algId* und der Länge des Modulus *n*. Wenn *algId* gleich
   1. rsaEncipherOaep ist, dann DARF OctetLength( *M* ) NICHT größer als OctetLength( *n* ) – 66 sein.
   2. rsaEncipherPKCS1\_V1\_5 ist, dann DARF OctetLength( *M* ) NICHT größer als OctetLength( *n* ) – 11 sein.
2. (N091.422) K\_externeWelt {K\_COS}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
3. (N091.424) K\_externeWelt {K\_COS}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 198 verwendet werden.

Tabelle 198: PSO Encipher, Verschlüsseln mittels gespeichertem RSA-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´80´ | Beschreibung der Kommandodaten, hier Klartext |
| Data | ´XX…XX´ | *M* |
| Le | ´0000´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Verschlüsseln mittels gespeichertem ELC-Schlüssel

In dieser Variante wird ein in der Smartcard gespeicherter Schlüssel verwendet. Folgende Algorithmen sind zulässig: {elcSharedSecretCalculation}.

1. Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher Daten enthält.

In dieser Variante enthält die APDU des PSO Encipher-Kommandos zwei Parameter:

1. (N091.430) K\_externeWelt {K\_COS}   
   Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *M* MUSS so gewählt werden, dass für die Kommandonachricht die Vorgaben aus (N026.900) eingehalten werden und die Antwortnachricht nicht länger ist als *limitRspSecureMessaging*.
2. (N091.432) K\_externeWelt {K\_COS}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipher* in der Antwortnachricht enthalten ist.
3. (N091.434) K\_externeWelt {K\_COS}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 199 verwendet werden.

Tabelle 199: PSO Encipher, Verschlüsseln mittels gespeichertem ELC-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´80´ | Beschreibung der Kommandodaten, hier Klartext |
| Data | ´XX…XX´ | *M* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Verschlüsseln mittels symmetrischem Schlüssel

In dieser Variante wird ein in der Smartcard gespeicherter Schlüssel verwendet. Folgende Algorithmen sind zulässig: {aesSessionkey, desSessionkey (Option\_DES)}.

1. Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher Daten enthält.

In dieser Variante enthält die APDU des PSO Encipher-Kommandos zwei Parameter:

1. (N091.440) K\_externeWelt {K\_COS}   
   Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *M* MUSS so gewählt werden, dass für die Kommandonachricht die Vorgaben aus (N026.900) eingehalten werden und die Antwortnachricht nicht länger ist als *limitRspSecureMessaging*.
2. (N091.443) K\_externeWelt {K\_COS}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipher* in der Antwortnachricht enthalten ist.
3. (N091.446) K\_externeWelt {K\_COS}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 200 verwendet werden.

Tabelle 200: PSO Encipher, Verschlüsseln mittels symmetrischem Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´80´ | Beschreibung der Kommandodaten, hier Klartext |
| Data | ´XX…XX´ | *M* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Verschlüsseln von Daten

Tabelle 201: PSO Encipher Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *cipher* | Chiffrat |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Verschlüsselungsoperation |

Tabelle 202: PSO Encipher Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | EncipherError | Verschlüsselung fehlgeschlagen |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | KeyNotFound | Schlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N091.500) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N091.600) Kommandovarianten
   1. K\_COS   
      Das COS MUSS die PSO Encipher-Variante aus 14.8.4.1 und 14.8.4.2 unterstützen.
   2. K\_COS, Option\_Kryptobox   
      Das COS MUSS die PSO Encipher-Variante aus 14.8.4.3 und 14.8.4.4 unterstützen.
   3. K\_COS, Option\_Kryptobox, Option\_PACE\_PCD   
      Das COS MUSS die PSO Encipher-Variante aus 14.8.4.5 unterstützen.
   4. Das COS KANN weitere PSO Encipher-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N091.650) K\_COS   
   Wenn *channelContext.keyReferenceList.dataEncipher* nicht leer ist, genau dann MUSS das Kommando auf folgende Art mit einem in der Smartcard gespeicherten Schlüssel arbeiten:
   1. Es wird *affectedObject* = SearchKey(   
      *currentFolder,   
          keyReferenceList.dataEncipher.keyReference,   
          keyReferenceList.dataEncipher.algID*) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
   2. Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
   3. Das Chiffrat *cipher* MUSS wie folgt berechnet werden: Wenn *channelContext.keyReferenceList.dataEncipher.algID* den Wert

* + 1. aesSessionkey besitzt, dann gilt mit den Attributen *SSCmac* und *Kenc* aus dem Attribut *SessionkeyContext*
       1. Schritt 0: *SSCmac* = *SSCmac* + 1.
       2. Schritt 1: *SSCenc* = OS2I(AES\_ENC (*Kenc*, I2OS(*SSCmac*, 16))).
       3. Schritt 2: *P* = PaddingIso( *M*, 16 ).
       4. Schritt 3: *C* = AES\_CBC\_ENC(*Kenc*, *SSCenc*, *P* ).
       5. Schritt 3: *cipher* = ´01´ || *C*.
    2. Option\_DES, desSessionkey besitzt, dann gilt mit den Attributen *SSCenc* und *Kenc* aus dem Attribut *SessionkeyContext*
       1. Schritt 1: *SSCenc* = *SSCenc* + 1.
       2. Schritt 2: *P* = PaddingIso( *M*, 8 ).
       3. Schritt 3: *C* = 3TDES\_CBC\_ENC(*Kenc*, *SSCenc*, *P* ).
       4. Schritt 4: *cipher* = ´01´ || *C*.

* + 1. elcSharedSecretCalculation besitzt, dann MUSS gelten:
       1. ( *POA*, *C*, *T* ) = ELC\_ENC(   
              *M*,   
              *affectedObject.publicElcKey.P*,   
              *affectedObject*.*domainParameter*   
          ).   
          Wenn diese Funktion mit dem Fehler „*ERROR*“ terminiert, genau dann MUSS das Kommando mit dem Trailer EncipherError terminieren. Andernfalls wird ein DER-TLV codierter Oktettstring cipher wie folgt konstruiert: *Hinweis: cipher ist hier identisch zu (N090.300)c und (N094.400)c definiert.*
       2. Setze   *oidDO* = ´06–L06–*affectedObject*.*domainParameter*.OID´.
       3. Setze   *keyDO* = ´7F49–L7F49–( 86–L86–*POA* )´.
       4. Setze   *cipherDO* = ´86–L86–( 02   ||   *C* )´.
       5. Setze   *macDO* = ´8E–L8E–*T*´.
       6. Setze   *cipher* = ´A6–LA6–(*oidDO* || *keyDO* || *cipherDO* || *macDO* )´.
    2. rsaEncipherOaep besitzt, dann gilt:   
       Schritt 1: *C* = RSAES\_OAEP\_ENCRYPT( *affectedObject.publicKey*, *M* )   
       Schritt 2: *cipher* = ´85´ || *C*.
    3. rsaEncipherPKCS1\_V1\_5 besitzt, dann gilt:   
       Schritt 1: *C* = RSAES\_PKCS1\_V1\_5\_ENCRYPT(*affectedObject.publicKey*, *M*)  
       Schritt 2: *cipher* = ´81´ || *C*.

1. (N091.700) K\_COS   
   Wenn *channelContext.keyReferenceList.dataEncipher* leer ist, genau dann MUSS das Kommando auf folgende Art mit einem Schlüssel in den Kommanddaten arbeiten: Das Chiffrat *cipher* MUSS wie folgt berechnet werden:
   1. Suche in *plainDO* nach einem DO mit Tag = ´80´mit der Länge eins. Es gilt *algID\_enc* = Wertfeld dieses DO. Wenn *algID\_enc* den Wert

* 1. rsaEncipherPKCS1\_V1\_5 besitzt, dann gilt:
     1. *plainDO* = ´A0–LA0–( *algDO*   ||   *keyDO*   ||   *mDO* ).
     2. *algDO* = ´80 01   *algID\_enc*´.
     3. *keyDO* = ´7F49–L7F49–[(81–L81–*PuK.n*) || (82–L82–*PuK.e*)]´.
     4. *mDO* = ´80–L80–*M*´.
     5. *cipher* = ´00´ || RSAES\_PKCS1\_V1\_5\_ENCRYPT(*PuK*, *M* ).

* 1. rsaEncipherOaep besitzt, dann gilt:
     1. *plainDO* = ´A0–LA0–( *algDO*   ||   *keyDO*   ||   *mDO* ).
     2. *algDO* = ´80 01   *algID\_enc*´.
     3. *keyDO* = ´7F49–L7F49–[(81–L81–*PuK.n*) || (82–L82–*PuK.e*)]´.
     4. *mDO* = ´80–L80–*M*´.
     5. *cipher* = ´00´ || RSAES\_OAEP\_ENCRYPT( *PuK*, *M* ).

* 1. elcSharedSecretCalculation besitzt, dann MUSS gelten (*Hinweis:* cipher *ist hier identisch zu (N085.068)b.7, (N090.300)c und (N094.400)c definiert*):
     1. *plainDO* = ´A0–LA0–( *algDO*   ||   *oidDO*   ||   *keyDO*   ||   *mDO* ).
     2. *algDO* = ´80 01   *algID\_enc*´.
     3. *oidDO* = ´06–L06–*oid* ´.
     4. *keyDO* = ´7F49–L7F49–( 86–L86–*POB* )´.
     5. *mDO* = ´80–L80–*M*´.
     6. Die *oid* aus der Kommandonachricht wird gemäß Tabelle 271 in Domainparameter übersetzt, die im Folgenden mit *dP* bezeichnet werden.
     7. ( *POA*, *C*, *T* ) = ELC\_ENC( *M*, *POB*, *dP* )   
        Wenn diese Funktion mit dem Fehler „ERROR“ terminiert, genau dann MUSS das Kommando mit dem Trailer EncipherError terminieren. Andernfalls wird ein DER-TLV codierter Oktettstring *cipher* wie folgt konstruiert:
     8. Setze *keyDO* = ´7F49–L7F49–( 86–L86–*POA* )´.
     9. Setze *cipherDO* = ´86–L86–( 02   ||   *C* )´.
     10. Setze *macDO* = ´8E–L8E–*T*´.
     11. Setze *cipher* = ´A6–LA6–(*oidDO* || *keyDO* || *cipherDO* || *macDO* ).

1. (N091.800) Diese Anforderung ist absichtlich leer. Die in einer früheren Version hier enthaltenen Anforderungen sind nun in (N091.700) enthalten.
2. (N091.900) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS *cipher* verwendet werden.
3. (N092.000) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
4. (N092.100) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 202 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 202 MUSS eine höhere Priorität als NoError haben.

### PSO Hash

1. (N092.200) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-8]
   1. unterstützen oder
   2. ablehnen.

### PSO Transcipher

Das Kommando PSO Transcipher kombiniert die Kommandos PSO Decipher und PSO Encipher, ohne dass die Klartextdaten die Karte verlassen. Es werden also Daten mittels eines privaten Schlüssels entschlüsselt und sofort wieder mittels eines öffentlichen Schlüssels verschlüsselt. Der private Schlüssel und der zu verwendende Algorithmus werden vor der Entschlüsselungsoperation ausgewählt. Dies geschieht vor dem Senden dieses PSO Transcipher-Kommandos durch ein MSE-Set-Kommando (siehe 14.9.9.11). Die umzuschlüsselnden Daten und der öffentliche Schlüssel des Empfängers sind als Parameter in der Kommandonachricht enthalten.

#### Use Case Umschlüsseln von Daten mittels RSA-Schlüssel

In dieser Variante wird ein Chiffrat mittels eines RSA-Schlüssels entschlüsselt und anschließend mit einem anderen RSA-Schlüssels verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos vier Parameter:

1. (N092.300) K\_externeWelt {K\_Karte}   
   Der Parameter *Cin* enthält die umzuschlüsselnden Daten. Der Parameter *Cin* ist ein Oktettstring mit beliebigem Inhalt.
2. (N092.400) K\_externeWelt {K\_Karte}   
   Der Parameter *PuK* enthält den öffentlichen Schlüssel des Empfängers gemäß 8.2.4.1. Der Parameter *PuK* ist ein Oktettstring, dessen Inhalt so gewählt werden MUSS, dass bei der Decodierung kein Fehler auftritt (siehe (N094.400)b.3).
3. (N092.500) K\_externeWelt {K\_Karte}   
   Der Parameter *algID\_enc* enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {rsaEncipherPKCS1\_V1\_5, rsaEncipherOaep} verwendet werden.
4. (N092.600) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
5. (N092.700) K\_externeWelt {K\_Karte}   
   Die Parameter *Cin*, *PuK* und *algID\_enc* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.
6. (N092.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 203 verwendet werden.

Tabelle 203: PSO Transcipher, Umschlüsseln von Daten mittels RSA-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´B8´ | Beschreibung der Kommandodaten, hier CRT für Verschlüsselung |
| Data | ´XX…XX´ | *cipherIN*, DER codiertes Datenfeld |
| Le | ´0000´ | length, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Umschlüsseln von Daten von RSA-Schlüssel nach ELC-Schlüssel

In dieser Variante wird ein Chiffrat mittels eines RSA-Schlüssels entschlüsselt und anschließend mit einem ELC-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos fünf Parameter:

1. (N092.820) K\_externeWelt {K\_Karte}   
   Der Parameter *Cin* enthält die umzuschlüsselnden Daten. Der Parameter *Cin* ist ein Oktettstring mit beliebigem Inhalt.
2. (N092.821) K\_externeWelt {K\_Karte}   
   Der Parameter *oidout* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Verschlüsselung referenziert. Der Parameter *oidout* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.
3. (N092.822) K\_externeWelt {K\_Karte}   
   Der Parameter *POB* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Empfängers. Der Parameter *POB* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.500)a).
4. (N092.824) K\_externeWelt {K\_Karte}   
   Der Parameter *algID\_enc* enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {elcSharedSecretCalculation} verwendet werden.
5. (N092.826) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
6. (N092.828) K\_externeWelt {K\_Karte}   
   Die Parameter *Cin*, *oidout*, *POB* und *algID\_enc* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.
7. (N092.830) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 204 verwendet werden.

Tabelle 204: PSO Transcipher, Umschlüsseln von Daten von RSA nach ELC

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´B8´ | Beschreibung der Kommandodaten, hier CRT für Verschlüsselung |
| Data | ´XX…XX´ | *cipherIN* |
| Le | ´0000´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Umschlüsseln von Daten mittels ELC

In dieser Variante wird ein Chiffrat mittels eines ELC-Schlüssels entschlüsselt und anschließend mit einem anderen ELC-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos acht Parameter:

1. (N092.900) K\_externeWelt {K\_Karte}   
   Der Parameter *POA* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt wurde vom Sender der Nachricht gewählt. Der Parameter *POA* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).
2. (N092.902) K\_externeWelt {K\_Karte}   
   Der Parameter *oidin* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Entschlüsselung referenziert. Der Parameter *oidin* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.
3. (N093.000) K\_externeWelt {K\_Karte}   
   Der Parameter *Cin* enthält die umzuschlüsselnden Daten. Der Parameter *Cin* ist ein Oktettstring mit beliebigem Inhalt.
4. (N093.100) K\_externeWelt {K\_Karte}   
   Der Parameter *Tin* enthält einen MAC, der die Integrität von *Cin* schützt. Der Parameter *Tin* ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.
5. (N093.200) K\_externeWelt {K\_Karte}   
   Der Parameter *POB* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Empfängers. Der Parameter *POB* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).
6. (N093.202) K\_externeWelt {K\_Karte}   
   Der Parameter *oidout* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Verschlüsselung referenziert. Der Parameter *oidout* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.
7. (N093.300) K\_externeWelt {K\_Karte}   
   Der Parameter *algID\_enc* enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {elcSharedSecretCalculation} verwendet werden.
8. (N093.400) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipherOUT* in der Antwortnachricht enthalten ist.
9. (N093.500) K\_externeWelt {K\_Karte}   
   Die Parameter *POA*, *oidin*, *Cin*, *Tin*, *POB*, *oidout* und *algID\_enc* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.
10. (N093.600) K\_externeWelt {K\_Karte}   
    Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 205 verwendet werden.

Tabelle 205: PSO Transcipher, Umschlüsseln von Daten mittels ELC-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´B8´ | Beschreibung der Kommandodaten, hier CRT für Verschlüsselung |
| Data | ´XX…XX´ | *cipherIN* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Umschlüsseln von Daten von ELC-Schlüssel nach RSA-Schlüssel

In dieser Variante wird ein Chiffrat mittels eines ELC-Schlüssels entschlüsselt und anschließend mit einem anderen RSA-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos sieben Parameter:

1. (N093.620) K\_externeWelt {K\_Karte}   
   Der Parameter *POA* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt wurde vom Sender der Nachricht gewählt. Der Parameter *POA* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).
2. (N093.621) K\_externeWelt {K\_Karte}   
   Der Parameter *oidin* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Entschlüsselung referenziert. Der Parameter *oidin* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.
3. (N093.622) K\_externeWelt {K\_Karte}   
   Der Parameter *Cin* enthält die umzuschlüsselnden Daten. Der Parameter *Cin* ist ein Oktettstring mit beliebigem Inhalt.
4. (N093.624) K\_externeWelt {K\_Karte}   
   Der Parameter *Tin* enthält einen MAC, der die Integrität von *C* schützt. Der Parameter *Tn* ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.
5. (N093.626) K\_externeWelt {K\_Karte}   
   Der Parameter *PuK* enthält den öffentlichen Schlüssel des Empfängers gemäß 8.2.4.1. Der Parameter *PuK* ist ein Oktettstring, dessen Inhalt so gewählt werden MUSS, dass bei der Decodierung kein Fehler auftritt (siehe (N094.400)b.3).
6. (N093.628) K\_externeWelt {K\_Karte}   
   Der Parameter *algID\_enc* enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {rsaEncipherPKCS1\_V1\_5, rsaEncipherOaep} verwendet werden.
7. (N093.630) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
8. (N093.632) K\_externeWelt {K\_Karte}   
   Die Parameter *POA*, *oidn*, *Cin*, *Tn*, *PuK* und *algID\_enc* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.
9. (N093.634) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 206 verwendet werden.

Tabelle 206: PSO Transcipher, Umschlüsseln von Daten von ELC nach RSA

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´86´ | Beschreibung der Antwortdaten, hier Chiffrat |
| P2 | ´B8´ | Beschreibung der Kommandodaten, hier CRT für Verschlüsselung |
| Data | ´XX…XX´ | *cipherIN* |
| Le | ´0000´ | length, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Umschlüsseln von Daten

Tabelle 207: PSO Transcipher Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | cipherOUT | Chiffrat |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Umschlüsselungsoperation |

Tabelle 208: PSO Transcipher Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | KeyInvalid | Schlüsseldaten fehlen |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoKeyReference | Kein Schlüssel zur Entschlüsselung ausgewählt |
| ´6A 80´ | WrongCiphertext | Fehler beim Entschlüsseln des Chiffrats |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den geforderten Algorithmus nicht |
| ´6A 84´ | MessageTooLong | Klartext zu lang für Verschlüsselung |
| ´6A 88´ | KeyNotFound | Schlüssel für Entschlüsselung nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N093.700) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N093.800) K\_COS
   1. Das COS MUSS die PSO Transcipher-Variante aus 14.8.6.1, 14.8.6.2, 14.8.6.3 und 14.8.6.4 unterstützen.
   2. Das COS KANN weitere PSO Transcipher-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N093.900) K\_COS   
   Wenn *channelContext.keyReferenceList.dataDecipher*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* gleich dem Schlüssel gesetzt, der durch SearchSecretKey(   
          *currentFolder,*   
          *keyReferenceList.dataDecipher.keyReference*,   
          *keyReferenceList.dataDecipher.algID*   
      ) bezeichnet wird. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
3. (N094.000) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N094.010) K\_COS   
   Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.
5. (N094.100) K\_COS   
   Die Kommandodaten werden wie folgt aufgeteilt:
   1. *cipherIN* = *cipher*   ||   *plainDO*.
   2. *cipher* MUSS ein DO mit Tag = ´A6´ sein.
   3. *plainDO* MUSS ein DO mit Tag = ´A0´ sein.
6. (N094.200) K\_COS   
   Der Klartext wird wie folgt berechnet:   
   Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert

* 1. rsaDecipherPKCS1\_V1\_5 besitzt, dann gilt:
     1. *cipher* = ´A6–LA6–( *cipherDOin* )´.
     2. *cipherDOin* = ´86–L86–( 00   ||   *Cin* )´.
     3. *M* = RSAES\_PKCS1\_V1\_5\_DECRYPT( *affectedObject.privateRsaKey*, *Cin* ).      
        Wenn diese Funktion mit dem Fehler „ERROR“ terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.

* 1. rsaDecipherOaep besitzt, dann gilt:
     1. *cipher* = ´A6–LA6–( *cipherDOin* )´.
     2. *cipherDOin* = ´86–L86–( 00   ||   *Cin* )´.
     3. *M* = RSAES\_OAEP\_DECRYPT( *affectedObject.privateRsaKey*, *Cin* ).   
        Wenn diese Funktion mit dem Fehler „ERROR“ terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.

* 1. elcSharedSecretCalculation besitzt, dann gilt (*Hinweis:* cipher *ist hier identisch zu (N090.300)c und (N091.700)d definiert*):
     1. *cipher* = ´A6–LA6–( *oidDOin* || *keyDOA* || *cipherDOin* || *macDOin* )´.
     2. *oidDOin* = ´06–L06–*oidin* ´.
     3. *keyDOA* = ´7F49–L7F49–( 86–L86–*POA* )´.
     4. *cipherDOin* = ´86–L86–( 02   ||   *Cin* )´.
     5. *macDOin* = ´8E–L8E–*Tin*´.
     6. Falls *oidin* verschieden ist zur OID, die gemäß (N008.600)d zu *affectedObject*.*privateElcKey*.*domainParameter* gehört, dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.
     7. *M* = ELC\_DEC( *POA*, *affectedObject.privateElcKey*, *Cin*, *Tin* ).   
        Wenn diese Funktion mit dem Fehler „ERROR“ terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.

1. (N094.300) K\_COS   
   Suche in *plainDO* nach einem DO mit Tag = ´80´ dieses DO MUSS die Länge eins haben. Es gilt *algID\_enc* = Wertfeld des DO mit Tag = ´80´.
2. (N094.400) K\_COS   
   Das Chiffrat *cipherOUT* MUSS wie folgt aus *M* berechnet werden: Wenn *algID\_enc* den Wert

* 1. rsaEncipherPKCS1\_V1\_5 besitzt, dann gilt:
     1. *plainDO* = ´A0–LA0–( *algDO*   ||   *keyDO* ).
     2. *algDO* = ´80 01   ||   *algID\_enc*´.
     3. *keyDO* = ´7F49–L7F49–[(81–L81–*PuK.n*) || (82–L82–*PuK.e*)]´.
     4. *cipherOUT* = ´00´ || RSAES\_PKCS1\_V1\_5\_ENCRYPT(*PuK*, *M* ).   
        Wenn diese Funktion mit dem Fehler “ERROR” terminiert, dann MUSS das Kommando mit dem Trailer MessageTooLong terminieren.
  2. rsaEncipherOaep besitzt, dann gilt:
     1. *plainDO* = ´A0–LA0–( *algDO*   ||   *keyDO* ).
     2. *algDO* = ´80 01   ||  *algID\_enc*´.
     3. *keyDO* = ´7F49–L7F49–[(81–L81–*PuK.n*) || (82–L82–*PuK.e*)]´.
     4. *cipherOUT* = ´00´ || RSAES\_OAEP\_ENCRYPT( *PuK*, *M* ).   
        Wenn diese Funktion mit dem Fehler “ERROR” terminiert, dann MUSS das Kommando mit dem Trailer MessageTooLong terminieren.

* 1. elcSharedSecretCalculation besitzt, dann gilt (*Hinweis:* cipherOUT *ist hier identisch zu (N085.068)b.7, (N090.300)c und* (N091.700)*d definiert*):
     1. *plainDO* = ´A0–LA0–( *algDO*   ||   *oidDOout*   ||   *keyDOB* ).
     2. *algDO* = ´80 01   ||   *algID\_enc*´.
     3. *oidDOout* = ´06–L06–*oidout* ´.
     4. *keyDOB* = ´7F49–L7F49–( 86–L86–*POB* )´.
     5. Die *oidout* aus der Kommandonachricht wird gemäß Tabelle 271 in Domainparameter übersetzt, die im Folgenden mit *dP* bezeichnet werden.
     6. ( *POout*, *Cout*, *Tout* ) = ELC\_ENC( *M*, *POB*, *dP* )   
        Wenn diese Funktion mit dem Fehler „ERROR“ terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren. Andernfalls wird ein DER-TLV codierter Oktettstring *cipherOUT* wie folgt konstruiert:
     7. Setze *keyDOout* = ´7F49–L7F49–( 86–L86–*POout* )´.
     8. Setze *cipherDOout* = ´86–L86–( 02   ||   *Cout* )´.
     9. Setze *macDOout* = ´8E–L8E–*Tout*´.
     10. Setze *cipherOUT* = ´A6–LA6–(   
           *oidDOout*   ||   *keyDOout*   ||   *cipherDOout*   ||   *macDOout*   
           ).

1. (N094.500) K\_COS   
   Als Datenfeld der Antwortnachricht MUSS *cipherOUT* verwendet werden.
2. (N094.600) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
3. (N094.700) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 208 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 208 MUSS eine höhere Priorität als NoError haben.

### PSO Verify Certificate

Das Kommando PSO Verify Certificate überprüft die Signatur eines Zertifikates mittels eines öffentlichen Schlüssels. Der öffentliche Schlüssel wird vor der Verifikationsoperation ausgewählt. Dies geschieht vor dem Senden dieses PSO Verify Certificate-Kommandos durch ein MSE-Set-Kommando (siehe 14.9.9.10). Das Zertifikat ist als Parameter in der Kommandonachricht enthalten. Falls die Signatur des Zertifikates als gültig betrachtet wird, dann werden gewisse Attribute des im Zertifikat enthaltenen Schlüsselobjektes zur späteren Verwendung gespeichert.

#### Use Case Import RSA-Schlüssels mittels Zertifikat, Option\_RSA\_CVC

In dieser Variante enthält die APDU des PSO Verify Certificate-Kommandos zwei Parameter:

1. (N094.800) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der Parameter *certificateContent* enthält Attributsinformationen für das Schlüsselobjekt.
2. (N094.810) K\_Anwendungsspezifikation {K\_Karte}, Option\_RSA\_CVC   
   Die Anwendungsspezifikation einer Karte MUSS eine vollständige Liste aller CHA-Werte enthalten, die von dieser Karte zu unterstützen ist.
3. (N094.820) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der in *certificateContent* enthaltene Wert für CHA MUSS aus Element der Anwendungsspezifikation genannten Menge sein.
4. (N094.900) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Der Parameter *signature* enthält die von einer CA erstellte Signatur über die Attributsinformationen.
5. (N095.000) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Die Parameter *certificateContent* und *signature* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N095.900)a spezifiziert.
6. (N095.100) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 209 verwendet werden.

Tabelle 209: PSO Verify Certificate für RSA-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Keine Antwortdaten |
| P2 | ´AE´ | Kommandodaten mit Template, dessen Wertfelder zertifiziert sind |
| Data | ´XX…XX´ | *certificate* |

#### Use Case Import ELC-Schlüssels mittels Zertifikat

In dieser Variante enthält die APDU des PSO Verify Certificate-Kommandos zwei Parameter:

1. (N095.200) K\_externeWelt {K\_Karte}   
   Der Parameter *certificateContent* enthält alle Attribute für das Schlüsselobjekt.
2. (N095.300) K\_externeWelt {K\_Karte}   
   Der Parameter *signature* enthält die von einer CA erstellte Signatur über die Attribute.
3. (N095.400) K\_externeWelt {K\_Karte}   
   Die Parameter *certificateContent* und *signature* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N095.900)b spezifiziert.
4. (N095.410) K\_COS   
   Der öffentliche Punkt *P* in Parameter *certificateContent* liegt gemäß [gemSpec\_PKI] auf einer der Kurven brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1. Die Signature im Parameter *signature* ist gemäß [gemSpec\_PKI] mit einem offentlichen Schlüssel prüfbar, der auf einer der Kurven brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1 liegt. In einem CV‑Zertifikat für elliptische Kurven ist eine Kurve für *certificateContent* zu kombinieren mit einer Kurve für *signature*. Für die zu unterstützenden Kombinationen durch das COS MÜSSEN die in Tabelle 210 genannten Schlüsselwörter gelten.

Tabelle 210: Kombination von Kurvenparametern in CV‑Zertifikaten

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| certificateContent  signature | brainpoolP256r1 | brainpoolP384r1 | brainpoolP512r1 | andere |
| brainpoolP256r1 | MUSS | MUSS | MUSS | KANN |
| brainpoolP384r1 | MUSS | MUSS | MUSS | KANN |
| brainpoolP512r1 | MUSS | MUSS | MUSS | KANN |
| andere | KANN | KANN | KANN | KANN |

1. (N095.500) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 211 verwendet werden.

Tabelle 211: PSO Verify Certificate für ELC-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Keine Antwortdaten |
| P2 | ´BE´ | Kommandodaten mit zertifiziertem Template |
| Data | ´XX…XX´ | *certificate* |

#### Antwort der Karte auf Vergleich eines Benutzergeheimnisses

Tabelle 212: PSO Verify Certificate Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Zertifikatsprüfung |

Tabelle 213: PSO Verify Certificate Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | KeyExpired | Gültigkeitszeitraum des  Signaturprüfschlüssels ist abgelaufen  Zertifikates ist abgelaufen |
| ´69 85´ | NoKeyReference | Kein Signaturprüfschlüssel ausgewählt |
| ´6A 80´ | VerificationError | Prüfung des Zertifikates fehlgeschlagen |
| ´6A 88´ | InconsistentKeyReference | Signaturprüfschlüssel hat eine andere Referenz als CAR des Zertifikates |
| ´6A 88´ | KeyNotFound | Signaturprüfschlüssel nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N095.600) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N095.700) K\_COS
   1. Das COS MUSS die PSO Verify Certificate-Variante aus
      1. Option\_RSA\_CVC, 14.8.7.1 und
      2. 14.8.7.2 unterstützen.
   2. Das COS KANN weitere PSO Verify Certificate-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N095.800) K\_COS   
   Wenn *channelContext.keyReferenceList.verifyCertificate*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchPublicKey(   
          *currentFolder,*   
          *keyReferenceList.verifyCertificate.keyReference*,   
          *verifyCertificate*   
      ) gesetzt. Gemäß 9.2.3.2 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
3. (N095.810) K\_COS   
   Falls *affectedObject* ein Attribut *expirationDate* besitzt und *affectedObject*.CHAT.*flagList* nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. b0 b1 ist ungleich 112) und *affectedObject*.*expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
4. (N095.820) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
5. (N095.900) K\_COS   
   Extraktion von Schlüsselinformationen:

* 1. Option\_RSA\_CVC, Wenn *affectedObject.publicKey* vom Typ *publicRsaKey* ist, dann gilt:
     1. *certificate* = ´5F37–L5F37–*signature*   ||   5F38–L5F38–*certificateContent*´.
     2. ( *out*, *M* ) = RSA\_ISO9796\_2\_DS1\_VERIFY(   
          *affectedObject.publicRsaKey*,   
          *signature*,   
          *certificateContent*   
        )   
        Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
     3. Die Schlüsselattribute werden gemäß 7.1.2 aus der Nachricht *M* extrahiert und ein öffentliches Schlüsselobjekt *pukObj* gebildet.
     4. Falls CPI gleich ´21´ ist, dann MUSS *pukObj* ein öffentliches Signaturprüfobjekt sein (siehe 8.6.4.1), dessen Attribute mit den Definitionen aus 7.1.2.1 wie folgt zu setzen sind:
        1. *pukObj.oid* = OID.
        2. *pukObj.publicKey.n* = Modulus.
        3. *pukObj.publicKey.e* = öffentlicherExponent.
        4. *pukObj.keyIdentifier* = CHR.
        5. *pukObj.lifeCycleStatus* = „Operational state (active)“
        6. *pukObj*.*accessRules* = *affectedObject*. *accessRulesPublicSignatureVerificationObject*.
        7. *pukObj.accessRulesPublicSignatureVerificationObject* = *affectedObject. accessRulesPublicSignatureVerificationObject*.
        8. *pukObj. accessRulesPublicAuthenticationObject* = *affectedObject. accessRulesPublicAuthenticationObject*.
     5. Falls CPI gleich ´22´ ist, dann MUSS *pukObj* ein öffentliches Authentisierungsobjekt sein (siehe 8.6.4.2), dessen Attribute mit den Definitionen aus 7.1.2.2 wie folgt zu setzen sind:
        1. *pukObj.CHA* = CHA.
        2. *pukObj.oid* = OID.
        3. *pukObj.publicKey.n* = Modulus.
        4. *pukObj.publicKey.e* = öffentlicherExponent.
        5. *pukObj.keyIdentifier* = CHR.
        6. *pukObj.lifeCycleStatus* = „Operational state (active)“
        7. *pukObj*.*accessRules* = *affectedObject*. *accessRulesPublicAuthenticationObject*.
  2. Wenn *affectedObject.publicKey* vom Typ *publicElcKey* ist, dann gilt:
     1. CPI:
        1. Das COS MUSS den Wert CPI = ´70´ = 112 unterstützen.
        2. Das COS KANN weitere Werte für CPI
           1. unterstützen oder
           2. ablehnen.
     2. *certificate* = ´7F4E–L7F4E– *certificateContent*   ||   5F37–L5F37– *signature* ´.
     3. Falls *affectedObject*.*publicElcKey*.*domainParameter*.*L* gleich
        1. 32 ist, dann gilt: *hash* = SHA\_256(´7F4E–L7F4E– *certificateContent*´).
        2. 48 ist, dann gilt: *hash* = SHA\_384(´7F4E–L7F4E– *certificateContent*´).
        3. 64 ist, dann gilt: *hash* = SHA\_512(´7F4E–L7F4E– *certificateContent*´).
     4. *signature* wird wie folgt in *R* und *S* aufgeteilt:   
        *signature* =   *R*   ||   *S*, mit OctetLength( *R* ) = OctetLength( *S* ).
     5. *out* = ELC\_VER\_SIG(   
          *affectedObject.publicElcKey*,   
          *R*,   
          *S*,   
          *hash*   
        )   
        Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
     6. Falls die OID im CHAT aus *certificateContent* verschieden ist von der aus *affectedObject*.*publickey*.*accessRights*, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
     7. Das COS MUSS das Wertfeld von CED als vorzeichenlose ganze Zahl interpretieren, gemäß *ced* = OS2I( Wertfeld\_von\_CED ).
     8. Das COS MUSS das Wertfeld von CXD als vorzeichenlose ganze Zahl interpretieren, gemäß *cxd* = OS2I( Wertfeld\_von\_CXD ).
     9. Falls CED aus *certificateContent* größer ist als CXD aus *certificateContent*, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
     10. Das COS MUSS den Wert *effectiveFlagList* wie folgt bilden:   
             *effectiveFlagList*   = *affectedObject*.CHAT.*flagList* AND   
           (*flagList* aus *certificateContent)*
     11. Falls *effectiveFlagList*
         1. die Rolle Root-CA-Schlüssel anzeigt (d.h. b0 b1 ist gleich 112), dann DARF ein Import NICHT daran scheitern, CXD aus *certificateContent* größer ist als *affectedObject*.*expirationDate*.
         2. nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. b0 b1 ist ungleich 112) und CXD aus *certificateContent* größer ist als *affectedObject*.*expirationDate*, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
     12. Falls *effectiveFlagList*
         1. die Rolle Root-CA-Schlüssel anzeigt (d.h. b0 b1 ist gleich 112), dann DARF ein Import NICHT daran scheitern, dass CXD aus *certificateContent* kleiner als *pointInTime* ist (siehe (N019.900)j).
         2. nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. b0 b1 ist ungleich 112) und CXD aus *certificateContent* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
     13. Falls CED aus *certificateContent* größer als *pointInTime* ist (siehe (N019.900)j), dann MUSS *pointInTime* mit Transaktionsschutz auf den Wert CED gesetzt werden.
     14. Die Schlüsselattribute werden gemäß [gemSpec\_PKI] aus *certificateContent* extrahiert und ein öffentliches Schlüsselobjekt *pukObj* gebildet.
     15. Falls das höchstwertige Bit von *effectiveFlagList*)
         1. gesetzt ist, dann MUSS *pukObj* ein öffentliches Signaturprüfobjekt sein (siehe 8.6.4.1).
         2. nicht gesetzt ist, dann MUSS *pukObj* ein öffentliches Authentisierungsobjekt sein (siehe 8.6.4.2).
     16. Die Attribute von *pukObj* sind mit den Definitionen aus [gemSpec\_PKI] wie folgt zu setzen:
         1. Domainparameter: Falls das Wertfeld von DO´86´ eine Länge von
            1. ´41´= 65 hat, gilt *PuK.domainParameter* = brainpoolP256r1.
            2. ´61´= 97 hat, gilt *PuK.domainParameter* = brainpoolP384r1.
            3. ´81´= 129 hat, gilt *PuK.domainParameter* = brainpoolP512r1.
         2. *PuK*.*P* = OS2P(   
              Wertfeld von DO´86´,   
              *PuK.domainParameter*   
              ).
         3. *pukObj.keyIdentifier* = CHR gemäß [gemSpec\_PKI#6.7.2.4].
         4. *pukObj.lifeCycleStatus* = „Operational state (active)“
         5. *pukObj.publicElcKey* = *PuK*.
         6. *pukObj.oid* = OIDPuK.   
            *Hinweis: Anhand von* pukObj.oid *ist erkennbar, ob pukObj ein Signaturprüfobjekt oder ein Authentisierungsobjekt ist.*
         7. Falls *pukObj* ein öffentliches Signaturprüfobjekt ist, dann gilt:
            1. *pukObj*.*accessRules* = *affectedObject*.   
                 *accessRulesPublicSignatureVerificationObject*.
            2. *pukObj.accessRulesPublicSignatureVerificationObject* = *affectedObject. accessRulesPublicSignatureVerificationObject*.
            3. *pukObj. accessRulesPublicAuthenticationObject* = *affectedObject. accessRulesPublicAuthenticationObject*.
         8. Falls *pukObj* ein öffentliches Authentisierungsobjekt ist, dann gilt:   
             *pukObj*.*accessRules* = *affectedObject*.   
              *accessRulesPublicAuthenticationObject*.
         9. *pukObj*.CHAT.OIDflags = *affectedObject*.CHAT.OIDflags.
         10. *pukObj*.CHAT.*flagList* = *effectiveFlagList*
         11. *pukObj*.*expirationDate* = Wertfeld des Datenobjektes CXD.
  3. Falls *affectedObject.keyIdentifier* ungleich CAR aus dem imporiterten Zertifikat ist, genau dann MUSS das Kommando mit dem Trailer InconsistentKeyReference terminieren.
  4. Das Objekt *pukObj* wird wie folgt gespeichert:
     1. *persistent* = StoreInCache(„Ordner dem *affectedObject* zugeordnet ist“, *pukObj*)

1. (N096.000) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
2. (N096.100) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
3. (N096.200) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
4. (N096.300) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 213 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 213 MUSS eine höhere Priorität als UpdateRetryWarning haben.
   3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### PSO Verify Cryptographic Checksum

Das Kommando PSO Verify Cryptographic Checksum überprüft mittels eines symmetrischen Schlüssels, ob eine gegebene kryptographische Checksumme zu gegebenen Daten passt. Der symmetrische Schlüssel wird im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, 15.4.3, 15.4.4) ausgehandelt. Checksumme und geschützte Daten sind als Parameter in der Kommandonachricht enthalten.

#### Use Case Prüfung einer kryptographischen Checksumme

Diese Variante gilt für Algorithmen aus der folgenden Menge: {aesSessionkey, desSessionkey (Option\_DES)}.

In dieser Variante enthält die APDU des PSO Verify Cryptographic Checksum-Kommandos zwei Parameter:

1. (N096.340) K\_externeWelt {K\_Karte}   
   Der Parameter *data* enthält die geschützten Daten. Der Parameter *data* ist ein Oktettstring mit beliebigem Inhalt.
2. (N096.342) K\_externeWelt {K\_Karte}   
   Der Parameter *mac* enthält eine kryptographische Checksumme. Der Parameter *mac* ist ein Oktettstring mit beliebigem Inhalt, dessen Länge acht Oktette betragen MUSS.
3. Falls mac eine andere Länge als acht Oktett besitzt, dann schlägt gemäß (N002.700)e und (N002.810)h der Vergleich in (N002.900)b bzw. (N003.010)b oder (N003.020)b stets fehl.
4. (N096.344) K\_externeWelt {K\_Karte}   
   Die Parameter *data* und *mac* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N096.366)a spezifiziert.
5. (N096.346) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 214 verwendet werden.

Tabelle 214: PSO Verify Cryptographic Checksum, Prüfen MAC

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Beschreibung der Antwortdaten, hier keine Antwortdaten |
| P2 | ´A2´ | Beschreibung der Kommandodaten, hier Input Template für MAC |
| Data | ´XX…XX´ | *inputTemplate*, DER codiertes Datenfeld |

#### Antwort der Karte auf Berechnen einer kryptographischen Checksumme

Tabelle 215: PSO Verify Cryptographic Checksum Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | erfolgreiche Verifizierung eines MAC |

Tabelle 216: PSO Verify Cryptographic Checksum Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | NoKeyReference | kein Schlüssel für MAC-Berechnung ausgewählt |
| ´6A 80´ | VerificationError | MAC-Prüfung fehlgeschlagen |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 88´ | KeyNotFound | kein Schlüssel für MAC-Berechnung vorhanden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N096.350) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N096.360) K\_COS, Option\_Kryptobox, Option\_PACE\_PCD
   1. Das COS MUSS die PSO Verify Cryptographic Checksum-Variante aus 14.8.8.1 unterstützen.
   2. Das COS KANN weitere PSO Verify Cryptographic Checksum-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N096.362) K\_COS   
   Wenn *channelContext.keyReferenceList.macCalculation*
   1. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
   2. nicht leer ist, dann wird *affectedObject* = SearchKey(   
       *currentFolder,   
          keyReferenceList.macCalculation.keyReference,   
          keyReferenceList.macCalculation.algID*) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
      1. KeyNotFound terminieren.
      2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
3. (N096.364) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N096.366) K\_COS   
   Es gilt:
   1. *inputTemplate* = ´80–L80–*data*   ||   8E–L8E–*mac*´
5. (N096.368) K\_COS   
   Mit den Attributen *SSCmac* und *Kmac* aus dem Attribut *SessionkeyContext* gilt: Falls *keyReferenceList.macCalculation.algID* den Wert
   1. aesSessionkey besitzt, gilt:
      1. *SSCmac* = *SSCmac* + 1.
      2. *result* = VerifyCMAC\_IsoPadding(*Kmac*, *mac*, I2OS(*SSCmac*, 16) || *data* ).
   2. Option\_DES, desSessionkey besitzt, gilt:
      1. *SSCmac* = *SSCmac* + 1.
      2. *result* = VERIFY\_Retail\_MAC(*Kmac*, *mac*, I2OS(*SSCmac*, 8) || *data* ).
6. (N096.370) K\_COS   
   Falls *result* den Wert
   1. INVALID besitzt, dann
      1. MUSS als Trailer VerificationError verwendet werden und
      2. die Sessionkeys MÜSSEN mittels clearSessionkeys( ) gelöscht werden.
   2. VALID besitzt, genau dann MUSS als Trailer NoError verwendet werden.
7. (N096.372) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 216 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 216 MUSS eine höhere Priorität als NoError haben.

### PSO Verify Digital Signature

Das Kommando PSO Verify Digital Signature überprüft eine Signatur mittels eines öffentlichen Schlüssels. Sowohl die Signatur, als auch der öffentliche Schlüssel und der zur Prüfung zu verwendende Algorithmus sind als Parameter in der Kommanodnachricht enthalten.

#### Use Case Prüfen einer ELC-Signatur

Diese Variante prüft Signaturen, die mittels signECDSA erstellt wurden. Als Domainparameter sind alle Kurven zulässig, die vom COS unterstützt werden (siehe (N002.500)).

In dieser Variante enthält die APDU des PSO Verify Digital Signature vier Parameter:

1. (N096.380) K\_externeWelt {K\_Karte}   
   Der Parameter *oid* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve referenziert. Der Parameter *oid* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.
2. (N096.381) K\_externeWelt {K\_Karte}   
   Der Parameter *POB* enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Signierenden. Der Parameter *POB* ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.500)a).
3. (N096.382) K\_externeWelt {K\_Karte}   
   Der Parameter *hash* enthält den im Rahmen der Signaturerstellung erzeugten Hashwert. Der Parameter *hash* ist ein Oktettstring mit beliebigem Inhalt. Für den Zusammenhang zwischen *oid* und der Länge von *hash* MUSS gelten:
   1. ansix9p256r1 =>   OctetLength(*hash*) gleich 32.
   2. ansix9p384r1 =>   OctetLength(*hash*) gleich 48.
   3. brainpoolP256r1 =>   OctetLength(*hash*) gleich 32.
   4. brainpoolP384r1 =>   OctetLength(*hash*) gleich 48.
   5. brainpoolP512r1 =>   OctetLength(*hash*) gleich 64.
4. (N096.383) K\_externeWelt {K\_Karte}   
   Der Parameter *signature* enthält die zu prüfende Signatur. Der Parameter *signature* ist ein Oktettstring mit beliebigem Inhalt. Für den Zusammenhang zwischen *oid* und der Länge von *signature* MUSS gelten:
   1. ansix9p256r1 =>   OctetLength(*signature*) gleich 64.
   2. ansix9p384r1 =>   OctetLength(*signature*) gleich 96.
   3. brainpoolP256r1 =>   OctetLength(*signature*) gleich 64.
   4. brainpoolP384r1 =>   OctetLength(*signature*) gleich 96.
   5. brainpoolP512r1 =>   OctetLength(*signature*) gleich 128.
5. (N096.386) K\_externeWelt {K\_Karte}   
   Die Parameter *oid*, *POB*, *hash* und *signature* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N096.394) spezifiziert.
6. (N096.388) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 217 verwendet werden.

Tabelle 217: PSO Verify Digital Signature mittels übergebenem ELC-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´2A´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | Keine Antwortdaten |
| P2 | ´A8´ | Beschreibung der Kommandodaten, Template einer digitalen Signatur |
| Data | ´XX…XX´ | *signatureTemplate*, DER codiertes Datenfeld |

#### Antwort der Karte auf Prüfen einer digitalen Signatur

Tabelle 218: PSO Verify Digital Signature Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | erfolgreiche Verifizierung einer Signatur |

Tabelle 219: PSO Verify Digital Signature Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´6A 80´ | VerificationError | Signaturprüfung fehlgeschlagen |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N096.390) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N096.392) K\_COS
   1. Das COS MUSS die PSO Verify Digital Signature-Variante aus 14.8.9.1 unterstützen.
   2. Das COS KANN weitere PSO Verify Digital Signature-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N096.394) K\_COS   
   Mit den Festlegungen aus [ISO/IEC 7816-8#Table 6] MUSS gelten:
   1. *signatureTemplate* = ´06‑L06‑*oid* || 90‑L90‑*hash* || 9C–L9C–*publicKey* ||   
        9E–L9E‑*signature*´
   2. *publicKey* = ´7F49‑L7F49‑( 86–L86–*POB* )´
   3. *signature* wird wie folgt auf *R* und *S* aufgeteilt:
      1. *signature* = *R*   ||   *S*.
      2. OctetLength( *R* ) = OctetLength( *S* ).
   4. Die *oid* wird gemäß Tabelle 271 in Domainparameter übersetzt, die im Folgenden mit *dP* bezeichnet werden.
3. (N096.396) K\_COS   
   Falls ELC\_VER\_SIG( OS2P( *POB*, *dP*), *R*, *S*, *hash* ) den Wert
   1. False besitzt, genau dann MUSS als Trailer VerificationError verwendet werden.
   2. True besitzt, genau dann MUSS als Trailer NoError verwendet werden.
4. (N096.398) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 219 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 219 MUSS eine höhere Priorität als NoError haben.

## Verschiedenes

### Envelope

1. (N096.400) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]
   1. unterstützen oder
   2. ablehnen.

### Fingerprint

Das Kommando Fingerprint dient der Überprüfung der Integrität und Authentizität des COS. Dazu wird im Kommando ein Präfix übergeben. Über das Präfix und das COS wird ein Fingerprint berechnet. Stimmen berechneter Fingerprint und der Fingerprint eines authentischen COS überein, so ist die Authentizität des COS nachgewiesen.

1. Typischerweise werden für Tests im Rahmen einer funktionalen Zulassung und im Rahmen einer Sicherheitsevaluierung verschiedene Images erzeugt. Zudem werden typischerweise im Rahmen einer Fehlerbeseitigung Patches entwickelt. Mit Hilfe dieses Kommandos lässt sich der Nachweis führen, dass in sämtlichen Images, die für ein Zulassungsverfahren relevant sind, derselbe Softwarestand (im Allgemeinen ROM Code plus Patches) zu Grunde liegt.

#### Use Case Fingerprint über das COS berechnen.

In dieser Variante enthält die APDU des Fingerprint-Kommandos zwei Parameter:

1. (N096.450) K\_externeWelt {K\_Karte}   
   Der Parameter *prefix* enthält den Präfix. Der Parameter *prefix* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *prefix* beträgt 128 Oktett.
2. (N096.452) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* beträgt WildCardShort.
3. (N096.454) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 220 verwendet werden.

Tabelle 220: Fingerprint über das COS

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´FA´ | Instruction Byte |
| P1 | ´00´ | ‑ |
| P2 | ´00´ | – |
| Data | ´XX…XX´ | *prefix* |
| Le | ´00´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Fingerprintberechnung

Tabelle 221: Fingerprint Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *fingerprint* | Fingerprint des COS |
| ´90 00´ | NoError | Erfolgreiche Fingerprintberechnung |

Tabelle 222: Fingerprint Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N096.469) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N096.470) K\_COS
   1. Das COS MUSS die Fingerprint-Variante aus 14.9.2.1 unterstützen.
   2. Das COS KANN weitere Fingerprint-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N096.472) K\_COS   
   Als *affectedObject* MUSS *currentFolder* verwendet werden.
3. (N096.474) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N096.476) K\_COS   
   Repräsentant *M* des COS:
   1. Das COS MUSS einen Oktettstring *M* konstruieren, der sämtliche Bestandteile des COS beinhaltet.
   2. Die Funktion, die das COS auf *M* abbildet ist herstellerspezifisch und
      1. MUSS zeitinvariant sein,
      2. MUSS reproduzierbar sein und
      3. DARF NICHT abhängig sein von Veränderungen am Objektsystem, die mit Kommandos möglich sind, die im Rahmen dieser Spezifikation möglich und gemäß Objektsystemspezifikation erlaubt sind.
5. (N096.478) K\_COS   
   Basierend auf dem Parameter *prefix* aus den Kommandodaten und *M* MUSS das Datenfeld *fingerprint* der Antwortnachricht mit einem der folgenden Verfahren berechnet werden:
   1. *fingerprint* = SHA\_256( *prefix* || *M* ) gesetzt werden.
   2. *fingerprint* = SHA\_384( *prefix* || *M* ) gesetzt werden.
   3. *fingerprint* = SHA\_512( *prefix* || *M* ) gesetzt werden.
   4. *fingerprint* = CalculateCMAC\_IsoPadding( *key*, *prefix* || *M* ) gesetzt werden,   
      wobei *key* ein herstellerspezifischer Schlüssel ist.
6. (N096.480) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
7. (N096.482) K\_COS   
   Das Datenfeld der Antwortnachricht MUSS *fingerprint* sein.

### Generate Asymmetric Key Pair

Das Kommando Generate Asymmetric Key Pair (GAKP) dient dem Erzeugen von asymmetrischen Schlüsselpaaren und dem Auslesen eines dabei erzeugten öffentlichen Schlüssels. Das betroffene Schlüsselobjekt wird zuvor ausgewählt. Dies geschieht durch ein Manage Security Environment-Kommando (siehe 14.9.9.9). Zusätzlich ist es möglich, den zu generierenden Schlüssel im Kommando zu referenzieren.

#### Use Case Generierung, ohne Überschreiben, ohne Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein Manage Security Environment-Kommando ausgewählt und mit Schlüsseldaten befüllt, falls diese fehlen. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier einen Parameter:

1. (N096.500) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´84´ gewählt werden.
2. (N096.600) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 223 verwendet werden.

Tabelle 223: GAKP, ohne Überschreiben, ohne Referenz, ohne Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´84´ | *operationMode* = Schlüsselgenerierung, falls kein Schlüssel vorhanden |
| P2 | ´00´ | betroffenes Objekt via *channelContext*.*keyReferenceList* |

#### Use Case Generierung, ohne Überschreiben, mit Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt und mit Schlüsseldaten befüllt, falls diese fehlen. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

1. (N096.640) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´84´ gewählt werden.
2. (N096.642) K\_externeWelt {K\_Karte}   
   Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.
3. (N096.644) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 224 verwendet werden.

Tabelle 224: GAKP, ohne Überschreiben, mit Schlüsselreferenz, ohne Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´84´ | *operationMode* = Schlüsselgenerierung, falls kein Schlüssel vorhanden |
| P2 | ´XX´ | *keyReference* bestimmt betroffenes Schlüsselobjekt |

#### Use Case Generierung, ggf. Überschreiben, ohne Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein Manage Security Environment-Kommando ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Die APDU des GAKP-Kommandos enthält hier einen Parameter:

1. (N096.650) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´C4´ gewählt werden.
2. (N096.652) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 225 verwendet werden.

Tabelle 225: GAKP, ggf. Überschreiben, ohne Referenz, ohne Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´C4´ | *operationMode* = Schlüsselgenerierung, ggf. Überschreiben |
| P2 | ´00´ | betroffenes Objekt via *channelContext*.*keyReferenceList* |

#### Use Case Generierung, ggf. Überschreiben, mit Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

1. (N096.660) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´C4´ gewählt werden.
2. (N096.662) K\_externeWelt {K\_Karte}   
   Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.
3. (N096.664) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 226 verwendet werden.

Tabelle 226: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, ohne Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´C4´ | *operationMode* = Schlüsselgenerierung, ggf. Überschreiben |
| P2 | ´XX´ | *keyReference* bestimmt betroffenes Schlüsselobjekt |

#### Use Case Auslesen vorhandener Schlüssel, ohne Referenz

In dieser Variante wird das betroffene Schlüsselobjekt durch ein Manage Security Environment-Kommando ausgewählt. Die dort vorhandenen Schlüsseldaten werden ausgelesen. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

1. (N096.700) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N096.800) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* in der Antwortnachricht enthalten ist.
3. (N096.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 227 verwendet werden.

Tabelle 227: GAKP, Auslesen vorhandener Schlüssel ohne Schlüsselreferenz

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Auslesen eines öffentlichen Schlüssels |
| P2 | ´00´ | betroffenes Objekt via *channelContext*.*keyReferenceList* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Auslesen vorhandener Schlüssel, mit Referenz

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt. Die dort vorhandenen Schlüsseldaten werden ausgelesen. Die APDU des GAKP-Kommandos enthält hier drei Parameter:

1. (N096.940) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N096.942) K\_externeWelt {K\_Karte}   
   Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.
3. (N096.944) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* in der Antwortnachricht enthalten ist.
4. (N096.946) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando–APDU MÜSSEN die Angaben aus Tabelle 228 verwendet werden.

Tabelle 228: GAKP, Auslesen vorhandener Schlüssel mit Schlüsselreferenz

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Auslesen eines öffentlichen Schlüssels |
| P2 | ´XX´ | *keyReference* bestimmt betroffenes Schlüsselobjekt |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Generierung, ohne Überschreiben, ohne Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein Manage Security Environment-Kommando ausgewählt, mit Schlüsseldaten befüllt und die der erzeugte öffentliche Schlüssel wird exportiert. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

1. (N097.000) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´80´ gewählt werden.
2. (N097.100) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* in der Antwortnachricht enthalten ist.
3. (N097.200) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 229 verwendet werden.

Tabelle 229: GAKP, ohne Überschreiben, ohne Schlüsselreferenz, mit Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´80´ | *operationMode* = Schlüsselgenerierung falls nicht vorhanden, Ausgabe |
| P2 | ´00´ | betroffenes Objekt via *channelContext*.*keyReferenceList* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Generierung, ohne Überschreiben, mit Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt, mit Schlüsseldaten befüllt und der öffentliche Teil des erzeugten Schlüsselpaares wird exportiert. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier drei Parameter:

In dieser Variante enthält die APDU des GAKP-Kommandos drei Parameter:

1. (N097.240) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´80´ gewählt werden.
2. (N097.242) K\_externeWelt {K\_Karte}   
   Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.
3. (N097.244) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* in der Antwortnachricht enthalten ist.
4. (N097.246) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 230 verwendet werden.

Tabelle 230: GAKP, ohne Überschreiben, mit Schlüsselreferenz, mit Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´80´ | *operationMode* = Schlüsselgenerierung falls nicht vorhanden, Ausgabe |
| P2 | ´XX´ | *keyReference* bestimmt betroffenes Schlüsselobjekt |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Generierung, ggf. Überschreiben, ohne Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein Manage Security Environment-Kommando ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Der erzeugte öffentliche Schlüssel wird exportiert. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

1. (N097.250) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´C0´ gewählt werden.
2. (N097.252) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* in der Antwortnachricht enthalten ist.
3. (N097.254) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 231 verwendet werden.

Tabelle 231: GAKP, ggf. Überschreiben, ohne Schlüsselreferenz, mit Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´C0´ | *operationMode* = Schlüsselgenerierung, ggf. Überschreiben, Ausgabe |
| P2 | ´00´ | betroffenes Objekt via *channelContext*.*keyReferenceList* |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Generierung, ggf. Überschreiben, mit Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Der erzeugte öffentliche Schlüssel wird exportiert. Die APDU des GAKP-Kommandos enthält hier drei Parameter:

1. (N097.260) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´C0´ gewählt werden.
2. (N097.262) K\_externeWelt {K\_Karte}   
   Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.
3. (N097.264) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* in der Antwortnachricht enthalten ist.
4. (N097.266) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 232 verwendet werden.

Tabelle 232: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, mit Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´C0´ | *operationMode* = Schlüsselgenerierung, ggf. Überschreiben, Ausgabe |
| P2 | ´XX´ | *keyReference* bestimmt betroffenes Schlüsselobjekt |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Zusammenfassung der Generate Asymmetric Key Pair-Kommando-Varianten

Wegen der Vielzahl an Varianten für dieses Kommando werden hier alle auf einen Blick dargestellt. Es sei darauf hingewiesen, dass nicht alle Kombinationen der folgenden Tabelle in den vorangegangenen Kapiteln enthalten sind. Deshalb sind solche nicht zwingend zu unterstützen.

Tabelle 233: Generate Asymmetric Key Pair, Kommandoparameter im Überblick

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´46´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´  ´80´  ´84´  ´C0´  ´C4´ | keine Schlüsselgenerierung, nur Ausgabe  Schlüsselgenerierung, falls kein Schlüssel vorhanden, Ausgabe  Schlüsselgenerierung, falls kein Schlüssel vorhanden, keine Ausgabe  Schlüsselgenerierung, ggf. Überschreiben, Ausgabe  Schlüsselgenerierung, ggf. Überschreiben, keine Ausgabe |
| P2 | ´00´  sonst | betroffenes Objekt via *channelContext*.*keyReferenceList*  *keyReference* bestimmt betroffenes Schlüsselobjekt |
| Le | *length* | Bit b3 von P1 = 0: Anzahl der erwarteten Oktette in den Antwortdaten  Bit b3 von P1 = 1: abwesend |

#### Antwort der Karte auf Schlüsselgenerierung

Tabelle 234: GAKP Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *publicKeyDO* | Abwesend, oder öffentlicher Schlüssel |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´63 Cx´ | UpdateRetryWarning | Wie NoError, aber Schreibschwierigkeiten |
| ´90 00´ | NoError | Erfolgreiche Operation |

Tabelle 235: GAKP Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´64 00´ | KeyInvalid | Auszulesende Schlüsseldaten fehlen |
| ´65 81´ | MemoryFailure | Schreibvorgang nicht erfolgreich |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 85´ | KeyAlreadyPresent | Schlüsseldaten bereits gesetzt, Generierung unmöglich |
| ´6A 88´ | KeyNotFound | Referenziertes Schlüsselobjekt wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N097.300) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N097.400) K\_COS
   1. Das COS MUSS die GAKP-Varianten aus 14.9.3.1, 14.9.3.2, 14.9.3.3, 14.9.3.4, 14.9.3.5, 14.9.3.6, 14.9.3.7, 14.9.3.8 14.9.3.9 und 14.9.3.10 unterstützen.
   2. Das COS KANN weitere GAKP-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N097.500) K\_COS   
   Das vom Kommando betroffene Schlüsselobjekt wird wie folgt bestimmt:
   1. Falls der Parameter P2 gleich ´00´ ist, dann gilt:   
      Falls das Attribute *channelContext*.*keyReferenceList.signatureCreation*
      1. leer ist, dann MUSS das COS die Bearbeitung dieses Kommandos mit einem beliebigen Trailer beantworten. In diesem Fall ist es zulässig, dass das COS mit einem beliebigen anderen privaten Schlüsselobjekt arbeitet.
      2. nicht leer ist, gilt:   
         *keyReference* = *keyReferenceList.signatureCreation.keyReference*
   2. Falls der Parameter P2 ungleich ´00´ ist, MUSS *keyReference* = P2 gelten.
   3. Es wird *affectedObject* = SearchSecretKey(   
          *currentFolder,*   
          *keyReference*,   
          Wildcard   
      ) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren. Der Fehler notSupported ist wegen der Wildcard-Suche nicht möglich.
3. (N097.600) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N097.650) K\_COS   
   Wenn *operationMode* Element der Menge {´80´, ´84´} ist und das Attribute *keyAvailable* den Wert True hat, genau dann MUSS das Kommando mit dem Trailer KeyAlreadyPresent terminieren.
5. (N097.700) K\_COS   
   Wenn *operationMode* Element der Menge {´80´, ´84´, ´C0´, ´C4´} ist, dann MUSS
   1. ein Schlüsselpaar ( *PrK*, *PuK* ) erzeugt werden, dessen Eigenschaften zu den Attributen von *affectedObject* und zu (N002.100) passen.
   2. Anschließend MUSS das Attribute *keyAvailable* auf den Wert True geändert werden. Die Änderung von *keyAvailable* MUSS mit Transaktionsschutz ausgeführt werden.
6. (N097.800) K\_COS   
   Wenn *operationMode* Element der Menge {´81´} ist und das Attribute *keyAvailable* den Wert False hat, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.
7. (N097.900) K\_COS   
   Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
8. (N098.000) K\_COS   
   Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
   1. entweder als Trailer MemoryFailure verwendet werden,
   2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
9. (N098.100) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
10. (N098.200) K\_COS   
    Das DER-TLV codierte Datenobjekt *publicKeyDO* MUSS wie folgt berechnet werden: Falls *PuK* ein
    1. RSA-Schlüssel ist, dann gilt:
       1. Setze L82 größer gleich OctetLength( *PuK.e* ), aber kleiner als 128.
       2. Setze       *n* = I2OS( *PuK.n*, OctetLength( *PuK.n* ) ).
       3. Setze       *e* = I2OS( *PuK.e*, L82 ).
       4. *publicKeyDO* = ´7F49–L7F49–( 81–L81–*n*   ||   82–L82‑*e* )´.
    2. ELC-Schlüssel ist mit dem öffentlichen Punkt *P* und den Domainparametern *dP*, dann gilt   
       *publicKeyDO* = ´7F49–L7F49–( 86–L86–P2OS( *P* , *dP*.*L*) )´.
11. (N098.300) K\_COS   
    Für das Datenfeld der Antwortnachricht MUSS gelten:
    1. Wenn *operationMode* einen Wert aus der Menge {´80´, ´81´, ´C0´} hat, dann enthält das Datenfeld der Antwortnachricht *publicKeyDO*.
    2. Andernfalls fehlt das Datenfeld der Antwortnachricht.
12. (N098.400) K\_COS   
    Für die Priorität der Trailer gilt:
    1. Die Priorität der Trailer in Tabelle 235 ist herstellerspezifisch.
    2. Jeder Trailer in Tabelle 235 MUSS eine höhere Priorität als UpdateRetryWarning haben.
    3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

### Get Challenge

Das Kommando Get Challenge erzeugt eine Zufallszahl. Diese steht kartenintern mindestens bei der Ausführung des nächsten Kommandos zur Verfügung. Typischerweise beinhaltet dieses nächste Kommando die Authentisierung einer externen Komponente.

#### Use Case Zufallszahl für DES oder RSA Authentisierung

In dieser Variante enthält die APDU des Get Challenge-Kommandos einen Parameter:

1. (N098.500) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich 8 sein.
2. (N098.600) K\_externeWelt {K\_Karte}, Option\_DES, Option\_RSA\_CVC   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 236 verwendet werden.

Tabelle 236: Get Challenge für DES oder RSA Authentisierung

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´84´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | – |
| P2 | ´00´ | – |
| Le | ´08´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten, hier acht |

#### Use Case Zufallszahl für AES oder ELC Authentisierung

In dieser Variante enthält die APDU des Get Challenge-Kommandos einen Parameter:

1. (N098.620) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich 16 sein.
2. (N098.625) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 237 verwendet werden.

Tabelle 237: Get Challenge für AES oder ELC Authentisierung

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´84´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | – |
| P2 | ´00´ | – |
| Le | ´10´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten, hier 16 |

#### Antwort der Karte auf Erzeugen einer Zufallszahl

Tabelle 238: Get Challenge Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *rspData* | Zufallszahl |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Erzeugung einer Zufallszahl |

Tabelle 239: Get Challenge Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| – | – | Derzeit keine Fehlerfälle spezifiziert |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N098.700) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N098.800) K\_COS
   1. Das COS MUSS die Get Challenge-Variante aus 14.9.4.1 und 14.9.4.2 unterstützen.
   2. Das COS KANN weitere Get Challenge-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N098.900) K\_COS   
   Zugriffsregeln für das Get Challenge-Kommando:
   1. Das COS KANN die Auswertung von Zugriffsregeln für das Get Challenge-Kommando unterstützen.
   2. Das COS KANN als Zugriffsbedingung für das Get Challenge-Kommando stets ALWAYS verwenden.
3. (N099.000) K\_COS   
   Es MUSS *rspData* = RAND( Ne ) gesetzt werden.
4. (N099.100) K\_COS   
   Als Trailer MUSS NoError gewählt werden.
5. (N099.200) K\_COS   
   Das Datenfeld der Antwortnachricht MUSS *rspData* sein.
6. (N099.300) K\_COS   
   Das Datenfeld der Antwortnachricht *rspData* MUSS zwecks Verwendung in nachfolgenden Kommandos in *RND.ICC* (siehe (N029.900)b) gespeichert werden.

### Get Random

Das Kommando Get Random erzeugt eine Zufallszahl. Im Unterschied zu Get Challenge steht diese Zufallszahl nach Abschluss des Kommandos kartenintern nicht für weitere Aktionen zur Verfügung. Dafür erfüllt die mittels Get Random erzeugte Zufallszahl gewisse Sicherheitsanforderungen (siehe (N099.356)b).

#### Use Case Erzeugen kryptographisch sicherer Zufallszahl

In dieser Variante enthält die APDU des Get Random-Kommandos einen Parameter:

1. (N099.320) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {1, 2, …, 255, WildCardShort} gewählt werden.
2. (N099.322) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2S Kommando-APDU MÜSSEN die Angaben aus Tabelle 240 verwendet werden.

Tabelle 240: Get Random

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´84´ | Instruction Byte gemäß [ISO/IEC 7816-4] (identisch zu Get Challenge) |
| P1 | ´00´ | – |
| P2 | ´00´ | – |
| Le | *length* | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte Erzeugen kryptographisch sichere Zufallszahl

Tabelle 241: Get Random Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *random* | Zufallszahl |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | erfolgreiche Erzeugung einer Zufallszahl |

Tabelle 242: Get Random Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N099.340) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N099.344) K\_COS, Option\_logische\_Kanäle
   1. Das COS MUSS die Get Random-Variante aus 14.9.5.1 unterstützen.
   2. Das COS KANN weitere Get Random-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N099.348) K\_COS   
   Als *affectedObject* MUSS *currentFolder* verwendet werden.
3. (N099.352) K\_COS   
   Wenn AccessRuleEvaluation( *affectedObject*, *CLA*, *INS*, *P1*, *P2* ) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
4. (N099.356) K\_COS
   1. Es MUSS *random* = RAND( Ne ) gesetzt werden.
   2. Die Güte der Zufallszahl in *random* MUSS mindestens [BSI-TR-03116#3.4 PTG.2] entsprechen.
5. (N099.360) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
6. (N099.364) K\_COS   
   Das Datenfeld der Antwortnachricht MUSS *random* sein.

### Get Response

1. (N099.400) K\_COS   
   Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]
   1. unterstützen oder
   2. ablehnen.

### List Public Key

Das Kommando List Public Key liefert eine Liste der in einer Karte gespeicherten öffentlichen Schlüsselobjekte. Die Ausführung dieses Kommandos ist an keine Vorbedingung geknüpft. Insbesondere ist die Ausführung dieses Kommandos unabhängig vom konkreten Wert von *currentFolder*.

#### Use Case Auslesen der Liste öffentlicher Schlüsselobjekte

In dieser Variante enthält die Liste alle Arten von öffentlichen Schlüsselobjekten.

1. (N099.450) K\_externeWelt {K\_Karte}   
   Die APDU des List Public Key-Kommandos enthält zwei Parameter.
   1. Der Parameter *intendedAction* zeigt an, dass in die Liste alle Arten von öffentlichen Schlüsselobjekten aufzunehmen sind.
   2. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.
2. (N099.452) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2E-Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2E-Kommando-APDU MÜSSEN die Angaben aus Tabelle 243 verwendet werden.

Tabelle 243: List Public Key mit allen Arten öffentlicher Schlüsselobjekte

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´80´ | CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt |
| INS | ´CA´ | Instruction Byte von Get Data gemäß [ISO/IEC 7816-4] |
| P1 | ´0100´ | gemäß [ISO/IEC 7816-4] proprietärer Wert, hier: Alle Arten von öffentlichen Schlüsselobjekten |
| P2 |
| Le | ´0000´ | *length*, Anzahl der erwarteten Oktette in den Antwortdaten |

#### Antwort der Karte auf Auslesen einer Schlüsselliste

Tabelle 244: List Public Key Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| ´xx…xx´ | *keyReferenceList* | Liste mit Referenzen öffentlicher Schlüsselobjekte |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´62 00´ | DataTruncated | Antwortdaten unvollständig |
| ´90 00´ | NoError | Erfolgreiche Leseoperation |

Tabelle 245: List Public Key Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ‑ | ‑ | derzeit ist kein Fehlerfall spezifiziert |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N099.458) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N099.460) K\_COS
   1. Das COS MUSS die List Public Key-Variante aus 14.9.7.1 unterstützen.
   2. Das COS KANN weitere List Public Key-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N099.461) K\_COS   
   Zugriffsregeln für das List Public Key Kommando:
   1. Das COS KANN die Auswertung von Zugriffsregeln für das List Public Key Kommando unterstützen.
   2. Das COS KANN als Zugriffsbedingung für das List Public Key Kommando stets ALWAYS verwenden.
3. (N099.462) K\_COS   
   Das Datenfeld *keyReferenceList* der Antwortnachricht MUSS wie folgt erzeugt werden:
   1. Falls keine Referenz auf ein öffentliches Schlüsselobjekt zurückgemeldet wird, dann MUSS *keyReferenceList* = ´´ leer sein.
   2. Für jedes öffentliche Schlüsselobjekt in *persistentPublicKeyList* MUSS ein Eintrag in *keyReferenceList* erfolgen.
   3. Für ein öffentliches Schlüsselobjekt in *volatileCache* KANN ein Eintrag in *keyReferenceList*
      1. erfolgen oder
      2. unterbleiben.
   4. Jeder Eintrag in *keyReferenceList* MUSS ein DER-TLV-Datenobjekt *puKReference* sein.
   5. Alle *pukReference* Datenobjekte MÜSSEN konkateniert werden.
   6. Jedes Datenobjekt *pukReference* MUSS wie folgt aufgebaut sein:
      1. Als Tag MUSS ´E0´ verwendet werden.
      2. Das erste Datenobjekt im DO´E0´ MUSS ein Application Identifier DO´4F´ sein und die Applikation angeben, in dessen Unterstruktur das öffentliche Schlüsselobjekt gespeichert ist.
      3. Falls das öffentliche Schlüsselobjekt
         1. der im DO´4F´ referenzierten Applikation zugeordnet ist, dann DARF DO´E0´ KEIN DO´51´enthalten.
         2. einem Unterordner der im DO´4F´referenzierten Applikation zugeordnet ist, dann MUSS das DO´E0´ an zweiter Stelle ein File Reference DO´51´ enthalten.
            1. Das Wertfeld des DO´51´ MUSS einen Pfad von der Applikation zu dem DF enthalten, welches das öffentliche Schlüsselobjekt enthält.
            2. Die Länge des Werfeldes von DO´51´ MUSS gerade sein.
            3. Die ersten beiden Oktette in DO´51´ MÜSSEN den File Identifier eines DF enthalten, welches im Attribut *children* der Applikation enthalten ist.
      4. Als nächstes MUSS DO´E0´ ein CRT enthalten, dessen Tag gleich
         1. ´A4´ ist, falls es sich um ein öffentliches Authentisierungsobjekt handelt.
         2. ´B6´ ist, falls es sich um ein öffentliches Signaturprüfobjekt handelt.
         3. ´B8´ ist, falls es sich um ein öffentliches Verschlüsselungsobjekt handelt.
         4. Das CRT MUSS ein DO´83´ mit dem Attribut *keyIdentifier* des öffentlichen Schlüsselobjektes enthalten.
      5. Falls es für ein öffentliches Schlüsselobjekt mehr als eine Möglichkeit gibt *pukReference* zu codieren, dann MUSS das COS aus der Menge der möglichen Werte genau einen auswählen.
4. (N099.463) K\_COS   
   Das COS MUSS für *keyReferenceList* eine Mindestlänge von *limitRspSecureMessaging* unterstützen. Falls *keyReferenceList* länger ist als vom COS unterstüzt, dann MUSS als Trailer DataTruncated verwendet werden.
   1. MÜSSEN so viele beliebige Einträge wie nötig aus *keyReferenceList* entfernt werden, damit die vom COS unterstützte Länge eingehalten wird.
   2. MUSS als Trailer DataTruncated verwendet werden.
5. (N099.464) K\_COS   
   Für den Trailer der Antwort-APDU gilt:
   1. Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
   2. Für die Priorität der Trailer gilt: DataTruncated MUSS eine höhere Priorität als NoError haben.
6. (N099.466) FunktionaleZulassung   
   Es ist unzulässig mehr Einträge aus *keyReferenceList* zu entfernen, als nötig. Im Rahmen funktionaler Zulassungstests MUSS eine Abweichung zur Testerwartungshaltung ausgewiesen werden, falls von allen möglichen Codierungen von *pukReference* aller in *keyReferenceList* fehlender Schlüssel die längste dieser Codierungen zu *keyReferenceList* hinzugefügbar ist, ohne dadurch *limitRspSecureMessaging* zu überschreiten.

### Manage Channel

Das Kommando Manage Channel dient dem Öffnen und Schließen von logischen Kanälen mit einer von null verschiedenen Kanalnummer sowie dem Rücksetzen eines beliebigen logischen Kanals. Ob ein Kanal geöffnet, welcher Kanal geschlossen oder zurückgesetzt wird, bestimmen Parameter, die diesem Manage Channel-Kommando beigefügt sind.

Es ist möglich, dass ein zusätzlich zum Basiskanal 0 geöffneter logischer Kanal Einfluss auf die Ausführung eines Kommandos hat, wenn dieses Kommando auf Objekte zugreift, welche in anderen logischen Kanälen aktiv sind. Zwecks Vermeidung von Seiteneffekten gilt folgende Anforderung, die von der Kommando-APDU schickenden Einheit einzuhalten ist:

1. (N099.500) K\_externeWelt {K\_Karte}   
   Das Kommando Delete DARF NICHT an das COS gesendet werden, wenn außer dem Basiskanal noch weitere logische Kanäle geöffnet sind.

#### Use Case Öffnen eines logischen Kanals

In dieser Variante enthält die APDU des Manage Channel zwei Parameter:

1. (N099.504) K\_externeWelt {K\_Karte}   
   Der Parameter *intendedAction* zeigt an, dass ein logischer Kanal zu öffnen ist, wobei die Kanalnummer vom COS bestimmt wird. Der Wert von *intendedAction* MUSS 0 = ´0000´ sein.
2. (N099.506) K\_externeWelt {K\_Karte}   
   Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS 1 = ´01´ sein.
3. (N099.508) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 246 verwendet werden.

Tabelle 246: Manage Channel zum Öffnen eines logischen Kanals

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´70´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´00´ | *intendedAction*, hier Öffnen eines Kanals,  Kanalnummer wird vom COS bestimmt. |
| P2 | ´00´ |
| Le | ´01´ | Anzahl der erwarteten Oktette in den Antwortdaten |

#### Use Case Schließen eines logischen Kanals

In dieser Variante enthält die APDU des Manage Channel-Kommandos zwei Parameter:

1. (N099.510) K\_externeWelt {K\_Karte}   
   Der Parameter *logicalChannelNumber* MUSS die Nummer eines geöffneten Kanals enthalten, der zu schließen ist. Der Wert von *logicalChannelNumber* MUSS von null verschieden sein und MUSS gemäß [ISO/IEC 7816-4#5.4.1] ins CLA-Byte der APDU eingestellt werden, die an der Schnittstelle „Interface I/O“ sichtbar ist.
2. (N099.512) K\_externeWelt {K\_Karte}   
   Der Parameter *intendedAction* zeigt an, dass ein logischer Kanal zu schließen ist, wobei die Kanalnummer im CLA-Byte übertragen wird. Der Wert von *intendedAction* MUSS 32.768 = ´8000´ sein.
3. (N099.514) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interface I/O“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 247 verwendet werden.

Tabelle 247: Manage Channel zum Schließen eines logischen Kanals

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´XX´ | CLA-Byte gemäß [ISO/IEC 7816-4] mit einer von 0 verschiedenen Kanalnummer |
| INS | ´70´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´80´ | *intendedAction*, hier Schließen eines Kanals,  betroffener Kanal wird im CLA-Byte angezeigt. |
| P2 | ´00´ |

#### Use Case Zurücksetzen eines logischen Kanals

In dieser Variante enthält die APDU des Manage Channel-Kommandos zwei Parameter:

1. (N099.520) K\_externeWelt {K\_Karte}   
   Der Parameter *logicalChannelNumber* MUSS die Nummer eines geöffneten Kanals enthalten, der zurückzusetzen ist. Der Wert von *logicalChannelNumber* MUSS gemäß [ISO/IEC 7816-4#5.4.1] ins CLA-Byte der APDU eingestellt werden, die an der Schnittstelle „Interface I/O“ sichtbar ist.
2. (N099.522) K\_externeWelt {K\_Karte}   
   Der Parameter *intendedAction* zeigt an, dass ein logischer Kanal zurückzusetzen ist, wobei die Kanalnummer im CLA-Byte übertragen wird. Der Wert von *intendedAction* MUSS 16.384 = ´4000´ sein.
3. (N099.524) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interface I/O“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 248 verwendet werden.

Tabelle 248: Manage Channel zum Zurücksetzen eines logischen Kanals

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´XX´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´70´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´40´ | *intendedAction*, hier Zurücksetzen eines Kanals,  betroffener Kanal wird im CLA-Byte angezeigt. |
| P2 | ´00´ |

#### Use Case logischer Reset der Applikationsebene

In dieser Variante enthält die APDU des Manage Channel-Kommandos einen Parameter:

1. (N099.530) K\_externeWelt {K\_Karte}   
   Der Parameter *intendedAction* zeigt an, dass der Basiskanal zurückzusetzen ist und alle anderen logischen Kanäle zu schließen sind. Der Wert von *intendedAction* MUSS 16.385 = ´4001´ sein.
2. (N099.532) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interface I/O“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 249 verwendet werden.

Tabelle 249: Manage Channel zum logischen Reset der Applikationsebene

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´70´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´40´ | *intendedAction*, hier logischer Reset der Applikationsebene, d.h. Basiskanal zurücksetzen und alle anderen logischen Kanäle schließen. |
| P2 | ´01´ |

#### Antwort der Karte auf Kanalmanagementoperationen

Tabelle 250: Manage Channel Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Daten** | **Inhalt** | **Beschreibung** |
| Kanalnr. | ´XX´ | Nummer des soeben geöffneten Kanals |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | erfolgreiche Operation |

Tabelle 251: Manage Channel Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´69 81´ | NoMoreChannelsAvailable | kein weiterer logischen Kanäle verfügbar |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes in Abbildung 1 entdeckt wurden.
2. (N099.540) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N099.542) Kommandounterstützung
   1. K\_COS   
      Das COS MUSS die Manage Channel-Variante aus 14.9.8.3 unterstützen.
   2. K\_COS, Option\_logische\_Kanäle   
      Das COS MUSS die Manage Channel-Varianten aus 14.9.8.1, 14.9.8.2 und 14.9.8.4 unterstützen.
   3. Das COS KANN weitere Manage Channel-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N099.545) K\_COS   
   Zugriffsregeln für das Manage Channel Kommando:
   1. Das COS KANN die Auswertung von Zugriffsregeln für das Manage Channel Kommando unterstützen.
   2. Das COS KANN als Zugriffsbedingung für das Manage Channel Kommando stets ALWAYS verwenden.
3. (N099.548) K\_COS   
   Wenn *intendedAction* das
   1. Öffnen eines logischen Kanals (*indendedAction* = P1P2 = ´0000´) anzeigt
      1. und bereits alle verfügbaren logischen Kanäle geöffnet sind, dann MUSS das Kommando mit dem Trailer NoMoreChannelsAvailable terminieren,
      2. andernfalls MUSS das COS
         1. eine nicht verwendete Kanalnummer *newChannelNumber* allokieren, wobei die Zahl *newChannelNumber* aus dem Intervall [1, 19] gewählt werden MUSS.
         2. einen weiteren logischen Kanal öffnen,
         3. diesem die allokierte Nummer *newChannelNumber* zuweisen und
         4. dessen Kanalkontext entsprechend (N030.100) initialisieren.
      3. Als Datenfeld der Antwortnachricht MUSS I2OS( *newChannelNumber*, 1) verwendet werden.
   2. Schließen eines logischen Kanals (*indendedAction* = P1P2 = ´8000´) anzeigt
      1. dann MUSS der entsprechende logische Kanal geschlossen werden und
      2. die freiwerdende Kanalnummer MUSS für zukünftige Allokationen verfügbar sein.
      3. Das Datenfeld der Antwortnachricht MUSS leer sein.
   3. Zurücksetzen eines logischen Kanals (*indendedAction* = P1P2 = ´4000´) anzeigt,
      1. dann MUSS der *channelContext* des entsprechenden logischen Kanals auf den in (N030.100) definierten Wert gesetzt werden, und
      2. das Datenfeld der Antwortnachricht MUSS leer sein.
   4. logische Resetten (*indendedAction* = P1P2 = ´4001´) anzeigt
      1. dann MÜSSEN bis auf den Basiskanal alle anderen offenen logischen Kanäle geschlossen werden und
      2. die freiwerdenden Kanalnummern MÜSSEN für zukünftige Allokationen verfügbar sein, und
      3. der *channelContext* des Basiskanals auf den in (N030.100) definierten Wert gesetzt werden, und
      4. das Datenfeld der Antwortnachricht MUSS leer sein.
4. (N099.551) K\_COS   
   Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
5. (N099.554) K\_COS   
   Für die Priorität der Trailer gilt:
   1. Die Priorität der Trailer in Tabelle 251 ist herstellerspezifisch.
   2. Jeder Trailer in Tabelle 251 MUSS eine höhere Priorität als NoError haben.

### Manage Security Environment

Das Kommando Manage Security Environment (MSE) verändert im *currentFolder* die Attribute *seIdentifier* und Elemente von *keyReferenceList*. Welche Aktion durchzuführen ist, welches Attribut oder Listenelement betroffen ist und auf welchen Wert sie zu ändern sind, wird durch Parameter bestimmt, die in der Kommandonachricht enthalten sind.

1. (N099.600) K\_externeWelt {K\_Karte}   
   Falls ein symmetrisches Authentisierungsobjekt oder ein symmetrisches Kartenverbindunsobjekt oder ein privates Schlüsselobjekt referenziert wird, dann besteht der Parameter *keyReference* aus den zwei Teilen *location* und *identifier*. *location* zeigt an, ob ein globaler oder DF-spezifischer Schlüssel von der Aktion betroffen ist. Als Wert für *location* MUSS ein Element der Menge {´00´, ´80´} verwendet werden. Dabei gilt:
   1. Der Wert *location* = ´00´ MUSS verwendet werden, wenn ein globaler Schlüssel betroffen ist.
   2. Der Wert *location* = ´80´ MUSS verwendet werden, wenn ein DF-spezifischer Schlüssel betroffen ist.
   3. Der Parameter *identifier* bestimmt das betroffene Schlüsselobjekt. Der Wert von *identifier* MUSS konform zu (N016.400) bzw. (N017.100) gewählt werden.
   4. Der Parameter *keyReference* MUSS in einem Oktett mit folgendem Wert codiert werden: *keyReference* = *location* + *identifier*.

#### Use Case Ändern des SE-Identifiers

In dieser Variante enthält die APDU des MSE Kommandos zwei Parameter:

1. (N099.700) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´F3´ gewählt werden.
2. (N099.800) K\_externeWelt {K\_Karte}   
   Der Parameter *seNo* MUSS gemäß (N007.900) gewählt werden.
3. (N099.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 252 verwendet werden.

Tabelle 252: MSE, Restore Variante

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´F3´ | *operationMode* = Auswahl eines SE–Identifiers |
| P2 | ´XX´ | *seNo* = Wert des auszuwählenden SE–Identifiers |

#### Use Case Schlüsselauswahl zur internen, symmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N100.000) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´41´ gewählt werden.
2. (N100.100) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´A4´ gewählt werden.
3. (N100.200) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
4. (N100.300) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {   
          aesSessionkey4TC,   
          desSessionkey4TC (Option\_DES),   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N100.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 253 verwendet werden.

Tabelle 253: MSE, Selektion symmetrischer Internal Authenticate-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´41´ | *operationMode* = Setzen eines „internen“ Schlüssels  *crtTag* = betroffenes Listenelement ist internalAuthenticate |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´83 – L83 – *keyRef* || 80 01 || *algId* ´ |

1. L83 ist ein Oktett mit dem Wert I2OS( OctetLength(keyRef), 1).

#### Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N100.500) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´41´ gewählt werden.
2. (N100.600) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´A4´ gewählt werden.
3. (N100.700) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
4. (N100.800) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 oder Tabelle 270 gewählt werden, wobei ein Wert aus der Menge {     
           elcAsynchronAdmin,   
           elcRoleAuthentication,   
           elcSessionkey4SM,   
           elcSessionkey4TC,   
           rsaClientAuthentication,   
           rsaRoleAuthentication (Option\_RSA\_CVC),   
           rsaSessionkey4SM (Option\_DES),   
           rsaSessionkey4TC (Option\_DES),   
           signPKCS1\_V1\_5   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N100.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 254 verwendet werden.

Tabelle 254: MSE, Selektion privater Internal Authenticate-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´41´ | *operationMode* = Setzen eines „internen“ Schlüssels  *crtTag* = betroffenes Listenelement ist internalAuthenticate |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´84 01 || *keyRef* || 80 01 || *algId* ´ |

#### Use Case Schlüsselauswahl zur externen, symmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N101.000) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N101.100) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´A4´ gewählt werden.
3. (N101.200) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
4. (N101.300) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {   
          aesSessionkey4TC,   
          desSessionkey4TC (Option\_DES),   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N101.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 255 verwendet werden.

Tabelle 255: MSE, Selektion symmetrischer External Authenticate-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Setzen eines „externen“ Schlüssels  *crtTag* = betroffenes Listenelement ist externalAuthenticate |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´83 – L83 – *keyRef* || 80 01 || *algId* ´ |

1. L83 ist ein Oktett mit dem Wert I2OS( OctetLength(keyRef), 1).

#### Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N101.500) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N101.600) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´A4´ gewählt werden.
3. (N101.700) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N019.500) gewählt werden.
4. (N101.800) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {   
          elcRoleCheck,   
          rsaRoleCheck (Option\_RSA\_CVC),   
          rsaSessionkey4SM (Option\_DES),   
          rsaSessionkey4TC (Option\_DES)   
      } verwendet werden MUSS.   
      *Hinweis: Die Schlüsselselektion für die Algorithmen elcSessionkey4SM und elcSessionkey4TC erfolgt im* General Authenticate*‑Kommando, siehe 14.7.2.2.1.*
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N101.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 256 verwendet werden.

Tabelle 256: MSE, Selektion öffentlicher External Authenticate-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Setzen eines öffentlichen Schlüssels  *crtTag* = betroffenes Listenelement ist externalAuthenticate |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´83‑0C‑*keyRef*   ||   80‑01‑*algId*´ |

#### Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N102.000) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N102.100) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´A4´ gewählt werden.
3. (N102.200) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
4. (N102.300) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {   
          aesSessionkey4SM   
          desSessionkey4SM (Option\_DES)   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N102.400) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 257 verwendet werden.

Tabelle 257: MSE, Selektion symmetrischer Mutual Authenticate-Schlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Setzen eines symmetrischen Schlüssels  *crtTag* = betroffenes Listenelement ist externalAuthenticate |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´83 01 *keyRef* || 80 01 *algId* ´ |

#### Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe

In dieser Variante wird ein symmetrisches Kartenverbindungsobjekt gemäß [BSI-TR-03110-3#B.11.1] ausgewählt ohne Referenzierung einer elliptischen Kurve. Anschließend ist es möglich eine Authentisierung mit dem PACE Authentisierungsprotokoll gemäß 15.4.2 durchzuführen. In dieser Variante enthält das MSE Kommando zwei Parameter:

1. (N102.440) K\_externeWelt {K\_Karte}   
   Der Parameter *OID* bestimmt, welche PACE Variante vom COS verwendet wird und enthält den neuen Wert für das Element *algortihmIdentifier* in den Listenelementen *externalAuthenticate* und *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 271 gewählt werden. Falls die

* 1. Option\_kontaktlose\_Schnittstelle unterstützt wird, dann MUSS für *OID* eine Wert aus der Menge {
     1. id-PACE-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
     2. id-PACE-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
     3. id-PACE-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)

} verwendet werden.

* 1. Option\_PACE\_PCD unterstützt wird, dann MUSS für *OID* ein Wert aus der Menge {
     1. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
     2. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
     3. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)

} verwendet werden.

1. (N102.444) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* in den Listenelementen *externalAuthenticate* und *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
2. (N102.448) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 258 verwendet werden.

Tabelle 258: MSE, Selektion sym. Kartenverbindungsobjekt ohne Kurvenangabe

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´C1´ | *operationMode* = Setzen eines geheimen Schlüsselobjektes  *crtTag* = betr. Listenelemente: *externalAuthenticate*, *internalAuthenticate* |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´80 I2OS(OctetLength(*OID*), 1) || *OID*   ||   83 01 || *keyRef* ´ |

#### Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe

In dieser Variante wird ein symmetrisches Kartenverbindungsobjekt gemäß [BSI-TR-03110-3#B.11.1] ausgewählt mit Referenzierung einer elliptischen Kurve. Anschließend ist es möglich eine Authentisierung mit dem PACE-Authentisierungsprotokoll gemäß 15.4.2 durchzuführen. In dieser Variante enthält das MSE Kommando drei Parameter:

1. (N102.450) K\_externeWelt {K\_Karte}   
   Der Parameter *OID* hat dieselbe Bedeutung und Codierung wie in (N102.440).
2. (N102.452) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* hat dieselbe Bedeutung und Codierung wie in (N102.444).
3. (N102.454) K\_externeWelt {K\_Karte}   
   Der Parameter *idDomainParameter* enthält einen Identifier für Domainparameter der zu verwendenden elliptischen Kurve mit der in Tabelle 259 gezeigten Zuordnung gemäß [BSI-TR-03110-3#A.2.1.1], wobei *idDomainparameter* passend zum Attribut *algorithmIdentifier* des referenzierten symmetrischen Kartenverbindungsobjektes gewählt werden MUSS:

Tabelle 259: Zuordnung von *idDomainParameter* zu Domainparameter

|  |  |  |
| --- | --- | --- |
| **ID** | **Domainparameter** | ***algorithmIdentifier* im sym. Kartenverbindungsobjekt** |
| 13 = ´0D´ | brainpoolP256r1 | id-PACE-ECDH-GM-AES-CBC-CMAC-128 oder  id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128 |
| 16 = ´10´ | brainpoolP384r1 | id-PACE-ECDH-GM-AES-CBC-CMAC-192 oder  id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192 |
| 17 = ´11´ | brainpoolP512r1 | id-PACE-ECDH-GM-AES-CBC-CMAC-256 oder  id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256 |

1. (N102.458) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S-Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3-Kommando-APDU MÜSSEN die Angaben aus Tabelle 260 verwendet werden.

Tabelle 260: MSE, Selektion symmetrisches Kartenverbindungsobjekt

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´C1´ | *operationMode* = Setzen eines geheimen Schlüsselobjektes  *crtTag* = betr. Listenelemente: *externalAuthenticate*, *internalAuthenticate* |
| P2 | ´A4´ |
| Data | ´XX…XX´ | ´80 I2OS(OctetLength(*OID*), 1) || *OID* ||   83 01 || *keyRef* ||   84 01 || *idDomainparameter* ´ |

#### Use Case Schlüsselauswahl für Signierschlüssel

Dieser Use Case wird verwendet, um einen Signierschlüssel zu selektieren. Anschließend ist es möglich diesen Schlüssel zu erzeugen oder seinen öffentlichen Teil auszulesen (siehe 14.9.3) oder mit diesem Schlüssel Signaturen zu erzeugen (siehe 14.8.2). In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N102.500) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´41´ gewählt werden.
2. (N102.600) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´B6´ gewählt werden.
3. (N102.700) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *signatureCreation*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
4. (N102.800) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *signatureCreation*. Wert und Codierung MÜSSEN gemäß Tabelle 268 oder Tabelle 270 gewählt werden, wobei ein Wert aus der Menge {   
          rsaClientAuthentication,   
          sign9796\_2\_DS2,   
          signECDSA,   
          signPKCS1\_V1\_5,   
          signPSS   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N102.900) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 261 verwendet werden.

Tabelle 261: MSE, Selektion privater Signaturschlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´41´ | *operationMode* = Setzen eines privaten Schlüssels  *crtTag* = betroffenes Listenelement ist signatureCreation |
| P2 | ´B6´ |
| Data | ´XX…XX´ | ´84 01 || *keyRef* || 80 01 || *algId* ´ |

#### Use Case Schlüsselauswahl zum Prüfen von CV–Zertifikaten

In dieser Variante enthält die APDU des MSE Kommandos drei Parameter:

1. (N103.000) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N103.100) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´B6´ gewählt werden.
3. (N103.200) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *verifyCertificate*. Wert und Codierung MÜSSEN gemäß (N019.100) gewählt werden.
4. (N103.300) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 262 verwendet werden.

Tabelle 262: MSE, Selektion öffentlicher Zertifikatsprüfschlüssel

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Setzen eines öffentlichen Schlüssels  *crtTag* = betroffenes Listenelement ist verifyCertificate |
| P2 | ´B6´ |
| Data | ´XX…XX´ | ´83 08 || *keyRef* ´ |

#### Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N103.400) K\_externeWelt {K\_Karte}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´41´ gewählt werden.
2. (N103.500) K\_externeWelt {K\_Karte}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´B8´ gewählt werden.
3. (N103.600) K\_externeWelt {K\_Karte}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *dataDecipher*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.
4. (N103.700) K\_externeWelt {K\_Karte}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *dataDecipher*. Wert und Codierung MÜSSEN gemäß Tabelle 269 gewählt werden, wobei ein Wert aus der Menge {   
          elcSharedSecretCalculation,   
          rsaDecipherPKCS1\_V1\_5,   
          rsaDecipherOaep   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N103.800) K\_externeWelt {K\_Karte}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 263 verwendet werden.

Tabelle 263: MSE, Schlüsselselektion zur Entschlüsselung

|  |  |  |
| --- | --- | --- |
|  | **Inhalt** | **Beschreibung** |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´41´ | *operationMode* = Setzen eines privaten Schlüssels  *crtTag* = betroffenes Listenelement ist dataDecipher |
| P2 | ´B8´ |
| Data | ´XX…XX´ | ´84 01 || *keyRef* || 80 01 || *algId* ´ |

#### Use Case Schlüsselauswahl für Verschlüsselung

Dieser Use Case wird verwendet um einen Verschlüsselungsschlüssel zu selektieren. Anschließend ist es möglich diesen Schlüssel zum Verschlüsseln von Daten einzusetzen (siehe Kapitel 14.8.4.3 und 14.8.4.4). In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

1. (N103.830) K\_externeWelt {K\_COS}   
   Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ´81´ gewählt werden.
2. (N103.835) K\_externeWelt {K\_COS}   
   Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = ´B8´ gewählt werden.
3. (N103.840) K\_externeWelt {K\_COS}   
   Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *dataEncipher*. Wert und Codierung MÜSSEN gemäß (N019.822) gewählt werden.
4. (N103.845) K\_externeWelt {K\_COS}
   1. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *dataEncipher*. Wert und Codierung MÜSSEN gemäß Tabelle 269 gewählt werden, wobei ein Wert aus der Menge {   
          elcSharedSecretCalculation,   
          rsaEncipherOaep,   
          rsaEncipherPKCS1\_V1\_5   
      } verwendet werden MUSS.
   2. Das COS KANN weitere Werte für *algId*
      1. akzeptieren oder
      2. ablehnen.
5. (N103.850) K\_externeWelt {K\_COS}   
   Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 261 verwendet werden.

Tabelle 264: MSE, Selektion Verschlüsselungsschlüssel

|  | **Inhalt** | **Beschreibung** |
| --- | --- | --- |
| CLA | ´00´ | CLA-Byte gemäß [ISO/IEC 7816-4] |
| INS | ´22´ | Instruction Byte gemäß [ISO/IEC 7816-4] |
| P1 | ´81´ | *operationMode* = Setzen eines öffentlichen Schlüssels  *crtTag* = betroffenes Listenelement ist dataEncipher |
| P2 | ´B8´ |
| Data | ´XX…XX´ | ´83 0C || *keyRef* || 80 01 || *algId* ´ |

#### Antwort der Karte auf Management des Security Environments

Tabelle 265: MSE Antwort-APDU im Erfolgsfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´90 00´ | NoError | Erfolgreiche Übernahme der Kommandodatenparameter |

Tabelle 266: MSE Antwort-APDU im Fehlerfall

|  |  |  |
| --- | --- | --- |
| **Trailer** | **Inhalt** | **Beschreibung** |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegebenen Algorithmus nicht |
| ´6A 88´ | KeyNotFound | Zu selektierendes Schlüsselobjekt wurde nicht gefunden |

1. Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.
2. (N103.900) K\_COS   
   Ein COS KANN zusätzliche Trailer verwenden.

#### Kommandoabarbeitung innerhalb der Karte

1. (N104.000) Kommandounterstützung
   1. K\_COS   
      Das COS MUSS die Manage Security Environment-Varianten aus 14.9.9.1, 14.9.9.3, 14.9.9.5, 14.9.9.6, 14.9.9.9, 14.9.9.10 und 14.9.9.11 unterstützen.
   2. K\_COS, Option\_Kryptobox   
      Das COS MUSS die Manage Security Environment-Varianten aus 14.9.9.2, 14.9.9.4 und 14.9.9.12 unterstützen.
   3. K\_COS, Option\_kontaktlose\_Schnittstelle   
      Das COS MUSS die Manage Security Environment-Varianten aus 14.9.9.7 und 14.9.9.8 unterstützen.
   4. K\_COS, Option\_PACE\_PCD   
      Das COS MUSS die Manage Security Environment-Variante aus 14.9.9.7 und 14.9.9.8 unterstützen.
   5. Das COS KANN weitere Manage Security Environment-Varianten
      1. unterstützen oder
      2. ablehnen.
2. (N104.100) K\_COS   
   Zugriffsregeln für das Manage Security Environment Kommando:
   1. Das COS KANN die Auswertung von Zugriffsregeln für das Manage Security Environment Kommando unterstützen.
   2. Das COS KANN als Zugriffsbedingung für das Manage Security Environment Kommando stets ALWAYS verwenden.
3. (N104.200) K\_COS   
   Wenn der Parameter *operationMode* in P1 den Wert ´F3´ besitzt, dann MUSS
   1. in *currentFolder* das Attribut *seIdentifier* (siehe (N030.000)a) auf den Wert des Parameters *seNo* gesetzt werden.
   2. jedes Listenelement in *keyReferenceList* (siehe (N029.900)c), welches auf ein Schlüsselobjekt in *currentFolder* zeigt, auf den Wert „leer“ gesetzt werden.
   3. jedes Element in *globalPasswordList* (siehe (N029.900)i) und in *dfSpecificPasswordList* (siehe (N029.900)j) mittels clearPasswordStatus(…) aus den genannten Listen entfernt werden, falls es *currentFolder* zugeordnet ist.
   4. der Wert von *currentEF* unverändert bleiben.
   5. clearSecurityStatusFolder(*currentFolder* ) ausgeführt werden.
   6. Attribute, die übergeordneten Ordnern zugeordnet sind, DÜRFEN sich NICHT ändern.
4. (N104.300) K\_COS   
   Wenn der Parameter *operationMode* in P1 einen Wert aus der Menge {´41´, ´81´, ´C1´} besitzt und
   1. im Falle einer ein Oktett langen Schlüsselreferenz *keyRef*
      1. SOLL entweder mittels SearchKey(*channelContext*.*currentFolder*, *keyRef*, *algId*) nach dem Schlüssel gesucht werden. Wenn diese Funktion
         1. den Fehler keyNotFound meldet, dann MUSS dass Kommando mit dem Trailer KeyNotFound terminieren.
         2. den Fehler notSupported meldet, dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
         3. keinen Fehler meldet dann MUSS als Trailer NoError gewählt werden.
      2. oder KANN keine Schlüsselsuche durchgeführt und als Trailer NoError gewählt werden.
   2. im Falle einer acht Oktett (öffentliches Signaturprüfobjekt) langen Schlüsselreferenz, MUSS *keyRef* mittels   
      SearchKey(*channelContext*.*currentFolder*, *keyRef*, verifyCertificate) nach dem Schlüssel gesucht werden. Wenn diese Funktion
      1. den Fehler keyNotFound meldet, dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. den Fehler notSupported meldet, dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
      3. keinen Fehler meldet, dann MUSS als Trailer NoError gewählt werden.
   3. im Falle einer zwölf Oktett (öffentliches Authentisierungsobjekt, öffentliches Verschlüsselungsobjekt) langen Schlüsselreferenz, MUSS *keyRef* mittels SearchKey(*channelContext*.*currentFolder*, *keyRef*, *algId*) nach dem Schlüssel gesucht werden. Wenn diese Funktion
      1. den Fehler keyNotFound meldet, dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
      2. den Fehler notSupported meldet, dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
      3. keinen Fehler meldet, dann MUSS als Trailer NoError gewählt werden.
5. (N104.320) K\_COS   
   Falls als Trailer
   1. NoError verwendet wird, dann MÜSSEN in *channelContext.keyReferenceList* die durch die Parameter *operationMode* und *crtTag* gekennzeichneten Listenelemente mit den Parametern aus dem Datenfeld der Kommandonachricht gefüllt werden.
   2. nicht NoError verwendet wird, dann DARF *channelContext.keyReferenceList* NICHT verändert werden.
6. (N104.400) Dieser Punkt ist absichtlich leer.
7. (N104.500) Dieser Punkt ist absichtlich leer.

# Authentisierungsprotokolle (normativ)

Dieses Kapitel beschreibt, wie eine schlüsselbasierte Authentisierung zwischen einer externen Entität und dem COS abläuft. Die externe Entität wird als Gegenstelle bezeichnet. Aus Symmetriegründen und der leichteren Referenzierbarkeit innerhalb dieses Dokumentes wird eine sprachliche Darstellung gewählt, welche die Gegenstelle als Smartcard mit einem Funktionsumfang beschreibt, der dem Funktionsumfang dieses Dokumentes analog ist. Diese Darstellungsform beschränkt nicht die Allgemeingültigkeit, weil hier lediglich die Schnittstelle zwischen Gegenstelle und COS betrachtet wird, und für die Gegenstelle aus Sicht dieses Dokumentes auch eine beliebige andere Komponente mit entsprechender Funktionalität möglich ist.



Abbildung 3: Kommunikationsmodell Gegenstelle, Steuersoftware und COS

In diesem Kapitel wird folgendes Kommunikationsmodell verwendet:

* Eine Komponente COS besitze die in diesem Dokument spezifizierten Schnittstelleneigenschaften.
* Eine Komponente Gegenstelle besitze zur Komponente COS analoge Eigenschaften.
* Eine Komponente Steuersoftware besitze eine Ablauflogik. Zudem kommuniziert die Steuersoftware mit der Gegenstelle über den Kommunikationskanal Channel\_a und mit dem COS über den Kommunikationskanal Channel\_b. Diese beiden Kanäle entsprechen der physikalischen Schnittstelle in Abbildung 1.

Jedem, der hier behandelten Verfahren, ist ein eigenes Unterkapitel gewidmet. Im Allgemeinen besteht die Authentisierung aus einer Sequenz von einem oder mehreren Kommandos, die über die Kommunikationskanäle gesendet werden. Für alle Authentisierungsprotokolle gilt:

1. (N104.600) K\_externeWelt {K\_Karte}   
   Falls die Sequenz eines Authentisierungsprotokolls aus mehr als einem Kommando-Antwort-Paar besteht (siehe 11.1), so DARF diese Sequenz an der Schnittstelle „Channel\_b“ (siehe Abbildung 3) NICHT durch Kommandos unterbrochen werden, welche nicht zu dieser Sequenz gehören.
2. (N104.700) K\_COS   
   Falls die Anforderung (N104.600) von der externen Entität nicht eingehalten wird, mithin also die Sequenz durch ein unterbrechendes Kommando unterbrochen wird, dann KANN das COS
   1. das unterbrechende Kommando akzeptieren.
   2. das unterbrechende Kommando ablehnen.
   3. ein Fortsetzen der unterbrochenen Sequenz akzeptieren.
   4. ein Fortsetzen der unterbrochenen Sequenz ablehnen.
3. (N104.800) K\_COS   
   Es MUSS möglich sein, dass jede, der hier genannten Sequenzen, ungeschützt (ohne Secure Messaging) übertragen wird. Falls auch die geschützte Übertragung der Sequenz zu unterstützen ist, so ist dies im entsprechenden Kapitel vermerkt.
4. (N104.850) K\_externeWelt {K\_Karte}   
   Wenn das erste Kommando dieser Sequenz
   1. geschützt übertragen wird, dann MÜSSEN auch alle anderen Kommandos dieser Sequenz geschützt übertragen werden.
   2. ungeschützt übertragen wird, dann MÜSSEN auch alle anderen Kommandos dieser Sequenz ungeschützt übertragen werden.
5. (N104.900) K\_COS   
   Falls die Anforderung (N104.800) von der externen Entität nicht eingehalten wird, dann KANN das COS dies
   1. akzeptieren oder
   2. ablehnen.

## Externe Authentisierung

Dieses Kapitel behandelt die Authentisierung einer „Gegenstelle“ gegenüber dem COS.



Abbildung 4: Sequenzdiagramm zur externen Authentisierung

1. Die Bedeutung der Bezeichnungen in Abbildung 4 ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß Kapitel 14.

### Externe Authentisierung mittels symmetrischer Schlüssel

In einer früheren Dokumentenversion wurde in diesem Unterkapitel die einseitige symmetrische Authentisierung ohne Vereinbarung von Sessionkeys behandelt. Weil diese Art der Authentisierung nicht mehr zum normativen Umfang gehört, sind alle Anforderungen dieses Unterkapitels leer.

1. (N105.000) Diese Anforderung ist absichtlich leer.
2. (N105.100) Diese Anforderung ist absichtlich leer.
3. (N105.200) Diese Anforderung ist absichtlich leer.

### RSA, asymmetrische Rollenauthentisierung, Option\_RSA\_CVC

Für den Fall, dass eine externe Entität ihre Authentizität mittels eines RSA-Schlüsselpaares nachweisen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 4):

1. (N105.300) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Das erste Kommando der Sequenz MUSS Get Challenge gemäß 14.9.4.1 sein und über Channel\_b geschickt werden.
2. (N105.400) K\_externeWelt {K\_Karte}, Option\_RSA\_CVC   
   Das zweite Kommando über Channel\_b ist das letzte dieser Sequenz und MUSS ein External Authenticate gemäß 14.7.1.1 mit *algId* gleich rsaRoleCheck sein.
3. (N105.500) K\_COS   
   Es MUSS möglich sein, die Kommandos dieser Sequenz, welche über Channel\_b gesendet werden, geschützt (mit Secure Messaging) zu übertragen.

### ELC, asymmetrische Berechtigungsnachweis

Für den Fall, dass eine externe Entität ihre Authentizität mittels eines ELC-Schlüsselpaares nachweisen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 4):

1. (N105.600) K\_externeWelt {K\_Karte}   
   Das erste Kommando der Sequenz MUSS Get Challenge gemäß 14.9.4.2 sein und über Channel\_b geschickt werden.
2. (N105.700) K\_externeWelt {K\_Karte}   
   Das zweite Kommando über Channel\_b ist das letzte dieser Sequenz und MUSS ein External Authenticate gemäß 14.7.1.1 mit *algId* aus der Menge {elcRoleCheck} sein.
3. (N105.800) K\_COS   
   Es MUSS möglich sein, die Kommandos dieser Sequenz, welche über Channel\_b gesendet werden, geschützt (mit Secure Messaging) zu übertragen.

## Interne Authentisierung

Für den Fall, dass eine externe Entität die Authentizität des COS mittels eines Schlüssels überprüfen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 5):

1. (N105.900) K\_externeWelt {K\_Karte}   
   Das erste Kommando über Channel\_b ist das letzte dieser Sequenz für Channel\_b und MUSS ein Internal Authenticate gemäß 14.7.4.1 mit einer *algId* aus folgender Menge sein: {   
       elcRoleAuthentication,   
       rsaClientAuthentication,   
       rsaRoleAuthentication (Option\_RSA\_CVC),   
       signPKCS1\_V1\_5   
   }.
2. (N106.000) K\_COS   
   Es MUSS möglich sein, die Kommandos dieser Sequenz, welche über Channel\_b gesendet werden, geschützt (mit Secure Messaging) zu übertragen.



Abbildung 5: Sequenzdiagramm zur internen Authentisierung

1. Die Bedeutung der Bezeichnungen in Abbildung 5 ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß Kapitel 14.

## Card-2-Card-Authentisierung ohne Sessionkey-Aushandlung

Die Card-2-Card-Authentisierung ohne Sessionkey Aushandlung ist eine spezielle Variante der in 15.1 und 15.2 behandelten Verfahren, wobei die Gegenstelle definitiv eine Smartcard ist.

Falls eine einseitige Authentisierung (nur eine der beiden Komponenten Gegenstelle oder COS authentisiert sich) beabsichtigt ist, wird je nach Richtung der Authentisierung entweder ein Algorithmus aus 15.1 oder aus 15.2 gewählt.

Falls eine gegenseitige Authentisierung (beide Komponenten, Gegenstelle und COS, authentisieren sich) beabsichtigt ist, wird sowohl ein Algorithmus aus 15.1 als auch ein Algorithmus aus 15.2 gewählt. Typischerweise legen Zugriffsregeln der beteiligten Schlüssel fest, welche Komponente sich zuerst zu authentisieren hat.

## Aushandlung von Sessionkey

Dieses Unterkapitel behandelt die gegenseitige Authentisierung zweier Entitäten, wobei gleichzeitig Sessionkeys ausgehandelt werden.

### Sessionkeys mittels symmetrischer Authentisierungsobjekte

Für den Fall, dass eine gegenseitige Authentisierung mittels symmetrischer Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 6):

1. (N106.100) K\_externeWelt {K\_Karte}   
   Das erste Kommando der Sequenz MUSS Get Challenge gemäß 14.9.4.1 sein und über Channel\_b geschickt werden. Die dabei vom COSb erzeugte Zufallszahl wird mit RND.1 bezeichnet.
2. (N106.200) K\_externeWelt {K\_Karte}   
   Das zweite Kommando der Sequenz MUSS über Channel\_a zum COSa gesendet werden und MUSS ein Internal Authenticate gemäß 14.7.4.1 mit *algId* gleich symSessionkey4TC sein. Dabei MUSS *cmdData* RND.1 und *iccsn8* des COSb gemäß M2 aus (N084.400)b.1 enthalten. Die Antwortdaten des COSa werden mit rspData.2 bezeichnet.
3. (N106.300) K\_externeWelt {K\_Karte}   
   Das dritte Kommando der Sequenz MUSS über Channel\_b zum COSb gesendet werden. Es MUSS ein Mutual Authenticate gemäß 14.7.1.2 mit *algId* gleich symSessionkey4SM sein. Als Kommandodaten MÜSSEN rspData.2 verwendet werden. Die Antwortdaten des COSb werden mit rspData.3 bezeichnet.
4. (N106.400) K\_externeWelt {K\_Karte}   
   Das vierte Kommando ist das letzte dieser Sequenz und MUSS über Channel\_a zum COSa gesendet werden. Es MUSS ein External Authenticate gemäß 14.7.1.1 mit *algId* gleich symSessionkey4TC sein. Als Kommandodaten MÜSSEN rspData.3 verwendet werden.
5. (N106.500) K\_COS   
   Sowohl COSa als auch COSb KANN die geschützte Übertragung der Sequenz
   1. erlauben oder
   2. ablehnen.



Abbildung 6: Sequenzdiagramm symmetrische Sessionkey-Aushandlung

1. Aus Gründen der Übersichtlichkeit wurden in Abbildung 6 kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in Kapitel 14. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.

### Sessionkeys mittels symmetrischer Kartenverbindungsobjekte

Für den Fall, dass eine gegenseitige Authentisierung mittels symmetrischer Kartenverbindungsobjekte durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 7):

1. (N106.540) K\_COS, Option\_kontaktlose\_Schnittstelle   
   Das COS MUSS das PACE-Protokoll in der Version 2 gemäß [BSI-TR-03110-2#3.2] und [BSI-TR-03110-3] für OID aus folgender Menge unterstützen:
   1. id-PACE-ECDH-GM-AES-CBC-CMAC-128 mit brainpoolP256r1
   2. id-PACE-ECDH-GM-AES-CBC-CMAC-192 mit brainpoolP384r1
   3. id-PACE-ECDH-GM-AES-CBC-CMAC-256 mit brainpoolP512r1
2. (N106.545) K\_COS   
   Verfügbarkeit PACE Authentisierungsprotokoll:
   1. Option\_kontaktlose\_Schnittstelle: Das COS MUSS die PACE-Varianten aus (N106.540) in der Rolle „MRTD Chip (PICC)“ unterstützen.

* 1. Option\_PACE\_PCD: Das COS MUSS die PACE-Varianten aus (N106.540) in der Rolle „Terminal (PCD)“ unterstützen.

1. In (N106.545)b ist absichtlich nur die Option\_PACE\_PCD enthalten, nicht aber die Option\_kontaktlose\_Schnittstelle.
2. (N106.547) K\_COS   
   Das COS KANN die geschützte Übertragung der Sequenz
   1. erlauben oder
   2. ablehnen.
3. (N106.550) K\_externeWelt {K\_Karte}   
   Im ersten Schritt MUSS zur Komponente
   1. COSa ein General Authenticate-Kommando gemäß 14.7.2.1.1 geschickt werden. Die Antwortdaten enthalten das Kryptogramm *z*, welches im zweiten Schritt auf Seiten der Komponente COSb benötigt wird.
   2. COSb ein General Authenticate-Kommando gemäß 14.7.2.4.1 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel *~PK1PCD*, der im zweiten Schritt auf Seiten der Komponente COSa benötigt wird.
4. (N106.552) K\_externeWelt {K\_Karte}   
   Im zweiten Schritt MUSS zur Komponente
   1. COSa ein General Authenticate-Kommando mit *~PK1PCD* aus (N106.550)b gemäß 14.7.2.1.2 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel *~PK1PICC* welcher im dritten Schritt auf Seiten der Komponente COSb benötigt wird.
   2. COSb ein General Authenticate-Kommando mit *z* aus (N106.550)a gemäß 14.7.2.4.2 geschickt werden. Die Antwortnachricht enthält kein Datenfeld.
5. (N106.554) K\_externeWelt {K\_Karte}   
   Im dritten Schritt MUSS zum COSb ein General Authenticate-Kommando mit *~PK1PICC* aus (N106.552)a gemäß 14.7.2.4.3 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel *~PK2PCD* welcher im vierten Schritt auf Seiten der Komponente COSa benötigt wird.
6. (N106.556) K\_externeWelt {K\_Karte}   
   Im vierten Schritt MUSS zum COSa ein General Authenticate-Kommando mit *~PK2PCD* aus (N106.554) gemäß 14.7.2.1.3 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel *~PK2PICC* welcher im fünften Schritt auf Seiten der Komponente COSb benötigt wird.
7. (N106.558) K\_externeWelt {K\_Karte}   
   Im fünften Schritt MUSS zum COSb ein General Authenticate-Kommando mit *~PK2PICC* aus (N106.556) gemäß 14.7.2.4.4 geschickt werden. Die Antwortdaten enthalten einen CMAC *TPCD* über *~PK2PICC* welcher im sechsten Schritt auf Seiten der Komponente COSa benötigt wird.
8. (N106.560) K\_externeWelt {K\_Karte}   
   Im sechsten Schritt MUSS zum COSa ein General Authenticate-Kommando mit *TPCD* aus (N106.558) gemäß 14.7.2.1.4 geschickt werden. Die Antwortdaten enthalten einen CMAC *TPICC* über *~PK2PCD* welcher im siebten Schritt auf Seiten der Komponente COSb benötigt wird.
9. (N106.562) K\_externeWelt {K\_Karte}   
   Im siebten Schritt MUSS zum COSb ein General Authenticate-Kommando mit *TPICC* aus (N106.560) gemäß 14.7.2.4.5 geschickt werden. Die Antwortnachricht enthält kein Datenfeld.



Abbildung 7: Sequenzdiagramm PACE Authentisierung

1. Aus Gründen der Übersichtlichkeit wurden in Abbildung 7 kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in Kapitel 14. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.
2. Aus Performanzgründen ist es für die Steuersoftware ratsam die Kommandos des ersten Schrittes zeitlich parallel an die Komponenten schicken. Analog ist es ratsam die Kommandos des zweiten Schrittes zeitlich parallel an die Komponenten zu schicken.

### Sessionkeyaushandlung mittels RSA-Schlüssel, Option\_DES

Für den Fall, dass eine gegenseitige Authentisierung mittels asymmetrischer RSA-Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 8):

1. (N106.600) K\_externeWelt {K\_Karte}, Option\_DES   
   Das erste Kommando MUSS zum COSa gesendet werden und MUSS Get Challenge gemäß 14.9.4.1 sein. Die dabei vom COSa erzeugten Antwortdaten werden mit rspData.1 bezeichnet.
2. (N106.700) K\_externeWelt {K\_Karte}, Option\_DES   
   Das zweite Kommando MUSS zum COSb gesendet werden und MUSS ein Internal Authenticate gemäß 14.7.4.1 mit *algId* gleich rsaSessionkey4SM sein. Dabei MUSS cmdData.2 = rspData.1 || ICCSN8.User gelten mit ICCSN8.User gleich *iccsn8* (siehe (N019.900)c) aus COSa. Wie die Instanz, welche die Kommando-APDU sendet, Kenntnis von ICCSN8.User erhält, ist nicht Gegenstand dieses Dokumentes. Die korrespondierenden Antwortdaten des COSb werden mit rspData.2 bezeichnet.
3. (N106.800) K\_externeWelt {K\_Karte}, Option\_DES   
   Das dritte Kommando MUSS zum COSa gesendet werden. Es MUSS ein External Authenticate gemäß 14.7.1.1 mit *algId* gleich rsaSessionkey4TC sein. Als Kommandodaten MÜSSEN rspData.2 verwendet werden.
4. (N106.900) K\_externeWelt {K\_Karte}, Option\_DES   
   Das vierte Kommando MUSS zum COSb gesendet werden und MUSS Get Challenge gemäß 14.9.4.1 sein. Die dabei vom COSb erzeugten Antwortdaten werden mit rspData.4 bezeichnet.
5. (N107.000) K\_externeWelt {K\_Karte}, Option\_DES   
   Das fünfte Kommando MUSS zum COSa gesendet werden und MUSS ein Internal Authenticate gemäß 14.7.4.1 mit *algId* gleich rsaSessionkey4TC sein. Dabei MUSS cmdData.5 = rspData.4 || ICCSN8.COS gelten mit ICCSN8.COS gleich *iccsn8* (siehe (N019.900)c) aus COSb. Wie die Instanz, welche die Kommando-APDU sendet, Kenntnis von ICCSN8.COS erhält, ist nicht Gegenstand dieses Dokumentes. Die korrespondierenden Antwortdaten vom COSa werden mit rspData.5 bezeichnet.
6. (N107.100) K\_externeWelt {K\_Karte}, Option\_DES   
   Das sechste Kommando ist das letzte der Sequenz und MUSS zum COSb gesendet werden. Es MUSS ein External Authenticate gemäß 14.7.1.1 mit *algId* gleich rsaSessionkey4SM sein. Als Kommandodaten MÜSSEN rspData.5 verwendet werden.
7. (N107.200) K\_COS\_G1, Option\_DES   
   Sowohl COSa als auch COSb KANN die geschützte Übertragung der Sequenz
   1. erlauben oder
   2. ablehnen.



Abbildung 8: Sequenzdiagramm RSA-Sessionkey-Aushandlung

1. Aus Gründen der Übersichtlichkeit wurden in Abbildung 8 kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in Kapitel 14. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.

### Sessionkeys mittels ELC-Schlüssel

1. Das hier vorgestellte Protokoll geht auf einen Vorschlag zurück, der im DIN NIA 17.4 vorgestellt wurde. Ein Sicherheitsbeweis liegt bislang nicht vor.

Beim hier beschriebenen Authentisierungsprotokoll authentisieren sich zwei Protokollpartner *A* und *B* gegenseitig mittels asymmetrischer Kryptographie, die auf elliptischen Kurven basiert und handeln dabei Sessionkeys aus. Die Sessionkeys haben die Verwendungszwecke „Secure Messaging“ und „Trusted Channel“. Das Protokoll ist so aufgebaut, dass eine erfolgreiche Nutzung der Sessionkeys nur möglich ist, wenn die Verwendungszwecke in den Protokollpartnern unterschiedlich ist. Ohne Beschränkung der Allgemeingültigkeit sei hier angenommen, dass Protokollpartner *A* die Sessionkeys anschließend für den Zweck „Secure Messaging“ nutzt. Dann ist es für eine erfolgreiche Nutzung der Sessionkeys zwingend erforderlich, dass der andere Protokollpartner *B* die Sessionkeys für den anderen Zweck „Trusted Channel“ nutzt. Das Protokoll geht von folgenden Voraussetzungen aus:

1. Beide Protokollpartner *A* und *B* verfügen über einen statischen (d. h. persistent gespeicherten) privaten Schlüssel *PrK*.
2. Die beiden privaten Schlüssel *PrK.a* und *PrK.b* verwenden dieselben Domainparameter *D*.
3. Zu beiden privaten Schlüsseln existiert jeweils ein CV-Zertifikat, welches den korrespondierenden öffentlichen Schlüssel enthält *PuK*.
4. Die öffentlichen Schlüssel *PuK.a* und *PuK.b* werden der jeweiligen Gegenseite bekanntgegeben (d. h. die CV-Zertifikate werden mittels PSO Verify Certificate importiert).
5. In der Komponente *A* wird der private Schlüssel *PrK.a* selektiert. Als *algorithmIdentifier* wird dabei elcSessionkey4SM verwendet.
6. In der Komponente *B* wird der private Schlüssel *PrK.b* selektiert. Als *algorithmIdentifier* wird dabei elcSessionkey4TC verwendet.

Im Rahmen des Authentisierungsprotokolls werden folgende Schritte durchlaufen:

1. Im ersten Schritt wird jede Komponente aufgefordert, den importierten öffentlichen Schlüssel zu selektieren und ein ephemeres Schlüsselpaar zu erzeugen. Die öffentlichen Schlüssel der ephemeren Schlüsselpaare werden aus den Komponenten exportiert. Im Einzelnen:
   1. In Komponente *A* wird
      1. *PuK.b* selektiert und
      2. das ephemere Schlüsselpaar *SK.a*, *PK.a* generiert und
      3. *PK.a* exportiert.
   2. Gleichzeitig wird in der Komponente *B*
      1. *PuK.a* selektiert und
      2. das ephemere Schlüsselpaar *SK.b*, *PK.b* generiert und
      3. *PK.b* exportiert.
2. Im zweiten Schritt werden die öffentlichen Teile der ephemeren Schlüsselpaare in die jeweils andere Komponente importiert und in beiden Komponenten werden zwei gemeinsame Geheimnisse berechnet und konkateniert. Im Einzelnen:
   1. In Komponente A wird
      1. der ephemere Schlüssel *PK.b* importiert
      2. das erste gemeinsame Geheimnis K1 = ECKA(*PrK.a*, *PK.b*, *D*) berechnet.
      3. das zweite gemeinsame Geheimnis K2 = ECKA(*SK.a*, *PuK.b*, *D*) berechnet.
      4. Komponente *A* bildet die Schlüsselableitungsdaten K = K1 || K2.
      5. Der Sicherheitszustand der Komponente *A* wird entsprechend den Informationen aus dem CV-Zertifkat zu *PuK.b* gesetzt.
      6. Mittels der Schlüsselableitungsdaten K werden Sessionkeys berechnet.
   2. Gleichzeitig wird in der Komponente *B*
      1. der ephemere Schlüssel *PK.a* importiert
      2. das erste gemeinsame Geheimnis K1 = ECKA(*SK.b*, *PuK.a*, *D*) berechnet.
      3. das zweite gemeinsame Geheimnis K2 = ECKA(*PrK.b*, *PK.a*, *D*) berechnet.
      4. Komponente *B* bildet die Schlüsselableitungsdaten K = K1 || K2.
      5. Der Sicherheitszustand der Komponente *B* wird entsprechend den Informationen aus dem CV-Zertifkat zu *PuK.a* gesetzt.
      6. Mittels der Schlüsselableitungsdaten K werden Sessionkeys berechnet.

Komponente *A* bildet K1 mittels des statischen, privaten Schlüssels *PrK.a* und des ephemeren, öffentlichen Schlüssels *PK.b*. Demgegenüber bildet Komponente *B* K1 mittels des ephemeren, privaten Schlüssels *SK.b* und des statischen, öffentlichen Schlüssels *PuK.a*. Diese Asymmetrie ist der Grund, weshalb eine erfolgreiche Etablierung von nutzbaren Sessionkeys nur möglich ist, wenn die *algorithmIdentifier*, welche *A* und *B* bei der Selektion der privaten Schlüssel übergeben wurden, anzeigen, dass die Sessionkeys in *A* und *B* unterschiedliche Verwendungszwecke haben.

Der Komponente *A* ist es nur dann mit an Sicherheit grenzender Wahrscheinlichkeit möglich dasselbe K1 wie Komponente *B* zu berechnen, wenn sie über *PrK.a* verfügt.

Der Komponente *B* ist es nur dann mit an Sicherheit grenzender Wahrscheinlichkeit möglich dasselbe K2 wie Komponente *A* zu berechnen, wenn sie über *PrK.b* verfügt.

Falls ein Angreifer *C* entweder Komponente *A* oder *B* simuliert, fände trotzdem ein vollständiger Protokolldurchlauf statt. Weder die andere Komponente noch die Steuersoftware haben die Möglichkeit festzustellen, dass anstelle einer authentischen Komponente ein Angreifer am Protokoll teilnimmt. Allerdings wird der Angreifer mit an Sicherheit grenzender Wahrscheinlichkeit andere Sessionkeys berechnen, als der authentische Protokollpartner.

Simuliert der Angreifer die Komponente *B*, so wird nach dem Protokolldurchlauf in *A* ein zu *B* gehörender Sicherheitszustand gesetzt. Da der Angreifer nicht über die korrekten Sessionkeys verfügt, wird dieser Sicherheitszustand in *A* spätestens mit dem nächsten Kommando entweder wegen (N031.600)a.1 oder wegen (N031.700)a.2.i zurückgesetzt.

Simuliert der Angreifer die Komponente *A*, so wird nach dem Protokolldurchlauf in *B* ein zu *A* gehörender Sicherheitszustand gesetzt. Gemäß der derzeitigen Spezifikationslage könnte der Angreifer dann mittels PSO Encipher beliebige Paare von „plaintext“ und „ciphertext“ erzeugen lassen, oder mittels PSO Compute Cryptographic Checksum beliebige Paare von Nachrichten und zugehörigem MAC. Nützlich wären diese Paare nicht. Deshalb wird hier aus Performanzgründen darauf verzichtet, von der Komponente *A* einen Nachweis über den Besitz der Sessionkeys zu verlangen, bevor in Komponente *B* der Sicherheitszustand gesetzt wird.

Mit an Sicherheit grenzender Wahrscheinlichkeit sind *PK.a* und *PK.b* verschieden, wenn *A* und *B* verschieden sind. Falls ein Angreifer *C* die Kommunikation so manipuliert, dass *A* identisch zu *B* ist (technisch wird dabei eine Smartcard auf unterschiedlichen logischen Kanälen angesprochen), dann ist dieser Angriff durch Vergleich von *PK.a* und *PK.b* erkennbar.



Abbildung 9: Sequenzdiagramm ELC-Sessionkey-Aushandlung

Für den Fall, dass eine gegenseitige Authentisierung mittels asymmetrischer ELC-Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 9):

1. (N107.220) K\_externeWelt {K\_Karte}   
   Im ersten Schritt MUSS zu jeder der beiden Komponenten ein General Authenticate-Kommando gemäß 14.7.2.2.1 geschickt werden. Die beiden Antwortdaten enthalten einen ephemeren, öffentlichen Schlüssel.
2. (N107.230) K\_externeWelt {K\_Karte}   
   Im zweiten und letzten Schritt dieser Sequenz MUSS zu jeder der beiden Komponente ein General Authenticate-Kommando gemäß 14.7.2.2.2 geschickt werden.
3. (N107.235) K\_COS   
   Sowohl COSa als auch COSb KANN die geschützte Übertragung der Sequenz
   1. erlauben oder
   2. ablehnen.
4. Aus Performanzgründen ist es für die Steuersoftware ratsam die Kommandos des ersten Schrittes zeitlich parallel an die Komponenten schicken. Analog ist es ratsam die Kommandos des zweiten Schrittes zeitlich parallel an die Komponenten zu schicken.

## Statisches Secure Messaging

Dieses Kapitel behandelt die Übertragung von Sessionkeys an ein COS.



Abbildung 10: Sequenzdiagramm zur Übertragung von Sessionkeys

1. Die Bedeutung der Bezeichnungen in Abbildung 10 ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß Kapitel 14.

Für den Fall, dass eine externe Entität Sessionkeys übertragen will, ist folgende Sequenz zu wählen (vergleiche Abbildung 10):

1. (N107.250) K\_externeWelt {K\_Karte}   
   Das erste Kommando der Sequenz MUSS über Channel\_a geschickt werden und ein General Authenticate Kommando
   1. gemäß 14.7.2.3.1 sein, falls der in (N085.051)b beschriebene Fall vorliegt, oder
   2. gemäß 14.7.2.5.1 sein, falls der in (N085.051)d beschriebene Fall vorliegt.
2. (N107.252) K\_externeWelt {K\_Karte}   
   Das zweite Kommando über Channel\_a ist das letzte dieser Sequenz und MUSS ein General Authenticate Kommando
   1. gemäß 14.7.2.3.2 sein, falls der in (N085.051)b beschriebene Fall vorliegt, oder
   2. gemäß 14.7.2.5.2 sein, falls der in (N085.051)d beschriebene Fall vorliegt.
3. (N107.254) K\_COS   
   Das COS KANN die geschützte Übertragung
   1. erlauben oder
   2. ablehnen.
4. Hintergründe zu dieser Art der Sessionkeyübertragung finden sich in Anhang F.

# Verschiedenes (normativ)

## Identifier

Tabelle 267: Generische AlgorithmIdentifier für Authentisierungszwecke

|  |  |  |
| --- | --- | --- |
| **Name** | **Oberbegriff für** | |
| asymClientAuthentication |  | rsaClientAuthentication |
| asymRoleAuthentication | elcRoleAuthentication | rsaRoleAuthentication |
| asymRoleCheck | elcRoleCheck | rsaRoleCheck |
| asymSessionkey4SM | elcSessionkey4SM | rsaSessionkey4SM |
| asymSessionkey4TC | elcSessionkey4TC | rsaSessionkey4TC |
| symSessionkey4SM | aesSessionkey4SM | desSessionkey4SM |
| symSessionkey4TC | aesSessionkey4TC | desSessionkey4TC |

Tabelle 268: Konkrete AlgorithmIdentifier für Authentisierungszwecke

|  |  |  |
| --- | --- | --- |
| **Name** | **Codierung** | **Verwendung** |
| aesSessionkey4SM  desSessionkey4SM | 0101 01002 = ´54´ | Symmetrisch, Mutual Authenticate  Sessionkeys für Secure Messaging |
| aesSessionkey4TC  desSessionkey4TC | 0111 01002 = ´74´ | External Authenticate, Internal Authenticate, Sessionkeys für PSO-Kommandos |
| elcAsynchronAdmin | 1111 01002 = ´F4´ | Asymmetrisch, General Authenticate,  asynchrone Administration |
| elcRoleAuthentication | 0000 00002 = ´00´ | Asymmetrisch Internal Authenticate |
| elcRoleCheck | Asymmetrisch External Authenticate |
| elcSessionkey4SM | 0101 01002 = ´54´ | Asymmetrisch, General Authenticate,  Sessionkeys für SecureMessaging |
| elcSessionkey4TC | 1101 01002 = ´D4´ | Asymmetrisch, General Authenticate,  Sessionkeys für PSO-Kommandos |
| rsaClientAuthentication | 0000 01012 = ´05´ | Internal Authenticate |
| rsaRoleAuthentication, Option\_RSA\_CVC | 0000 00002 = ´00´ | Asymmetrisch Internal Authenticate |
| rsaRoleCheck, Option\_RSA\_CVC | 0000 00002 = ´00´ | Asymmetrisch External Authenticate |
| rsaSessionkey4SM, Option\_DES | 0101 01002 = ´54´ | External Authenticate, Internal Authenticate, Sessionkeys für SecureMessaging |
| rsaSessionkey4TC, Option\_DES | 0111 01002 = ´74´ | External Authenticate, Internal Authenticate, Sessionkeys für-PSO-Kommandos |

Tabelle 269: AlgorithmIdentifier für Ver- und Entschlüsselung

|  |  |  |
| --- | --- | --- |
| **Name** | **Codierung** | **Verwendung** |
| aesSessionkey | 0000 00002 = ´00´ | PSO Decipher + PSO Encipher |
| desSessionkey, Option\_DES |
| rsaDecipherOaep | 1000 01012 = ´85´ | PSO Decipher mit RSA |
| rsaDecipherPKCS1\_V1\_5 | 1000 00012 = ´81´ |
| rsaEncipherOaep | 0000 01012 = ´05´ | PSO Encipher + Transcipher mit RSA, gleich AlgorithmIdentifier für Decipher mod 128 |
| rsaEncipherPKCS1\_V1\_5 | 0000 00012 = ´01´ |
| elcSharedSecretCalculation | 0000 10112 = ´0B´ | PSO Decipher mit ELC |

Tabelle 270: AlgorithmIdentifier für Integrität und Authentizität

|  |  |  |
| --- | --- | --- |
| **Name** | **Codierung** | **Verwendung** |
| aesSessionkey | 0000 00002 = ´00´ | PSO Compute Cryptographic Checksum +  PSO Verify Cryptographic Checksum |
| desSessionkey, Option\_DES |
| sign9796\_2\_DS2 | 0000 01112 = ´07´ | PSO Compute Digital Signature |
| signPKCS1\_V1\_5 | 0000 00102 = ´02´ | PSO Compute Digital Signature |
| signPSS | 0000 01012 = ´05´ | PSO Compute Digital Signature |
| signECDSA | 0000 00002 = ´00´ | PSO Compute Digital Signature |
| verifyCertificate | ´XX´ | PSO Verify Certificate, da dieser Identifier nie an der physikalischen Schnittstelle verwendet wird, wird er hier nicht festgelegt |

Tabelle 271: Object Identifier, alphabetisch sortiert (informativ)

| **Name und Codierung** | **Bemerkung** |
| --- | --- |
| ansix9p256r1  {1.2.840.10045.3.1.7}  ´2A8648CE3D030107´ | Domainparameter einer elliptischen Kurve gemäß [ANSI X9.62#L.6.4.3] |
| ansix9p384r1  {1.3.132.0.34}  ´2B81040022´ | Domainparameter einer elliptischen Kurve gemäß [ANSI X9.62#L.6.5.2] |
| authS\_gemSpec-COS-G2\_ecc-with-sha256  {1.3.36.3.5.3.1}  ´2B2403050301´ | öffentlicher Schlüssel in einem CV‑Zertifikat, Verwendungszweck Authentisierung, SHA\_256 |
| authS\_gemSpec-COS-G2\_ecc-with-sha384  {1.3.36.3.5.3.2}  ´2B2403050302´ | öffentlicher Schlüssel in einem CV‑Zertifikat, Verwendungszweck Authentisierung, SHA\_384 |
| authS\_gemSpec-COS-G2\_ecc-with-sha512  {1.3.36.3.5.3.3}  ´2B2403050303´ | öffentlicher Schlüssel in einem CV‑Zertifikat, Verwendungszweck Authentisierung, SHA\_512 |
| authS\_ISO9796-2Withrsa\_sha256\_mutual  {1.3.36.3.5.2.4}, Option\_RSA\_CVC  ´2B2403050204´ | authentication scheme with RSA signature and DSI according to [ISO/IEC 9796-2] and SHA-256 for mutual authentication with or without establishment of a Trusted Channel |
| brainpoolP256r1  {1.3.36.3.3.2.8.1.1.7}  ´2B2403030208010107´ | Domainparameter einer elliptischen Kurve gemäß [RFC5639#3.4] |
| brainpoolP384r1  {1.3.36.3.3.2.8.1.1.11}  ´2B240303020801010b´ | Domainparameter einer elliptischen Kurve gemäß [RFC5639#3.6] |
| brainpoolP512r1  {1.3.36.3.3.2.8.1.1.13}  ´2B240303020801010d´ | Domainparameter einer elliptischen Kurve gemäß [RFC5639#3.7] |
| ecdsa-with-SHA256  {1.2.840.10045.4.3.2}  ´2A8648CE3D040302´ | öffentlicher Schlüssel in einem CV‑Zertifikat,  Verwendungszweck Prüfung von CVC, SHA\_256 |
| ecdsa-with-SHA384  {1.2.840.10045.4.3.3}  ´2A8648CE3D040303´ | öffentlicher Schlüssel in einem CV‑Zertifikat, Verwendungszweck Prüfung von CVC, SHA\_384 |
| ecdsa-with-SHA512  {1.2.840.10045.4.3.4}  ´2A8648CE3D040304´ | öffentlicher Schlüssel in einem CV‑Zertifikat, Verwendungszweck Prüfung von CVC, SHA\_512 |
| id-ELC-shared-secret-calculation | Verschlüsselung gemäß (N004.500) |
| id-PACE-ECDH-GM-AES-CBC-CMAC-128  {0.4.0.127.0.7.2.2.4.2.2}  ´04007f00070202040202´ | [BSI-TR-03110-3] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys für Secure Messaging verwendet werden. |
| id-PACE-ECDH-GM-AES-CBC-CMAC-192  {0.4.0.127.0.7.2.2.4.2.3}  ´04007f00070202040203´ | [BSI-TR-03110-3] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys für Secure Messaging verwendet werden. |
| id-PACE-ECDH-GM-AES-CBC-CMAC-256  {0.4.0.127.0.7.2.2.4.2.4}  ´04007f00070202040204´ | [BSI-TR-03110-3] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys für Secure Messaging verwendet werden. |
| id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128  {0.4.0.127.0.7.2.3.4.2.2}  ´04007f00070203040202´ | [BSI-TR-03110-3] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys im Rahmen von PSO-Kommandos verwendet werden. |
| id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192  {0.4.0.127.0.7.2.3.4.2.3}  ´04007f00070203040203´ | [BSI-TR-03110-3] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys im Rahmen von PSO-Kommandos verwendet werden. |
| id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256  {0.4.0.127.0.7.2.3.4.2.4}  ´04007f00070203040204´ | [BSI-TR-03110-3] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys im Rahmen von PSO-Kommandos verwendet werden. |
| id-RSAES-OAEP  {1.2.840.113549.1.1.7}  ´2A864886F70d010107´ | Verschlüsselung gemäß [PKCS#1] Kapitel 7.1.1 |
| oid\_cvc\_fl\_cms  {1.2.276.0.76.4.153}  ´2A8214004C048119´ | Werte gemäß [gemSpec\_OID#Tab\_PKI\_408], Interpretation der *flagList* gemäß [gemSpec\_PKI#Tab\_PKI\_911] |
| oid\_cvc\_fl\_ti  {1.2.276.0.76.4.152}  ´2A8214004C048118´ | Werte gemäß [gemSpec\_OID#Tab\_PKI\_408], Interpretation der *flagList* gemäß [gemSpec\_PKI#Tab\_PKI\_910] |
| rsaEncryption  {1.2.840.113549.1.1.1}  ´2A864886F70d010101´ | Verschlüsselung gemäß [PKCS#1] Kapitel 7.2.1 |
| secp256r1 | identisch zu ansix9p256r1, siehe dort |
| secp384r1 | identisch zu ansix9p384r1, siehe dort |
| sigS\_ISO9796-2Withrsa\_sha256  {1.3.36.3.4.2.2.4}, Option\_RSA\_CVC  ´2B240304020204´ | signature scheme with RSA signature and DSI according to [ISO/IEC 9796-2] and SHA-256 |

## Codierungen für Trailer

Tabelle 272: Trailer 🡪 Fehlername

| **Wert** | **Name** | **Bedeutung** |
| --- | --- | --- |
| ´62 00´ | DataTruncated | Antwortdaten unvollständig |
| ´62 81´ | CorruptDataWarning | Die Integrität der Antwortdaten ist nicht gewährleistet |
| ´62 82´ | EndOfFileWarning | Es wurden mehr Daten angefordert als die Datei enthält |
| ´62 82´ | EndOfRecordWarning | Es wurden mehr Daten angefordert als der Rekord enthält |
| ´62 82´ | UnsuccessfulSearch | Pattern wurde in keinem der adressierten Rekords gefunden |
| ´62 83´ | FileDeactivated | File, auf welches sich die Operation bezieht, ist deaktiviert |
| ´62 85´ | FileTerminated | File, auf welches sich die Operation bezieht, ist terminiert |
| ´62 87´ | RecordDeactivated | Rekord, auf welchen sich Operation bezieht, ist deaktiviert |
| ´62 Cx´ | TransportStatus | Indikation des Transportschutzverfahrens, siehe Tabelle 144 |
| ´62 D0´ | PasswordDisabled | Passwortobjekt ausgeschaltet, Verifikation nicht erforderlich |
| ´63 00´ | AuthenticationFailure | Authentisierung fehlgeschlagen |
| ´63 CF´ | NoAuthentication | Keine Authentisierung mit dem referenzierten Schlüssel |
| ´63 Cx´ | RetryCounter | Wert des Fehlbedienungszählers |
| ´63 Cx´ | UpdateRetryWarning | Schreibschwierigkeiten |
| ´63 Cx´ | WrongSecretWarning | Falsches Passwort in den Kommandodaten |
| ´64 00´ | EncipherError | Fehlerhafte Verschlüsselungsoperation |
| ´64 00´ | KeyInvalid | Schlüsseldaten fehlen, Generierung erforderlich |
| ´64 00´ | ObjectTerminated | Objekt befindet sich im Zustand „Termination state" |
| ´64 00´ | ParameterMismatch | Domainparameter passen nicht zusammen |
| ´65 81´ | MemoryFailure | Schreibfehler |
| ´67 00´ | WrongRecordLength | Falsche Rekordlänge |
| ´68 81´ | ChannelClosed | Logischer Kanal nicht geöffnet |
| ´69 81´ | NoMoreChannelsAvailable | kein weiterer logischer Kanal verfügbar |
| ´69 81´ | VolatileKeyWithoutLCS | volatile Schlüssel werden vom Kommando nicht unterstützt |
| ´69 81´ | WrongFileType | Datei unterstützt das aktuelle Kommando nicht |
| ´69 82´ | SecurityStatusNotSatisfied | Zugriffsregel nicht erfüllt |
| ´69 83´ | CommandBlocked | Rücksetzen des Fehlbedienungszählers nicht mehr möglich |
| ´69 83´ | KeyExpired | Der Gültigkeitsbereich des Schlüssels ist abgelaufen |
| ´69 83´ | PasswordBlocked | Fehlbedienungszähler abgelaufen |
| ´69 85´ | KeyAlreadyPresent | Schlüsseldaten bereits gesetzt, Generierung unmöglich |
| ´69 85´ | LongPassword | Neues Passwort zu lang |
| ´69 85´ | NoKeyReference | Schlüsselreferenz fehlt, MSE-Set ist notwendig |
| ´69 85´ | NoPrkReference | Schlüsselreferenz fehlt, MSE-Set ist notwendig |
| ´69 85´ | NoPukReference | Schlüsselreferenz fehlt, MSE-Set ist notwendig |
| ´69 85´ | NoRandom | Keine Zufallszahl, Get Challenge ist notwendig |
| ´69 85´ | NoRecordLifeCycleStatus | Datei unterstützt das aktuelle Kommando nicht |
| ´69 85´ | PasswordNotUsable | Transportschutz aktiv, CHANGE REF. DATA notwendig |
| ´69 85´ | WrongRandomLength | Zufallszahl hat falsche Länge, Get Challenge erforderlich |
| ´69 85´ | ShortPassword | Neues Passwort zu kurz |
| ´69 86´ | NoCurrentEF | Kommandobearbeitung unmöglich, da keine Datei selektiert |
| ´69 88´ | IncorrectSmDo | Fehlerhaftes Secure Messaging |
| ´6A 80´ | NewFileSizeWrong | *newFileSize* kein Vielfaches der Rekordlänge |
| ´6A 80´ | NumberPreconditionWrong | Vorbedingung zum Laden des Scenarios nicht erfüllt |
| ´6A 80´ | NumberScenarioWrong | Scenario wurde bereits geladen |
| ´6A 80´ | VerificationError | Fehlerhaftes CV-Zertifikat |
| ´6A 80´ | WrongCiphertext | Fehlerhaftes Chiffrat |
| `6A 80´ | WrongToken | Token ist fehlerhaft |
| ´6A 81´ | UnsupportedFunction | Schlüssel unterstützt den angegeben Algorithmus nicht |
| ´6A 82´ | FileNotFound | Referenzierte Datei nicht gefunden |
| ´6A 83´ | RecordNotFound | Referenzierter Rekord nicht verwendbar |
| ´6A 84´ | DataTooBig | Zu viele Daten |
| ´6A 84´ | FullRecordList | Rekordliste bereits komplett gefüllt |
| ´6A 84´ | MessageTooLong | Klartext zu lang für Verschlüsselung |
| ´6A 84´ | OutOfMemory | Zu wenig Speicherplatz |
| ´6A 88´ | InconsistentKeyReference | Schlüsselreferenz im CV-Zertifikat fehlerhaft |
| ´6A 88´ | KeyNotFound | Referenzierten Schlüssel nicht gefunden |
| ´6A 88´ | PasswordNotFound | Referenziertes Passwort nicht gefunden |
| ´6A 88´ | PrKNotFound | Referenzierten Schlüssel nicht gefunden |
| ´6A 88´ | PukNotFound | Referenzierten Schlüssel nicht gefunden |
| ´6A 89´ | DuplicatedObject | Neu anzulegendes Objekt existiert bereits |
| ´6A 8A´ | DfNameExists | Neu anzulegende Applikation existiert bereits |
| ´6B 00´ | OffsetTooBig | Offset zu groß |
| ´6D 00´ | InstructionNotSupported | Der im INS-Byte angezeigte Befehl wird nicht unterstützt |
| ´90 00´ | NoError | Normale Kommandoausführung, kein Fehler, keine Warnung |

# Anhang A Hinweise zur Sicherheitsevaluierung (informativ)

Der Hauptteil des Dokumentes enthält im Wesentlichen funktionale Anforderungen. Darüber hinaus sind bei der COS-Entwicklung auch Sicherheitsanforderungen zu beachten. Grundsätzlich sind Sicherheitsanforderungen in einem Protection Profile oder Security Target zu finden. In diesem Anhang werden Aspekte aufgelistet, die im Rahmen einer Evaluierung zu prüfen sind. Die hier genannte Liste erhebt keinen Anspruch auf Vollständigkeit.

1. Gemäß (N034.400) ist es funktional zulässig, dass ein COS das Kommando Get Challenge auch für andere, insbesondere sehr kleine Werte für Ne unterstützt. Im Rahmen einer Sicherheitsbetrachtung ist dann nachzuweisen, dass die in *RND.ICC* gespeicherte Zufallszahl nicht weniger zufällig erzeugte Bits enthält, als die in 14.9.4 behandelten Varianten.
2. Dieses Dokument beschreibt in (N001.200) lediglich die funktionalen Anforderungen an einen Zufallszahlengenerator. Die dort erzeugten Zufallszahlen werden für diverse Zwecke eingesetzt. Im Rahmen einer Sicherheitsbetrachtung ist nachzuweisen, dass die Zufallszahlen die in [BSI-TR-03116#3.4 PTG.2] geforderte Qualität aufweisen.
3. Dieses Dokument beschreibt in 14.9.2 lediglich die funktionalen Anforderungen an die Fingerprintberechnung. Im Rahmen einer Sicherheitsbetrachtung ist nachzuweisen, dass
   1. der Repräsentant *M* des COS tatsächlich alle Bestandteile des COS umfasst.
   2. bei der Berechnung des Fingerprints gemäß (N096.478) keine Informationen über den Objektcode des COS über Seitenkanäle die Smartcard verlassen.

# Anhang B – Vorgaben zur Performanz

## Einführung (informativ)

Die Akzeptanz der Smartcards im Gesundheitswesen und der damit verbundenen Abläufe in den medizinischen Einrichtungen (Praxis, Apotheke, Krankenhaus, …) hängt stark von den Zeiten ab, die für einzelne Aktionen benötigt werden. Um im Gesamtsystem zu akzeptablen Werten zu kommen, sind für die einzelnen Komponenten Vorgaben zu treffen, die als zulassungsrelevant in die jeweiligen Spezifikationen aufgenommen werden. In diesem Dokument werden entsprechende Vorgaben für Smartcards im Gesundheitswesen festgelegt.

Es werden sowohl die Randbedingungen als auch die Messparameter festgelegt, um reproduzierbare und aussagekräftige Messungen zu ermöglichen. Es werden Vorgaben für verschiedene Szenarien getroffen. Die ermittelten Messwerte werden über ein Punktesystem bewertet. Die Gesamtpunktzahl setzt sich aus einer gewichteten Addition aller Einzelpunktzahlen zusammen. Das COS wird nur dann von der gematik zugelassen, wenn es eine definierte Mindestpunktzahl (siehe (N109.700)) erreicht.

## Messaufbau (normativ)

Die Bearbeitungszeiten eines Kommandos werden mit dem im Folgenden beschriebenen Messaufbau durchgeführt:



Abbildung 11: Komponentendiagramm Performanzmessplatz

Die Komponente **Steuersoftware** führt die Performanzmessung durch und protokolliert dabei die Messergebnisse. Diese Komponente besitzt logisch gesehen zwei Kommunikationskanäle zum Interface Device (IFD, Kartenleser). Der eine Kanal transportiert Kommando und Antwort-APDUs, die vom IFD zum Prüfling weitergeleitet werden. Der zweite Kanal dient der Steuerung des IFD sowie dem Auslesen der Laufzeiten, die vom IFD gemessen werden.

Die Komponente **Interface Device (IFD)** transferiert APDUs gemäß der elektrischen Schnittstelle vom und zum Prüfling. Dabei ermittelt das IFD die Bearbeitungszeit eines Kommando-Antwort-APDU-Paares. Wichtig ist dabei, dass das IFD die maximale Übertragungskapazität des Prüflings unterstützt (siehe B.4).

Die Komponente **„Device under Test“ (DuT)** repräsentiert den Prüfling, dessen Performanz zu ermitteln ist.

## Anforderungen an die Steuersoftware (normativ)

1. (N107.300) K\_externeWelt {K\_Karte}   
   Die Steuersoftware MUSS in der Lage sein, die einzelnen Schritte der Performanzmessung auszuführen, zu protokollieren und das Gesamtergebnis der Messung aus den einzelnen Resultaten zu ermitteln.

## Anforderungen an das Interface Device (IFD) (normativ)

1. (N107.350) K\_externeWelt {K\_Karte}   
   Bezüglich der Schnittstelle des Interface Devices IFD zum Prüfling DuT gilt:
   1. Das IFD MUSS das Übertragungsprotokoll T=1 gemäß 11.2.1 unterstützen.
   2. Das IFD SOLL das Übertragungsprotokoll [ISO/IEC 7816-12] gemäß 11.2.2 unterstützen.
   3. Das IFD SOLL das kontaktlose Übertragungsprotokoll gemäß 11.2.3 unterstützen.

### Anforderungen an das IFD bezüglich T=1

Dieses Unterkapitel enthält die Anforderungen an das IFD bezüglich des Übertragungsprotokolls T=1 gemäß 11.2.1.

1. (N107.400) K\_externeWelt {K\_Karte}   
   Der Übertragungskanal TPDU\_Channel des IFD SOLL die maximale Übertragungskapazität des Prüflings DuT unterstützen. Im Einzelnen bedeutet das:
2. (N107.500) K\_externeWelt {K\_Karte}   
   Das IFD MUSS im Rahmen einer PPS-Sequenz alle Werte für PPS1 aus der Menge {´18´, ´95´, ´96´, ´97´} unterstützen.

Tabelle 273: Bedeutung PPS1 gemäß [ISO/IEC 7816-3#Table 7 und 8]

|  |  |  |  |
| --- | --- | --- | --- |
| **PPS1** | **Teilerfaktor** | **fmax / [MHz]** | **C / [kBaud]** |
| ´18´ | 372/12 = 31 | 5 | 161 |
| ´95´ | 512/16 = 32 | 5 | 156 |
| ´96´ | 512/32 = 16 | 5 | 313 |
| ´97´ | 512/64 = 8 | 5 | 625 |

1. (N107.600) K\_externeWelt {K\_Karte}   
   Das IFD MUSS eine Versorgungsspannung *Ucc* gemäß [ISO/IEC 7816-3#Table 1] für Class A und Class B liefern.
2. (N107.700) K\_externeWelt {K\_Karte}   
   Das IFD MUSS einen Versorgungsstrom *Icc* gemäß [ISO/IEC 7816-3#Table 1] für Class A und Class B liefern.
3. (N107.800) K\_externeWelt {K\_Karte}   
   Das IFD MUSS während der Performanzmessung ein Clocksignal mit einer Frequenz aus dem Intervall [4,95, 5,05] MHz liefern.
4. (N107.900) K\_externeWelt {K\_Karte}   
   Das IFD MUSS „direct convention“ (siehe [ISO/IEC 7816-3#8.1]) unterstützen.
5. (N108.000) K\_externeWelt {K\_Karte}   
   Falls das Byte TC1 im ATR fehlt, dann MUSS das IFD beim Senden eine „extra guard time“ von 12 etu verwenden (siehe [ISO/IEC 7816-3#8.3]).
6. (N108.100) K\_externeWelt {K\_Karte}   
   Falls das Byte TC1 im ATR vorhanden ist, dann MUSS das IFD beim Senden eine „extra guard time“ von 11 etu verwenden (siehe [ISO/IEC 7816-3#8.3 und 11.2]).
7. (N108.200) K\_externeWelt {K\_Karte}   
   Das IFD MUSS beim Empfang eine „character guard time“ von 11 etu unterstützen (siehe [ISO/IEC 7816-3#11.2]).
8. (N108.300) K\_externeWelt {K\_Karte}   
   Das IFD MUSS *BGT* = 22 etu verwenden (siehe [ISO/IEC 7816-3#11.2]).
9. (N108.400) K\_externeWelt {K\_Karte}   
   Das IFD MUSS für IFSC einen Wert von 254 unterstützen (siehe [ISO/IEC 7816-3#11.4.2]).
10. (N108.500) K\_externeWelt {K\_Karte}   
    Das IFD MUSS für IFSD einen Wert von 254 unterstützen (siehe [ISO/IEC 7816-3#11.4.2]).
11. (N108.600) K\_externeWelt {K\_Karte}   
    Das IFD
    1. MUSS für eine Kommandonachricht 4096 und SOLL 32.768 als APDU Länge unterstützen.
    2. MUSS für eine Antwortnachricht 65.638 als APDU Länge unterstützen.

### Anforderungen an das IFD für [ISO/IEC 7816-12] Datenübertragung

Dieses Unterkapitel enthält die Anforderungen an das IFD des Performanztest bezüglich einer Datenübertragung gemäß [ISO/IEC 7816-12].

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

### Anforderungen an das IFD bezüglich kontaktloser Datenübertragung

Dieses Unterkapitel enthält die Anforderungen an das IFD des Performanztest bezüglich des kontaktlosen Übertragungsprotokolls gemäß 11.2.3.

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

## Allgemeines (normativ)

### Normale Zeitmessung

#### Normale Zeitmessung für das Übertragungsprotokoll T=1

Hier wird die standardmäßig verwendete Zeitmessung für das Übertragungsprotokoll T=1 beschrieben. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

1. (N108.700) K\_externeWelt {K\_Karte}   
   Die normale Zeitmessung MUSS die Zeitspanne *tRun* = *tEnd* – *tStart* – *tIO* ermitteln, wobei folgende Definitionen gelten:
   1. Der Zeitpunkt *tStart* ist durch den Beginn des ersten Startbits des ersten Characters (siehe [ISO/IEC 7816-3#Figure 7]) der ersten TPDU (siehe [ISO/IEC 7816-3#Figure 23]) gekennzeichnet, welche zur Übertragung einer Kommando-APDU verwendet wird.
   2. Der Zeitpunkt *tEnd* ist durch das Ende des letzten Paritybits des letzten Characters der letzten TPDU gekennzeichnet, welche zur Übertragung der zugehörigen Antwort-APDU verwendet wird.
   3. Die Zeitspanne *tIO* berücksichtigt die Übertragungszeit von Kommando und Antwort-APDU. Sie MUSS wie folgt berechnet werden:
      1. Sei *Ncmd\_APDU* die Anzahl Oktette in der Kommando-APDU, dann gilt für die Anzahl übertragener Oktette *Ncmd\_TPDU* auf TPDU-Ebene falls *Ncmd\_APDU* größer gleich 1 ist und das Epilogue Field einen „longitudinal redundance code“ enthält (siehe [ISO/IEC 7816-3#11.3.4]):   
          *Ncmd\_TPDU* = *Ncmd\_APDU* + 8 ceiling( *Ncmd\_APDU* / 254 ) – 4.
      2. Für die Anzahl *Ncmd\_etu* folgt unter der Vorraussetzung, dass pro Oktett 11 etu benötigt werden und eine „block guard time“ von 22 etu berücksichtigt wird (siehe [ISO/IEC 7816-3#11.2]):   
          *Ncmd\_etu* = 11 *Ncmd\_TPDU* + 22 ceiling( *Ncmd\_APDU* / 254 ).
      3. Analog gilt für die Antwort-APDU:   
           *Nrsp\_TPDU* = *Nrsp\_APDU* + 8 ceiling( *Nrsp\_APDU* / 254 ) – 4   
         und *Nrsp\_etu* = 11 *Nrsp\_TPDU* + 22 floor( *Nrsp\_APDU* / 254 ).
      4. Daraus folgt: *tIO* = ( *Ncmd\_etu* + *Nrsp\_etu* ) Teilerfaktor / *fmax* mit Teilerfaktor und *fmax* in Abhängigkeit von PPS1 aus Tabelle 273.

#### Normale Zeitmessung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12]

Hier wird die standardmäßig verwendete Zeitmessung für Übertragungsprotokoll gemäß [ISO/IEC 7816-12] beschrieben. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

#### Normale Zeitmessung für die kontaktlose Datenübertragung

Hier wird die standardmäßig verwendete Zeitmessung für die kontaktlose Übertragungsprotokoll beschrieben. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

### Reguläre Aktivierung der Smartcard

#### Reguläre Aktivierung für das Übertragungsprotokoll T=1

Hier wird die standardmäßige Aktivierung der Karte für das Übertragungsprotokoll T=1 beschrieben, die der Testvorbereitung dient. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

1. (N108.800) K\_externeWelt {K\_Karte}   
   Schritt 1: Im Rahmen der regulären Aktivierung der Smartcard MUSS zunächst eine Aktivierung gemäß (N023.920)a und (N023.920)b erfolgen.
2. (N108.900) K\_externeWelt {K\_Karte}   
   Schritt 2: Es MUSS eine PPS-Sequenz gemäß [ISO/IEC 7816-3] erfolgen. Die Bits b4 bis b1 in PPS0 MÜSSEN das Übertragungsprotokoll T=1 anzeigen. Als PPS1 MUSS der Wert von TA1 aus dem ATR verwendet werden.
3. (N109.000) K\_externeWelt {K\_Karte}   
   Schritt 3: Das IFD MUSS der Smartcard den Wert von IFSD = 254 präsentieren.
4. (N109.100) K\_externeWelt {K\_Karte}   
   Schritt 4: Es MUSS eine beliebige Kommando-APDU bearbeitet werden. Einzige Anforderung an diese Kommando-APDU ist, dass sie so gewählt werden MUSS, dass der Trailer NoError anzeigt und der auszuführende Testfall nicht beeinflusst wird. Hier SOLL eine Selektion von *root* gemäß 14.2.6.1 verwendet werden.
5. Die hier beschriebene reguläre Aktivierung hat das Ziel, die Smartcard vollständig zu booten. Typischerweise sind bei aktuellen Smartcard-Betriebssystemen die Initialisierungen so umfangreich, dass sie nicht vollständig innerhalb der Sendezeit des ATR ausführbar sind. Deshalb wird der Smartcard durch das abschließende Kommando genügend Zeit zur Verfügung gestellt, die typischerweise nicht relevant für die Performanzmessung ist.

#### Reguläre Aktivierung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12]

Hier wird die standardmäßige Aktivierung der Karte für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12] beschrieben, die der Testvorbereitung dient. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

#### Reguläre Aktivierung für die kontaktlose Datenübertragung

Hier wird die standardmäßige Aktivierung der Karte für das kontaktlose Übertragungsprotokoll beschrieben, die der Testvorbereitung dient. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

### Punkteermittlung

|  |  |  |
| --- | --- | --- |
| Input: | t | n-Tupel mit Zeiten von n Einzelmessungen |
| TR | Referenzzeit |
| Output: | P | Gewichtete Qualität der Messzeit |
| Errors: | – | Keine |
| Notation: |  | P = points( t, TR ) |

Es gelten folgende Definitionen:   
  *X* = Mittelwert (Erwartungswert) aller Einzelmessungen im *n*-Tupel *t*.   
  σ = Standardabweichung aller Einzelmessungen im *n*-Tupel *t*.

1. (N109.200) K\_externeWelt {K\_Karte}   
   Es gilt *f*1 .
2. (N109.300) K\_externeWelt {K\_Karte}   
   Es gilt *f2* .
3. (N109.400) K\_externeWelt {K\_Karte}   
   Es gilt *P* .

Die Funktion *P* besteht aus den Faktoren *f*1 und *f*2 und wird durch *TR* gewichtet.

Falls die Standardabweichung σ null ist, dann ist der Faktor *f*1 exakt eins. Größere Standardabweichungen führen zu kleineren Faktoren und damit zu geringeren Punktzahlen *P*.

Der Faktor *f*2 setzt den Mittelwert *X* in Relation zur Referenzzeit *TR*. Deshalb wird *f*2 als Funktion von *X* aufgefasst, die durch *TR* parametrisiert wird. Trivialerweise ist es wünschenswert, dass ein kleinerer Wert von *X* zu einem größeren Wert von *f*2 führt. Daraus folgt, dass *f*2 streng monoton fallend ist, mithin also die Ableitung von *f*2 nach *X* kleiner null ist. Zudem ist es ratsam, auch die zweite Ableitung von *f*2 nach *X* kleiner gleich null zu wählen, weil es dann eher lohnt, schlechte Mittelwerte zu verbessern, als gute Mittelwerte weiter zu optimieren.

Falls der Mittelwert *X* der Messergebnisse gleich dem Referenzwert *TR* ist, dann ist der Faktor *f*2 gleich 0,75.

Das Produkt aus *f*1 und *f*2 wird zur Ermittlung des Wertes *P* mit der Referenzzeit *TR* gewichtet. Durch diese Gewichtung korreliert das Gewicht eines Prüfpunktes mit dem Beitrag des Prüfpunkt im Rahmen zusammengesetzter Kommandosequenzen.

### Gesamtbewertung

Die Gesamtbewertung wird durch gewichtetes Addieren der Einzelpunktzahlen ermittelt. Die Gewichte korrelieren mit der Häufigkeit des Auftretens der Einzelpositionen im Feld.

Tabelle 274: Gesamtbewertung für das Basisbetriebssystem

| **Prüfpunkt** |  |  |  | ***TRi* / [ms]** | ***gi*** | ***gi TRi* / [ms]** |
| --- | --- | --- | --- | --- | --- | --- |
| Kanalkapazität | (N024.100) | B.6.1 | *PIO* | 17 | 2.000 | 34.000 |
| Karte starten | (N023.920) | B.7 | *PKarteStarten* | 60 | 500 | 30.000 |
| Activate Datei | 14.2.1.1 | B.8.1 | *Pactivate\_EF* | 30 | 1 | 30 |
| Activate Ordner |  |  | *Pactivate\_DF* | 30 | 1 | 30 |
| Activate private Key | 14.2.1.2 |  | *Pactivate\_PrK* | 30 | 1 | 30 |
| Activate sym. Key |  |  | *Pactivate\_SK* | 30 | 1 | 30 |
| Activate PuK | 14.2.1.3 |  | *Pactivate\_PuK* | 30 | 1 | 30 |
| Activate Pwd | 14.2.1.4 |  | *Pactivate\_Pwd* | 30 | 1 | 30 |
| Deactivate Datei | 14.2.3.1 |  | *Pdeactivate\_EF* | 30 | 1 | 30 |
| Deactivate Ordner |  |  | *Pdeactivate\_DF* | 30 | 1 | 30 |
| Deactivate private Key | 14.2.3.2 |  | *Pdeactivate\_PrK* | 30 | 1 | 30 |
| Deactivate sym. Key |  |  | *Pdeactivate\_SK* | 30 | 1 | 30 |
| Deactivate PuK | 14.2.3.3 |  | *Pdeactivate\_PuK* | 30 | 1 | 30 |
| Deactivate Pwd | 14.2.3.4 |  | *Pdeactivate\_Pwd* | 30 | 1 | 30 |
| Delete Datei | 14.2.4.1 |  | *Pdelete\_EF* | 100 | 1 | 100 |
| Delete Ordner |  |  | *Pdelete\_DF* | 500 | 1 | 500 |
| Delete private Key | 14.2.4.2 |  | *Pdelete\_PrK* | 60 | 1 | 60 |
| Delete sym. Key |  |  | *Pdelete\_SK* | 60 | 1 | 60 |
| Delete PuK | 14.2.4.3 |  | *Pdelete\_PuK* | 60 | 1 | 60 |
| Delete Pwd | 14.2.4.4 |  | *Pdelete\_Pwd* | 60 | 1 | 60 |
| Load Application | 14.2.5 |  | *PLoadApp* | 200 | 1 | 200 |
| Select Datei | 14.2.6.13 | B.8.2 | *Pselect\_EF* | 10 | 800 | 8.000 |
| Select Ordner | 14.2.6.9 | B.8.17 | *Pselect\_DF* | 10 | 350 | 3.500 |
| Terminate Card Usage | 14.2.7.1 | B.8.4 | *PterminateCard* | 30 | 1 | 30 |
| Terminate DF | 14.2.8.1 | B.8.1 | *Pterminate\_DF* | 30 | 1 | 30 |
| Terminate EF | 14.2.9.1 |  | *Pterminate\_EF* | 30 | 1 | 30 |
| Terminate privateKey | 14.2.9.2 |  | *Pterminate\_PrK* | 30 | 1 | 30 |
| Terminate sym. Key |  |  | *Pterminate\_SK* | 30 | 1 | 30 |
| TerminatePuK | 14.2.9.3 |  | *Pterminate\_PuK* | 30 | 1 | 30 |
| Terminate Pwd | 14.2.9.4 |  | *Pterminate\_Pwd* | 30 | 1 | 30 |
| Set Logical Eof | 14.3.4.1 | B.8.5 | *PSetEOF* | 600 | 1 | 600 |
| Erase Binary wipe | 14.3.1.1 | B.8.6 | *PWipeBin* | 600 | 1 | 600 |
| Read Binary *b* | 14.3.2.1 | B.8.7 | *PReadBinary,b* | 18 | 400 | 7.200 |
| Read Binary *m* |  |  | *PReadBinary,m* | 60 | 400 | 24.000 |
| Update Binary *b* | 14.3.5.1 | B.8.6 | *PUpdateBin,b* | 50 | 200 | 10.000 |
| Update Binary *m* |  |  | *PUpdateBin,m* | 70 | 200 | 14.000 |
| Write Binary *b* | 14.3.6.1 | B.8.5 | *Pwirte,b* | 50 | 1 | 50 |
| Write Binary *m* |  |  | *Pwirte,m* | 70 | 1 | 70 |
| Activate Record | 14.4.1.1 | B.8.8 | *PActivateRec* | 30 | 1 | 30 |
| Append Record *b* | 14.4.2.1 |  | *PAppendRecord,b* | 70 | 200 | 14.000 |
| Append Record *m* |  |  | *PAppendRecord,m* | 50 | 200 | 10.000 |
| Deactivate Record | 14.4.3.1 |  | *PDeactivateRec* | 30 | 1 | 30 |
| Delete Record | 14.4.4.1 |  | *PDeleteRec* | 80 | 1 | 80 |
| Erase Record wipe *b* | 14.4.5.1 |  | *PWipeRecord,b* | 40 | 1 | 40 |
| Erase Record wipe *m* |  |  | *PWipeRecord,m* | 80 | 1 | 80 |
| Read Record *b* | 14.4.6.1 |  | *PReadRecord,b* | 18 | 400 | 7.200 |
| Read Record *m* |  |  | *PReadRecord,m* | 60 | 400 | 24.000 |
| Search Record | 14.4.7.1 |  | *PSearchRec* | 120 | 1 | 120 |
| Update Record *b* | 14.4.8.1 |  | *PUpdateRecord,b* | 70 | 100 | 7.000 |
| Update Record *m* |  |  | *PUpdateRecord,m* | 50 | 100 | 5.000 |
| Change Reference Data | 14.6.1.1 | B.8.1 | *PChRefData* | 90 | 1 | 90 |
|  | 14.6.1.2 |  | *PSetPIN* | 50 | 1 | 50 |
| DISABLE Ver. Req. | 14.6.2.1 |  | *PDisablePIN* | 50 | 1 | 50 |
| ENABLE Ver. Req. | 14.6.3.1 |  | *PEnablePIN* | 50 | 1 | 50 |
| Get Pin Status | 14.6.4.1 |  | *PGetPinStatus* | 10 | 1 | 10 |
| Reset Retry Counter | 14.6.5.1 |  | *PResetRC* | 70 | 5 | 350 |
| Verify | 14.6.6.1 |  | *PVerify* | 15 | 5 | 75 |
| External Authenticate | 14.7.1.1 | B.8.11.4 |  | 0 | 0 | 0 |
|  |  | B.8.11.1 | *PRoleCheck,ELC256* | 150 | 300 | 45.000 |
|  |  | B.8.11.2 | *PRoleCheck,ELC384* | 200 | 100 | 20.000 |
|  |  | B.8.11.3 | *PRoleCheck,ELC512* | 300 | 50 | 15.000 |
| Get Security Status Key | 14.7.3.1 | B.8.10 |  | 0 | 0 | 0 |
|  | 14.7.3.2 | B.8.11.4 |  | 0 | 0 | 0 |
|  | 14.7.3.3 | B.8.11.1 | *PSesKey,ELC256* | 10 | 3 | 30 |
| Internal Authenticate | 14.7.4.1 | B.8.12 | *PRoleAuth,ELC256* | 120 | 300 | 36.000 |
|  |  |  | *PRoleAuth,ELC384* | 190 | 100 | 19.000 |
|  |  |  | *PRoleAuth,ELC512* | 280 | 50 | 14.000 |
|  |  |  |  | 0 | 0 | 0 |
| Sessionkeyaushandlung für Secure Messaging | 15.4.1 | B.8.10 | *PSK4SM,AES128* | 140 | 200 | 28.000 |
|  |  |  | *PSK4SM,AES192* | 160 | 50 | 8.000 |
|  |  |  | *PSK4SM,AES25* | 180 | 10 | 1.800 |
|  |  |  |  | 0 | 0 | 0 |
|  | 15.4.4 | B.8.11.1 | *PSesKey,ELC256* | 400 | 210 | 84.000 |
|  |  | B.8.11.2 | *PSesKey,ELC384* | 600 | 30 | 18.000 |
|  |  | B.8.11.3 | *PSesKey,ELC512* | 900 | 15 | 13.500 |
| PSO Compute Digital Signature | 14.8.2.1 | B.8.13 | *PsignPSS,2048* | 270 | 100 | 27.000 |
|  |  |  | *PsignPSS,3072* | 1.000 | 25 | 25.000 |
|  |  | B.8.14 | *PsignECDSA,256* | 100 | 100 | 10.000 |
|  |  |  | *PsignECDSA,384* | 180 | 25 | 4.500 |
|  |  |  | *PsignECDSA,512* | 280 | 10 | 2.800 |
| PSO Decipher | 14.8.3.1 | B.8.15 | *Pdec,2048* | 250 | 60 | 15.000 |
|  |  |  | *Pdec,3072* | 1.100 | 30 | 33.000 |
|  | 14.8.3.2 | B.8.16 | *Pdec,256* | 160 | 200 | 32.000 |
|  |  |  | *Pdec,384* | 230 | 50 | 11.500 |
|  |  |  | *Pdec,512* | 260 | 25 | 6.500 |
| PSO Encipher | 14.8.4.1 | B.8.15 | *Penc,2048* | 50 | 2,5 | 125 |
|  | 14.8.4.2 | B.8.16 | *Penc,256* | 270 | 200 | 54.000 |
|  |  |  | *Penc,384* | 430 | 50 | 21.500 |
|  |  |  | *Penc,512* | 650 | 25 | 16.250 |
| PSO Verify Certificate |  |  |  | 0 | 0 | 0 |
|  | 14.8.7.2 | B.8.11.1 | *PImport,ELC256* | 540 | 400 | 216.000 |
|  |  | B.8.11.2 | *PImport,ELC384* | 680 | 200 | 136.000 |
|  |  | B.8.11.3 | *PImport,ELC512* | 1.000 | 51 | 51.000 |
| PSO Verify Digital Signature | 14.8.9.1 | B.8.14 | *PverifyECDSA,256* | 100 | 5 | 500 |
|  |  |  | *PverifyECDSA,384* | 170 | 1 | 170 |
|  |  |  | *PverifyECDSA,512* | 270 | 1 | 270 |
| Fingerprint | 14.9.2.1 | B.8.3 | *Pfingerprint* | 18.000 | 0,4 | 7.200 |
| Generate Asymmetric Key Pair | 14.9.3.8 | B.8.15 | *PGAKP,2048* | 10.000 | 0,09 | 900 |
|  |  |  | *PGAKP,3072* | 60.000 | 0,01 | 600 |
|  |  | B.8.16 | *PGAKP,256* | 180 | 100 | 18.000 |
|  |  |  | *PGAKP,384* | 270 | 50 | 13.500 |
|  |  |  | *PGAKP,512* | 380 | 20 | 7.600 |
| Get Challenge | 14.9.4.2 | B.8.17 | *Pchallenge* | 10 | 200 | 2.000 |
| Manage Channel rst Ch | 14.9.8.3 |  | *Preset\_Ch* | 5 | 10 | 50 |
| MSE Restore | 14.9.9.1 | B.8.18 | *PMSE\_Restore* | 5 | 40 | 200 |
| MSE Set | 14.9.9 |  | *PMSE\_Set* | 10 | 300 | 3.000 |
| **Spaltensummen** |  |  |  | 104.738 | 10.000 | 1.220.380 |

Tabelle 275: Gesamtbewertung für Option\_kontaktlose\_Schnittstelle

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prüfpunkt** |  |  |  | ***TRi* / [ms]** | ***gi*** | ***gi TRi* / [ms]** |
| General Authenticate | 14.7.2.1 | B.8.19 | *PPACE* | 1.000 | 10 | 10.000 |
| **Spaltensummen** |  |  |  | 1.000 | 10 | 10.000 |

Tabelle 276: Gesamtbewertung für Option\_Kryptobox

| **Prüfpunkt** |  |  |  | ***TRi* / [ms]** | ***gi*** | ***gi TRi* / [ms]** |
| --- | --- | --- | --- | --- | --- | --- |
| Sessionkeyaushandlung für Trusted Channel | 15.4.1 | B.8.20 | *PSK4TC,AES128* | 90 | 15 | 1.350 |
|  |  |  | *PSK4TC,AES192* | 110 | 10 | 1.100 |
|  |  |  | *PSK4TC,AES256* | 120 | 15 | 1.800 |
| PSO Compute Cryptographic Checksum | 14.8.1.1 | B.8.21 | *Pcompute128,b* | 5 | 200 | 1.000 |
|  |  |  | *Pcompute128,m* | 10 | 200 | 2.000 |
|  |  |  | *Pcompute192,b* | 5 | 100 | 500 |
|  |  |  | *Pcompute192,m* | 15 | 100 | 1.500 |
|  |  |  | *Pcompute256,b* | 5 | 70 | 350 |
|  |  |  | *Pcompute256,m* | 20 | 70 | 1.400 |
| PSO Decipher | 14.8.3.3 |  | *Pdec128,b* | 5 | 200 | 1.000 |
|  |  |  | *Pdec128,m* | 10 | 200 | 2.000 |
|  |  |  | *Pdec192,b* | 5 | 100 | 500 |
|  |  |  | *Pdec192,m* | 15 | 100 | 1.500 |
|  |  |  | *Pdec256,b* | 5 | 70 | 350 |
|  |  |  | *Pdec256,m* | 20 | 70 | 1.400 |
| PSO Encipher | 14.8.4.5 |  | *Penc128,b* | 5 | 200 | 1.000 |
|  |  |  | *Penc128,m* | 10 | 200 | 2.000 |
|  |  |  | *Penc192,b* | 5 | 100 | 500 |
|  |  |  | *Penc192,m* | 15 | 100 | 1.500 |
|  |  |  | *Penc256,b* | 5 | 70 | 350 |
|  |  |  | *Penc256,m* | 20 | 70 | 1.400 |
| PSO Verify Cryptographic Checksum | 14.8.8.1 |  | *Pverify128,b* | 5 | 200 | 1.000 |
|  |  |  | *Pverify128,m* | 10 | 200 | 2.000 |
|  |  |  | *Pverify192,b* | 5 | 100 | 500 |
|  |  |  | *Pverify192,m* | 15 | 100 | 1.500 |
|  |  |  | *Pverify256,b* | 5 | 70 | 350 |
|  |  |  | *Pverify256,m* | 20 | 70 | 1.400 |
| **Spaltensummen** |  |  |  | 560 | 3.000 | 31.250 |

Tabelle 277: Gesamtbewertung für Option\_logische\_Kanäle

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prüfpunkt** |  |  |  | ***TRi* / [ms]** | ***gi*** | ***gi TRi* / [ms]** |
| Get Random | 14.9.5.1 | B.8.22 | *PRandom,b* | 10 | 10 | 100 |
|  |  |  | *PRandom,m* | 1.000 | 1 | 1.000 |
| Manage Channel Open | 14.9.8.1 | B.8.23 | *POpen* | 10 | 500 | 5.000 |
| Manage Channel Close | 14.9.8.2 |  | *PClose* | 5 | 400 | 2.000 |
| Manage Channel rst ICC | 14.9.8.4 |  | *PRST* | 5 | 89 | 445 |
| **Spaltensummen** |  |  |  | 1.030 | 1.000 | 8.545 |

Tabelle 278: Gesamtbewertung je nach Kombination der Optionen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Basis** | **Dual** | **Krypto** | **Kanal** | **Summe / [s]** | **Zulassungsgrenze / [s]** |
| 1.220.380 | 10.000 | 31.250 | 8.545 | 1270,2 | 555,7 |
| 1.220.380 | 10.000 | 31.250 | 0 | 1261,6 | 552,0 |
| 1.220.380 | 10.000 | 0 | 8.545 | 1238,9 | 542,0 |
| 1.220.380 | 10.000 | 0 | 0 | 1230,4 | 538,3 |
| 1.220.380 | 0 | 31.250 | 8.545 | 1260,2 | 551,3 |
| 1.220.380 | 0 | 31.250 | 0 | 1251,6 | 547,6 |
| 1.220.380 | 0 | 0 | 8.545 | 1228,9 | 537,6 |
| 1.220.380 | 0 | 0 | 0 | 1220,4 | 533,9 |

1. Bedeutung der Spaltenüberschriften in Tabelle 278:
   1. Basis: COS, welches keines der optionalen Funktionspakete enthält
   2. Dual: COS enthält Option\_kontaktlose\_Schnittstelle
   3. Krypto: COS enthält Option\_Kryptobox
   4. Kanal: COS enthält Option\_logische\_Kanäle
2. (N109.500) K\_externeWelt {K\_Karte}   
   Für die Gesamtbewertung gilt:   
     *Pgesamt* ,   
   wobei nur die Prüfpunkte zu berücksichtigen sind, die gemäß der vom COS unterstützten Optionen möglich sind.

Unter den vereinfachenden Annahmen, dass die Messwerte innerhalb einer Reihe identisch sind (Standardabweichung σ ist null) und das Verhältnis *Xi* / *TRi* in allen Messreihen konstant ist, folgt dann für das Basisbetriebssystem:

1. *Pgesamt\_einfach* 1220,4s, mit *f*2 als Funktion von *Xi / TRi*.

Abbildung 12 zeigt diesen Zusammenhang graphisch mit dem Verhältnis *Xi* / *TRi* als Abszisse und *Pgesamt\_einfach* als Ordinate. Demnach erreicht ein unendlich schnelles Basisbetriebssystem 1220,4s Punkte. Eine Karte, die in allen Prüfpunkten die Referenzzeit benötigt, erreicht 915,3s Punkte. Eine Karte, die überall die doppelte Referenzzeit benötigt, null Punkte.

1. (N109.600) K\_externeWelt {K\_Karte}   
   Ein COS DARF KEINE Zulassung erhalten, wenn in wenigstens einem Prüfpunkt das Verhältnis *Xi* / *TRi* größer als vier ist.
2. (N109.700) K\_externeWelt {K\_Karte}   
   Ein COS DARF KEINE Zulassung erhalten, wenn die gemäß (N109.500) ermittelte Punktzahl kleiner ist als in Tabelle 278 für die Kombination von Optionen angegeben. Dies entspricht einer relativen Ausführungszeit, die 1,5-mal so groß ist wie die Referenzzeit.



Abbildung 12: Graphische Darstellung von *Pgesamt\_einfach*

## Übertragungsgeschwindigkeit

### Übertragungsgeschwindigkeit für das Übertragungsprotokoll T=1

Dieser Prüfpunkt berücksichtigt die Kanalkapazität des Kanals TPDU\_Channel für das Übertragungsprotokoll T=1 in Abbildung 11. Diese wird wegen der starken Abhängigkeit von der externen Taktfrequenz des IFD nicht experimentell, sondern rechnerisch aus den Bytes TA1 und TC1 im ATR und der Kapazität *C* aus Tabelle 273 ermittelt. Für die (rechnerische) Übertragungszeit von 1.000 Oktett zur Karte gilt:

*tT* = 

Mit *C* aus Tabelle 273 für den Fall PPS1 = TA1 und *CGT* in Abhängigkeit von TC1 im ATR gemäß folgender Tabelle:

Tabelle 279: Character Guard Time (CGT) gemäß [ISO/IEC 7816-3#11.2]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TC1** | ´FF´ | ´00´ | ´01´ | ´02´ | … | ´FD´ | ´FE´ |
| **CGT** | 11 | 12 | 13 | 14 | … | 265 | 266 |

Es gilt: *PIO* = points( (*tT*, *tT* ), *TIO* ).

### Übertragungsgeschwindigkeit für das Protokoll [ISO/IEC 7816-12]

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

### Übertragungsgeschwindigkeit für kontaktlose Datenübertragung

1. Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

Die folgenden Anforderungen sind absichtlich leer:   
(N109.800), (N109.900), (N110.000), (N110.100), (N110.200), (N110.300), (N110.400), (N110.500), (N110.600), (N110.700), (N110.800), (N110.900), (N111.000), (N111.100), (N111.200), (N111.300), (N111.400), (N111.500), (N111.600), (N111.700), (N111.800), (N111.900), (N112.000), (N112.100), (N112.200), (N112.300), (N112.400), (N112.500), (N112.600), (N112.700), (N112.800), (N112.900), (N113.000), (N113.100), (N113.200), (N113.300), (N113.400), (N113.500), (N113.600), (N113.700), (N113.800), (N113.900), (N114.000), (N114.100), (N114.200), (N114.300), (N114.400), (N114.500), (N114.600), (N114.700), (N114.800), (N114.900), (N115.000), (N115.100), (N115.200), (N115.300), (N115.400), (N115.500), (N115.600), (N115.700), (N115.800), (N115.900), (N116.000), (N116.100), (N116.200), (N116.300), (N116.400), (N116.500), (N116.600), (N116.700), (N116.800), (N116.900), (N117.000), (N117.100), (N117.200), (N117.300), (N117.400), (N117.500), (N117.600), (N117.700), (N117.800), (N117.900), (N118.000), (N118.100), (N118.200), (N118.300), (N118.400), (N118.500), (N118.600), (N118.700), (N118.800), (N118.900), (N119.000), (N119.100), (N119.200), (N119.300), (N119.400), (N119.500), (N119.600), (N119.700), (N119.800), (N119.900), (N120.000), (N120.100), (N120.200), (N120.300), (N120.400), (N120.500), (N120.600), (N120.700), (N120.800), (N120.900), (N121.000), (N121.100), (N122.000), (N122.100), (N122.200), (N122.300), (N122.400), (N122.500), (N122.600), (N122.700), (N122.800), (N122.900), (N123.000), (N123.100), (N123.200), (N123.300), (N123.400), (N123.500), (N123.600), (N123.700), (N123.800), (N123.900), (N124.000), (N124.100), (N124.200), (N124.300), (N124.400), (N124.500), (N124.600), (N124.700), (N124.710), (N124.800), (N124.900), (N125.000), (N125.100), (N125.200), (N125.300), (N125.400), (N125.500), (N125.600), (N125.700), (N125.800), (N125.900), (N126.000), (N126.100), (N126.200), (N126.300), (N126.400), (N126.500), (N126.600), (N126.700), (N126.800), (N126.900)

## Startsequenz für das Übertragungsprotokoll T=1

Dieser Prüfpunkt behandelt eine Sequenz, die bei der Aktivierung einer Smartcard durchlaufen wird.

Der Prüfpunkt beinhaltet:

* Die Aktivierung (Bootvorgang) des Betriebssystems,
* die Aushandlung einer höheren Übertragungsrate und
* die Aushandlung einer Puffergröße für die Datenübertragung.

Testvorbereitung:

Keine.

Testdurchführung:

1. (N200.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung durchläuft eine Schleife 100-mal. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt. Abweichend von den Festlegungen in B.5.1 wird hier die Zeit pro Schleifendurchlauf anders ermittelt.
2. (N200.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Die Zeit *tStart* gibt den Zeitpunkt an, zu welchem die Smartcard gemäß (N108.800) aktiviert wird. Genauer, den Zeitpunkt des Wechsels von RST von L nach H (siehe [ISO/IEC 7816-3#Figure 1]). Anschließend wird der ATR empfangen.
3. (N200.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Es MUSS eine PPS-Sequenz gemäß [ISO/IEC 7816-3] erfolgen. Die Bits 4 bis 1 in PPS0 MÜSSEN das Übertragungsprotokoll T=1 anzeigen. Als PPS1 MUSS der Wert von TA1 aus dem ATR verwendet werden.
4. (N200.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Das IFD MUSS der Smartcard den Wert von IFSD = 254 präsentieren.
5. (N200.240) K\_externeWelt {K\_Karte}   
   Schritt 4: Der Zeitpunkt *tEnd* ist definiert durch das Ende der letzten TPDU, welche in (N200.230) übertragen wird. Damit gilt für den *i*-ten Schleifendurchlauf:   
     *tRun*4,*i* = *tEnd* – *tStart*.

Testauswertung:

Es gilt: *PKarteStarten* = points( (*tRun4*,1, *t Run4*,2, …, *t Run4*,100 ), *TKarteStarten* ).

Testnachbereitung:

Keine.

Besser wäre es, wenn auch hier die Messung aus B.5.1 anwendbar wäre. Es ist zu prüfen, ob dafür passendes Equipment zur Verfügung steht.

## Messverfahren für Einzelkommandos (normativ)

### Activate, Deactivate, Delete, Load Application, Terminate

In diesem Kapitel werden Kommandos zur Bearbeitung eines Life Cycle Status betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.LCS.

Testvorbereitung:

1. (N201.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N201.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.LCS gesetzt werden.

Testdurchführung:

1. (N201.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 10 ausgeführt.
2. (N201.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Die Datei / MF / DF.LCS / EF.LCS MUSS mittels Use Case aus 14.2.6.13 selektiert werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N201.220) K\_externeWelt {K\_Karte}   
   Schritt 2: LCS für Dateien:
   1. *currentEF* MUSS mittels Use Case aus 14.2.3.1 deaktiviert werden. Die Laufzeit *tdeactivate\_EF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. *currentEF* MUSS mittels Use Case aus 14.2.1.1 aktiviert werden. Die Laufzeit *tactivate\_EF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   3. *currentEF* MUSS mittels Use Case aus 14.2.9.1 terminiert werden. Die Laufzeit *tterminate\_EF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   4. *currentEF* MUSS mittels Use Case aus 14.2.4.1 gelöscht werden. Die Laufzeit *tdelete\_EF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
4. (N201.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Arbeiten mit Passwortobjekten:
   1. Der Status von PIN.LCS MUSS mittels Use Case aus 14.6.4.1 abgefragt werden. Die Laufzeit *tGetStatus,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. Der Transportschutz von PIN.LCS MUSS mittels Use Case aus 14.6.1.2 aufgehoben werden, wobei für *newSecret* eine zufällig gewählte PIN verwendet werden MUSS, deren Länge gleichverteilt aus dem Bereich [6, 12] gezogen wird. Die Laufzeit *tSetPIN,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   3. Das Attribut PIN.LCS.*flagEnabled* MUSS mittels Use Case aus 14.6.2.1 auf den Wert False gesetzt werden. Die Laufzeit *tDisablePIN,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   4. Das Attribut PIN.LCS.*flagEnabled* MUSS mittels Use Case aus 14.6.3.1 auf den Wert True gesetzt werden. Die Laufzeit *tEnablePIN,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   5. Das Attribut PIN.LCS.*secret* MUSS mittels Use Case aus 14.6.1.1 auf einen anderen Wert gesetzt werden, wobei für *newSecret* eine zufällig gewählte PIN verwendet werden MUSS, deren Länge gleichverteilt aus dem Bereich [6, 12] gezogen wird. Die Laufzeit *tChRefData,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   6. Verify Kommando gemäß 14.6.6.1 mit korrekten *verificationData* für das Objekt PIN.LCS. Die Laufzeit *tVerify\_1,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   7. Verify Kommando gemäß 14.6.6.1 mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit *tVerify\_2,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   8. Verify Kommando gemäß 14.6.6.1 mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit *tVerify\_3,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   9. Verify Kommando gemäß 14.6.6.1 mit korrekten *verificationData* für das Objekt PIN.LCS. Die Laufzeit *tVerify\_4,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   10. Verify Kommando gemäß 14.6.6.1 mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
   11. Reset Retry Counter gemäß 14.6.5.1 mit korrekter PUK für das Objekt PIN.LCS, wobei für *newSecret* eine zufällig gewählte PIN verwendet werden MUSS, deren Länge gleichverteilt aus dem Bereich [6, 12] gezogen wird. Die Laufzeit *tResetRC,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
5. (N201.240) K\_externeWelt {K\_Karte}   
   Schritt 4: LCS für Passwortobjekte:
   1. PIN.LCS MUSS mittels Use Case aus 14.2.3.4 deaktiviert werden. Die Laufzeit *tdeactivate\_Pwd,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. PIN.LCS MUSS mittels Use Case aus 14.2.1.4 aktiviert werden. Die Laufzeit *tactivate\_Pwd,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   3. PIN.LCS MUSS mittels Use Case aus 14.2.9.4 terminiert werden. Die Laufzeit *tterminate\_Pwd,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   4. PIN.LCS MUSS mittels Use Case aus 14.2.4.4 gelöscht werden. Die Laufzeit *tdelete\_Pwd,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
6. (N201.250) K\_externeWelt {K\_Karte}   
   Schritt 5: LCS für private Schlüssel:
   1. PrK.LCS MUSS mittels Use Case aus 14.2.3.2 deaktiviert werden. Die Laufzeit *tdeactivate\_PrK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. PrK.LCS MUSS mittels Use Case aus 14.2.1.2 aktiviert werden. Die Laufzeit *tactivate\_PrK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   3. PrK.LCS MUSS mittels Use Case aus 14.2.9.2 terminiert werden. Die Laufzeit *tterminate\_PrK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   4. PrK.LCS MUSS mittels Use Case aus 14.2.4.2 gelöscht werden. Die Laufzeit *tdelete\_PrK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
7. (N201.260) K\_externeWelt {K\_Karte}   
   Schritt 6: LCS für öffentliche Schlüssel:
   1. PuK.LCS MUSS mittels Use Case aus 14.2.3.3 deaktiviert werden. Die Laufzeit *tdeactivate\_PuK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. PuK.LCS MUSS mittels Use Case aus 14.2.1.3 aktiviert werden. Die Laufzeit *tactivate\_PuK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   3. PuK.LCS MUSS mittels Use Case aus 14.2.9.3 terminiert werden. Die Laufzeit *tterminate\_PuK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   4. PuK.LCS MUSS mittels Use Case aus 14.2.4.3 gelöscht werden. Die Laufzeit *tdelete\_PuK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
8. (N201.270) K\_externeWelt {K\_Karte}   
   Schritt 7: LCS für symmetrische Schlüssel:
   1. SK.LCS MUSS mittels Use Case aus 14.2.3.2 deaktiviert werden. Die Laufzeit *tdeactivate\_SyK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. SK.LCS MUSS mittels Use Case aus 14.2.1.2 aktiviert werden. Die Laufzeit *tactivate\_SyK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   3. SK.LCS MUSS mittels Use Case aus 14.2.9.2 terminiert werden. Die Laufzeit *tterminate\_SyK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   4. SK.LCS MUSS mittels Use Case aus 14.2.4.2 gelöscht werden. Die Laufzeit *tdelete\_SyK,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
9. (N201.280) K\_externeWelt {K\_Karte}   
   Schritt 8: Der Ordner / MF / DF.LCS MUSS mittels Use Case aus 14.2.6.9 selektiert werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
10. (N201.290) K\_externeWelt {K\_Karte}   
    Schritt 9: LCS für Ordner:
    1. *currentFolder* MUSS mittels Use Case aus 14.2.3.1 deaktiviert werden. Die Laufzeit *tdeactivate\_DF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
    2. *currentFolder* MUSS mittels Use Case aus 14.2.1.1 aktiviert werden. Die Laufzeit *tactivate\_DF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
    3. *currentFolder* MUSS mittels Use Case aus 14.2.8.1 terminiert werden. Die Laufzeit *tterminate\_DF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
    4. *currentFolder* MUSS mittels Use Case aus 14.2.4.1 gelöscht werden. Die Laufzeit *tdelete\_DF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
11. (N201.300) K\_externeWelt {K\_Karte}   
    Schritt 10: Laden von DF.LCS.
    1. Die Anwdendung DF.LCS MUSS mittels Load Application in den Prüfling geladen werden. Dazu sind ein oder mehrere Load Application Kommandos erforderlich. Gemäß (N040.100) ist *currentFolder* anschließend auf DF.LCS gesetzt.
    2. Die Ausführungszeit jedes Load Application Kommandos MUSS gemäß B.5.1 gemessen werden.
    3. Die Summe aller Load Application Kommandoausführungszeiten in der *i*-ten Schleifeniteration wird mit *tLoadApp,i* bezeichnet.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Pactivate\_DF* | = points( (*tactivate\_DF*,1, | *tactivate\_DF*,2, …, | *tactivate\_DF*,100 ), | *Tactivate\_DF* ) |
| *Pactivate\_EF* | = points( (*tactivate\_EF*,1, | *tactivate\_EF*,2, …, | *tactivate\_EF*,100 ), | *Tactivate\_EF* ) |
| *Pactivate\_Pwd* | = points( (*tactivate\_Pwd*,1, | *tactivate\_Pwd*,2, …, | *tactivate\_Pwd*,100 ), | *Tactivate\_Pwd* ) |
| *Pactivate\_PrK* | = points( (*tactivate\_PrK*,1, | *tactivate\_PrK*,2, …, | *tactivate\_PrK*,100 ), | *Tactivate\_PrK* ) |
| *Pactivate\_PuK* | = points( (*tactivate\_PuK*,1, | *tactivate\_PuK*,2, …, | *tactivate\_PuK*,100 ), | *Tactivate\_PuK* ) |
| *Pactivate\_SK* | = points( (*tactivate\_SK*,1, | *tactivate\_SK*,2, …, | *tactivate\_SK*,100 ), | *Tactivate\_SK* ) |
| *Pdeactivate\_DF* | = points( (*tdeactivate\_DF*,1, | *tdeactivate\_DF*,2, …, | *tdeactivate\_DF*,100 ), | *Tdeactivate\_DF* ) |
| *Pdeactivate\_EF* | = points( (*tdeactivate\_EF*,1, | *tdeactivate\_EF*,2, …, | *tdeactivate\_EF*,100 ), | *Tdeactivate\_EF* ) |
| *Pdeactivate\_Pwd* | = points( (*tdeactivate\_Pwd*,1, | *tdeactivate\_Pwd*,2, …, | *tdeactivate\_Pwd*,100 ), | *Tdeactivate\_Pwd* ) |
| *Pdeactivate\_PrK* | = points( (*tdeactivate\_PrK*,1, | *tdeactivate\_PrK*,2, …, | *tdeactivate\_PrK*,100 ), | *Tdeactivate\_PrK* ) |
| *Pdeactivate\_PuK* | = points( (*tdeactivate\_PuK*,1, | *tdeactivate\_PuK*,2, …, | *tdeactivate\_PuK*,100 ), | *Tdeactivate\_PuK* ) |
| *Pdeactivate\_SK* | = points( (*tdeactivate\_SK*,1, | *tdeactivate\_SK*,2, …, | *tdeactivate\_SK*,100 ), | *Tdeactivate\_SK* ) |
| *Pdelete\_DF* | = points( (*tdelete\_DF*,1, | *tdelete\_DF*,2, …, | *tdelete\_DF*,100 ), | *Tdelete\_DF* ) |
| *Pdelete\_EF* | = points( (*tdelete\_EF*,1, | *tdelete\_EF*,2, …, | *tdelete\_EF*,100 ), | *Tdelete\_EF* ) |
| *Pdelete\_Pwd* | = points( (*tdelete\_Pwd*,1, | *tdelete\_Pwd*,2, …, | *tdelete\_Pwd*,100 ), | *Tdelete\_Pwd* ) |
| *Pdelete\_PrK* | = points( (*tdelete\_PrK*,1, | *tdelete\_PrK*,2, …, | *tdelete\_PrK*,100 ), | *Tdelete\_PrK* ) |
| *Pdelete\_PuK* | = points( (*tdelete\_PuK*,1, | *tdelete\_PuK*,2, …, | *tdelete\_PuK*,100 ), | *Tdelete\_PuK* ) |
| *Pdelete\_SK* | = points( (*tdelete\_SK*,1, | *tdelete\_SK*,2, …, | *tdelete\_SK*,100 ), | *Tdelete\_SK* ) |
| *Pterminate\_DF* | = points( (*tterminate\_DF*,1, | *tterminate\_DF*,2, …, | *tterminate\_DF*,100 ), | *Tterminate\_DF* ) |
| *Pterminate\_EF* | = points( (*tterminate\_EF*,1, | *tterminate\_EF*,2, …, | *tterminate\_EF*,100 ), | *Tterminate\_EF* ) |
| *Pterminate\_Pwd* | = points( (*tterminate\_Pwd*,1, | *tterminate\_Pwd*,2, …, | *tterminate\_Pwd*,100 ), | *Tterminate\_Pwd* ) |
| *Pterminate\_PrK* | = points( (*tterminate\_PrK*,1, | *tterminate\_PrK*,2, …, | *tterminate\_PrK*,100 ), | *Tterminate\_PrK* ) |
| *Pterminate\_PuK* | = points( (*tterminate\_PuK*,1, | *tterminate\_PuK*,2, …, | *tterminate\_PuK*,100 ), | *Tterminate\_PuK* ) |
| *Pterminate\_SK* | = points( (*tterminate\_SK*,1, | *tterminate\_SK*,2, …, | *tterminate\_SK*,100 ), | *Tterminate\_SK* ) |
| *PLoadApp* | = points( (*tLoadApp*,1, | *tLoadApp*,2, …, | *tLoadApp*,100 ), | *TLoadApp* ) |
| *PGetPinStatus* | = points( (*tGetStatus*,1, | *tGetStatus*,2, …, | *tGetStatus*,100 ), | *TGetStatus* ) |
| *PSetPIN* | = points( (*tSetPIN*,1, | *tSetPIN*,2, …, | *tSetPIN*,100 ), | *TSetPIN* ) |
| *PDisablePIN* | = points( (*tDisablePIN*,1, | *tDisablePIN*,2, …, | *tDisablePIN*,100 ), | *TDisablePIN* ) |
| *PEnablePIN* | = points( (*tEnablePIN*,1, | *tEnablePIN*,2, …, | *tEnablePIN*,100 ), | *TEnablePIN* ) |
| *PChRefData* | = points( (*tChRefData*,1, | *tChRefData*,2, …, | *tChRefData*,100 ), | *TChRefData* ) |
| *PVerify\_1* | = points( (*tVerify\_1*,1, | *tVerify\_1*,2, …, | *tVerify\_1*,100 ), | *TVerify\_1* ) |
| *PVerify\_2* | = points( (*tVerify\_2*,1, | *tVerify\_2*,2, …, | *tVerify\_2*,100 ), | *TVerify\_2* ) |
| *PVerify\_3* | = points( (*tVerify\_3*,1, | *tVerify\_3*,2, …, | *tVerify\_3*,100 ), | *TVerify\_3* ) |
| *PVerify\_4* | = points( (*tVerify\_4*,1, | *tVerify\_4*,2, …, | *tVerify\_4*,100 ), | *TVerify\_4* ) |
| *PVerify* | = 0,8 *PVerify\_1 +* 0,08 *PVerify\_2 +* 0,02 *PVerify\_3 +* 0,1 *PVerify\_4* | | | |
| *PResetRC* | = points( (*tResetRC*,1, | *tResetRC*,2, …, | *tResetRC*,100 ), | *TResetRC* ) |

Testnachbereitung:

Keine.

### Select Datei

In diesem Kapitel wird die Selektion von Dateien betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.SelectEF.

Testvorbereitung:

1. (N202.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N202.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.SelectEF gesetzt werden.
3. (N202.130) K\_externeWelt {K\_Karte}   
   Schritt 3: *currentEF* MUSS auf / MF / DF.SelectEF / EF.00 gesetzt werden.

Testdurchführung:

1. (N202.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.
2. (N202.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Aus der Menge der zu Anfang 100 in DF.SelectEF gültigen Fileidentifier wird ein bislang noch nicht verwendeter gezogen (ziehen ohne zurücklegen).
3. (N202.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Datei mit dem in Schritt 1 ausgewählten Fileidentifier MUSS mittels Use Case aus 14.2.6.13 selektiert werden. Die Laufzeit *tselect\_EF,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Pselect\_EF* | = points( (*tselect\_EF*,1, | *tselect\_EF*,2, …, | *tselect\_EF*,100 ), | *Tselect\_EF* ) |

Testnachbereitung:

Keine.

### Fingerprint

In diesem Abschnitt wird die Berechnung des COS Fingerprints betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

1. (N203.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

1. (N203.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.
2. (N203.210) K\_externeWelt {K\_Karte}   
   Schritt 1: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *prefix* = RAND( 128 ) erzeugt werden.
3. (N203.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Fingerprint Kommando gemäß 14.9.2.1, wobei als *prefix* der im vorherigen Schritt erzeugte Wert dient. Die Laufzeit *tfp,i* dieses Kommandos MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Pfingerprint* | = points( (*tfp*,1, | *tfp*,2, …, | *tfp*,100 ), | *Tfingerprint* ) |

Testnachbereitung:

Keine.

### Terminate Card Usage

In diesem Kapitel wird das Terminieren einer Smartcard betrachtet.

Testvorbereitung:

1. (N204.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

1. (N204.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife zehnmal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.
2. (N204.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Die Smartcard MUSS mittels Use Case aus 14.2.7.1 terminiert werden. Die Laufzeit *tterminateCard,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
3. (N204.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Smartcard MUSS so reinitialisiert werden, dass die in B.9 definierte Kartenkonfiguration vorliegt.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PterminateCard* | = points( (*tterminateCard*,1, | *tterminateCard*,2, …, | *tterminateCard*,10 ), | *TterminateCard* ) |

Testnachbereitung:

Keine.

### Set Logical Eof, Write Binary

In diesem Kapitel wird das Anfügen von Daten in transparenten EF und das Setzen des Attributes *positionLogicalEndOfFile* betrachtet. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.SelectEF / EF.transparent.

Testvorbereitung:

1. (N205.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N205.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.
3. (N205.130) K\_externeWelt {K\_Karte}   
   Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.
4. (N205.140) K\_externeWelt {K\_Karte}   
   Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen *MNc* = {1, 2, 3, …, 1000} erstellt werden.
5. (N205.150) K\_externeWelt {K\_Karte}   
   Schritt 5: Es MUSS eine leere Menge *MSetEOF* = {} erstellt werden.

Testdurchführung:

1. (N205.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung durchläuft eine Schleife solange, bis die Menge *MNc* leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 4 fortgefahren.
2. (N205.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Aus der Menge *MNc* wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen) und der Oktettstring *newData* = RAND( *e* ) erzeugt.
3. (N205.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Der *body* von EF.transparent wird gemäß Use Case aus 14.3.6.1 erweitert wobei *newData* als Kommandonachricht verwendet wird. Die Laufzeit *twrite,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
4. (N205.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Falls das Attribute EF.transparent.*positionLogicalEndOfFile* nach Schritt 2 größer als ´7800´ = 30.720 ist, dann MUSS dieses Attribut mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt werden. Die Laufzeit *tset* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *MSetEOF* hinzugefügt werden.
5. (N205.240) K\_externeWelt {K\_Karte}   
   Schritt 4: Das Attribute EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt. Die Laufzeit *tset* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *MSetEOF* hinzugefügt werden.

Testauswertung:

Durch die Messpunkt (*x*, *y*) 🡪 (e, *twrite,i*) wird eine Ausgleichsgerade *y* = *m* *x* + *b* gelegt.

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Pwirte,b* | = points( | (*b*, *b*) | , | *Twrite,b* ) |
| *Pwirte,m* | = points( | (1000*m*, 1000*m*) | , | *Twirte,b* ) |
| *PSetEOF* | = points( | *MSetEOF* | , | *TSetEOF* ) |

1. Die Steigung m der Ausgleichsgeraden gibt die Schreibrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von Pm aufgeführt ist, rechnet dies um in Sekunden pro kByte.

Testnachbereitung:

Keine.

### Erase Binary, Update Binary

In diesem Kapitel wird das Schreiben und Löschen von Daten in transparenten EF betrachtet. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.SelectEF / EF.transparent.

Testvorbereitung:

1. (N206.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N206.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.
3. (N206.130) K\_externeWelt {K\_Karte}   
   Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.
4. (N206.140) K\_externeWelt {K\_Karte}   
   Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen *MNc* = {1, 2, 3, …, 1000} erstellt werden.
5. (N206.150) K\_externeWelt {K\_Karte}   
   Schritt 5: Es MUSS eine leere Menge *Mwipe* = {} erstellt werden.
6. (N206.160) K\_externeWelt {K\_Karte}   
   Schritt 6: Das Attribute EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.5.1 auf den Wert EF.transparent.*numberOfOctet* gesetzt, wobei für den Kommandoparameter *newData* = ´00´ gilt.
7. (N206.170) K\_externeWelt {K\_Karte}   
   Schritt 7: Die Variable *index* wird auf den Wert null gesetzt.

Testdurchführung:

1. (N206.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung durchläuft eine Schleife solange, bis die Menge *MNc* leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 4 fortgefahren.
2. (N206.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Aus der Menge *MNc* wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen) und der Oktettstring *newData* = RAND( *e* ) erzeugt.
3. (N206.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Der *body* von EF.transparent wird gemäß Use Case aus 14.3.5.1 beschrieben, wobei *newData* als Kommandonachricht und *index* als Kommandoparameter *offset* verwendet wird. Die Laufzeit *tupdate,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden. Nach Kommandoausführung wird *index* += *e* gesetzt.
4. (N206.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Falls *index* nach Schritt 2 größer als ´7800´ = 30.720 ist, dann MUSS *index* auf den Wert null gesetzt werden und der Inhalt von EF.transparent mittels Use Case aus 14.3.1.1 gelöscht werden, wobei der Kommandoparameter *offset* = 0 gesetzt wird. Die Laufzeit *twipe* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *Mwipe* hinzugefügt werden.
5. (N206.240) K\_externeWelt {K\_Karte}   
   Schritt 4: Der Inhalt von EF.transparent wird mittels Use Case aus 14.3.1.1 gelöscht, wobei der Kommandoparameter *offset* = 0 gesetzt wird. Die Laufzeit *twipe* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *Mwipe* hinzugefügt werden.

Testauswertung:

Durch die Messpunkt (*x*, *y*) 🡪 (e, *tupdate,i*) wird eine Ausgleichsgerade *y* = *m* *x* + *b* gelegt.

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PUpdateBin,b* | = points( | (*b*, *b*) | , | *Tupdate,b* ) |
| *PUpdateBin,m* | = points( | (1000*m*, 1000*m*) | , | *Tupdate,m* ) |
| *PWipeBin* | = points( | *Mwipe* | , | *Twipe*) |

1. Die Steigung m der Ausgleichsgeraden gibt die Schreibrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von Pm aufgeführt ist, rechnet dies um in Sekunden pro kByte.

Testnachbereitung:

1. (N206.300) K\_externeWelt {K\_Karte}   
   Das Attribute EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt.

### Read Binary

In diesem Kapitel wird das Lesen von Daten in transparenten EF betrachtet. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.SelectEF / EF.transparent.

Testvorbereitung:

1. (N207.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N207.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.
3. (N207.130) K\_externeWelt {K\_Karte}   
   Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.
4. (N207.140) K\_externeWelt {K\_Karte}   
   Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen *MNe*= {1, 2, 3, …, 1000} erstellt werden.
5. (N207.150) K\_externeWelt {K\_Karte}   
   Schritt 5: Das Attribute EF.transparent.*body* MUSS vollständig mit zufälligen Werten befüllt werden.

Testdurchführung:

1. (N207.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung durchläuft eine Schleife solange, bis die Menge *MNe* leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.
2. (N207.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Aus der Menge *MNe* wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen).
3. (N207.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Teile von EF.transparent.*body* werden gemäß Use Case aus 14.3.2.1 ausgelesen, wobei *e* als Kommandoparameter *length* und eine zufällige Zahl aus dem Bereich [0, 30.720] = [´0000´, ´7800´] als Kommandoparameter *offset* verwendet wird. Die Laufzeit *tread,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Durch die Messpunkt (*x*, *y*) 🡪 (e, *tread,i*) wird eine Ausgleichsgerade *y* = *m* *x* + *b* gelegt.

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PReadBinary,b* | = points( | (*b*, *b*) | , | *TReadBinary,b* ) |
| *PReadBinary,m* | = points( | (1000*m*, 1000*m*) | , | *TReadBinary,m* ) |

1. Die Steigung m der Ausgleichsgeraden gibt die Leserate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von Pm aufgeführt ist, rechnet dies um in Sekunden pro kByte.

Testnachbereitung:

1. (N207.300) K\_externeWelt {K\_Karte}   
   Das Attribute EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt.

### Rekord orientierte Kommandos

In diesem Abschnitt werden folgende Kommandos betrachtet: Activate Record, Append Record, Deactivate Record, Delete Record, Erase Record, Read Record, Update Record. Das Kommando Search Record wird in B.8.9 behandelt. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.strukturiert / EF.strukturiert.

Testvorbereitung:

1. (N208.100) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N208.110) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.strukturiert gesetzt werden.
3. (N208.120) K\_externeWelt {K\_Karte}   
   Schritt 3: Die Datei / MF / DF.strukturiert / EF.strukturiert MUSS mittels Use Case aus 14.2.6.13 selektiert werden.

Testdurchführung:

1. (N208.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 255 mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.
2. (N208.210) K\_externeWelt {K\_Karte}   
   Schritt 1, Append Record: Im *i*-ten Schleifendurchlauf wird ein Oktettstring *recordData* = RAND( *i* ) erzeugt. In *currentEF* MUSS mittels Use Case aus 14.4.2.1 ein neuer Rekord angelegt werden, wobei als Datenteil der Kommandonachricht *recordData* verwendet wird. Die Laufzeit *tappend,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
3. (N208.220) K\_externeWelt {K\_Karte}   
   Schritt 2, Deactivate Record: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.3.1 deaktiviert werden. Die Laufzeit *tdeactivate,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
4. (N208.230) K\_externeWelt {K\_Karte}   
   Schritt 3, Activate Record: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.1.1 aktiviert werden. Die Laufzeit *tactivate,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
5. (N208.240) K\_externeWelt {K\_Karte}   
   Schritt 4, Update Record: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.8.1 auf den Wert *newData* = RAND( *i* ) werden. Die Laufzeit *tupdate,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
6. (N208.250) K\_externeWelt {K\_Karte}   
   Schritt 5, Read Record: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.6.1 gelesen werden. Die Laufzeit *tread,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
7. (N208.260) K\_externeWelt {K\_Karte}   
   Schritt 6, Erase Record wipe: Der Inhalt von Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.5.1 gelöscht werden. Die Laufzeit *twipe,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
8. (N208.270) K\_externeWelt {K\_Karte}   
   Schritt 7, Delete Record: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.4.1 aus *currentEF*.*recordList* entfernt werden. Die Laufzeit *tdelete,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Durch die Messpunkt (*x*, *y*) 🡪 (*i*, *ta,i*) wird eine Ausgleichsgerade *y* = *ma* *x* + *ba* gelegt.

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PAppendRecord,b* | = points( | (*bAppendRecord*, | *bAppendRecord*), | *TAppendRecord,b* ) |
| *PAppendRecord,m* | = points( | (1000*mAppendRecord*, | 1000*mAppendRecord*), | *TAppendRecord,m* ) |
| *PUpdateRecord,b* | = points( | (*bUpdateRecord*, | *bUpdateRecord*), | *TUpdateRecord,b* ) |
| *PUpdateRecord,m* | = points( | (1000*mUpdateRecord*, | 1000*mUpdateRecord*), | *TUpdateRecord,m* ) |
| *PReadRecord,b* | = points( | (*bReadRecord*, | *bReadRecord*), | *TReadRecord,b* ) |
| *PReadRecord,m* | = points( | (1000*mReadRecord*, | 1000*mReadRecord*), | *TReadRecord,m* ) |
| *PWipeRecord,b* | = points( | (*bWipeRecord*, | *bWipeRecord*), | *TWipeRecord,b* ) |
| *PWipeRecord,m* | = points( | (1000*mWipeRecord*, | 1000*mWipeRecord*), | *TWipeRecord,m* ) |

1. Die Steigung m der Ausgleichsgeraden gibt die Schreib- oder Leserate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von Pa,m aufgeführt ist, rechnet dies um in Sekunden pro kByte.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PActivateRec* | = points( (*tActivateRec*,1, | *tActivateRec*,2, …, | *tActivateRec*,1000 ), | *TActivateRec* ) |
| *PDeactivateRec* | = points( (*tDeactivateRec*,1, | *tDeactivateRec*,2, …, | *tDeactivateRec*,1000 ), | *TDeactivateRec* ) |
| *PDeleteRec* | = points( (*tDeleteRec*,1, | *tDeleteRec*,2, …, | *tDeleteRec*,1000 ), | *TDeleteRec* ) |

Testnachbereitung:

Keine.

### Search Record

Hier wird lediglich das Kommando Search Record betrachtet. Die übrigen rekordorientierten Kommandos werden in B.8.8 behandelt. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.strukturiert / EF.strukturiert.

Testvorbereitung:

1. (N209.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N209.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.strukturiert gesetzt werden.
3. (N209.130) K\_externeWelt {K\_Karte}   
   Schritt 3: Die Datei / MF / DF.strukturiert / EF.strukturiert MUSS mittels Use Case aus 14.2.6.13 selektiert werden.
4. (N209.140) K\_externeWelt {K\_Karte}   
   Schritt 4: Das Attribut *currentEF*.*recordList* MUSS mittels Use Case aus 14.4.2.1 wie folgt mit 254 Rekords gefüllt werden, die alle 255 Oktette lang sind:
   1. Rekord     1 = ´0100FF…FF´,
   2. Rekord     2 = ´FF0200FF…FF´,
   3. Rekord     3 = ´FFFF0300FF…FF´,
   4. …
   5. Rekord   16 = ´FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1000FF…FF´,
   6. Rekord 252 = ´FF…FFFC00FFFF´,
   7. Rekord 253 = ´FF…FFFD00FF´,
   8. Rekord 254 = ´FF…FFFE00´.
5. Ein Rekord i enthält seine Rekordnummer n an der Position n. Der Rekordnummer folgt ein Oktett mit dem Wert ´00´. Die übrigen Oktette besitzen den Wert ´FF´. Für ein Oktett lange Pattern gilt somit:
   1. Pattern = ´00´: Dieses Pattern ist in jedem Rekord enthalten. Der offset dieses Patterns variiert mit der Rekordnummer.
   2. Pattern = ´FF´: Dieses Pattern ist in jedem Rekord im Wesentlichen am Rekordanfang enthalten.
   3. Alle übrigen Pattern mit einer Länge von einem Oktett sind in genau einem Rekord enthalten. Der offset dieser Pattern variiert mit der Rekordnummer.

Testdurchführung:

1. (N209.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 254-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 3 fortgefahren.
2. (N209.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Im *i*-ten Schleifendurchlauf gilt: *searchString* = I2OS( *i*, 1).
3. (N209.220) K\_externeWelt {K\_Karte}   
   Schritt 2: In *currentEF* MUSS mittels Use Case aus 14.4.7.1 gesucht werden, wobei als Parameter *recordNumber* = 1 und als Datenteil der Kommandonachricht *searchString* aus Schritt 1 verwendet wird. Die Laufzeit *tsearch,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
4. (N209.230) K\_externeWelt {K\_Karte}   
   Schritt 3: In *currentEF* MUSS mittels Use Case aus 14.4.7.1 gesucht werden, wobei als Parameter *recordNumber* = 1 und als Datenteil der Kommandonachricht *searchString* = ´00´ verwendet wird. Die Laufzeit *tsearch,255* dieses Kommandos MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PSearchRec* | = points( (*tSearchRec*,1, | *tSearchRec*,2, …, | *tSearchRec*,255 ), | *TSearchRec* ) |

Testnachbereitung:

1. (N209.300) K\_externeWelt {K\_Karte}   
   Aus / MF / DF.strukturiert / EF.strukturiert MÜSSEN mittels Use Case aus 14.4.4.1 MÜSSEN alle Rekords entfernt werden.

### Symmetrische Sessionkeyaushandlung für Secure Messaging

In diesem Abschnitt wird lediglich die symmetrische Aushandlung von Sessionkeys für Secure Messaging betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln SK.AES128, SK.AES192 und SK.AES256 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

1. (N210.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N210.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.
3. (N210.130) K\_externeWelt {K\_Karte}   
   Schritt 3: Es MUSS eine leere Menge *MGetStatus* = {} erstellt werden.

Testdurchführung:

1. (N210.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {SK.AES128, SK.AES192, SK.AES256} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.
2. (N210.210) K\_externeWelt {K\_Karte}   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.6, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4SM gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N210.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 6 ausgeführt. Anschließend wird mit Schritt 7 fortgefahren.
4. (N210.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Es wird eine Zufallszahl mittels Use Case aus 14.9.4.2 vom Prüfling abgeholt. Die Laufzeit *trand,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
5. (N210.240) K\_externeWelt {K\_Karte}   
   Schritt 4: Es wird eine erfolgreiche gegenseitige Authentisierung gemäß Use Case aus 14.7.1.2 und (N084.410)a durchgeführt. Die Laufzeit *tAuth,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
6. (N210.250) K\_externeWelt {K\_Karte}   
   Schritt 5: Die Ausführungszeiten werden wie folgt zusammengefasst:   
     *tSK,i* = *trand,i* + *tAuth,i*.
7. (N210.260) Dieser Punkt ist absichtlich leer.
8. (N210.270) K\_externeWelt {K\_Karte}   
   Schritt 7: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PSK4SM,AES128* | = points( (*tSK128*,1, | *tSK128*,2, …, | *tSK128*,100 ), | *TSK4SM,AES128* ) |
| *PSK4SM,AES192* | = points( (*tSK192*,1, | *tSK192*,2, …, | *tSK192*,100 ), | *TSK4SM,AES192* ) |
| *PSK4SM,AES25* | = points( (*tSK56*,1, | *tSK256*,2, …, | *tSK256*,100 ), | *TSK4SM,AES256* ) |

Testnachbereitung:

Keine.

### Schlüsselimport und asymmetrische Authentisierungsprotokolle

In diesem Kapitel wird der Import von Authentisierungsschlüsseln mittels CV–Zertifikaten behandelt, wobei die öffentlichen Schlüssel aller verwendeten CA bei Produktion des Prüflings bekannt sind. Zudem wird die Rollenüberprüfung (eine andere Komponente authentisiert sich gegenüber der eGK) sowie die asymmetrische Aushandlung von Sessionkeys betrachtet.

Der Prüfpunkt beinhaltet einerseits:

* 1. Den Import eines Authentisierungsschlüssels.
  2. Eine Rollenprüfung zur Erlangung eines Sicherheitszustandes.

Zudem wird hier ebenfalls geprüft:

* 1. Der asymmetrische Aufbau eines Trusted–Channels zur Karte.

#### ELC 256

Es werden zehn Test CA benutzt, denen von der Root–CA (siehe PuK.RCA\_ELC256) folgende CV–Zertifikate zugeordnet werden:

|  |  |
| --- | --- |
| Test  CA0 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d0403028641043dc8968bf65157711a3547714230e7d80667d1451897204a9fcd91e5e53b1aeb960a4257e36656f06da638d50d0a4095fbfa11c99c8481da49fe5245a47638855f200844455858586f01127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3740960632d2f613697ccbb05b3b21ae6ddb1fbb146deb5cc3f7787a28d485c2e7042b5001db598eac960055c56e3489c568302ff3e638a8b5bece070995859629a9' |
| Test  CA1 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104068fb3f80118c9a1390b6f3d6d84a5dd5f5c1228e57268afd09297b71c5d8d451879b7097be706fb9a9e6952f599ef0f7eaa13f111d9c1ec656fdc93163ed9575f200844455858586f11127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37406fc05554871d4f0cc60386b500e21975471265ca08a47b932082b596e8be8dbc6e68a729051c47b38db23e16667ff4fae724e39529c7ebd8f05aab249e824c9b' |
| Test  CA2 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d0403028641045fe0d65046d587cf0d494766ed0f4acd646f00820c7d98458df25f49aab891cd2b80e5d51c18719354bf8c8f3a83a41ec8819e894a55a27b8acfe212393f4e685f200844455858586f21127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37404f73ecc1733ce8acba3c539b45ba3ba79829a65d6d1057f3f3732fd32c1920ca8efb4489072cc4943b4b948421b146053f39bd51c5275d534ec395333b9f6430' |
| Test  CA3 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104a2f12f29f8c45da68c2d9d484ef4f764b861934f19c8ad829edad541050b1ad9323d830837144714bac3ff9ffa9b9f2ec5288e93373744ba2923c7afb72e6fde5f200844455858586f31127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f374048a2a0e899ac08817249726f1f13bb983ee5728d3439869953b46b8b69c41d02690ebe1be8a04a5897c82ca04eedc527a016d798f07a776d75fa99f2c7f88e7f' |
| Test  CA4 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104156028235673196c4d54e4b78205c9979d2bdb39362d05568ee5c11671ac9010284bb19e24444b6f174167cd50efd5e6ef9a5780788ac02312f6961dd180e9785f200844455858586f41127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f374081d4965d98f35605fcb6bb6303378787810af9182d2b9fb495b22161a29e553a60ea87f7182c96153dc7ef1dfae79e32ad5dae554eca1bc244ab73f57d76cf14' |
| Test  CA5 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d04030286410462082556bbab7c0387eb3ee7a2767a2277beaf9a02ccdc255a07baf97bd232a74c18c584888ac8f655c05d56ac7afe399c79bfb865e6fe71ee8a76f35257ff0d5f200844455858586f51127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37408db6a883d4e575c1667edd7fc1b3e1b41e7a5d8f73c8ce81b29232f671f4b5178ea62299d06eae35c09794233ad1979b2e263449dbf4c9d68d74331a9a3d94eb' |
| Test  CA6 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d0403028641041932b906841a94133354ef5fb2878d29612e83e3f55b43aceb44d809b1eebecd6c0c515dece9c483273b5f070c3ab2924a65e26900e73506ed46d96be09bffad5f200844455858586f61127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3740948249b88e2d1db43237cdc5c814865aed1ac1f8685cfece3222c2bd169c23c15fe199585d714e74fd468893299f739f80c916e57dff8ffb1747624809122fa3' |
| Test  CA7 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104827bc42213603730397a0d1a25981fbf3d4c4326ebf943784fba310eebd0af39153d457fb56d7dacfe7d3acc0397fa29160c67f0fc11caf4d9074b79213355bc5f200844455858586f71127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37402bca5198dceee074b1a1ed84dddb52c06ea11e190aaa938636844aaba27741207a66ac35471fa81fca9d04dea2e410fcf1757328fe9d8deec645e5707bdb8cad' |
| Test  CA8 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d04030286410427ed1d352e39fcf029f07314414df5a2070587e316f42dee798cf117e9dbfabb56a77fb65169d1014b950eeb12efdcdd6e63d44e351386457b53ec58375d10c65f200844455858586f81127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3740953e492d4fcd165cf0aa658aa0c3ccecf4bf4877ec52acdb3eb1b2ba2810e8f30928563f2ce1e3666dac086ef84c749c29d6dfb21b448c67b435fe85bdfd126d' |
| Test  CA9 | '7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d04030286410437b7e999de3102cfab44162d204b1bcab1d495da3f3c1293db90198b2399f42920d2b716e8e5fceeba0a760df6a27009b4f5b5a51a4dae74c4e0aa03d9ab16515f200844455858586f91127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f374006c12b8bb2affd0e2dc64dc9a51df59ac7d0685b736c58a5af92c0e7c0147e151f937657c6500bb3361c9b504ec39f3867990201ff78d238aa6e797aa74f59c7' |

Test:

1. (N211.110) K\_externeWelt {K\_Karte}   
   Dieser Prüfpunkt arbeitet mit folgenden Parametern:
   1. *MCVC* = {TestCA0, TestCA1, …, TestCA9}
   2. *PuK.RCA =* / MF / DF.Auth / PuK.RCA\_ELC256
   3. *PrK.Auth =* / MF / DF.Auth / PrK.Auth\_ELC256
2. (N211.120) K\_externeWelt {K\_Karte}   
   Die Performanzmessung wird wie folgt durchgeführt:   
   (*MImport*, *MRoleCheck*, *MSesKey*, *MGetSecSta*) = PerformanceAuth(*MCVC*, *PuK.RCA*, *PrK.Auth*)

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PImport,ELC256* | = points( | *MImport* | , | *TImport,ELC256* ) |
| *PRoleCheck,ELC256* | = points( | *MRoleCheck* | , | *TRoleCheck,ELC256* ) |
| *PSesKey,ELC256* | = points( | *MSesKey* | , | *TSesKey,ELC256* ) |
| *PGetSecStat,ELC256* | = points( | *MGetSecStat* | , | *TGetSecStat,ELC256* ) |

#### ELC 384

Es werden zehn Test CA benutzt, denen von der Root–CA (siehe PuK.RCA\_ELC384) folgende CV–Zertifikate zugeordnet werden:

|  |  |
| --- | --- |
| Test  CA0 | '' |
| Test  CA1 | '' |
| Test  CA2 | '' |
| Test  CA3 | '' |
| Test  CA4 | '' |
| Test  CA5 | '' |
| Test  CA6 | '' |
| Test  CA7 | '' |
| Test  CA8 | '' |
| Test  CA9 | '' |

Test:

1. (N211.210) K\_externeWelt {K\_Karte}   
   Dieser Prüfpunkt arbeitet mit folgenden Parametern:
   1. *MCVC* = {TestCA0, TestCA1, …, TestCA9}
   2. *PuK.RCA =* / MF / DF.Auth / PuK.RCA\_ELC384
   3. *PrK.Auth =* / MF / DF.Auth / PrK.Auth\_ELC384
2. (N211.220) K\_externeWelt {K\_Karte}   
   Die Performanzmessung wird wie folgt durchgeführt:   
   (*MImport*, *MRoleCheck*, *MSesKey*, *MGetSecSta*) = PerformanceAuth(*MCVC*, *PuK.RCA*, *PrK.Auth*)

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PImport,ELC384* | = points( | *MImport* | , | *TImport,ELC384* ) |
| *PRoleCheck,ELC384* | = points( | *MRoleCheck* | , | *TRoleCheck,ELC384* ) |
| *PSesKey,ELC384* | = points( | *MSesKey* | , | *TSesKey,ELC384* ) |

1. Das Ergebnis MGetSecSta wird hier nicht weiter verwendet, weil dieser Prüfpunkt bereits in B.8.11.1 betrachtet wird.

#### ELC 512

Es werden zehn Test CA benutzt, denen von der Root–CA (siehe PuK.RCA\_ELC512) folgende CV–Zertifikate zugeordnet werden:

|  |  |
| --- | --- |
| Test  CA0 | '' |
| Test  CA1 | '' |
| Test  CA2 | '' |
| Test  CA3 | '' |
| Test  CA4 | '' |
| Test  CA5 | '' |
| Test  CA6 | '' |
| Test  CA7 | '' |
| Test  CA8 | '' |
| Test  CA9 | '' |

Test:

1. (N211.310) K\_externeWelt {K\_Karte}   
   Dieser Prüfpunkt arbeitet mit folgenden Parametern:
   1. *MCVC* = {TestCA0, TestCA1, …, TestCA9}
   2. *PuK.RCA =* / MF / DF.Auth / PuK.RCA\_ELC512
   3. *PrK.Auth =* / MF / DF.Auth / PrK.Auth\_ELC512
2. (N211.320) K\_externeWelt {K\_Karte}   
   Die Performanzmessung wird wie folgt durchgeführt:   
   (*MImport*, *MRoleCheck*, *MSesKey*, *MGetSecSta*) = PerformanceAuth(*MCVC*, *PuK.RCA*, *PrK.Auth*)

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PImport,ELC512* | = points( | *MImport* | , | *TImport,ELC512* ) |
| *PRoleCheck,ELC512* | = points( | *MRoleCheck* | , | *TRoleCheck,ELC512* ) |
| *PSesKey,ELC512* | = points( | *MSesKey* | , | *TSesKey,ELC512* ) |

1. Das Ergebnis MGetSecSta wird hier nicht weiter verwendet, weil dieser Prüfpunkt bereits in B.8.11.1 betrachtet wird.

#### RSA 2048

Dieses Kapitel ist absichtlich leer.

#### Testablauf Schlüsselimport und asymmetrische Authentisierung

Dieser Abschnitt beschreibt auf generischer Ebene den Ablauf der Performanzmessung für Schlüsselimport (PSO Verify Certificate) sowie für eine asymmetrische Authentisierung mit und ohne Sessionkeyaushandlung. Der Ablauf wird sowohl für RSA als auch für ELC durchlaufen, wobei ELC Schlüssel unterschiedlicher Länge geprüft werden.

|  |  |  |
| --- | --- | --- |
| Input: | MCVC | Menge von CV‑Zertifikaten die einer CA zugeordnet sind |
| PuK.RCA | Sicherheitsanker der PKI, mit welchem sich die Zertifikate in MCVC prüfen lassen. |
| PrK.Auth | privater Schlüssel des Prüflings, der im Rahmen der Sessionkeyaushandlung benötigt wird |
| Output: | MImport | Tupel mit Ausführungszeiten zum Schlüsselimport |
| MRoleCheck | Tupel mit Ausführungszeiten zur Rollenüberprüfung |
| MSesKey | Tupel mit Ausführungszeiten zur Sessionkeyaushandlung |
| MGetSecSta | Tupel mit Ausführungszeiten zur Abfrage Sicherheitszustand |
| Errors: | ‑ | keine |
| Notation: |  | (MImport, MRoleCheck, MSesKey, MGetSecSta ) =    PerformanceAuth( MCVC, PuK.RCA, PrK.Auth ) |

Testvorbereitung:

Von jeder Test\_CA aus *MCVC* werden zehn CV–Zertifikate erzeugt. Insgesamt ergeben sich so einhundert CV–Zertifikate für Authentisierungsschlüssel.

1. (N211.510) K\_externeWelt {K\_eGK}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N211.520) K\_externeWelt {K\_eGK}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

1. (N211.600) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 100–mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 0 bis 12 ausgeführt.
2. (N211.602) K\_externeWelt {K\_Karte}   
   Schritt 0: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1. Dadurch werden alle Elemente aus der Liste *dfSpecificSecurityList* (siehe (N029.900) entfernt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N211.610) K\_externeWelt {K\_Karte}   
   Schritt 1: Aus den einhundert CV–Zertifikaten mit Authentisierungsschlüssel wird ein bislang noch nicht verwendetes gezogen (Ziehen ohne Zurücklegen). Durch die Ziehung wird folgende CV–Zertifikatskette gebildet:   
     *PuK.RCA* 🡪 CVC\_Test\_CAx 🡪 CVC\_ICCy.   
   Die erste Ziehung ist beliebig. Bei allen weiteren Ziehungen MUSS die Nebenbedingung beachtet werden, dass CVC\_Test\_CAx verschieden ist vom unmittelbar vorher verwendeten Zertifikat CVC\_Test\_CAx. Dann MUSS mit Schritt 4 fortgefahren werden.
4. (N211.620) K\_externeWelt {K\_Karte}   
   Schritt 2: MSE Set Kommando gemäß 14.9.9.10, wobei als *keyRef* der Wert *keyIdentifier* aus *PuK.RCA* verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*2,*i* bezeichnet.
5. (N211.630) K\_externeWelt {K\_Karte}   
   Schritt 3: Falls das im vorherigen Schritt selektierte Schlüsselobjekt ein
   1. ELC Schlüssel ist, wird ein PSO Verify Certificate Kommando gemäß 14.8.7.2 verwendet, wobei als Parameter *certificate* CVC\_Test\_CAx verwendet wird.
   2. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*3,*i* bezeichnet.
6. (N211.640) K\_externeWelt {K\_Karte}   
   Schritt 4: MSE Set Kommando gemäß 14.9.9.10, wobei als *keyRef* der Wert CAR aus CVC\_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*4,*i* bezeichnet. Falls dieses Kommando nicht mit NoError beendet wird, fahre mit Schritt 2 fort, sonst mit Schritt 5.
7. (N211.650) K\_externeWelt {K\_Karte}   
   Schritt 5: PSO Verify Certificate Kommando gemäß 14.8.7.1, wobei als Parameter *certificate* CVC\_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*5,*i* bezeichnet.
8. (N211.660) K\_externeWelt {K\_Karte}   
   Schritt 6: MSE Set Kommando gemäß 14.9.9.5, wobei als *keyRef* der Wert CHR aus CVC\_ICCy und als *algId* elcRoleCheck verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*6,*i* bezeichnet.
9. (N211.670) K\_externeWelt {K\_Karte}   
   Schritt 7: Falls im vorherigen Schritt
   1. elcRoleCheck verwendet wurde: Get Challenge Kommando gemäß 14.9.4.2.
   2. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*7,*i* bezeichnet.
10. (N211.680) K\_externeWelt {K\_Karte}   
    Schritt 8: External Authenticate Kommando gemäß 14.7.1.1 und (N084.400){a, b}. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*8,*i* bezeichnet.
11. (N211.690) K\_externeWelt {K\_Karte}   
    Schritt 9: Abfrage eines Sicherheitsstatus:
    1. Falls *MCVC* ELC Schlüssel enthält, dann werden folgende Schritte ausgeführt:
       1. Get Security Status Key gemäß 14.7.3.3 mit *oid* = oid\_cvc\_fl\_ti und *cmdData* = ´0FFF…FF´. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*9.1,*i* bezeichnet.
       2. Get Security Status Key gemäß 14.7.3.3 mit *oid* = oid\_cvc\_fl\_ti und *cmdData* = *flagList* aus CVC\_ICCy. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*9.2,*i* bezeichnet.
    2. Es gilt: *tGetSecStat*,*i* = *tRun*9.1,*i* + *tRun*9.2,*i*.
12. (N211.700) K\_externeWelt {K\_Karte}   
    Schritt 10: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1. Dadurch werden alle Elemente aus der Liste *dfSpecificSecurityList* entfernt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
13. (N211.710) K\_externeWelt {K\_Karte}   
    Schritt 11: MSE Set Kommando gemäß 14.9.9.3, wobei als *keyRef* der Wert *keyIdentifier* von *PrK.Auth* und als *algId* elcSessionkey4SM verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*11,*i* bezeichnet.
14. (N211.720) K\_externeWelt {K\_Karte}   
    Schritt 12: Aushandlung von Sessionkeys:
    1. Falls *MCVC* ELC Schlüssel enthält, dann werden folgende Schritte ausgeführt:

* + 1. General Authenticate gemäß 14.7.2.2.1, wobei als *keyRef* der Wert CHR aus CVC\_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*12.1,*i* bezeichnet.
    2. General Authenticate gemäß 14.7.2.2.2, wobei als *ephemeralPK\_oponent* die Anttwort des vorherigen Kommandos verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit *tRun*12.2,*i* bezeichnet.
    3. Es gilt: *tRun*12,*i* = *tRun*12.1,*i* + *tRun*12.2,*i*.

Testauswertung:

Bei der Bearbeitung einer Schleifeniteration sind folgende Fälle denkbar:

1. Der öffentliche Schlüssel der CA ist bereits in der Karte gespeichert, weil er bei der Kartenproduktion in *persistentPublicKeyList* oder durch einen früheren Zertifikatsimport in *persistentCache* gespeichert wurde. In diesem Fall besteht der *i*–te Schleifendurchlauf aus der Schrittfolge 4, 5, 6, 7, …. Die Laufzeiten der Schritte 4 und 5 werden summiert zu *tImport,i*.
2. Der öffentliche Schlüssel der CA ist nicht in der Karte gespeichert und dies wird bereits während der Schlüsselselektion bemerkt. Dann besteht der *i*–te Schleifendurchlauf aus der Schrittfolge 4, 2, 3, 4, 5, 6, 7, …. Die Laufzeiten der Schritte 4, 2, 3, 4 und 5 werden summiert zu *tImport,i*.
3. Die Laufzeiten der Schritte 6, 7 und 8 werden summiert zu *tRoleCheck,i*.
4. Die Laufzeiten der Schritte 11 und 12 werden aufsummiert zu *tSK*,*i*.
5. Die gemessenen Zeiten werden zu folgenden Tupeln zusammengefasst:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *MImport* | = (*tImport*,1, | *tImport*,2, | …, | *tImport*,100 ) |
| *MRoleCheck* | = (*tRoleCheck*,1, | *tRoleCheck*,2, | …, | *tRoleCheck*,100 ) |
| *MSesKey* | = (*tSesKey*,1, | *tSesKey*,2, | …, | *tSesKey*,100 ) |
| *MGetSecStat* | = (*tGetSecStat*,1, | *tGetSecStat*,2, | …, | *tGetSecStat*,100 ) |

### Internal Authenticate zur Rollenauthentisierung

In diesem Abschnitt wird die Rollenauthentisierung mit privaten RSA und ELC Schlüsseln betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.Auth\_ELC256, PrK.Auth\_ELC384 und PrK.Auth\_ELC512 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

1. (N212.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N212.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

1. (N212.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.Auth\_ELC256, PrK.Auth\_ELC384, PrK.Auth\_ELC512} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.
2. (N212.210) K\_externeWelt {K\_Karte}   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.3, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N212.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 4 ausgeführt.
4. (N212.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Es wird eine Zufallszahl *token* = RAND(16) erzeugt.
5. (N212.240) K\_externeWelt {K\_Karte}   
   Schritt 4: Internal Authenticate gemäß 14.7.4.1 und (N086.900){a, c}. Die Laufzeit *tAuth,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PRoleAuth,ELC256* | = points( (*tAuth*,1, | *tAuth*,2, …, | *tAuth*,100 ), | *TroleAuth,ELC256* ) |
| *PRoleAuth,ELC384* | = points( (*tAuth*,1, | *tAuth*,2, …, | *tAuth*,100 ), | *TroleAuth,ELC384* ) |
| *PRoleAuth,ELC512* | = points( (*tAuth*,1, | *tAuth*,2, …, | *tAuth*,100 ), | *TroleAuth,ELC512* ) |

Testnachbereitung:

Keine.

### PSO Compute Digital Signature mittels signPSS

In diesem Abschnitt wird die Signaturberechnung mit privaten RSA Schlüsseln und dem Signaturverfahren signPSS betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509\_RSA2048 und PrK.X509\_RSA3072 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

1. (N213.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N213.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.
3. (N213.130) K\_externeWelt {K\_Karte}   
   Schritt 3: Generate Asymmetric Key Pair Kommando gemäß 14.9.3.10, wobei als *keyRef* = ´8B´ aus dem Attribut *keyIdentifier* von PrK.X509\_RSA2048 gebildet wird und *operationMode* = ´C0´ gesetzt wird.
4. (N213.140) K\_externeWelt {K\_Karte}   
   Schritt 4: Generate Asymmetric Key Pair Kommando gemäß 14.9.3.10, wobei als *keyRef* = ´8C´ aus dem Attribut *keyIdentifier* von PrK.X509\_RSA3072 gebildet wird und *operationMode* = ´C0´ gesetzt wird.

Testdurchführung:

1. (N213.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509\_RSA2048, PrK.X509\_RSA3072} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.
2. (N213.220) K\_externeWelt {K\_Karte}   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.9, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = signPSS gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N213.230) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Testdurchführung MUSS eine innere Schleife 64-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 4 ausgeführt.
4. (N213.240) K\_externeWelt {K\_Karte}   
   Schritt 3: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *dataToBeSigned* = RAND( *i* ) erzeugt werden.
5. (N213.250) K\_externeWelt {K\_Karte}   
   Schritt 4: PSO Compute Digital Signature gemäß 14.8.2.1 und (N088.600)a. Die Laufzeit *trun,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PsignPSS,2048* | = points( (*trun*,1, | *trun*,2, …, | *trun*,64 ), | *TsignPss,RSA2048* ) |
| *PsignPSS,3072* | = points( (*trun*,1, | *trun*,2, …, | *trun*,64 ), | *TsignPss,RSA3072* ) |

Testnachbereitung:

Keine.

### Signaturerzeugung und –verifikation mittles signECDSA

In diesem Abschnitt wird die Signaturberechnung und –verifiaktion mit privaten ELC Schlüsseln betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509\_ELC256, PrK.X509\_ELC384 und PrK.X509\_ELC512 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

1. (N214.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N214.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

Testdurchführung:

1. (N214.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509\_ELC256, PrK.X509\_ELC384, PrK.X509\_ELC512} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 6 ausgeführt.
2. (N214.210) K\_externeWelt {K\_Karte}   
   Schritt 1: Generate Asymmetric Key Pair Kommando gemäß 14.9.3.10, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *operationMode* = ´C0´ gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N214.220) K\_externeWelt {K\_Karte}   
   Schritt 2: MSE Set Kommando gemäß 14.9.9.9, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = signECDSA gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
4. (N214.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 4 bis 6 ausgeführt.
5. (N214.240) K\_externeWelt {K\_Karte}   
   Schritt 4: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *dataToBeSigned* = RAND(*domainParameter*.τ / 8) erzeugt werden.
6. (N214.250) K\_externeWelt {K\_Karte}   
   Schritt 5: PSO Compute Digital Signature gemäß 14.8.2.1 und (N088.600)c. Die Laufzeit *tsign,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
7. (N214.260) K\_externeWelt {K\_Karte}   
   Schritt 6: PSO Verify Digital Signature gemäß 14.8.9.1 zur erfolgreichen Verifikation der im vorherigen Schritt erzeugten Signatur. Die Laufzeit *tverify,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PsignECDSA,256* | = points( (*tsign*,1, | *tsign*,2, …, | *tsign*,100 ), | *TsignECDSA,ELC256* ) |
| *PsignECDSA,384* | = points( (*tsign*,1, | *tsign*,2, …, | *tsign*,100 ), | *TsignECDSA,ELC384* ) |
| *PsignECDSA,512* | = points( (*tsign*,1, | *tsign*,2, …, | *tsign*,100 ), | *TsignECDSA,ELC512* ) |
| *PverifyECDSA,256* | = points( (*tverify*,1, | *tverify*,2, …, | *tverify*,100 ), | *TverifyECDSA,ELC256* ) |
| *PverifyECDSA,384* | = points( (*tverify*,1, | *tverify*,2, …, | *tverify*,100 ), | *TverifyECDSA,ELC384* ) |
| *PverifyECDSA,512* | = points( (*tverify*,1, | *tverify*,2, …, | *tverify*,100 ), | *TverifyECDSA,ELC512* ) |

Testnachbereitung:

Keine.

### PSO Encipher und PSO Decipher mittels rsaDecipherOaep

In diesem Abschnitt wird die Ver- und Entschlüsselung mit privaten RSA Schlüsseln und dem Entschlüsselungsverfahren rsaDecipherOaep betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509\_RSA2048 und PrK.X509\_RSA3072 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

1. (N215.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N215.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

Testdurchführung:

1. (N215.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509\_RSA2048, PrK.X509\_RSA3072} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 6 ausgeführt.
2. (N215.210) K\_externeWelt {K\_Karte}   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.11, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = rsaDecipherOaep gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N215.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Testdurchführung MUSS eine innere Schleife 190-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt.
4. (N215.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Generate Asymmetric Key Pair Kommando gemäß 14.9.3.10, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *operationMode* = ´C0´ gesetzt wird. Die Laufzeit *tGAKP,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
5. (N215.240) K\_externeWelt {K\_Karte}   
   Schritt 4: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *M* = RAND( *i* ) erzeugt.
6. (N215.250) K\_externeWelt {K\_Karte}   
   Schritt 5: Falls *keyRef* gleich dem *keyIdentifier* von
   1. PrK.X509\_RSA2048 ist, dann PSO Encipher gemäß 14.8.4.1 und (N091.700)c, wobei *PuK* der öffentlich Schlüssel ist, der in Schritt 3 erzeugt wurde und *algID* = rsaEncipherOaep. Die Laufzeit *tenc,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
   2. PrK.X509\_RSA3072 ist, dann wird *M* außerhalb des Prüflings äquivalent zu PSO Encipher gemäß 14.8.4.1 und (N091.700)c verschlüsselt, wobei *PuK* der öffentlich Schlüssel ist, der in Schritt 3 erzeugt wurde und *algID* = rsaEncipherOaep.
7. (N215.260) K\_externeWelt {K\_Karte}   
   Schritt 6: PSO Decipher gemäß 14.8.3.1 und (N090.300)b. Die Laufzeit *tdec,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Penc,2048* | = points( (*tenc*,1, | *tenc*,2, …, | *tenc*,190 ), | *Tenc,RSA2048* ) |
| *Pdec,2048* | = points( (*tdec*,1, | *tdec*,2, …, | *tdec*,190 ), | *Tdec,RSA2048* ) |
| *PGAKP,2048* | = points( (*tGAKP*,1, | *tGAKP*,2, …, | *tGAKP*,190 ), | *TGAKP,RSA2048* ) |
| *Pdec,3072* | = points( (*tdec*,1, | *tdec*,2, …, | *tdec*,190 ), | *Tdec,RSA3072* ) |
| *PGAKP,3072* | = points( (*tGAKP*,1, | *tGAKP*,2, …, | *tGAKP*,190 ), | *TGAKP,RSA3072* ) |

Testnachbereitung:

Keine.

### PSO Encipher und PSO Decipher mittels elcSharedSecretCalculation

In diesem Abschnitt wird die Ver- und Entschlüsselung mit privaten ELC Schlüsseln und dem Entschlüsselungsverfahren elcSharedSecretCalculation betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509\_ELC256, PrK.X509\_ELC384 und PrK.X509\_ELC512 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

1. (N216.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N216.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

Testdurchführung:

1. (N216.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509\_ELC256, PrK.X509\_ELC384, PrK.X509\_ELC512} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 6 ausgeführt.
2. (N216.210) K\_externeWelt {K\_Karte}   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.11, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = elcSharedSecretCalculation gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N216.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Die Testdurchführung MUSS eine innere Schleife 256-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt.
4. (N216.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Generate Asymmetric Key Pair Kommando gemäß 14.9.3.10, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *operationMode* = ´C0´ gesetzt wird. Die Laufzeit *tGAKP,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
5. (N216.240) K\_externeWelt {K\_Karte}   
   Schritt 4: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *M* = RAND( *i* ) erzeugt.
6. (N216.250) K\_externeWelt {K\_Karte}   
   Schritt 5: PSO Encipher gemäß 14.8.4.2, wobei *POB* der öffentlich Schlüssel ist, der in Schritt 3 erzeugt wurde. Die Laufzeit *tenc,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
7. (N216.260) K\_externeWelt {K\_Karte}   
   Schritt 6: PSO Decipher gemäß 14.8.3.2. Die Laufzeit *tdec,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Penc,256* | = points( (*tenc*,1, | *tenc*,2, …, | *tenc*,256 ), | *Tenc,ELC256* ) |
| *Pdec,256* | = points( (*tdec*,1, | *tdec*,2, …, | *tdec*,256 ), | *Tdec,ELC256* ) |
| *PGAKP,256* | = points( (*tGAKP*,1, | *tGAKP*,2, …, | *tGAKP*,256 ), | *TGAKP,ELC256* ) |
| *Penc,384* | = points( (*tenc*,1, | *tenc*,2, …, | *tenc*,256 ), | *Tenc,ELC384* ) |
| *Pdec,384* | = points( (*tdec*,1, | *tdec*,2, …, | *tdec*,256 ), | *Tdec,ELC384* ) |
| *PGAKP,384* | = points( (*tGAKP*,1, | *tGAKP*,2, …, | *tGAKP*,256 ), | *TGAKP,ELC384* ) |
| *Penc,512* | = points( (*tenc*,1, | *tenc*,2, …, | *tenc*,256 ), | *Tenc,ELC512* ) |
| *Pdec,512* | = points( (*tdec*,1, | *tdec*,2, …, | *tdec*,256 ), | *Tdec,ELC512* ) |
| *PGAKP,512* | = points( (*tGAKP*,1, | *tGAKP*,2, …, | *tGAKP*,256 ), | *TGAKP,ELC512* ) |

Testnachbereitung:

Keine.

### Selektieren von Ordnern und Logical Channel Reset

In diesem Abschnitt wird die Selektion eines Ordners und das Rücksetzen des Basiskanals betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.Auth.

Testvorbereitung:

1. (N217.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

1. (N217.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt.
2. (N217.210) K\_externeWelt {K\_Karte}   
   Schritt 1: *currentFolder* MUSS gemäß 14.2.6.9 auf / MF / DF.Auth gesetzt werden. Die Laufzeit *tselect,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
3. (N217.220) K\_externeWelt {K\_Karte}   
   Schritt 2: Get Challenge wird gemäß 14.9.4.2 ausgeführt. Die Laufzeit *trnd,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
4. (N217.230) K\_externeWelt {K\_Karte}   
   Schritt 3: Der Basiskanal MUSS gemäß Use Case aus 14.9.8.3 zurückgesetzt werden, wobei der Kommandoparameter *logicalChannelNumber* gleich null zu setzen ist. Die Laufzeit *treset,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Pselect\_DF* | = points( (*tselect*,1, | *tselect*,2, …, | *tselect*,100 ), | *Tselect\_DF* ) |
| *Pchallenge* | = points( (*trnd*,1, | *trnd*,2, …, | *trnd*,100 ), | *Tchallenge* ) |
| *Preset\_Ch* | = points( (*treset*,1, | *treset*,2, …, | *treset*,100 ), | *Treset\_Ch* ) |

Testnachbereitung:

Keine.

### Manage Security Environment

In diesem Abschnitt wird die Selektion kryptographishcer Objekte betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.IAS.

Testvorbereitung:

1. (N218.110) K\_externeWelt {K\_Karte}   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N218.120) K\_externeWelt {K\_Karte}   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.
3. (N218.130) K\_externeWelt {K\_Karte}   
   Schritt 3: Es MUSS eine leere Menge *MSet* = {} erstellt werden.

Testdurchführung:

1. (N218.200) K\_externeWelt {K\_Karte}   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.
2. (N218.210) K\_externeWelt {K\_Karte}   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.3, wobei als *keyRef* PrK.X509\_ELC256.*keyIdentifier* verwendet und *algId* = gesetzt wird. Die Laufzeit *tSet* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *MSet* hinzugefügt werden.
3. (N218.220) K\_externeWelt {K\_Karte}   
   Schritt 2: MSE Set Kommando gemäß 14.9.9.9, wobei als *keyRef* PrK.X509\_ELC384.*keyIdentifier* verwendet und *algId* = signECDSA gesetzt wird. Die Laufzeit *tSet* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *MSet* hinzugefügt werden.
4. (N218.230) K\_externeWelt {K\_Karte}   
   Schritt 3: MSE Set Kommando gemäß 14.9.9.11, wobei als *keyRef* PrK.X509\_ELC512.*keyIdentifier* verwendet und *algId* = elcSharedSecretCalculation gesetzt wird. Die Laufzeit *tSet* dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge *MSet* hinzugefügt werden.
5. (N218.240) K\_externeWelt {K\_Karte}   
   Schritt 4: MSE Restore Kommando gemäß 14.9.9.1, wobei *seNo* = 1 gesetzt wird. Die Laufzeit *trestore,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PMSE\_Set* | = points( | *MSet* | , | *TMSE\_Set*) |
| *PMSE\_Restore* | = points( (*trestore*,1, | *trestore*,2, …, | *trestore*,100 ), | *TMSE\_restore* ) |

Testnachbereitung:

Keine.

### General Authenticate, PACE

Dieser Prüfpunkt ist nur relevant, wenn Option\_kontaktlose\_Schnittstelle vorhanden ist.

In diesem Abschnitt wird die Etablierung mit einem symmetrischen Kartenverbindungsobjekt betrachtet. Dieser Prüfpunkt arbeitet mit dem Objekt / MF / DF.LCS / CAN\_256.

Testvorbereitung:

1. (N219.110) K\_externeWelt {K\_Karte}, Option\_kontaktlose\_Schnittstelle   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N219.120) K\_externeWelt {K\_Karte}, Option\_kontaktlose\_Schnittstelle   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.LCS gesetzt werden.

Testdurchführung:

1. (N219.200) K\_externeWelt {K\_Karte}, Option\_kontaktlose\_Schnittstelle   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.
2. (N219.210) K\_externeWelt {K\_Karte}, Option\_kontaktlose\_Schnittstelle   
   Schritt 1: MSE Set Kommando gemäß 14.9.9.7 zur Auswahl von CAN\_256. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N219.220) K\_externeWelt {K\_Karte}, Option\_kontaktlose\_Schnittstelle   
   Schritt 2: Etablierung eines vertrauenswürdigen Kanals gemäß 14.7.2.1. Die Laufzeit aller daran beteiligten Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und zu *tPACE,i* addiert werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PPACE* | = points( (*tPACE*,1, | *tPACE*,2, …, | *tPACE*,100 ), | *TPACE* ) |

Testnachbereitung:

Keine.

### Symmetrische Sessionkeyaushandlung für Trusted Channel

Dieser Prüfpunkt ist nur relevant, wenn Option\_Kryptobox vorhanden ist.

In diesem Abschnitt wird lediglich die symmetrische Aushandlung von Sessionkeys für Secure Messaging betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln TC.AES128, TC.AES192 und TC.AES256 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

1. (N220.110) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N220.120) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

1. (N220.200) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {TC.AES128, TC.AES192, TC.AES256} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.
2. (N220.210) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 1: MSE Set Kommandos gemäß 14.9.9.2 und 14.9.9.4, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4TC gesetzt wird. Die Laufzeiten dieser Kommandos sind für diesen Prüfpunkt irrelevant.
3. (N220.215) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 2: MSE Set Kommando gemäß 14.9.9.4, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4TC gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
4. (N220.220) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 3: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 4 bis 6 ausgeführt. Anschließend wird mit Schritt 7 fortgefahren.
5. (N220.230) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 4: Es wird ein Internal Authenticate Kommando gemäß 14.7.4.1 und (N086.902)a ausgeführt. Die Laufzeit *tIntAuth,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
6. (N220.240) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 5: Ein erfolgreiches External Authenticate Kommando 14.7.1.1 und (N084.402)a wird durchgeführt. Die Laufzeit *tExtAuth,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
7. (N220.250) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 6: Die Ausführungszeiten werden wie folgt zusammengefasst:   
     *tSK,i* = *tIntAuth,i* + *tExtAuth,i*.
8. (N220.260) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 7: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PSK4TC,AES128* | = points( (*tSK128*,1, | *tSK128*,2, …, | *tSK128*,100 ), | *TSK4TC,AES128* ) |
| *PSK4TC,AES192* | = points( (*tSK192*,1, | *tSK192*,2, …, | *tSK192*,100 ), | *TSK4TC,AES192* ) |
| *PSK4TC,AES256* | = points( (*tSK256*,1, | *tSK256*,2, …, | *tSK256*,100 ), | *TSK4TC,AES256* ) |

Testnachbereitung:

Keine.

### Sessionkeynutzung im Trusted Channel

Dieser Prüfpunkt ist nur relevant, wenn Option\_Kryptobox vorhanden ist.

In diesem Abschnitt wird die Nutzung von Sessionkeys im Rahmen eines Trusted Channels betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln TC.AES128, TC.AES192 und TC.AES256 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

1. (N221.110) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N221.120) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

1. (N221.200) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {TC.AES128, TC.AES192, TC.AES256} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 9 ausgeführt.
2. (N221.210) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 1: Etablierung von Sessionkeys zur Nutzung in einem Trusted Channel gemäß der Schritte (N220.210) bis (N220.240). Die Laufzeit aller dabei verwendeten Kommandos ist für diesen Prüfpunkt irrelevant.
3. (N221.220) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 2: Die Testdurchführung MUSS eine innere Schleife 1000-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 8 ausgeführt. Anschließend wird mit Schritt 9 fortgefahren.
4. (N221.230) K\_externeWelt {K\_Karte}   
   Schritt 3: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *cmdData* = RAND( *i* ) erzeugt werden.
5. (N221.240) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 4: Es wird ein PSO Encipher Kommando gemäß 14.8.4.5 und (N091.650)c.1 ausgeführt, wobei als Kommandoparameter *M* der im vorherigen Schritt erzeugte Oktettstring *cmdData* verwendet wird. Die Laufzeit *tenc,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
6. (N221.250) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 5: Es wird ein PSO Compute Cryptographic Checksum Kommando gemäß 14.8.1.1 und (N087.248)a durchgeführt, wobei als Kommandoparameter *data* das Chiffrat *cipher* aus der Antwortnachricht des vorherigen Kommandos verwendet wird. Die Laufzeit *tcomputeCC,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
7. (N221.260) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 6: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *rspData* = RAND( *i* ) erzeugt werden. Diese Daten werden so verschlüsselt, dass das Chiffrat *C* im folgenden PSO Decipher Kommando erfolgreich entschlüsselt wird. Zum Chiffrat *C* wird ein MAC *mac* so berechnet, dass dieser im folgenden PSO Verify Cryptographic Checksum erfolgreich verifiziert wird.
8. (N221.270) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 7: Es wird ein PSO Verify Cryptographic Checksum Kommando gemäß 14.8.8.1 durchgeführt, wobei im inputTemplate das Chiffrat C in den MAC mac aus dem vorherigen Schritt enthält. Die Laufzeit *tverifyCC,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
9. (N221.280) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
   Schritt 8: Es wird ein PSO Decipher Kommando gemäß 14.8.3.3 und (N090.302)a durchgeführt, wobei das Chiffrat *C* aus Schritt 6 als Kommandoparameter verwendet wird. Die Laufzeit *tdec,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
10. (N221.290) K\_externeWelt {K\_Karte}, Option\_Kryptobox   
    Schritt 9: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

Testauswertung:

Durch die Messpunkt (*x*, *y*) 🡪 (*i*, *ta,i*) wird eine Ausgleichsgerade *y* = *m* *x* + *b* gelegt.

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Pcompute128,b* | = points( | (*b*, *b*) | , | *Tcompute128,b* ) |
| *Pcompute128,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tcompute128,b* ) |
| *Pdec128,b* | = points( | (*b*, *b*) | , | *Tde128c,b* ) |
| *Pdec128,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tdec128,b* ) |
| *Penc128,b* | = points( | (*b*, *b*) | , | *Tenc128,b* ) |
| *Penc128,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tenc128,b* ) |
| *Pverify128,b* | = points( | (*b*, *b*) | , | *Tverify128,b* ) |
| *Pverify128,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tverify128,b* ) |
| *Pcompute192,b* | = points( | (*b*, *b*) | , | *Tcompute192,b* ) |
| *Pcompute192,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tcompute192,b* ) |
| *Pdec192,b* | = points( | (*b*, *b*) | , | *Tde192c,b* ) |
| *Pdec192,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tdec192,b* ) |
| *Penc192,b* | = points( | (*b*, *b*) | , | *Tenc192,b* ) |
| *Penc192,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tenc192,b* ) |
| *Pverify192,b* | = points( | (*b*, *b*) | , | *Tverify192,b* ) |
| *Pverify192,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tverify192,b* ) |
| *Pcompute256,b* | = points( | (*b*, *b*) | , | *Tcompute256,b* ) |
| *Pcompute256,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tcompute256,b* ) |
| *Pdec256,b* | = points( | (*b*, *b*) | , | *Tde256c,b* ) |
| *Pdec256,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tdec256,b* ) |
| *Penc256,b* | = points( | (*b*, *b*) | , | *Tenc256,b* ) |
| *Penc256,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tenc256,b* ) |
| *Pverify256,b* | = points( | (*b*, *b*) | , | *Tverify256,b* ) |
| *Pverify256,m* | = points( | (1000 *m*, 1000 *m*) | , | *Tverify256,b* ) |

1. Die Steigung m der Ausgleichsgeraden gibt die Verarbeitungsrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von Pm aufgeführt ist, rechnet dies um in Sekunden pro kByte.

Testnachbereitung:

Keine.

### Get Random

Dieser Prüfpunkt ist nur relevant, wenn Option\_logische\_Kanäle**~~Fehler! Es wurde kein Textmarkenname vergeben.~~**vorhanden ist.

In diesem Abschnitt wird die Erzeugung einer sicheren Zufallszahl betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

1. (N222.110) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle**~~Fehler! Es wurde kein Textmarkenname vergeben.~~**   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

1. (N222.200) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle**~~Fehler! Es wurde kein Textmarkenname vergeben.~~**   
   Die Testdurchführung MUSS eine Schleife 256-mal durchlaufen. In jedem Schleifendurchlauf wird Schritt 1 ausgeführt.
2. (N222.210) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle**~~Fehler! Es wurde kein Textmarkenname vergeben.~~**   
   Schritt 1: Get Random wird gemäß 14.9.5.1 ausgeführt, wobei im *i*-ten Schleifendurchlauf *i* zufällige Oktette vom Prüfling zu generieren sind. Die Laufzeit *trnd,i* dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Durch die Messpunkt (*x*, *y*) 🡪 (*i*, *trnd,i*) wird eine Ausgleichsgerade *y* = *m* *x* + *b* gelegt.

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *PRandom,b* | = points( | (*b*, *b*) | , | *TRandom,b* ) |
| *PRandom,m* | = points( | (1000 *m*, 1000 *m*) | , | *TRandom,m* ) |

1. Die Steigung m der Ausgleichsgeraden gibt die Erzeugungsrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von Pm aufgeführt ist, rechnet dies um in Sekunden pro kByte.

Testnachbereitung:

Keine.

### Öffnen und Schließen logischer Kanäle

Dieser Prüfpunkt ist nur relevant, wenn Option\_logische\_Kanäle vorhanden ist.

In diesem Abschnitt wird das Öffnen, Schließen und Rücksetzen logischer Kanäle betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

1. (N223.110) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle   
   Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
2. (N223.120) K\_externeWelt {K\_Karte}   
   Schritt 2: Es MÜSSEN folgende leere Mengen erstellt werden:
   1. *MOpen* = {},
   2. *MClose* = {},
   3. *MRST* = {}.

Testdurchführung:

1. (N223.200) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle   
   Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.
2. (N223.210) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle   
   Schritt 1: Zusätzlich zum Basiskanal MÜSSEN mittels Manage Channel gemäß 14.9.8.1 drei weitere logische Kanäle geöffnet werden. Die Laufzeit dieser drei Kommandos MUSS gemäß B.5.1 gemessen und in die Menge *MOpen* eingestellt werden.
3. (N223.220) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle   
   Schritt 2: Die zusätzlich zum Basiskanal geöffneten logischen Kanäle MÜSSEN mittels Manage Channel gemäß 14.9.8.2 geschlossen werden, wobei die Reihenfolge, in welcher die Kanäle geschlossen werden zufällig bestimmt wird.. Die Laufzeit dieser drei Kommandos MUSS gemäß B.5.1 gemessen und in die Menge *MClose* eingestellt werden.
4. (N223.230) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle   
   Schritt 3: Zusätzlich zum Basiskanal MÜSSEN mittels Manage Channel gemäß 14.9.8.1 drei weitere logische Kanäle geöffnet werden. Die Laufzeit dieser drei Kommandos MUSS gemäß B.5.1 gemessen und in die Menge *MOpen* eingestellt werden.
5. (N223.240) K\_externeWelt {K\_Karte}, Option\_logische\_Kanäle   
   Schritt 4: Die Applikationsebene MUSS mittels Manage Channel gemäß 14.9.8.4 zurückgesetzt werden. Die Laufzeit dieses Kommandos MUSS gemäß B.5.1 gemessen und in die Menge *MRST* eingestellt werden.

Testauswertung:

Es gilt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *POpen* | = points( | *MOpen* | , | *TOpen* ) |
| *PClose* | = points( | *MClose* | , | *TClose* ) |
| *PRST* | = points( | *MRST* | , | *TRST* ) |

Testnachbereitung:

Keine.

## Kartenkonfiguration für Performanztests (normativ)

Dieser Abschnitt beschreibt die Konfiguration des Prüflings für den Performanztest. Dabei ist es zulässig, die gesamte Konfiguration so auf mehrere Images aufzuteilen, dass in den einzelnen Images einige Anwendungen fehlen. Wegen des Speicherbedarfes von DF.strukturiert und DF.transparent ist dies gegebenenfalls erforderlich.

### Attribute des Objektsystems

1. (N251.100) K\_Personalisierung   
   Das Objektsystem gemäß (N019.900) MUSS folgende Attribute enthalten:
   1. Der Wert des Attributes *root* MUSS die Anwendung gemäß B.9.3 sein.
   2. Der Wert des Attributes *answerToReset* MUSS gemäß B.9.1.1 sein.
   3. Der Wert des Attributes *iccsn8* MUSS identisch zu den letzten acht Oktetten im *body* von / MF / EF.GDO sein.
   4. Das Attribut *applicationPublicKeyList* MUSS alle Schlüssel der folgenden Menge enthalten: {   
         / MF / DF.Auth / PuK.RCA\_ELC256,   
         / MF / DF.Auth / PuK.RCA\_ELC384,   
         / MF / DF.Auth / PuK.RCA\_ELC512}
   5. In *persistentCache* MUSS Platz für mindestens zehn CA-Schlüssel sein.
   6. Das Attribut *pointInTime* MUSS den Wert 2012.08.22 = ´010200080202´ besitzen.

#### Answer To Reset

1. (N251.200) K\_Personalisierung   
   Für das Attribut *answerToRest* MUSS gelten:
   1. Der *answerToRest* MUSS den Vorgaben aus Tabelle 280 entsprechen.
   2. Der ATR SOLL ein TC1 Byte mit dem Wert ´FF´ enthalten. In diesem Fall MUSS T0 auf den Wert ´Dx´ gesetzt werden.
   3. Die Historical Bytes MÜSSEN gemäß [ISO/IEC 7816-4] codiert werden.

Tabelle 280: ATR Codierung

| Zeichen | Wert | Bedeutung |
| --- | --- | --- |
| TS | ´3B´ | Initial Character (direct convention) |
| T0 | ´9x´ | Format Character (TA1/TD1 indication, x = no. of HB) |
| TA1 | ´xx´ | Interface Character (FI/DI, erlaubte Werte: siehe (N024.100)) |
| TD1 | ´81´ | Interface Character, (T=1, TD2 indication) |
| TD2 | ´B1´ | Interface Character, (T=1, TA3/TB3/TD3 indication) |
| TA3 | ´FE´ | Interface Character (IFSC coding) |
| TB3 | ´45´ | Interface Character, (BWI/CWI coding) |
| TD3 | ´1F´ | Interface Character, (T=15, TA4 indication) |
| TA4 | ´xx´ | Interface Character (XI/UI coding) |
| Ti | HB | Historical Bytes (HB, imax. = 15) |
| TCK | XOR | Check Character (exclusive OR) |

#### Allgemeine Festlegungen zu Attributstabellen

1. (N252.100) K\_Personalisierung   
   Für Zugriffsbedingungen gilt:
   1. Sofern nicht anders angegeben MUSS die Zugriffsbedingung für alle in diesem Dokument genannten Kommandos und Kommandovarianten ALWAYS sein und zwar für alle unterstützen Security Environments, alle unterstützten Schnittstellen und alle Life Cycle Status.
   2. Unterstützt ein COS weitere Kommandos oder Kommandovarianten, so KÖNNEN die Zugriffsbedingungen dafür herstellerspezifisch sein.
2. (N252.200) K\_Personalisierung   
   SE abhängige Attribute:
   1. Alle Objekte MÜSSEN sich in SE#1 wie angegeben verwenden lassen.
   2. Objekte KÖNNEN in anderen SE verwendet werden
   3. Falls Objekte in anderen SE verwendbar sind, dann MÜSSEN sie dort dieselben Eigenschaften wie in SE#1 besitzen.
3. (N252.300) K\_Personalisierung   
   Enthält eine Tabelle mit Ordnerattributen
   1. keinen *applicationIdentifier* (AID), so KANN diesem Ordner herstellerspezifisch ein beliebiger AID zugeordnet werden.
   2. einen oder mehrere AID, dann MUSS sich dieser Ordner mittels aller angegebenen AID selektieren lassen.
   3. keinen *fileIdentifier* (FID),
      1. so DARF dieser Ordner NICHT mittels eines *fileIdentifier* aus dem Intervall gemäß 8.1.1 selektierbar sein, es sei denn, es handelt sich um den Ordner *root*, dessen optionaler *fileIdentifier* den Wert ´3F00´ besitzen MUSS.
      2. so KANN diesem Ordner ein beliebiger *fileIdentifier* außerhalb des Intervalls gemäß 8.1.1 zugeordnet werden.
4. (N252.400) K\_Personalisierung   
   Enthält eine Tabelle mit Attributen einer Datei keinen *shortFileIdentifier*, so DARF sich dieses EF NICHT mittels *shortFileIdentifier* aus dem Intervall gemäß 8.1.2 selektieren lassen.

#### *Root*, die Wurzelapplikation

1. (N253.050) K\_Personalisierung   
   Die Anwendung *root* MUSS die in Tabelle 281 dargestellten Attribute besitzen.

Tabelle 281: Attribute / MF

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | Ein Wert aus der Menge {Application, ADF} |  |
| *applicationIdentifier* | ´F000 0000 03´ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *fileIdentifier* | *objektType* = Application 🡪 kein *fileIdentifier*  *objectType* = ADF 🡪 *fileIdentifier* = ´3F 00´ |  |
| *children* | * / MF / DF.Auth * / MF / DF.IAS * / MF / DF.LCS * / MF / DF.SelectEF * / MF / DF.strukturiert * / MF / DF.transparent * / MF / EF.ATR * / MF / EF.DIR * / MF / EF.GDO |  |

#### / MF / EF.ATR

1. (N253.110) K\_Personalisierung   
   Die Datei EF.ATR MUSS die in Tabelle 282 dargestellten Attribute besitzen.

Tabelle 282: Attribute / MF / EF.ATR

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | transparentes Elementary File |  |
| *fileIdentifier* | ´2F 01´ |  |
| *shortFileIdentifier* | ´1D´= 29 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | False |  |
| *flagChecksum* | True |  |
| *numberOfOctet* | herstellerspezifisch |  |
| *positionLogicalEndOfFile* | herstellerspezifisch |  |
| *body* | ´XX…YY´ | siehe unten |

Für das Attribut *body* gelten folgende Festlegungen:

1. (N253.120) K\_Personalisierung   
   Der Oktettstring *body* MUSS DER–TLV codierte Datenobjekte (DO) enthalten, welche lückenlos hintereinander konkateniert werden MÜSSEN.
2. (N253.130) K\_Personalisierung   
   In *body* MUSS an erster Stelle genau ein DO\_BufferSize mit folgenden Eigenschaften enthalten sein:
   1. Tag = ´E0´.
   2. DO\_Buffersize MUSS genau vier DO mit einem Tag ´02´ enthalten.
   3. Das erste DO mit Tag ´02´ gibt die maximale Anzahl der Oktette in einer ungesicherten Kommando APDU an.
   4. Das zweite DO mit Tag ´02´ gibt die maximale Anzahl der Oktette in einer ungesicherten Antwort an.
   5. Das dritte DO mit Tag ´02´ gibt die maximale Anzahl der Oktette in einer gesicherten Kommando APDU an.
   6. Das vierte DO mit Tag ´02´ gibt die maximale Anzahl der Oktette in einer gesicherten Antwort an.
3. (N253.140) K\_Personalisierung   
   In *body* KÖNNEN weitere DER–TLV codierte Datenobjekte enthalten sein.

#### / MF / EF.DIR

Die Datei EF.DIR enthält eine Liste mit Anwendungstemplates gemäß [ISO/IEC 7816-4]. Diese Liste wird dann angepasst, wenn sich die Applikationsstruktur durch Löschen oder Anlegen von Anwendungen verändert.

1. (N253.210) K\_Personalisierung   
   Die Datei EF.DIR MUSS die in Tabelle 283 dargestellten Attribute besitzen.
2. (N253.220) K\_Personalisierung   
   Für jede im Objektsystem vorhandene Anwendung MUSS ein eigener Rekord in EF.DIR enthalten sein, der diese Anwendung beschreibt.

Tabelle 283: Attribute / MF / EF.DIR

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | linear variables Elementary File |  |
| *fileIdentifier* | ´2F 00´ |  |
| *shortFileIdentifier* | ´1E´= 30 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | True |  |
| *flagChecksum* | True |  |
| *maximumNumberOfRec.* | 10 Rekord |  |
| *maximumRecordLength* | 36 Oktett |  |
| *flagRecordLifeCycleStatus* | False |  |
| *numberOfOctet* | ´00BE´ Oktett = 190 Oktett |  |
| *recordList*    Rekord 1      Rekord 2   … | ´61‑L61‑{ 4F‑05‑F000000000   50‑00   53‑L53‑COS\_Identifier }´ ´61‑L61‑{4F‑L4F‑AID}´ … | *root*, siehe B.9.3    weitere Anwendungstemplates |

1. Der Oktettstring COS\_Identifier wird von der gematik herstellerspezifisch festgelegt.

#### / MF / EF.GDO

1. (N253.310) K\_Personalisierung   
   Die Datei EF.GDO MUSS die in Tabelle 284 dargestellten Attribute besitzen.

Tabelle 284: Attribute / MF / EF.GDO

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | transparentes Elementary File |  |
| *fileIdentifier* | ´2F 02´ |  |
| *shortFileIdentifier* | ´02´= 2 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | False |  |
| *flagChecksum* | True |  |
| *numberOfOctet* | ´000C´ Oktett = 12 Oktett |  |
| *positionLogicalEndOfFile* | 12 |  |
| *body* | ´5A‑0A‑80276…´ | wird personalisiert |

1. (N253.320) K\_Personalisierung   
   In *body* MUSS genau ein DER–TLV codiertes Datenobjekt DO\_ICCSN mit folgenden Eigenschaften enthalten sein:
   1. Tag = ´5A´ und Längenfeld = ´0A´.
   2. Für das Wertfeld MUSS gelten:
      1. Das erste Oktett MUSS den Major Industry Identifer (MII) mit dem Wert ´80´ enthalten.
      2. Die nächsten drei Nibble MÜSSEN den Country Code Deutschlands mit dem Wert ´276´enthalten.
      3. Die nächsten fünf Nibble MÜSSEN den Issuer Identifer enthalten.
      4. Die restlichen fünf Oktette MÜSSEN BCD codiert eine Seriennummer enthalten.

### Anwendung für Authentisierungsprotokolle, DF.Auth

1. (N254.005) K\_Personalisierung   
   Die Anwendung DF.Auth MUSS die in Tabelle 285 dargestellten Attribute besitzen.

Tabelle 285: Attribute / MF / DF.Auth

| **Attribute** | **Wert** | **Bemerkung** |
| --- | --- | --- |
| *objectType* | Ordner |  |
| *applicationIdentifier* | ´F000 0000 04´ |  |
| *fileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *children* | * / MF / DF.Auth / PrK.Auth\_ELC256 * / MF / DF.Auth / PrK.Auth\_ELC384 * / MF / DF.Auth / PrK.Auth\_ELC512 * / MF / DF.Auth / PuK.RCA\_ELC256 * / MF / DF.Auth / PuK.RCA\_ELC384 * / MF / DF.Auth / PuK.RCA\_ELC512 * / MF / DF.Auth / SK.AES128 * / MF / DF.Auth / SK.AES192 * / MF / DF.Auth / SK.AES256 * / MF / DF.Auth / TC.AES128 * / MF / DF.Auth / TC.AES192 * / MF / DF.Auth / TC.AES256 |  |

* + - 1. / MF/ DF.Auth / PrK.Auth\_ELC256

1. (N254.010) K\_Personalisierung   
   Der Schlüssel PrK.Auth\_ELC256 MUSS die in Tabelle 286 dargestellten Attribute besitzen.

Tabelle 286: Attribute / MF / DF.Auth / PrK.Auth\_ELC256

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates Authentisierungsobjekt, ELC256 |
| *keyIdentifier* | ´11´ = 17 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP256r1, *privateElcKey.d* = ´3b47afc75fafcf62ea3546efbe0b8d4a9295892e19acf562556441c34f374810´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE:  *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSessionkey4SM} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:   
   ´040597cf45e706318271161744c8a2df37daf696c3feb0e3244de79d3a7472d8eb07d37f751e9494d4e9b12f9cea5a2105724f742d1e444a5553ec26da4e0556e9´.
   * + 1. / MF/ DF.Auth / PrK.Auth\_ELC384
2. (N254.020) K\_Personalisierung   
   Der Schlüssel PrK.Auth\_ELC384 MUSS die in Tabelle 287 dargestellten Attribute besitzen.

Tabelle 287: Attribute / MF / DF.Auth / PrK.Auth\_ELC384

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates Authentisierungsobjekt, ELC384 |
| *keyIdentifier* | ´12´ = 18 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP384r1, *privateElcKey.d* = ´5055f4c2260871efbf7eac69119597bb582be3210df5075629311a065d1328720f1982dc9d99ea1a4f3f4ad16c857ce8´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE:  *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSessionkey4SM} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:   
   ´04808efcb3b017f52f46bafbca6ad5860d0f378ffe260edb0cbab2eff3f57c93c006958e0648e824c8fc211b9d33d564d70698419b30919ce79cdee85b06cd445146f254faa5df04debb56bf45355728b3e8a0d90560b002601d74c50f78d67aa6´.
   * + 1. / MF/ DF.Auth / PrK.Auth\_ELC512
2. (N254.030) K\_Personalisierung   
   Der Schlüssel PrK.Auth\_ELC512 MUSS die in Tabelle 288 dargestellten Attribute besitzen.

Tabelle 288: Attribute / MF / DF.Auth / PrK.Auth\_ELC512

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates Authentisierungsobjekt, ELC512 |
| *keyIdentifier* | ´13´ = 19 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP512r1, *privateElcKey.d* = ´4ef41d5494649f0a214dded44a77d617d41ca1f56795038c70e1b222ab6d6d703c61dd6a6a2fa26e79db5848dedf1208ad2180afab576b23f23b31084e03edc4´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE:  *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSessionkey4SM} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:   
   ´0488ef0a9023aae89e687c9ab56b1e16d9dc5bdb7b4773dc1f883c4f257e917faba22c2f8f1c20f9c283aa8197d70c592b2db4abcace1acf26a0f83f8ccd71665b88d5cb73df9b8ed006cd463683e00a0bbc982c0f470543d4bf7190d13c8ecd21cc1b83db0421faa737d2fad35e0391d48ecd11b25b3584fb691e66fd8786ab78´.
   * + 1. / MF/ DF.Auth / PrK.Auth\_RSA2048

Dieses Kapitel ist absichtlich leer.

1. (N254.040) Dieser Punkt ist absichtlich leer.
   * + 1. / MF/ DF.Auth / PuK.RCA\_ELC256
2. (N254.050) K\_Personalisierung   
   Der Schlüssel PuK.RCA\_ELC256 MUSS die in Tabelle 289 dargestellten Attribute besitzen.

Tabelle 289: Attribute / MF / DF.Auth / PuK.RCA\_ELC256

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | Öffentliches Signaturprüfobjekt, ELC256 |
| *keyIdentifier* | ´4445585858600112´ |
| *lifeCycleStatus* | „Operational state (active)“ |
| *publicKey* | *domainParameter* gemäß brainpoolP256r1, P = ´0434dc65068c3633671828f73adeb8fbc01952e3de3511823448f15afc28ebea3829be1ac9d4d10869ce5fa675275087603e4e9fc86ff3c127b9e94a14c670164d´ |
| *oid* | ecdsa-with-SHA256 |
| *CHAT* | * OIDflags = oid\_cvc\_fl\_ti * flagList = ´FF FFFF FFFF FFFF´ |
| *expirationDate* | 2023.04.15: YYMMDD = ´0203 0004 0105´ |

1. Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = ´74320bea049777bc663c27cfb2c488b7a49b05af348d9186f86a7196c2f6bb43´.
   * + 1. / MF/ DF.Auth / PuK.RCA\_ELC384
2. (N254.060) K\_Personalisierung   
   Der Schlüssel PuK.RCA\_ELC384 MUSS die in Tabelle 290 dargestellten Attribute besitzen.

Tabelle 290: Attribute / MF / DF.Auth / PuK.RCA\_ELC384

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | Öffentliches Signaturprüfobjekt, ELC384 |
| *keyIdentifier* | ´4445585858600212´ |
| *lifeCycleStatus* | „Operational state (active)“ |
| *publicKey* | *domainParameter* gemäß brainpoolP384r1, P = ´043e84af62d1330a9a6f04a2791e0935e7cfe01eb8f7ae8d96eef77c036c0569928e53700322a28454e647c85ea9c89c1d3051ce803c21e5c27a73bd8da121524d4679ded5144368523855e522cfe9a7d1ba31e691900cd02740c399ffcc10f260´ |
| *oid* | ecdsa-with-SHA384 |
| *CHAT* | * OIDflags = oid\_cvc\_fl\_ti * flagList = ´F F FFFF FFFF FFFF´ |
| *expirationDate* | 2023.04.15: YYMMDD = ´0203 0004 0105´ |

1. Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = ´3dbf52550829cec527f91de05fc3d70b47c38d1c8504623174c912afc7941afbe104b9b3d34a4eb0dd35452e2b67884a´.
   * + 1. / MF/ DF.Auth / PuK.RCA\_ELC512
2. (N254.070) K\_Personalisierung   
   Der Schlüssel PuK.RCA\_ELC512 MUSS die in Tabelle 291 dargestellten Attribute besitzen.

Tabelle 291: Attribute / MF / DF.Auth / PuK.RCA\_ELC512

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | Öffentliches Signaturprüfobjekt, ELC512 |
| *keyIdentifier* | ´4445585858600312´ |
| *lifeCycleStatus* | „Operational state (active)“ |
| *publicKey* | *domainParameter* gemäß brainpoolP512r1, P = ´043b5e694c154f4dd0812a91273869d382f5c81748ef5797416369b445f60dd2480d2400d1e7dd896b054ba615d8927fb10911e02753a22b1f6ec1ba18d0c1fdaa905712f33003bec6a51c179006e7b98f8d1d1219db49944f05602b46ac511570b136907081a5f93a0e000b7f335f76a7f8c18584fc7a7d11fddc96844ff8cb5b´ |
| *oid* | ecdsa-with-SHA512 |
| *CHAT* | * OIDflags = oid\_cvc\_fl\_ti * flagList = ´F F FFFF FFFF FFFF´ |
| *expirationDate* | 2023.04.15: YYMMDD = ´0203 0004 0105´ |

1. Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = ´285927d1d7469dc69c83d25380dd51bbbd39cc69cf74219a53700eb44d1d0827130c6881ea13836d454eae6972b5702b1da75e9ba39c6050539c5173129d2d0b´.
   * + 1. / MF/ DF.Auth / PuK.RCA\_RSA2048

Dieses Kapitel ist absichlich leer.

1. (N254.080) Dieser Punkt ist absichtlich leer.
   * + 1. / MF / DF.Auth / SK.AES128
2. (N254.090) K\_Personalisierung   
   Der Schlüssel SK.AES128 MUSS die in Tabelle 292 dargestellten Attribute besitzen.

Tabelle 292: Attribute / MF / DF.Auth / SK.AES128

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | symmetrisches Authentisierungsobjekt, AES128 |  |
| *keyIdentifier* | ´02´ = 2 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *encKey* | ´0102030405060708 090A0B0C0D0E0F10´ |  |
| *macKey* | ´100F0E0D0C0B0A09 0807060504030201´ |  |
| *numberScenario* | 0 |  |
| *algorithmIdentifier* | aesSessionkey4SM |  |

* + - 1. / MF / DF.Auth / SK.AES192

1. (N254.100) K\_Personalisierung   
   Der Schlüssel SK.AES192 MUSS die in Tabelle 293 dargestellten Attribute besitzen.

Tabelle 293: Attribute / MF / DF.Auth / SK.AES192

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | Symmetrisches Authentisierungsobjekt, AES192 |
| *keyIdentifier* | ´03´ = 3 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *encKey* | ´0102030405060708 090A0B0C0D0E0F10 1112131415161718´ |
| *macKey* | ´1817161514131211 100F0E0D0C0B0A09 0807060504030201´ |
| *numberScenario* | 0 |
| *algorithmIdentifier* | aesSessionkey4SM |

* + - 1. / MF / DF.Auth / SK.AES256

1. (N254.110) K\_Personalisierung   
   Der Schlüssel SK.AES256 MUSS die in Tabelle 294 dargestellten Attribute besitzen.

Tabelle 294: Attribute / MF / DF.Auth / SK.AES256

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | symmetrisches Authentisierungsobjekt, AES256 |  |
| *keyIdentifier* | ´04´ = 4 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *encKey* | ´0102030405060708 090A0B0C0D0E0F10  1112131415161718 191A1B1C1D1E1F20´ |  |
| *macKey* | ´100F0E0D0C0B0A09 0807060504030201  201F1E1D1C1B1A19 1817161514131211´ |  |
| *numberScenario* | 0 |  |
| *algorithmIdentifier* | aesSessionkey4SM |  |

* + - 1. / MF / DF.Auth / TC.AES128

Dieses Objekt ist genau dann verfügbar, wenn Option\_Kryptobox implementiert ist.

1. (N254.120) K\_Personalisierung, Option\_Kryptobox   
   Der Schlüssel TC.AES128 MUSS die in Tabelle 295 dargestellten Attribute besitzen.

Tabelle 295: Attribute / MF / DF.Auth / TC.AES128

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | symmetrisches Authentisierungsobjekt, AES128 |  |
| *keyIdentifier* | ´05´ = 5 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *encKey* | ´0102030405060708 090A0B0C0D0E0F10´ |  |
| *macKey* | ´100F0E0D0C0B0A09 0807060504030201´ |  |
| *algorithmIdentifier* | aesSessionkey4TC |  |

* + - 1. / MF / DF.Auth / TC.AES192

Dieses Objekt ist genau dann verfügbar, wenn Option\_Kryptobox implementiert ist.

1. (N254.130) K\_Personalisierung   
   Der Schlüssel TC.AES192 MUSS die in Tabelle 296 dargestellten Attribute besitzen.

Tabelle 296: Attribute / MF / DF.Auth / TC.AES192

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | Symmetrisches Authentisierungsobjekt, AES192 |
| *keyIdentifier* | ´06´ = 6 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *encKey* | ´0102030405060708 090A0B0C0D0E0F10 1112131415161718´ |
| *macKey* | ´1817161514131211 100F0E0D0C0B0A09 0807060504030201´ |
| *algorithmIdentifier* | aesSessionkey4TC |

* + - 1. / MF / DF.Auth / TC.AES256

Dieses Objekt ist genau dann verfügbar, wenn Option\_Kryptobox implementiert ist.

1. (N254.140) K\_Personalisierung   
   Der Schlüssel TC.AES256 MUSS die in Tabelle 297 dargestellten Attribute besitzen.

Tabelle 297: Attribute / MF / DF.Auth / TC.AES256

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | symmetrisches Authentisierungsobjekt, AES256 |  |
| *keyIdentifier* | ´07´ = 7 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *encKey* | ´0102030405060708 090A0B0C0D0E0F10  1112131415161718 191A1B1C1D1E1F20´ |  |
| *macKey* | ´100F0E0D0C0B0A09 0807060504030201  201F1E1D1C1B1A19 1817161514131211´ |  |
| *algorithmIdentifier* | aesSessionkey4TC |  |

### Anwendung für IAS Services, DF.IAS

1. (N255.050) K\_Personalisierung   
   Die Anwendung DF.IAS MUSS die in Tabelle 298 dargestellten Attribute besitzen.

Tabelle 298: Attribute / MF / DF.IAS

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | Ordner |  |
| *applicationIdentifier* | ´F000 0000 05´ |  |
| *fileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *children* | * / MF / DF.IAS / PrK.X509\_ELC256 * / MF / DF.IAS / PrK.X509\_ELC384 * / MF / DF.IAS / PrK.X509\_ELC512 * / MF / DF.IAS / PrK.X509\_RSA2048 * / MF / DF.IAS / PrK.X509\_RSA3072 |  |

* + - 1. / MF/ DF.IAS / PrK.X509\_ELC256

1. (N255.110) K\_Personalisierung   
   Der Schlüssel PrK.X509\_ELC256 MUSS die in Tabelle 299 dargestellten Attribute besitzen.

Tabelle 299: Attribute / MF / DF.IAS / PrK.X509\_ELC256

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates ELC Schlüsselobjekt, ELC256 |
| *keyIdentifier* | ´18´ = 24 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP256r1, *privateElcKey.d* = ´295fcd16a620cb6fe34f7b0326e9b059c0b5229057c04ed9a7ed13a542967ee0´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE: *setAlgorithmIdentifier* =  {elcRoleAuthentication, elcSharedSecretCalculation, signECDSA} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet: ´04855978a5c0237c53404d6de6b9626145494feca14591dcfcbdb03850977598729ba33a5ae2dd510878c4edd669fbf6b55fddcbeffdbacfbec3b88e151fc78556´.
   * + 1. / MF/ DF.IAS / PrK.X509\_ELC384
2. (N255.120) K\_Personalisierung   
   Der Schlüssel PrK.X509\_ELC384 MUSS die in Tabelle 300 dargestellten Attribute besitzen.

Tabelle 300: Attribute / MF / DF.IAS / PrK.X509\_ELC384

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates ELC Schlüsselobjekt, ELC384 |
| *keyIdentifier* | ´19´ = 25 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP384r1, *privateElcKey.d* = ´6fbeecf714222b4a6ce80f035a00d80508cce63c73f4ad778db108f8e556f8a9f7625cce92fc6d24a2de0d8e998bee58´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE: *setAlgorithmIdentifier* =  {elcSharedSecretCalculation, signECDSA} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet: ´0435229000a37cc393cd04d8192ef25b910720a11453600a568b7f7adf3efebe984bc4893e3782aedfa9d4deeb602e757402e591fa8bc4cfe150e0b7a7504616f19dba2e7b55c6c11a89b6061e0b2e831a7ccefd8b306400c2a0fe6e8bc5ebff84´.
   * + 1. / MF/ DF.IAS / PrK.X509\_ELC512
2. (N255.130) K\_Personalisierung   
   Der Schlüssel PrK.X509\_ELC512 MUSS die in Tabelle 301 dargestellten Attribute besitzen.

Tabelle 301: Attribute / MF / DF.IAS / PrK.X509\_ELC512

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates ELC Schlüsselobjekt, ELC512 |
| *keyIdentifier* | ´1A´ = 26 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP512r1, *privateElcKey.d* = ´2c24d847750f8321f875880deb583a7f4afdc8ebfd1fd6f587e6876d594bffbaf284814019156c9efbafdac25ec426b6c842e82c5e4a657fee934c21b0447810´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE: *setAlgorithmIdentifier* =  {elcSharedSecretCalculation, signECDSA} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet: ´043e4e30df5028cdc67a4aa3b95c4803a278e6ada8dd6d37ead2816fcf9507001d03929a692137e376ee60c7318b3bb219d84f1bfb4429ed32b549c7544e006f226c787e46f2d0ed40e15e2baec59b5f9bcd0dc8d189dc84c27c82aeb4de14183d95c5b2899f1530c8cdadddc75b262658513d5716725745f101953d10f2e8a544´.
   * + 1. / MF/ DF.IAS / PrK.X509\_RSA2048
2. (N255.140) K\_Personalisierung   
   Der Schlüssel PrK.X509\_RSA2048 MUSS die in Tabelle 302 dargestellten Attribute besitzen.

Tabelle 302: Attribute / MF / DF.IAS / PrK.X509\_RSA2048

| **Attribute** | **Wert** |
| --- | --- |
| *objectType* | privates RSA Schlüsselobjekt, RSA2048 |
| *keyIdentifier* | ´0B´ = 11 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *modulusLength* | 2048 |
| *privateKey* | 308204bd020100300d06092a864886f70d0101010500048204a7308204a30201000282010100a20d2aa5d  0241c4cadd5db90509a3030a4f24f99c6ad2604ca96fd06986345a48eb74f1b9c7be26e18295fd58dd1a3  b6d715580ad27d2b5db93c168b8cdc261d92e12eee344624317a20d2221cbbbcf1bd376d20126980f470f  f2b48f65d686d9cc5e0f7383dcab0881a7de198f1a7b1a2717d1c03ad86973dcefdb0425ed9bb98344549  eac891c9e25c3a02e873aafb5e2adc4afa1c526081bd1f61e75c43fbcbf135f572073d18d1a30c69d6aac  6956761c02a36d74de4fc6bb96c7529528a60628d99dea4abcdb209da139398a68f563ab3595bd882b583  097a167b590932c9db5d47df32c10877bd5fe3cf46a31826259ef1329b5b99a2c87a40e78334590203010  0010282010034e0c8fecc394c46b51ea883a1d97e4a1138c442b072c58a20b53e214dce6ee6306f9e4fab  333d82a13db6f8cf4b0df9d69b2f5c70acc95eced7cd9f81ba4071bc00e0b877b43f912981d62fad62915  7ec5f4eab7d7691ceb1a481f24ff9d0de9b9e723719520876ac22c0dac176713ae47fd5cc3363071d08dd  2728c1af83aa4f0209fe69356338dbdff20a82cc55b8bd0096332538cc7f2357f78aff6ef73d78d618f46  773ccba18cc0b77f1258d395aecb3ba6af3e197111b08a2244cee1b5aa5d9cd76e50db7ebbabf61e36b92  51c77b3e7e91c12bc090c4c1bbd7f82952932f6f2a3c68f492de5d00af0be71214de359ae9a78740d21e8  6a4804ce5a50102818100c27b9a4daa01ef082498e84ced6d90bddd42c8682a7b359314cefedde7abb43a  098f93b75541d05b48d446c0fdc1c375fad17da4cf288dc6388c198739915edcbda3d6dc4c4df2ed74626  d006f73ab8acf1c8842cdd58a217a3972fab547fa377fcaab37dc0dc6799228e204705e8acd5f31419715  703c4dde4a87ae61311af902818100d54f682555c9966346e198495568238a74dc1dc0d821224311070a3  ffde4fc8b4d61c497e9fcdc40652b16defc1c6c595999cafd3a0588a5ab890789cf9d3cf92cfbd5186017  19d456173ef4701b62062572347fe2785d57652110c30c530d1af561d7c36104d0165047783bcdc7db6ea  69a6ccdb05132a60c79fc1d2b0adc6102818100ad83a7a4990486a5e6390230f83823c631ad4eae1910ba  38d27ce7c945d3dcc55718613b557695958c01d06a893c21ce960be8246dab09ea8b32fbe7b8c5e933da8  25dc548d2d6e5624c1a62240db843ed0ad0cb81677e88e5ca71ba1a98f03629eec119e25bebf33523029b  e14188673901f23a00a795360818c8bb1cfdb9c90281800217c45ad16ba7e91371f52f8b01f98f4b3439a  a81b45984b4ef0fbfaefb072b4ab811d8b7b7fe653758e3e18e31ad327739960f43f977ecc03118fcd627  a1c884137874c3c496414a12f2502da56721ce4e3f8b9daa36a83bdac63253b5a0e449d20aeee4cdca48d  dc2c5c03875a4868265f379552886c9b047a61c972685210281806f99aa16030050301c99ea08e9d121ba  6a3f9e0a32cd265cfc00200855f640416e0980ea47a196ecb5c48f69e8c58aa4cd2289758f7fcd384ee74  4e6b87d727f8d4585c7f9924afd197469cf9910ba847841f9f67d024de97013e3c34c61b9f315d5f6b5cb  8bfe7316193c37b4d9a99a59b962fd3c75245227b42b325d33ccf3 |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE: *setAlgorithmIdentifier* = {rsaDecipherOaep, rsaDecipherPKCS1\_V1\_5, sign9796\_2\_DS2, signPSS} |

* + - 1. / MF/ DF.IAS / PrK.X509\_RSA3072

1. (N255.150) K\_Personalisierung   
   Der Schlüssel PrK.X509\_RSA3072 MUSS die in Tabelle 303 dargestellten Attribute besitzen.

Tabelle 303: Attribute / MF / DF.IAS / PrK.X509\_RSA3072

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates RSA Schlüsselobjekt, RSA3072 |
| *keyIdentifier* | ´0C´ = 12 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *modulusLength* | 3072 |
| *privateKey* | 308206fe020100300d06092a864886f70d0101010500048206e8308206e40201000282018100c50ed1db3  704e8259e07d6d4433f4c2c45d0cf28a78dcb8512ebd4b09cd2a07f0ed22410b1188eb8e5faf29c724266  24c4d73deee856179eef171199d5c09cebb23a42b2866477e26c8ff34c8fd5a53320fd82e81abfa649b9b  68bf876bd2b180d8886811bb6f48e64eded13286e39a74f61ff410cf1f6537ca2c7397a7a12507ad8b5a4  30ba1cd2c0c79cb550bf422ee4a75115050164c31a5784f1db433464dbfdd8fafca3fb3e12f519eb2f92a  e1c67c88e5bd4fba8f10726317dd114ff64b644ba58b3a4cda9613219aa36d712300306b16f418c03f4ca  5f66ecf640808b98c547019d7b562a10b1f6d8ed9274a4e7159725ff5c20180434db6190235ae3ac905e4  a1981b3b488d11f78e1c586c85543be59bc2a313deb6914c14c8b4eb6838fdf1a145805aaf026d22362fa  1c2b5f7e686bf1ec35c59c15201a93c1c549de51e79f32fe371c826fc99e2137268de8993bd0c27801237  02fe62db3bbac9b94d05b6764de631e8d1074df47a896a36fd46dc800966e3c70c09cb2de726f9d020301  000102820181008f37c11fd0b1f942f5c2fec8d0831d8a83c749f86326740728844bd57c74bc7efbb790c  cacd98b4928a4a5553cd9b340ffeaebb14a1ce017fbeec935041c742ffd1dc8c1d55909a47f8073b46e49  39429ab48542c9c6064648fa3a6a7f69e6fb1942e376cc3cc12b0881bfaf0fd064888bf24e40e9f6a52f7  24320064f548f20fcd7a9d44631cda702c350ca9a8016cadc161aa0eb3d630d4584dc33a74705ba5404ff  8112c2df63f3053e2c272c34eca2252a09e6166d6797e99348bcffc4c917a402f30a61b59e870773401d1  1e4345892ea98c64da681089a449dceaa2f4e4a806d08762c5084e3c040b688ddfad316889bff178541eb  3a2781d37eed58fd24ac55928ae9db7f7419e6be825638df0b87f1a104912f1f0605e718d686e1864c7d5  4169d9df913cbf9ddb75a84a37640765037810f5aa74174c30ccc5aa2b9a895cbecbefc59fcd89a070a04  fdfb83e0bd4f6e4dcb540749eeb4291b553bd1703c8f585d439930b710069a2de833fdb42070d38e7c4be  6499d75699d52a7810281c100d570f346bc001fe691c060f81974d33d6c1a67b5c52b82891cb15a4faf22  df3342c3b417578cea4859a835b76f22c1ee83c50a6eca1f400b19586c82f84a2f8b1ab6489793cf24b4d  717dc65d73c7269577e298011c7cc3a21acd2c886d08cba8ecbd522f3d552b9c4ff0cba0b495e2a0ae446  61f7106721c75b6abd33d3d228025da96a4ea8c8b77743330f4cefaac73d037b008afd2422b32be64c9b4  bf69a94e6a41fb057718a5200bbcf69819efb2b68669684ddee6c3ceb8f3afe76fde10281c100ec59964e  422745a2f7a850cfbcf666ec4948e5ce5b2cebb1fe927723b92235a94fbda6cbebe158b6efcb2cc3c5e25  8afca945658e82c966fc742c43c2d8b3f824fa395a01de92c6fd125b929fc2fc493daace672cb2f4ca940  5273a12ae749554440e4f34ea4b4bfca973f98fe3116c765c4039391ca533273bd606a5611eebeb61a743  acc9525da1b12b4aa9db1bb74a6d690af77bf22105756883c35920ddd7651079c2b80272dc199dfee2bde  1bf3eb9406c8eaf32c19972f1c3064bc113d0281c0264622d9c44ca16bcf27183c651435ed05bd966b59f  7fb686ae4c2b5174ac18b87a5605def2d2db5db9df643f681dd14d03c3d2ab0c3f9d003b37f81dcd430c2  58ffe6f48764d5c837e9e773bbacf33740a136ffe83053c6f8d2ecd958937d28d702662de65b92820d794  ee6edaf325b87a87796899f3f5f27489553683fcba7b104eaa44d86a6b77f83655c91074930811866ae2a  3e4a848e6c88d7a3e1734f34f446ea07cd26670f5d625047331f371b09959bd82793e184eb18bc1375730  10281c035619e931a59e85e8075219b69e0752a94fac3fcb19719bcdff490b81ece5d345a704945f186ba  14c7602a42256594065048e49fbc00a611a4fdad78b208ab55714b4f3614cb5f5ff9eafd414056419deb5  bf5c8a39f913f00ea1e779e4b12c311615ed49d449216675975e1c637ac1f691ab67c14079076eab1311f  934505994963fc307398942e08c356ceda431445aa90f7a8c9f0ff6e956b53e5d56ba17a0d890ba0f82a3  faf13f4ff20ba03b6688722adcd9f5340f89eaee33d02e37f390281c1009f060963514741b1addc446bc2  b499010ad8a8893ae0c43b1acb0da212f80293a3a3c7b62f1d082ccf97fff0b8ccd9caaee64a293e5fd53  372cc7cc961158e81ba6eee86597c80d0ff890e7b7a3f5fce623548e1018b21c762a01d0aaa2aad8c2397  33fe7feb421e3caf25d347a9f45b78621ed808090e73e96acdb1da572d80f9e111125a86da90c7e7a81ec  9a0124d93d5943a8c77353dda990e85b85507336a73c1a43293a3698f0be81fc672d824510284108e860f  c16feabce2b99666c1 |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE: *setAlgorithmIdentifier* = {rsaDecipherOaep, rsaDecipherPKCS1\_V1\_5, sign9796\_2\_DS2, signPSS} |

### Anwendung für LCS Use Cases, DF.LCS

1. (N256.050) K\_Personalisierung   
   Die Anwendung DF.LCS MUSS die in Tabelle 304 dargestellten Attribute besitzen.

Tabelle 304: Attribute / MF / DF.LCS

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | Ordner |  |
| *applicationIdentifier* | ´F000 0000 06´ |  |
| *fileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *children* | * / MF / DF.LCS / CAN\_256 * / MF / DF.LCS / EF.LCS * / MF / DF.LCS / PIN.LCS * / MF / DF.LCS / PrK.LCS * / MF / DF.LCS / PuK.LCS * / MF / DF.LCS / SK.LCS |  |

* + - 1. / MF/ DF.LCS / CAN\_256

Dieses Objekt ist genau dann verfügbar, wenn Option\_kontaktlose\_Schnittstelle implementiert ist.

1. (N256.100) K\_Personalisierung, Option\_kontaktlose\_Schnittstelle   
   Der Schlüssel CAN\_256 MUSS die in Tabelle 305 dargestellten Attribute besitzen.

Tabelle 305: Attribute / MF / DF.LCS / CAN\_256

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | symmetrisches Kartenverbindungsobjekt |  |
| *keyIdentifier* | ´10´ = 16 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *can* | 123456 |  |
| *algorithmIdentifier* | für alle unterstützten SE:  id-PACE-ECDH-GM-AES-CBC-CMAC-256 |  |

* + - 1. / MF / DF.LCS / EF.LCS

1. (N256.200) K\_Personalisierung   
   Die Datei EF.LCS MUSS die in Tabelle 306 dargestellten Attribute besitzen.

Tabelle 306: Attribute / MF / DF.LCS / EF.LCS

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | transparentes Elementary File |  |
| *fileIdentifier* | ´EF 01´ |  |
| *shortFileIdentifier* | ´01´Oktett = 1 Oktett |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | False |  |
| *flagChecksum* | False |  |
| *numberOfOctet* | ´0001´ = 1 |  |
| *positionLogicalEndOfFile* | 1 |  |
| *body* | ´00´ |  |

* + - 1. / MF / DF.LCS / PIN.LCS

1. (N256.300) K\_Personalisierung   
   Das Passwortobjekt PIN.LCS MUSS die in Tabelle 307 dargestellten Attribute besitzen.

Tabelle 307: Attribute /  MF / DF.LCS / PIN.LCS

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** | |
| *objectType* | Reguläres Passwortobjekt |  |
| *pwdIdentifier* | ´01´ = 1 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *secret* | irrelevant |  |
| *minimumLength* | 4 |  |
| *maximumLength* | 12 |  |
| *startRetryCounter* | 3 |  |
| *retryCounter* | 3 |  |
| *transportStatus* | Leer-PIN |  |
| *flagEnabled* | True |  |
| *startSsecList* | für alle unterstützten SE  *startSsec* = unendlich |  |
| *PUK* | 12345678 |  |
| *pukUsage* | 10 |  |

* + - 1. / MF / DF.LCS / PrK.LCS

1. (N256.400) K\_Personalisierung   
   Der Schlüssel PrK.LCS MUSS die in Tabelle 308 dargestellten Attribute besitzen.

Tabelle 308: Attribute / MF / DF.LCS / PrK.LCS

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | privates ELC Schlüsselobjekt, ELC256 |
| *keyIdentifier* | ´11´ = 17 |
| *lifeCycleStatus* | „Operational state (active)“ |
| *privateKey* | *domainParameter* gemäß brainpoolP256r1, *privateElcKey.d* = ´7625964fa8ca41673ac8c88a69fecfe18df90496c61f38d29310ccc4fc1484a1´ |
| *keyAvailable* | True |
| *listAlgorithmIdentifier* | für alle unterstützten SE  *setAlgorithmIdentifier* = {elcRoleAuthentication} |

1. Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet: ´044814679aa37c8c05f6c673a2b6cb621574dfb0b79835d3399e1bf1336eebdf7f67dc4670fae6fa60d456b40a0539472d5f5dcc2bf4da6872a9a3a13717dc2dfa´.
   * + 1. / MF / DF.LCS / PuK.LCS
2. (N256.500) K\_Personalisierung   
   Der Schlüssel PuK.LCS MUSS die in Tabelle 309 dargestellten Attribute besitzen.

Tabelle 309: Attribute / MF / DF.LCS / PuK.LCS

|  |  |
| --- | --- |
| **Attribute** | **Wert** |
| *objectType* | Öffentliches Signaturprüfobjekt, ELC256 |
| *keyIdentifier* | ´0102030405060708´ |
| *lifeCycleStatus* | „Operational state (active)“ |
| *publicKey* | *domainParameter* gemäß brainpoolP256r1, P = ´049bbfb368605b45a6701d4e5fecbd52fbd43ee19340d802fa8792cac0a352f0e65fd6dd301fa7866a322d74c1cd0b634eac8a8e6fc600e1cd3361063107d54b5b´ |
| *oid* | ecdsa-with-SHA256 |
| *CHAT* | * OIDflags = oid\_cvc\_fl\_ti * flagList = ´BF FFFF FFFF FFFF´ |
| *expirationDate* | 2015.03.18: YYMMDD = ´0105 0003 0108´ |

1. Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = ´715babdef8bc5828ce2ba351e4531a1d7ae8f64c1694ff9fb07e433231f678c6´.
   * + 1. / MF / DF.LCS / SK.LCS
2. (N256.600) K\_Personalisierung   
   Der Schlüssel SK.LCS MUSS die in Tabelle 310 dargestellten Attribute besitzen.

Tabelle 310: Attribute / MF / DF.LCS / SK.LCS

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | symmetrisches Authentisierungsobjekt, AES128 |  |
| *keyIdentifier* | ´12´ = 18 |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *encKey* | ´0102030405060708090A0B0C0D0E0F10´ |  |
| *macKey* | ´100F0E0D0C0B0A090807060504030201´ |  |
| *numberScenario* | 0 |  |
| *algorithmIdentifier* | aesSessionkey4SM |  |

### Anwendung für Select EF Use Cases, DF.SelectEF

1. (N257.050) K\_Personalisierung   
   Die Anwendung DF.SelectEF MUSS die in Tabelle 311 dargestellten Attribute besitzen.

Tabelle 311: Attribute / MF / DF.SelectEF

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | Ordner |  |
| *applicationIdentifier* | ´F000 0000 07´ |  |
| *fileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *children* | * / MF / DF.SelectEF / EF.00 * / MF / DF.SelectEF / EF.… * / MF / DF.SelectEF / EF.99 |  |

* + - 1. / MF / DF.SelectEF / EF.xx

1. (N257.110) K\_Personalisierung   
   Die Dateien EF.xx MÜSSEN die in Tabelle 312 dargestellten Attribute besitzen.

Tabelle 312: Attribute / MF / DF.SelectEF / EF.xx

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | transparentes Elementary File |  |
| *fileIdentifier* | ´EF 00´ + *i*, mit *i* = 0, 1, 2, …, 99 |  |
| *shortFileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | False |  |
| *flagChecksum* | False |  |
| *numberOfOctet* | ´0001´ = 1 |  |
| *positionLogicalEndOfFile* | 1 |  |
| *body* | ´00´ |  |

1. Tabelle 312 spezifiziert 100 Dateien, welche sich lediglich im Attribut fileIdentifier unterscheiden. Gemäß der Tabelle gilt für die Menge der fileIdentifier: {´EF00´, ´EF01´, …, EF09´, ´EF0A´, …, EF0F´, ´EF10´, …, ´EF63´}.

### Anwendung für rekordorientierte Dateioperationen, DF.strukturiert

1. (N258.050) K\_Personalisierung   
   Die Anwendung DF.strukturiert MUSS die in Tabelle 313 dargestellten Attribute besitzen.

Tabelle 313: Attribute / MF / DF.strukturiert

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | Ordner |  |
| *applicationIdentifier* | ´F000 0000 08´ |  |
| *fileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *children* | * / MF / DF.strukturiert / EF.strukturiert |  |

#### / MF / DF.strukturiert / EF.strukturiert

1. (N258.110) K\_Personalisierung   
   Die Datei EF.strukturiert MUSS die in Tabelle 314 dargestellten Attribute besitzen.

Tabelle 314: Attribute / MF / DF.strukturiert / EF.strukturiert

| **Attribute** | **Wert** | **Bemerkung** |
| --- | --- | --- |
| *objectType* | linear variables Elementary File |  |
| *fileIdentifier* | ´EF 01´ |  |
| *shortFileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | True |  |
| *flagChecksum* | True |  |
| *maximumNumberOfRec.* | 254 |  |
| *maximumRecordLength* | ´FF´= 255 |  |
| *flagRecordLifeCycleStatus* | True |  |
| *numberOfOctet* | ´FD02´ = 64.770 |  |
| *recordList* | leere Liste |  |

### Anwendung für transparente Dateioperationen, DF.transparent

1. (N259.050) K\_Personalisierung   
   Die Anwendung DF.transparent MUSS die in Tabelle 315 dargestellten Attribute besitzen.

Tabelle 315: Attribute / MF / DF.transparent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | Ordner |  |
| *applicationIdentifier* | ´F000 0000 09´ |  |
| *fileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *children* | * / MF / DF.transparent / EF.transparent |  |

* + - 1. / MF / DF.transparent / EF.transparent

1. (N259.110) K\_Personalisierung   
   Die Datei EF.transparent MUSS die in Tabelle 316 dargestellten Attribute besitzen.

Tabelle 316: Attribute / MF / DF.transparent / EF.transparent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Wert** | **Bemerkung** |
| *objectType* | transparentes Elementary File |  |
| *fileIdentifier* | ´EF 01´ |  |
| *shortFileIdentifier* | ‑ |  |
| *lifeCycleStatus* | „Operational state (active)“ |  |
| *flagTransactionMode* | True |  |
| *flagChecksum* | True |  |
| *numberOfOctet* | ´8000´ = 32.768 |  |
| *positionLogicalEndOfFile* | 0 |  |
| *body* | irrelevant |  |

# Anhang C – Domainparameter elliptischer Kurven (informativ)

Dieser Anhang enthält die Domainparameter der elliptischen Kurven, die gemäß dieser Dokumentenversion vom COS zu unterstützen sind. Die Object Identifier der hier dargestellten Kurven sind Tabelle 271 entnommen.

## ansix9p256r1, OID = {1.2.840.10045.3.1.7} = ´2A8648CE3D030107´

Die hier dargestellten Domainparameter wurden [ANSI X9.62#L.6.4.3] entnommen und sind identisch zu [FIPS 186–4#D.1.2.3 P-256]:

|  |  |
| --- | --- |
| *p* | ´FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF´ |
| *a* | ´FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC´ |
| *b* | ´5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B´ |
| *G* | in komprimierter Form: ´03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296´  in Koordinatenform:  xg=´6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296´  yg=´4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5´ |
| *n* | ´FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551´ |
| *h* | 1 |

1. Gegenüber der Darstellung in [FIPS 186–4#D.1.2.3] ist zu beachten das gilt:   
   ‑3 ≡ a mod p.

## ansix9p384r1, OID = {1.3.132.0.34} = ´2B81040022´

Die hier dargestellten Domainparameter wurden [ANSI X9.62#L.6.5.2] entnommen und sind identisch zu [FIPS 186–4#D.1.2.4 P-384]::

|  |  |
| --- | --- |
| *p* | ´FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFF´ |
| *a* | ´FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFC´ |
| *b* | ´B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112  0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF´ |
| *G* | in komprimierter Form: ´03 AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98  59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7´  in Koordinatenform:  xg=´AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98  59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7´  yg=´3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD 289A147C  E9DA3113 B5F0B8C0 0A60B1CE 1D7E819D 7A431D7C 90EA0E5F´ |
| *n* | ´FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973´ |
| *h* | 1 |

1. Gegenüber der Darstellung in [FIPS 186–4#D.1.2.3] ist zu beachten das gilt:   
   ‑3 ≡ a mod p.

## brainpoolP256r1, OID={1.3.36.3.3.2.8.1.1.7}=´2B2403030208010107´

Die hier dargestellten Domainparameter wurden [RFC5639#3.4] entnommen:

|  |  |
| --- | --- |
| *p* | ´A9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377´ |
| *a* | ´7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9´ |
| *b* | ´26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B6´ |
| *G* | in Koordinatenform:  xg=´8BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE27E1E3BD23C23A4453BD9ACE3262´  yg=´547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C1D54C72F046997´ |
| *n* | ´A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7´ |
| *h* | 1 |

## brainpoolP384r1, OID={1.3.36.3.3.2.8.1.1.11}=´2B240303020801010B´

Die hier dargestellten Domainparameter wurden [RFC5639#3.6] entnommen:

|  |  |
| --- | --- |
| *p* | ´8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B4  12B1DA197FB71123ACD3A729901D1A71874700133107EC53´ |
| *a* | ´7BC382C63D8C150C3C72080ACE05AFA0C2BEA28E4FB22787  139165EFBA91F90F8AA5814A503AD4EB04A8C7DD22CE2826´ |
| *b* | ´04A8C7DD22CE28268B39B55416F0447C2FB77DE107DCD2A6  2E880EA53EEB62D57CB4390295DBC9943AB78696FA504C11´ |
| *G* | in Koordinatenform:  xg=´1D1C64F068CF45FFA2A63A81B7C13F6B8847A3E77EF14FE3  DB7FCAFE0CBD10E8E826E03436D646AAEF87B2E247D4AF1E´  yg=´8ABE1D7520F9C2A45CB1EB8E95CFD55262B70B29FEEC5864  E19C054FF99129280E4646217791811142820341263C5315´ |
| *n* | ´8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B3  1F166E6CAC0425A7CF3AB6AF6B7FC3103B883202E9046565´ |
| *h* | 1 |

## brainpoolP512r1, OID={1.3.36.3.3.2.8.1.1.13}=´2B240303020801010D´

Die hier dargestellten Domainparameter wurden [RFC5639#3.7] entnommen:

|  |  |
| --- | --- |
| *p* | ´AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330871  7D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A48F3´ |
| *a* | ´7830A3318B603B89E2327145AC234CC594CBDD8D3DF91610A83441CAEA9863BC  2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A72BF2C7B9E7C1AC4D77FC94CA´ |
| *b* | ´3DF91610A83441CAEA9863BC2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A7  2BF2C7B9E7C1AC4D77FC94CADC083E67984050B75EBAE5DD2809BD638016F723´ |
| *G* | in Koordinatenform:  xg=´81AEE4BDD82ED9645A21322E9C4C6A9385ED9F70B5D916C1B43B62EEF4D0098E  FF3B1F78E2D0D48D50D1687B93B97D5F7C6D5047406A5E688B352209BCB9F822´  yg=´7DDE385D566332ECC0EABFA9CF7822FDF209F70024A57B1AA000C55B881F8111  B2DCDE494A5F485E5BCA4BD88A2763AED1CA2B2FA8F0540678CD1E0F3AD80892´ |
| *n* | ´AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330870  553E5C414CA92619418661197FAC10471DB1D381085DDADDB58796829CA90069´ |
| *h* | 1 |

# Anhang D – Erläuterungen zu Wertebereichen (informativ)

1. Der Hauptteil des Dokumentes enthält im Wesentlichen normative Festlegungen, WIE etwas zu funktionieren hat. Dieser Abschnitt beschreibt WARUM gewisse Festlegungen so und nicht anders getroffen wurden. Derartige Begründungen und Designentscheidungen belasten damit einerseits nicht den Hauptteil und andererseits werden Entscheidungen und Begründungen nachvollziehbar.
2. Wertebereich *pwdIdentifier*, siehe (N015.000): Gemäß [ISO/IEC 7816-4] wird das Attribut *pwdIdentifier* an der Schnittstelle zur Smartcard in den Bits b5 bis b1 des Parameters P2 codiert. Daraus ergibt sich ein Wertebereich von 1 bis 31 bzw. 0 bis 31, wenn der Wert "no information given" explizit dem Wert *pwdIdentifier* = 0 zugeordnet wird. In den Generationen 0 und 1 wurde lediglich der Bereich [0, 3] genutzt, um den Overhead bei der Verwaltung von Sicherheitszuständen klein zu halten. Durch Einführung von "Multireferenz-PIN" erscheint es sinnvoll, den Wertebereich, wie in (N015.000) dargestellt, zu vergrößern.
3. Wertebereich *startSsec*, siehe (N015.800):
   1. Als Infimum wurde 1 gewählt, da der Wert 0 für *startSsec* sinnlos ist.
   2. Als Supremum wurde 250 gewählt, damit für 1-Byte-Codierungen noch genügend Codierungsmöglichkeiten für „unendlich“, Escape, RFU und ähnliches vorhanden sind.
4. Die Codierung für ShortPassword und LongPassword wurde absichtlich gleich gewählt, weil in [ISO/IEC 7816-4] nur ein begrenzter Vorrat an passenden Trailern vorhanden ist. Falls es für notwendig erachtet wird auch an der Schnittstelle zwischen den Fällen ShortPassword und LongPassword zu unterscheiden, dann ist es sinnvoll den Trailer für LongPassword zu ändern.

# Anhang E – Verzeichnisse (informativ)

## Abkürzungen

|  |  |
| --- | --- |
| **Kürzel** | **Erläuterung** |
| AID | Application Identifier, siehe [ISO/IEC 7816-5], Identifikator für eine Smartcard Applikation |
| APDU | Application Protocol Data Unit, siehe [ISO/IEC 7816-3], kleinste Nachrichteneinheit, die auf der Applikationsschicht mit der Smartcard ausgetauscht wird |
| BGT | „block guard time“, siehe [ISO/IEC 7816-3#11.2] |
| CHAT | Certificate Holder Authorisation Template, Datenobjekt gemäß [gemSpec\_PKI#6.7.2.5], Bestandteil eines CV‑Zertifikates |
| COS | Card Operating System, Betriebssystem einer Smartcard |
| DF | Dedicated File, siehe [ISO/IEC 7816-4], Bezeichnung für einen Ordner innerhalb des Objektsystems einer Smartcard |
| EF | Elementary File, siehe [ISO/IEC 7816-4], Bezeichnung für eine Datei innerhalb des Objektsystems einer Smartcard |
| ELC | Elliptic Curve Cryptography, Kryptographie mittels elliptischer Kurven |
| G1 | Abkürzung für Generation 1, bezeichnet eine frühere Version des Dokumentes, in der Regel ergänzt um den Zusatz „normativ“ |
| G2 | Abkürzung für Generation 2, bezeichnet diese Version dieses Dokumentes, in der Regel ergänzt um den Zusatz „normativ“ |
| IC | Integrated Chip, Halbleiter |
| ICC | Integrated Chip Card, Smartcard |
| IFD | Interface Device |
| IFSC | maximum Information Field Size for the Card, siehe [ISO/IEC 7816-3#11.4.2] |
| IFSD | maximum Information Field Size for the Interface Device, [ISO/IEC 7816-3#11.4.2] |
| MAC | Message Authentication Code |
| MSBit | Most Significant Bit |
| MSByte | Most Significant Byte |
| LSBit | Least Significant Bit |
| LSByte | Least Significant Byte |
| TPDU | Transport Protocol Data Unit, siehe [ISO/IEC 7816-3], kleinste Nachrichteneinheit, die auf dem Data Link Layer mit der Smartcard ausgetauscht wird |

## Glossar

|  |  |
| --- | --- |
| **Begriff** | **Erläuterung** |
| ephemer | Nur für kurze Zeit bestehend, flüchtig, ohne bleibende Bedeutung (siehe <http://de.wiktionary.org/wiki/ephemer>) |
| *flagList* | Bestandteil eines CHAT, der anzeigt, welche Zugriffsrechte durch eine erfolgreiche Authentisierung erworben werden |
| Nibble | Ein Nibble oder Nybble ist eine Datenmenge, die 4 Bits umfasst, es wird auch Halbbyte, Tetrade oder Quadrupel genannt (siehe <http://de.wikipedia.org/wiki/Nibble>). |
| schwacher DES-Schlüssel | Ein schwacher oder halbschwacher DES-Schlüssel senkt die Stärke einer Verschlüsselung. Zu weiteren Informationen und einer Liste schwacher und halbschwacher Schlüssel siehe:   * Schneier, B., "Applied Cryptography, Protocols, Algorithms, and Source Code in C", 2nd edition. * RFC 2409 Appendix A * <http://en.wikipedia.org/wiki/Weak_key> |

Das Glossar wird als eigenständiges Dokument zur Verfügung gestellt.

## Abbildungsverzeichnis

[Abbildung 1: Message Sequence Chart für die Kommandobearbeitung 131](#_Toc480895639)

[Abbildung 2: Zeitlicher Ablauf eines Roll-Back-Kommandos 165](#_Toc480895640)

[Abbildung 3: Kommunikationsmodell Gegenstelle, Steuersoftware und COS 389](#_Toc480895641)

[Abbildung 4: Sequenzdiagramm zur externen Authentisierung 390](#_Toc480895642)

[Abbildung 5: Sequenzdiagramm zur internen Authentisierung 392](#_Toc480895643)

[Abbildung 6: Sequenzdiagramm symmetrische Sessionkey-Aushandlung 393](#_Toc480895644)

[Abbildung 7: Sequenzdiagramm PACE Authentisierung 395](#_Toc480895645)

[Abbildung 8: Sequenzdiagramm RSA-Sessionkey-Aushandlung 396](#_Toc480895646)

[Abbildung 9: Sequenzdiagramm ELC-Sessionkey-Aushandlung 399](#_Toc480895647)

[Abbildung 10: Sequenzdiagramm zur Übertragung von Sessionkeys 400](#_Toc480895648)

[Abbildung 11: Komponentendiagramm Performanzmessplatz 407](#_Toc480895649)

[Abbildung 12: Graphische Darstellung von *Pgesamt\_einfach* 416](#_Toc480895650)

[Abbildung 13 - Synchrone Kommunikation zw. CMS und Smartcard 492](#_Toc480895651)

[Abbildung 14 - Asynchrone Kommunikation zw. CMS und Smartcard 492](#_Toc480895652)

[Abbildung 15: Zusammenhang Schlüsselspeicher 502](#_Toc480895653)

## Tabellenverzeichnis

[Tabelle 1: Präfixe, die auf Vielfachen von Zehnerpotenzen beruhen: 23](#_Toc480895654)

[Tabelle 2: Präfixe, die auf Vielfachen von Zweierpotenzen beruhen: 24](#_Toc480895655)

[Tabelle 3: Liste der Komponenten, an welche dieses Dokument Anforderungen stellt 25](#_Toc480895656)

[Tabelle 4: Liste der Schlüsselparameter eines RSA-Schlüssels 46](#_Toc480895657)

[Tabelle 5: CV-Zertifikat einer CA mit CPI = ´21´, SHA-256 68](#_Toc480895658)

[Tabelle 6: CV-Zertifikat zur Authentisierung mit CPI = ´22´, SHA-256 68](#_Toc480895659)

[Tabelle 7: Liste der Domainparameter einer elliptischen Kurve 74](#_Toc480895660)

[Tabelle 8: Aufheben des Transportschutzes 77](#_Toc480895661)

[Tabelle 9: Codierung des CLA-Bytes für Kanalnummern kleiner vier 132](#_Toc480895662)

[Tabelle 10: Codierung des CLA-Bytes für Kanalnummern größer gleich vier 133](#_Toc480895663)

[Tabelle 11: Case 1 Kommando-APDU 136](#_Toc480895664)

[Tabelle 12: Case 1 Antwort-APDU 136](#_Toc480895665)

[Tabelle 13: Case 2 Short Kommando-APDU 137](#_Toc480895666)

[Tabelle 14: Case 2 Extended Kommando-APDU 137](#_Toc480895667)

[Tabelle 15: Case 2 Antwort-APDU 138](#_Toc480895668)

[Tabelle 16: Case 3 Short Kommando-APDU 138](#_Toc480895669)

[Tabelle 17: Case 3 Extended Kommando-APDU 139](#_Toc480895670)

[Tabelle 18: Case 3 Antwort-APDU 140](#_Toc480895671)

[Tabelle 19: Case 4 Short Kommando-APDU 140](#_Toc480895672)

[Tabelle 20: Case 4 Extended Kommando-APDU 141](#_Toc480895673)

[Tabelle 21: Case 4 Antwort-APDU 142](#_Toc480895674)

[Tabelle 22: Kommandos, alphabetisch 164](#_Toc480895675)

[Tabelle 23: Kommandos, numerisch 164](#_Toc480895676)

[Tabelle 24: Activate aktuelles File 167](#_Toc480895677)

[Tabelle 25: Activate privates oder symmetrisches Schlüsselobjekt 167](#_Toc480895678)

[Tabelle 26: Activate öffentliches Schlüsselobjekt 168](#_Toc480895679)

[Tabelle 27: Activate Passwortobjekt 168](#_Toc480895680)

[Tabelle 28: Activate Antwort-APDU im Erfolgsfall 169](#_Toc480895681)

[Tabelle 29: Activate Antwort-APDU im Fehlerfall 169](#_Toc480895682)

[Tabelle 30: Deactivate aktuelles File 171](#_Toc480895683)

[Tabelle 31: Deactivate privates oder symmetrisches Schlüsselobjekt 172](#_Toc480895684)

[Tabelle 32: Deactivate öffentliches Schlüsselobjekt 172](#_Toc480895685)

[Tabelle 33: Deactivate Passwortobjekt 173](#_Toc480895686)

[Tabelle 34: Deactivate Antwort-APDU im Erfolgsfall 173](#_Toc480895687)

[Tabelle 35: Deactivate Antwort-APDU im Fehlerfall 173](#_Toc480895688)

[Tabelle 36: Delete aktuelles File 175](#_Toc480895689)

[Tabelle 37: Delete privates oder symmetrisches Schlüsselobjekt 176](#_Toc480895690)

[Tabelle 38: Delete öffentliches Schlüsselobjekt 176](#_Toc480895691)

[Tabelle 39: Delete Passwortobjekt 177](#_Toc480895692)

[Tabelle 40: Delete Antwort-APDU im Erfolgsfall 177](#_Toc480895693)

[Tabelle 41: Delete Antwort-APDU im Fehlerfall 177](#_Toc480895694)

[Tabelle 42: Load Application mit Command Chaining 180](#_Toc480895695)

[Tabelle 43: Load Application ohne Command Chaining 181](#_Toc480895696)

[Tabelle 44: Load Application Antwort-APDU im Erfolgsfall 181](#_Toc480895697)

[Tabelle 45: Load Application Antwort-APDU im Fehlerfall 181](#_Toc480895698)

[Tabelle 46: Select, kein AID, first occurrence, keine Antwortdaten 185](#_Toc480895699)

[Tabelle 47: Select, kein AID, first occurrence, Antwortdaten mit FCP 185](#_Toc480895700)

[Tabelle 48: Select, kein AID, next occurrence, keine Antwortdaten 186](#_Toc480895701)

[Tabelle 49: Select, kein AID, next occurrence, Antwortdaten mit FCP 187](#_Toc480895702)

[Tabelle 50: Select, AID, first occurrence, keine Antwortdaten 187](#_Toc480895703)

[Tabelle 51: Select, AID, first occurrence, Antwortdaten mit FCP 188](#_Toc480895704)

[Tabelle 52: Select, AID, next occurrence, keine Antwortdaten 189](#_Toc480895705)

[Tabelle 53: Select, AID, next occurrence, Antwortdaten mit FCP 190](#_Toc480895706)

[Tabelle 54: Select, DF oder ADF mit FID, keine Antwortdaten 190](#_Toc480895707)

[Tabelle 55: Select, DF oder ADF mit FID, Antwortdaten mit FCP 191](#_Toc480895708)

[Tabelle 56: Select, höhere Ebene keine Antwortdaten 192](#_Toc480895709)

[Tabelle 57: Select, höhere Ebene, Antwortdaten mit FCP 192](#_Toc480895710)

[Tabelle 58: Select, EF mit FID, keine Antwortdaten 193](#_Toc480895711)

[Tabelle 59: Select, EF mit FID, Antwortdaten mit FCP 194](#_Toc480895712)

[Tabelle 60: Select, Kommandoparameter im Überblick 194](#_Toc480895713)

[Tabelle 61: Select Antwort-APDU im Erfolgsfall 195](#_Toc480895714)

[Tabelle 62: Select Antwort-APDU im Fehlerfall 195](#_Toc480895715)

[Tabelle 63: Terminate Card Usage 198](#_Toc480895716)

[Tabelle 64: Terminate Card Usage Antwort-APDU im Erfolgsfall 198](#_Toc480895717)

[Tabelle 65: Terminate Card Usage Antwort-APDU im Fehlerfall 199](#_Toc480895718)

[Tabelle 66: Terminate DF aktueller Ordner 200](#_Toc480895719)

[Tabelle 67: Terminate DF Antwort-APDU im Erfolgsfall 200](#_Toc480895720)

[Tabelle 68: Terminate DF Antwort-APDU im Fehlerfall 201](#_Toc480895721)

[Tabelle 69: Terminate aktuelle Datei 202](#_Toc480895722)

[Tabelle 70: Terminate privates oder symmetrisches Schlüsselobjekt 203](#_Toc480895723)

[Tabelle 71: Terminate öffentliches Schlüsselobjekt 203](#_Toc480895724)

[Tabelle 72: Terminate Passwortobjekt 204](#_Toc480895725)

[Tabelle 73: Terminate Antwort-APDU im Erfolgsfall 204](#_Toc480895726)

[Tabelle 74: Terminate Antwort-APDU im Fehlerfall 204](#_Toc480895727)

[Tabelle 75: Erase Binary, logical EOF unverändert, ohne *shortFileIdentifier* 206](#_Toc480895728)

[Tabelle 76: Erase Binary, logical EOF unverändert, mit *shortFileIdentifier* 207](#_Toc480895729)

[Tabelle 77: Erase Binary Antwort-APDU im Erfolgsfall 207](#_Toc480895730)

[Tabelle 78: Erase Binary Antwort-APDU im Fehlerfall 207](#_Toc480895731)

[Tabelle 79: Read Binary ohne *shortFileIdentifier* 210](#_Toc480895732)

[Tabelle 80: Read Binary mit *shortFileIdentifier* 211](#_Toc480895733)

[Tabelle 81: Read Binary Antwort-APDU im Erfolgsfall 211](#_Toc480895734)

[Tabelle 82: Read Binary Antwort-APDU im Fehlerfall 211](#_Toc480895735)

[Tabelle 83: Set Logical Eof ohne *shortFileIdentifier* 214](#_Toc480895736)

[Tabelle 84: Set Logical Eof mit *shortFileIdentifier* 214](#_Toc480895737)

[Tabelle 85: Set Logical Eof Antwort-APDU im Erfolgsfall 214](#_Toc480895738)

[Tabelle 86: Set Logical Eof Antwort-APDU im Fehlerfall 214](#_Toc480895739)

[Tabelle 87: Update Binary ohne *shortFileIdentifier* 217](#_Toc480895740)

[Tabelle 88: Update Binary mit *shortFileIdentifier* 218](#_Toc480895741)

[Tabelle 89: Update Binary Antwort-APDU im Erfolgsfall 218](#_Toc480895742)

[Tabelle 90: Update Binary Antwort-APDU im Fehlerfall 218](#_Toc480895743)

[Tabelle 91: Write Binary ohne *shortFileIdentifier* 221](#_Toc480895744)

[Tabelle 92: Write Binary mit *shortFileIdentifier* 222](#_Toc480895745)

[Tabelle 93: Write Binary Antwort-APDU im Erfolgsfall 222](#_Toc480895746)

[Tabelle 94: Write Binary Antwort-APDU im Fehlerfall 222](#_Toc480895747)

[Tabelle 95: Activate Record, ein Rekord, ohne *shortFileIdentifier* 225](#_Toc480895748)

[Tabelle 96: Activate Record, ein Rekord, mit *shortFileIdentifier* 226](#_Toc480895749)

[Tabelle 97: Activate Record, alle Rekords ab P1, ohne *shortFileIdentifier* 226](#_Toc480895750)

[Tabelle 98: Activate Record, alle Rekords ab P1, mit *shortFileIdentifier* 227](#_Toc480895751)

[Tabelle 99: Activate Record Antwort-APDU im Erfolgsfall 227](#_Toc480895752)

[Tabelle 100: Activate Record Antwort-APDU im Fehlerfall 227](#_Toc480895753)

[Tabelle 101: Append Record, ohne *shortFileIdentifier* 230](#_Toc480895754)

[Tabelle 102: Append Record, mit *shortFileIdentifier* 230](#_Toc480895755)

[Tabelle 103: Append Record Antwort-APDU im Erfolgsfall 231](#_Toc480895756)

[Tabelle 104: Append Record Antwort-APDU im Fehlerfall 231](#_Toc480895757)

[Tabelle 105: Deactivate Record, ein Rekord, ohne *shortFileIdentifier* 234](#_Toc480895758)

[Tabelle 106: Deactivate Record, ein Rekord, mit *shortFileIdentifier* 235](#_Toc480895759)

[Tabelle 107: Deactivate Record, alle Rekords ab P1, ohne *shortFileIdentifier* 235](#_Toc480895760)

[Tabelle 108: Deactivate Record, alle Rekords ab P1, mit *shortFileIdentifier* 236](#_Toc480895761)

[Tabelle 109: Deactivate Record Antwort-APDU im Erfolgsfall 236](#_Toc480895762)

[Tabelle 110: Deactivate Record Antwort-APDU im Fehlerfall 236](#_Toc480895763)

[Tabelle 111: Delete Record, ohne *shortFileIdentifier* 239](#_Toc480895764)

[Tabelle 112: Delete Record, mit *shortFileIdentifier* 240](#_Toc480895765)

[Tabelle 113: Delete Record Antwort-APDU im Erfolgsfall 240](#_Toc480895766)

[Tabelle 114: Delete Record Antwort-APDU im Fehlerfall 240](#_Toc480895767)

[Tabelle 115: Erase Record, ohne *shortFileIdentifier* 242](#_Toc480895768)

[Tabelle 116: Erase Record, mit *shortFileIdentifier* 243](#_Toc480895769)

[Tabelle 117: Erase Record Antwort-APDU im Erfolgsfall 243](#_Toc480895770)

[Tabelle 118:Erase Record Antwort-APDU im Fehlerfall 243](#_Toc480895771)

[Tabelle 119: Read Record ohne *shortFileIdentifier* 246](#_Toc480895772)

[Tabelle 120: Read Record mit *shortFileIdentifier* 247](#_Toc480895773)

[Tabelle 121: Read Record Antwort-APDU im Erfolgsfall 247](#_Toc480895774)

[Tabelle 122: Read Record Antwort-APDU im Fehlerfall 247](#_Toc480895775)

[Tabelle 123: Search Record ohne *shortFileIdentifier* 250](#_Toc480895776)

[Tabelle 124: Search Record mit *shortFileIdentifier* 251](#_Toc480895777)

[Tabelle 125: Search Record Antwort-APDU im Erfolgsfall 251](#_Toc480895778)

[Tabelle 126: Search Record Antwort-APDU im Fehlerfall 251](#_Toc480895779)

[Tabelle 127: Update Record, ohne *shortFileIdentifier* 254](#_Toc480895780)

[Tabelle 128: Update Record mit *shortFileIdentifier* 255](#_Toc480895781)

[Tabelle 129: Update Record Antwort-APDU im Erfolgsfall 255](#_Toc480895782)

[Tabelle 130: Update Record Antwort-APDU im Fehlerfall 255](#_Toc480895783)

[Tabelle 131: Change Reference Data mit altem und neuem Benutzergeheimnis 260](#_Toc480895784)

[Tabelle 132: Change Reference Data, nur neues Benutzergeheimnis 260](#_Toc480895785)

[Tabelle 133: Change Reference Data Antwort-APDU im Erfolgsfall 261](#_Toc480895786)

[Tabelle 134: Change Reference Data Antwort-APDU im Fehlerfall 261](#_Toc480895787)

[Tabelle 135: Disable Verification Requirement mit Verifikationsdaten 264](#_Toc480895788)

[Tabelle 136: Disable Verification Requirement ohne Verifikationsdaten 264](#_Toc480895789)

[Tabelle 137: Disable Verification Requirement Antwort-APDU im Erfolgsfall 264](#_Toc480895790)

[Tabelle 138: Disable Verification Requirement Antwort-APDU im Fehlerfall 264](#_Toc480895791)

[Tabelle 139: Enable Verification Requirement mit Verifikationsdaten 267](#_Toc480895792)

[Tabelle 140: Enable Verification Requirement ohne Verifikationsdaten 267](#_Toc480895793)

[Tabelle 141: Enable Verification Requirement Antwort-APDU im Erfolgsfall 267](#_Toc480895794)

[Tabelle 142: Enable Verification Requirement Antwort-APDU im Fehlerfall 268](#_Toc480895795)

[Tabelle 143: Get Pin Status 270](#_Toc480895796)

[Tabelle 144: Get Pin Status Antwort-APDU im Erfolgsfall 270](#_Toc480895797)

[Tabelle 145: Get Pin Status Antwort-APDU im Fehlerfall 270](#_Toc480895798)

[Tabelle 146: Reset Retry Counter, mit PUK, mit *newSecret* 273](#_Toc480895799)

[Tabelle 147: Reset Retry Counter, mit PUK, ohne *newSecret* 273](#_Toc480895800)

[Tabelle 148: Reset Retry Counter, ohne PUK, mit *newSecret* 274](#_Toc480895801)

[Tabelle 149: Reset Retry Counter, ohne PUK, ohne *newSecret* 274](#_Toc480895802)

[Tabelle 150: Reset Retry Counter Antwort-APDU im Erfolgsfall 274](#_Toc480895803)

[Tabelle 151: Reset Retry Counter Antwort-APDU im Fehlerfall 275](#_Toc480895804)

[Tabelle 152: Verify, Vergleich eines Benutzergeheimnisses 277](#_Toc480895805)

[Tabelle 153: Verify Antwort-APDU im Erfolgsfall 277](#_Toc480895806)

[Tabelle 154: Verify Antwort-APDU im Fehlerfall 277](#_Toc480895807)

[Tabelle 155: External Authenticate ohne Antwortdaten 280](#_Toc480895808)

[Tabelle 156: Mutual Authenticate mit Antwortdaten 281](#_Toc480895809)

[Tabelle 157: External Authenticate Antwort-APDU im Erfolgsfall 281](#_Toc480895810)

[Tabelle 158: External Authenticate Antwort-APDU im Fehlerfall 281](#_Toc480895811)

[Tabelle 159: General Authenticate PACE Endnutzerkarte, Schritt 1a 288](#_Toc480895812)

[Tabelle 160: General Authenticate PACE Endnutzerkarte, Schritt 2a 289](#_Toc480895813)

[Tabelle 161: General Authenticate PACE Endnutzerkarte, Schritt 3a 290](#_Toc480895814)

[Tabelle 162: General Authenticate PACE Endnutzerkarte, Schritt 4a 290](#_Toc480895815)

[Tabelle 163: General Authenticate gegenseitige ELC-Authentisierung, Schritt 1 291](#_Toc480895816)

[Tabelle 164: General Authenticate gegenseitige ELC-Authentisierung, Schritt 2 291](#_Toc480895817)

[Tabelle 165: General Authenticate, asynchrone, sym. Admin., Schritt 1 292](#_Toc480895818)

[Tabelle 166: General Authenticate, asynchrone, sym. Admin., Schritt 2 293](#_Toc480895819)

[Tabelle 167: General Authenticate PACE Sicherheitsmodul, Schritt 1b 293](#_Toc480895820)

[Tabelle 168: General Authenticate PACE Sicherheitsmodul, Schritt 2b 294](#_Toc480895821)

[Tabelle 169: General Authenticate PACE Sicherheitsmodul, Schritt 3b 295](#_Toc480895822)

[Tabelle 170: General Authenticate PACE Sicherheitsmodul, Schritt 4b 295](#_Toc480895823)

[Tabelle 171: General Authenticate PACE Sicherheitsmodul, Schritt 5b 296](#_Toc480895824)

[Tabelle 172: General Authenticate, asynchrone, asym. Admin., Schritt 2 296](#_Toc480895825)

[Tabelle 173: General Authenticate Antwort-APDU im Erfolgsfall 297](#_Toc480895826)

[Tabelle 174: General Authenticate Antwort-APDU im Fehlerfall 297](#_Toc480895827)

[Tabelle 175: Get Security Status Key, symmetrischer Schlüssel 309](#_Toc480895828)

[Tabelle 176: Get Security Status Key, Rollenkennung 309](#_Toc480895829)

[Tabelle 177: Get Security Status Key, *bitSecurityList* 310](#_Toc480895830)

[Tabelle 178: Get Security Status Key Antwort-APDU im Erfolgsfall 310](#_Toc480895831)

[Tabelle 179: Get Security Status Key Antwort-APDU im Fehlerfall 310](#_Toc480895832)

[Tabelle 180: Internal Authenticate 312](#_Toc480895833)

[Tabelle 181: Internal Authenticate Antwort-APDU im Erfolgsfall 313](#_Toc480895834)

[Tabelle 182: Internal Authenticate Antwort-APDU im Fehlerfall 313](#_Toc480895835)

[Tabelle 183: Tabelle aller PSO Kommando Header 316](#_Toc480895836)

[Tabelle 184: PSO Compute Cryptographic Checksum, Berechnen eines MAC 317](#_Toc480895837)

[Tabelle 185: PSO Compute Cryptographic Checksum Antwort-APDU im Erfolgsfall 318](#_Toc480895838)

[Tabelle 186: PSO Compute Cryptographic Checksum Antwort-APDU im Fehlerfall 318](#_Toc480895839)

[Tabelle 187: PSO Compute Digital Signature, Signieren ohne „message recovery“ 320](#_Toc480895840)

[Tabelle 188: PSO Compute Digital Signature, Signieren „message recovery“ 321](#_Toc480895841)

[Tabelle 189: PSO Compute Digital Signature Antwort-APDU im Erfolgsfall 321](#_Toc480895842)

[Tabelle 190: PSO Compute Digital Signature Antwort-APDU im Fehlerfall 321](#_Toc480895843)

[Tabelle 191: PSO Decipher, Entschlüsseln mittels RSA 324](#_Toc480895844)

[Tabelle 192: PSO Decipher, Entschlüsseln mittels elliptischer Kurven 324](#_Toc480895845)

[Tabelle 193: PSO Decipher, Entschlüsseln mittels symmetrischem Schlüssel 325](#_Toc480895846)

[Tabelle 194: PSO Decipher Antwort-APDU im Erfolgsfall 325](#_Toc480895847)

[Tabelle 195: PSO Decipher Antwort-APDU im Fehlerfall 325](#_Toc480895848)

[Tabelle 196: PSO Encipher, Verschlüsseln mittels übergebenem RSA-Schlüssel 329](#_Toc480895849)

[Tabelle 197: PSO Encipher, Verschlüsseln mittels übergebenem ELC-Schlüssel 330](#_Toc480895850)

[Tabelle 198: PSO Encipher, Verschlüsseln mittels gespeichertem RSA-Schlüssel 331](#_Toc480895851)

[Tabelle 199: PSO Encipher, Verschlüsseln mittels gespeichertem ELC-Schlüssel 331](#_Toc480895852)

[Tabelle 200: PSO Encipher, Verschlüsseln mittels symmetrischem Schlüssel 332](#_Toc480895853)

[Tabelle 201: PSO Encipher Antwort-APDU im Erfolgsfall 332](#_Toc480895854)

[Tabelle 202: PSO Encipher Antwort-APDU im Fehlerfall 332](#_Toc480895855)

[Tabelle 203: PSO Transcipher, Umschlüsseln von Daten mittels RSA-Schlüssel 336](#_Toc480895856)

[Tabelle 204: PSO Transcipher, Umschlüsseln von Daten von RSA nach ELC 337](#_Toc480895857)

[Tabelle 205: PSO Transcipher, Umschlüsseln von Daten mittels ELC-Schlüssel 338](#_Toc480895858)

[Tabelle 206: PSO Transcipher, Umschlüsseln von Daten von ELC nach RSA 340](#_Toc480895859)

[Tabelle 207: PSO Transcipher Antwort-APDU im Erfolgsfall 340](#_Toc480895860)

[Tabelle 208: PSO Transcipher Antwort-APDU im Fehlerfall 340](#_Toc480895861)

[Tabelle 209: PSO Verify Certificate für RSA-Schlüssel 344](#_Toc480895862)

[Tabelle 210: Kombination von Kurvenparametern in CV‑Zertifikaten 344](#_Toc480895863)

[Tabelle 211: PSO Verify Certificate für ELC-Schlüssel 345](#_Toc480895864)

[Tabelle 212: PSO Verify Certificate Antwort-APDU im Erfolgsfall 345](#_Toc480895865)

[Tabelle 213: PSO Verify Certificate Antwort-APDU im Fehlerfall 345](#_Toc480895866)

[Tabelle 214: PSO Verify Cryptographic Checksum, Prüfen MAC 350](#_Toc480895867)

[Tabelle 215: PSO Verify Cryptographic Checksum Antwort-APDU im Erfolgsfall 350](#_Toc480895868)

[Tabelle 216: PSO Verify Cryptographic Checksum Antwort-APDU im Fehlerfall 351](#_Toc480895869)

[Tabelle 217: PSO Verify Digital Signature mittels übergebenem ELC-Schlüssel 353](#_Toc480895870)

[Tabelle 218: PSO Verify Digital Signature Antwort-APDU im Erfolgsfall 353](#_Toc480895871)

[Tabelle 219: PSO Verify Digital Signature Antwort-APDU im Fehlerfall 353](#_Toc480895872)

[Tabelle 220: Fingerprint über das COS 355](#_Toc480895873)

[Tabelle 221: Fingerprint Antwort-APDU im Erfolgsfall 355](#_Toc480895874)

[Tabelle 222: Fingerprint Antwort-APDU im Fehlerfall 355](#_Toc480895875)

[Tabelle 223: GAKP, ohne Überschreiben, ohne Referenz, ohne Ausgabe 357](#_Toc480895876)

[Tabelle 224: GAKP, ohne Überschreiben, mit Schlüsselreferenz, ohne Ausgabe 357](#_Toc480895877)

[Tabelle 225: GAKP, ggf. Überschreiben, ohne Referenz, ohne Ausgabe 358](#_Toc480895878)

[Tabelle 226: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, ohne Ausgabe 358](#_Toc480895879)

[Tabelle 227: GAKP, Auslesen vorhandener Schlüssel ohne Schlüsselreferenz 359](#_Toc480895880)

[Tabelle 228: GAKP, Auslesen vorhandener Schlüssel mit Schlüsselreferenz 360](#_Toc480895881)

[Tabelle 229: GAKP, ohne Überschreiben, ohne Schlüsselreferenz, mit Ausgabe 360](#_Toc480895882)

[Tabelle 230: GAKP, ohne Überschreiben, mit Schlüsselreferenz, mit Ausgabe 361](#_Toc480895883)

[Tabelle 231: GAKP, ggf. Überschreiben, ohne Schlüsselreferenz, mit Ausgabe 362](#_Toc480895884)

[Tabelle 232: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, mit Ausgabe 362](#_Toc480895885)

[Tabelle 233: Generate Asymmetric Key Pair, Kommandoparameter im Überblick 363](#_Toc480895886)

[Tabelle 234: GAKP Antwort-APDU im Erfolgsfall 363](#_Toc480895887)

[Tabelle 235: GAKP Antwort-APDU im Fehlerfall 363](#_Toc480895888)

[Tabelle 236: Get Challenge für DES oder RSA Authentisierung 366](#_Toc480895889)

[Tabelle 237: Get Challenge für AES oder ELC Authentisierung 366](#_Toc480895890)

[Tabelle 238: Get Challenge Antwort-APDU im Erfolgsfall 366](#_Toc480895891)

[Tabelle 239: Get Challenge Antwort-APDU im Fehlerfall 366](#_Toc480895892)

[Tabelle 240: Get Random 368](#_Toc480895893)

[Tabelle 241: Get Random Antwort-APDU im Erfolgsfall 368](#_Toc480895894)

[Tabelle 242: Get Random Antwort-APDU im Fehlerfall 368](#_Toc480895895)

[Tabelle 243: List Public Key mit allen Arten öffentlicher Schlüsselobjekte 369](#_Toc480895896)

[Tabelle 244: List Public Key Antwort-APDU im Erfolgsfall 370](#_Toc480895897)

[Tabelle 245: List Public Key Antwort-APDU im Fehlerfall 370](#_Toc480895898)

[Tabelle 246: Manage Channel zum Öffnen eines logischen Kanals 372](#_Toc480895899)

[Tabelle 247: Manage Channel zum Schließen eines logischen Kanals 373](#_Toc480895900)

[Tabelle 248: Manage Channel zum Zurücksetzen eines logischen Kanals 373](#_Toc480895901)

[Tabelle 249: Manage Channel zum logischen Reset der Applikationsebene 374](#_Toc480895902)

[Tabelle 250: Manage Channel Antwort-APDU im Erfolgsfall 374](#_Toc480895903)

[Tabelle 251: Manage Channel Antwort-APDU im Fehlerfall 374](#_Toc480895904)

[Tabelle 252: MSE, Restore Variante 377](#_Toc480895905)

[Tabelle 253: MSE, Selektion symmetrischer Internal Authenticate-Schlüssel 377](#_Toc480895906)

[Tabelle 254: MSE, Selektion privater Internal Authenticate-Schlüssel 378](#_Toc480895907)

[Tabelle 255: MSE, Selektion symmetrischer External Authenticate-Schlüssel 379](#_Toc480895908)

[Tabelle 256: MSE, Selektion öffentlicher External Authenticate-Schlüssel 380](#_Toc480895909)

[Tabelle 257: MSE, Selektion symmetrischer Mutual Authenticate-Schlüssel 381](#_Toc480895910)

[Tabelle 258: MSE, Selektion sym. Kartenverbindungsobjekt ohne Kurvenangabe 382](#_Toc480895911)

[Tabelle 259: Zuordnung von *idDomainParameter* zu Domainparameter 382](#_Toc480895912)

[Tabelle 260: MSE, Selektion symmetrisches Kartenverbindungsobjekt 383](#_Toc480895913)

[Tabelle 261: MSE, Selektion privater Signaturschlüssel 384](#_Toc480895914)

[Tabelle 262: MSE, Selektion öffentlicher Zertifikatsprüfschlüssel 384](#_Toc480895915)

[Tabelle 263: MSE, Schlüsselselektion zur Entschlüsselung 385](#_Toc480895916)

[Tabelle 264: MSE, Selektion Verschlüsselungsschlüssel 386](#_Toc480895917)

[Tabelle 265: MSE Antwort-APDU im Erfolgsfall 386](#_Toc480895918)

[Tabelle 266: MSE Antwort-APDU im Fehlerfall 386](#_Toc480895919)

[Tabelle 267: Generische AlgorithmIdentifier für Authentisierungszwecke 401](#_Toc480895920)

[Tabelle 268: Konkrete AlgorithmIdentifier für Authentisierungszwecke 401](#_Toc480895921)

[Tabelle 269: AlgorithmIdentifier für Ver- und Entschlüsselung 402](#_Toc480895922)

[Tabelle 270: AlgorithmIdentifier für Integrität und Authentizität 402](#_Toc480895923)

[Tabelle 271: Object Identifier, alphabetisch sortiert (informativ) 402](#_Toc480895924)

[Tabelle 272: Trailer 🡪 Fehlername 404](#_Toc480895925)

[Tabelle 273: Bedeutung PPS1 gemäß [ISO/IEC 7816-3#Table 7 und 8] 408](#_Toc480895926)

[Tabelle 274: Gesamtbewertung für das Basisbetriebssystem 412](#_Toc480895927)

[Tabelle 275: Gesamtbewertung für Option\_kontaktlose\_Schnittstelle 414](#_Toc480895928)

[Tabelle 276: Gesamtbewertung für Option\_Kryptobox 414](#_Toc480895929)

[Tabelle 277: Gesamtbewertung für Option\_logische\_Kanäle 415](#_Toc480895930)

[Tabelle 278: Gesamtbewertung je nach Kombination der Optionen 415](#_Toc480895931)

[Tabelle 279: Character Guard Time (CGT) gemäß [ISO/IEC 7816-3#11.2] 416](#_Toc480895932)

[Tabelle 280: ATR Codierung 457](#_Toc480895933)

[Tabelle 281: Attribute / MF 458](#_Toc480895934)

[Tabelle 282: Attribute / MF / EF.ATR 458](#_Toc480895935)

[Tabelle 283: Attribute / MF / EF.DIR 459](#_Toc480895936)

[Tabelle 284: Attribute / MF / EF.GDO 460](#_Toc480895937)

[Tabelle 285: Attribute / MF / DF.Auth 460](#_Toc480895938)

[Tabelle 286: Attribute / MF / DF.Auth / PrK.Auth\_ELC256 461](#_Toc480895939)

[Tabelle 287: Attribute / MF / DF.Auth / PrK.Auth\_ELC384 461](#_Toc480895940)

[Tabelle 288: Attribute / MF / DF.Auth / PrK.Auth\_ELC512 462](#_Toc480895941)

[Tabelle 289: Attribute / MF / DF.Auth / PuK.RCA\_ELC256 462](#_Toc480895942)

[Tabelle 290: Attribute / MF / DF.Auth / PuK.RCA\_ELC384 463](#_Toc480895943)

[Tabelle 291: Attribute / MF / DF.Auth / PuK.RCA\_ELC512 463](#_Toc480895944)

[Tabelle 292: Attribute / MF / DF.Auth / SK.AES128 464](#_Toc480895945)

[Tabelle 293: Attribute / MF / DF.Auth / SK.AES192 464](#_Toc480895946)

[Tabelle 294: Attribute / MF / DF.Auth / SK.AES256 464](#_Toc480895947)

[Tabelle 295: Attribute / MF / DF.Auth / TC.AES128 465](#_Toc480895948)

[Tabelle 296: Attribute / MF / DF.Auth / TC.AES192 465](#_Toc480895949)

[Tabelle 297: Attribute / MF / DF.Auth / TC.AES256 465](#_Toc480895950)

[Tabelle 298: Attribute / MF / DF.IAS 466](#_Toc480895951)

[Tabelle 299: Attribute / MF / DF.IAS / PrK.X509\_ELC256 466](#_Toc480895952)

[Tabelle 300: Attribute / MF / DF.IAS / PrK.X509\_ELC384 466](#_Toc480895953)

[Tabelle 301: Attribute / MF / DF.IAS / PrK.X509\_ELC512 467](#_Toc480895954)

[Tabelle 302: Attribute / MF / DF.IAS / PrK.X509\_RSA2048 468](#_Toc480895955)

[Tabelle 303: Attribute / MF / DF.IAS / PrK.X509\_RSA3072 469](#_Toc480895956)

[Tabelle 304: Attribute / MF / DF.LCS 469](#_Toc480895957)

[Tabelle 305: Attribute / MF / DF.LCS / CAN\_256 470](#_Toc480895958)

[Tabelle 306: Attribute / MF / DF.LCS / EF.LCS 470](#_Toc480895959)

[Tabelle 307: Attribute /  MF / DF.LCS / PIN.LCS 471](#_Toc480895960)

[Tabelle 308: Attribute / MF / DF.LCS / PrK.LCS 471](#_Toc480895961)

[Tabelle 309: Attribute / MF / DF.LCS / PuK.LCS 472](#_Toc480895962)

[Tabelle 310: Attribute / MF / DF.LCS / SK.LCS 472](#_Toc480895963)

[Tabelle 311: Attribute / MF / DF.SelectEF 472](#_Toc480895964)

[Tabelle 312: Attribute / MF / DF.SelectEF / EF.xx 473](#_Toc480895965)

[Tabelle 313: Attribute / MF / DF.strukturiert 473](#_Toc480895966)

[Tabelle 314: Attribute / MF / DF.strukturiert / EF.strukturiert 473](#_Toc480895967)

[Tabelle 315: Attribute / MF / DF.transparent 474](#_Toc480895968)

[Tabelle 316: Attribute / MF / DF.transparent / EF.transparent 474](#_Toc480895969)

## Referenzierte Dokumente

### Dokumente der gematik

Die nachfolgende Tabelle enthält die Bezeichnung der in dem vorliegenden Dokument referenzierten Dokumente der gematik zur Telematikinfrastruktur. Version und Stand der referenzierten Dokumente sind in der nachfolgenden Tabelle nicht aufgeführt. Deren zu diesem Dokument passende jeweils gültige Versionen sind in den von der gematik veröffentlichten Produkttypsteckbriefen enthalten, in denen die vorliegende Version aufgeführt wird.

| **[Quelle]** | **Herausgeber (Erscheinungsdatum): Titel** |
| --- | --- |
| [gemSpec\_eGK\_P1] | gematik: Teil 1 – Spezifikation der elektrischen Schnittstelle  (Vorgängerversion dieses Dokumentes für Karten der Generation 1) |
| [gemSpec\_Krypt] | gematik: Übergreifende Spezifikation - Verwendung kryptographischer Algorithmen in der Telematikinfrastruktur |
| [gemSpec\_eGK\_ObjSys] | gematik: Spezifikation der elektronischen Gesundheitskarte  eGK-Objektsystem |
| [gemSpec\_HBA\_ObjSys] | gematik: Spezifikation des elektronischen Heilberufsausweises  HBA-Objektsystem |
| [gemSpec\_OID] | gematik: Festlegung von OIDs |
| [gemSpec\_PKI] | gematik: Übergreifende Spezifikation, Spezifikation PKI |
| [gemSpec\_SMC-B\_ObjSys] | gematik: Spezifikation der Secure Module Card SMC-B  Objektsystem |

### Weitere Dokumente

| [Quelle] | Herausgeber (Erscheinungsdatum): Titel |
| --- | --- |
| [ANSI X3.92] | American National Standard X3.92 – 1981, Data Encryption Algorithm |
| [ANSI X9.62] | Public Key Cryptography for the Financial Services Industry, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 2005 |
| [ANSI X9.63] | Public Key Cryptography for the Financial Services Industry, *Key Agreement and Key Transport Using Elliptic Curve Cryptography*, 2001 |
| [BinPrefix] | Prefix for binary multiples  IEC INTERNATIONAL STANDARD 60027-2, Third edition, 2005-08, Letter symbols to be used in electrical technology –  Part 2: Telecommunications and electronics, clause 3.8.3  <http://physics.nist.gov/cuu/Units/binary.html>  <http://de.wikipedia.org/wiki/Bin%C3%A4rpr%C3%A4fix>  <http://en.wikipedia.org/wiki/Binary_prefix> |
| [BKryA] | Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen), Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, veröffentlicht am 18. Januar 2012 im Bundesanzeiger, Nr. 10, S. 243 (auch online verfügbar: <http://www.bundesnetzagentur.de>) |
| [CMAC] | NIST Special Publication 800-38B, Recommendation for Block, Cipher Modes of Operation: The CMAC Mode for Authentication, Morris Dworkin, May 2005,  <http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf> |
| [EMV®\_Book-1] | EMV Book 1, Application Independent ICC to Terminal Interface Requirements, version 4.3, November 2011 |
| [ETSI TS 102 222] | ETSI TS 102 222 V7.0.0 (2006-08) Integrated Circuit Cards (ICC); Administrative commands for telecommunications applications (Release 7) |
| [FIPS 180-4] | Federal Information Processing Standards Publication 180‑4 Secure Hash Standard (SHS), March 2012  <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf> |
| [FIPS 186–4] | FIPS PUB 186-4, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, July 2013 DIGITAL SIGNATURE STANDARD (DSS)  <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf> |
| [FIPS 197] | Federal Information Processing Standards Publication 197, (FIPS–197), November 26, 2001, Announcing the ADVANCED ENCRYPTION STANDARD (AES)  <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf> |
| [HBA\_P1] | Spezifikation des elektronischen Heilberufsausweises  Teil I: Kommandos, Algorithmen und Funktionen der Betriebssystemplattform, Version 2.3.2, 05.08.2009  (Vorgängerversion dieses Dokumentes für Karten der Generation 1) |
| [ISO/IEC 7816-3] | ISO/IEC 7816-3: 2006 (2nd edition),  Identification cards — Integrated circuit cards with contacts — Part 3: Electrical interface and transmission protocols |
| [ISO/IEC 7816-4] | ISO/IEC 7816-4: 2013 (3rd edition) Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange |
| [ISO/IEC 7816-5] | ISO/IEC 7816-5: 2004 (2nd edition) Identification cards — Integrated circuit cards — Part 5: Registration of application providers |
| [ISO/IEC 7816-8] | ISO/IEC 7816-8: 2004 (2nd edition) Identification cards — Integrated circuit cards — Part 8: Commands for security operations |
| [ISO/IEC 7816-9] | ISO/IEC 7816-9: 2004 (2nd edition) Identification cards — Integrated circuit cards — Part 9: Commands for card management |
| [ISO/IEC 7816-12] | ISO/IEC 7816-12: 2005-10-04 (1st edition) Identification cards — Integrated circuit cards — Part 12: Cards with contacts — USB electrical interface and operating procedures |
| [ISO/IEC 7816-13] | ISO/IEC 7816-13: 2007 (1st edition) Identification cards — Integrated circuit cards — Part 13: Commands for application management in a multi-application environment |
| [ISO/IEC 9796-2] | ISO/IEC 9796-2: 2010-12-15 (3rd edition) Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms |
| [ISO/IEC 14443-1] | ISO/IEC 14443-1: 2008-06-04 (2nd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 1: Physical characteristics |
| [ISO/IEC 14443-2] | ISO/IEC 14443-2: 2010-08-19 (2nd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 2: Radio frequency power and signal interface |
| [ISO/IEC 14443-3] | ISO/IEC 14443-3: 2011-04-12 (2nd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision |
| [ISO/IEC 14443-4] | ISO/IEC 14443-4: 2008-07-07 (2nd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 4: Transmission protocol |
| [ITU-T X.690] | ITU‑T X.690, 2008-11, entspricht ISO/IEC 8825‑1 Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) |
| [NIST sp800-38a] | Section 6.5 of NIST Special Publication 800-38A, *Recommendation for Block, Cipher Modes of Operation, Methods and Techniques*, Morris Dworkin, December 2001 Edition,  <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf> |
| [PKCS#1] | PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002, <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf> |
| [RFC2119] | Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, March 1997, <http://tools.ietf.org/html/rfc2119> |
| [RFC5639] | Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, RFC 5639, March 2010, <http://tools.ietf.org/html/rfc5639> |
| [BSI-TR-03110-2] | Technical Guideline TR-03110-2, Advanced Security Mechanisms for Machine Readable Travel Documents – Part 2 – Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI), Version 2.10, 2012-03-20 |
| [BSI-TR-03110-3] | Technical Guideline TR-03110-3, Advanced Security Mechanisms for Machine Readable Travel Documents – Part 3 – Common Specifications  Version 2.11, 2013-07-12 |
| [BSI-TR-03111] | Technical Guideline TR-03111, Elliptic Curve Cryptography,  Version 2.0, 2012-06-28 |
| [BSI-TR-03116] | BSI TR-03116, Technische Richtlinie für die eCard-Projekte der Bundesregierung, Version: 3.16, Datum: 07.08.2012, Status: Veröffentlichung, Fassung: August 2012 |

# Anhang F – Asynchrone, gesicherte APDU-Sequenzen (informativ)

## Einleitung

Gemäß einer Anforderung an das COS der Generation 2 soll es einem Kartenmanagement (CMS) möglich sein nicht nur synchron, sondern auch asynchron mit Karten der Generation 2 zu kommunizieren. Dieses Papier beleuchtet die diesbezügliche Anforderungslage und zeigt eine technische Umsetzungsmöglichkeit.

## CMS-Kommunikationsmuster

Hier werden das synchrone und das asynchrone Kommunikationsmuster zwischen einem Kartenmanagement und einer Smartcard aus technischer Sicht betrachtet.

### Synchrone Kommunikation zwischen CMS und Smartcard

Das wesentliche Merkmal der synchronen Kommunikation ist, dass der Sender einer Nachricht auf eine Antwort wartet, bevor die nächste Nachricht versendet wird.



Abbildung 13 - Synchrone Kommunikation zw. CMS und Smartcard

Aus technischer wie aus sicherheitstechnischer Sicht sind hier im Rahmen der synchronen Kommunikation lediglich die beiden Komponenten CMS und Smartcard relevant. Dass die Kommunikationsstrecke über Kartenleser, Router oder Internet führt, ist hier unerheblich. Für Karten der Generation 1 steht am Anfang einer CMS-Prozedur eine gegenseitige Authentisierung mit Sessionkey-Aushandlung. Da an der Sessionkey-Berechnung beide Kommunikationspartner beteiligt sind, ist es erst nach Abschluss der gegenseitigen Authentisierung möglich das vom CMS zu sendende Szenario (Abfolge von Kommando-APDU) mit den ausgehandelten Sessionkeys zu schützen.

### Asynchrone Kommunikation zwischen CMS und Smartcard

Das wesentliche Merkmal der asynchronen Kommunikation ist, dass der Sender einer Nachricht nicht auf eine Antwort wartet. Ein Blockieren des Senders bis zum Empfang der zugehörigen Antwort findet nicht statt.



Abbildung 14 - Asynchrone Kommunikation zw. CMS und Smartcard

Aus technischer, wie aus sicherheitstechnischer Sicht sind hier im Rahmen der asynchronen CMS-Kommunikation drei Komponenten beteiligt:

1. Das CMS erstellt Szenarien (Abfolgen von Kommando-APDU) und bietet diese über eine Schnittstelle an (beispielsweise zum Download).
2. Ein Puffer besorgt sich zu einem beliebigen späteren Zeitpunkt T1 beliebige Szenarien. Zu einem beliebigen Zeitpunkt T2 (wobei im Allgemeinen T1 ungleich T2 ist) werden ein oder mehrere Szenarien vom Puffer an eine Smartcard geschickt.
3. Die Smartcard verarbeitet die Szenarien und erstellt dabei typischerweise „Quittungen“, die eine erfolgreiche Verarbeitung anzeigen. Diese „Quittungen“ können vom Puffer zu einem beliebigen späteren Zeitpunkt an das CMS übertragen werden.

Für die Komponente Puffer gibt es diverse Realisierungsmöglichkeiten. Im einen Extrem befindet sich der Puffer möglichst nah an der Smartcard, beispielsweise versendet das CMS per Email Szenarien etwa an einen Karteninhaber. Dann wäre „Puffer“ aus Sicht des Karteninhabers ein lokales Programm. Im anderen Extrem befindet sich der Puffer möglichst nah am CMS, beispielsweise wäre der Puffer lediglich der per Internet erreichbare Online-Teil des CMS und der übrige CMS Teil (mit sensiblen, personenbezogenen Daten und Schlüsseln) wäre nicht per Internet erreichbar.

## Anforderungen an die asynchrone Kommunikation

Aus sicherheitstechnischer Sicht werden an die asynchrone Kommunikation folgende Anforderungen gestellt:

1. Asynchrone Kommunikation: Es MUSS dem CMS möglich sein komplette Szenarien so vorzuberechnen, dass das CMS beim Einspielen der Szenarien in eine Smartcard nicht beteiligt ist.
2. Quittungen: Es MUSS möglich sein, dass die Smartcard für einzelne oder alle Kommando-APDU eines Szenarios Quittungen so erzeugt, dass sich das CMS zweifelsfrei von einer erfolgreichen Kommandobearbeitung überzeugen kann.
3. Authentizität: Das CMS MUSS sich zweifelsfrei als Quelle eines Szenarios und jeder einzelnen Kommando-APDU innerhalb eines Szenarios identifizieren lassen.
4. Integrität eines Szenarios: Jede Veränderung eines Szenarios und jeder einzelnen Kommando-APDU MUSS sicher erkennbar sein. Typische Veränderungen sind:
   1. Verändern einer einzelnen Kommando-APDU
   2. Hinzufügen beliebiger Kommando-APDU (auch Duplizieren von Kommando-APDU)
   3. Weglassen beliebiger Kommando-APDU
   4. Verändern der Reihenfolge von Kommando-APDU
5. Vertraulichkeit: Es MUSS möglich sein, die Vertraulichkeit von Informationen zu gewährleisten, die in einem Szenario transportiert werden.
6. Reihenfolge von Szenarien: Es MUSS möglich sein, die Reihenfolge von Szenarien vorzugeben, beispielsweise: Szenario\_B nur, wenn auch Szenario\_A durchgeführt wurde. Relevant wird diese Anforderung etwa dann, wenn zunächst eine Anwendung geladen wird und später die Anwendungsdaten aktualisiert werden sollen. Eine Aktualisierung von Anwendungsdaten ist nur dann sinnvoll, wenn die Anwendung geladen wurde.
7. Resistenz gegen Replay Attacken: Es ist auszuschließen, dass ein Szenario ein zweites Mal erfolgreich durchgeführt wird.
8. Überspringen von Szenarien: Es MUSS dem CMS möglich sein Szenarien als obsolet zu kennzeichnen. Relevant wird diese Anforderung etwa dann, wenn sich Anwendungsdaten regelmäßig oder rasch ändern. Anstatt dann zunächst veraltete Szenarien an die Smartcard zu schicken, ist es dann möglich gleich das aktuelle Szenario auszuführen. Konkret:
   1. Regelmäßige Änderung von Anwendungsdaten: CV-Zertifikate mit Gültigkeitszeiträumen von wenigen Tagen sind regelmäßig zu aktualisieren. Nach einem mehrwöchigen Urlaub ist es sinnvoller gleich das aktuelle CV-Zertifikat zu laden und alle während des Urlaubs angefallenen Zwischenschritte der CVC-Aktualisierung zu überspringen.
   2. Rasche Änderung von Anwendungsdaten: In gewissen Situationen ändern sich die Versichertendaten möglicherweise mehrmals kurz hintereinander (etwa ändere sich durch eine Heirat zunächst der Familienstatus und durch den Bezug einer gemeinsamen Wohnung dann die Adresse, oder nach Abschluss einer Ausbildung / Studium ändert sich zunächst der Versichertenstatus und dann durch einen Wohnortwechsel im Zusammenhang mit einer Arbeitsaufnahme die Adresse, etc.). Dann erscheint es sinnvoller veraltete Szenarien zu überspringen.

## Lösungskonzept

Zur Umsetzung der in F.3 vorgeschlagenen Anforderungen wird folgendes Konzept vorge-schlagen:

1. Kryptographisches Material: Sowohl CMS, als auch jede vom CMS betreute Smartcard verfügen über kryptographisches Material. Das CMS weist damit seine Au-thentizität nach (Anforderung Authentizität). Zudem wird es zum sicheren Transport von Sessionkeys hin zur Smartcard verwendet (Anforderung asynchrone Kommunikation). Sessionkeys und SendSequenceCounter schützen die Kommando-APDU (Anforderungen Authentizität, Integrität eines Szenarios, Vertraulichkeit) eines Szenarios sowie die Antwort-APDU einer Smartcard (Anforderung Quittungen).
2. NumberScenario: Jedem Szenario wird vom CMS eine natürliche Zahl zugeordnet. Die Zuordnung zwischen Szenario und NumberScenario MUSS bijektiv (d.h. umkehrbar eindeutig) sein.
3. NumberPrecondition: Jedem Szenario wird vom CMS eine weitere natürliche Zahl oder die null zugeordnet. Diese gibt an, welches Szenario als Vorbedingung in eine Smartcard geladen worden sein muss, damit dieses Szenario akzeptiert wird (siehe Beispiel) (Anforderung Reihenfolge von Szenarien, Überspringen von Szenarien).
4. Die Smartcard speichert NumberScenario eines Szenarios. Szenarien MÜSSEN so an die Smartcard geschickt werden, dass ihr NumberScenario streng monoton steigt (Anforderungen Resistent gegen Replay Attacken). Zudem MUSS NumberPrecondition eines Szenarios kleiner oder gleich dem in der Smartcard gespeicherten Wert von NumberScenario sein (Anforderung Reihenfolge von Szenarien, Überspringen von Szenarien).

Im folgenden Beispiel gelte folgende Notation: S*NumberPrecondition*(*NumberScenario*). Dem Szenario S wird damit die natürliche Zahl *NumberScenario* und die Vorbedingung *NumberPrecondition* zugeordnet. Die folgende Tabelle stellt eine Chronologie dar, wobei die Zeit mit der Zeilennummer zunehme. Die Spalte CMS enthält Szenarien, die vom CMS zum betrachteten Zeitpunkt bereitgestellt werden. Die Spalte Puffer enthält die dort gespeicherten Szenarien. Szenarien sind grün hinterlegt, wenn sie von einer Smartcard akzeptiert werden und rot hinterlegt, wenn sie von einer Smartcard wegen unpassender *NumberPrecondition* oder unpassender *NumberScenario* nicht akzeptiert werden. Gelb hinterlegte Szenarien werden im betrachteten Zeitpunkt an die Smartcard geschickt.

| **CMS** | **Puffer** | **Smartcard** | **Bemerkung** |
| --- | --- | --- | --- |
| ‑ | ‑ | *NumberScenario* = 0 | Zu einem bestimmten Zeitpunkt sei eine Smartcard im Feld und der Wert von *NumberScenario* sei 0. |
| S0(1) | S0(1) | ‑ | Das CMS stellt ein Szenario zur Verfügung, welches vom Puffer heruntergeladen werde. Die Smartcard ist inaktiv. |
| S0(2) | S0(1) S0(2) | ‑ | Das CMS ersetzt S0(1) durch S0(2). Durch das Ersetzen und da *NumberPrecondition* unverändert auf 0 steht wird S0(1) als obsolet gekennzeichnet. Der Puffer lädt zusätzlich S0(2) herunter. Die Smartcard ist inaktiv. |
| S0(2) S2(3) | S0(1) S0(2) S2(3) | ‑ | Das CMS stellt S2(3) zusätzlich zu S0(2) zur Verfügung und der Puffer lädt auch dieses herunter. *NumberPrecondition* in S2(3) ist so gesetzt, dass erst S0(2) eingespielt werden muss, bevor S2(3) einspeilbar ist. Die Smartcard ist weiter inaktiv. |
| S0(2) S2(4) | S0(1) S0(2) S2(3) S2(4) | ‑ | Das CMS ersetzt S2(3) durch S2(4). Durch das Ersetzen und da *NumberPrecondition* unverändert auf 2 steht, wird S2(3) als obsolet gekennzeichnet. Der Puffer lädt zusätzlich S2(4) herunter. Die Smartcard ist inaktiv. |
| S0(2) S2(4) | S0(1) S0(2) S2(3) S2(4) | S0(1) zur Karte *NumberScenario* = 1 | Der Puffer schicke S0(1) an die Smartcard. Dies ist suboptimal (aber unschädlich), da S0(1) durch S0(2) obsolet wurde. |
| S0(2) S2(4) | S0(1) S0(2) S2(3) S2(4) | S0(2) zur Karte *NumberScenario* = 2 | Der Puffer schicke das einzig akzeptable Szenario S0(2) an die Smartcard.  S0(1) ist nicht akzeptabel, weil dessen *NumberScenario* nicht größer ist als *NumberScenario* der Smartcard.  S2(3) ist nicht akzeptabel, da dessen *NumberPrecondition* nicht kleiner ist als *NumberScenario* der Smartcard. |
| S0(2) S2(4) | S0(1) S0(2) S2(3) S2(4) | S2(4) zur Karte *NumberScenario* = 4 | Diesmal arbeite der Puffer optimal und schicke das aktuellste akzeptable Szenario an die Smartcard.  Zudem schicke der Puffer nach Abschluss des Szenarios die zu S2(4) gehörende Quittung an das CMS. |
| ‑ | S0(1) S0(2) S2(3) S2(4) | *NumberScenario* = 4 | Nach Empfang der Quittung können die Szenarien aus dem Downloadbereich des CMS entfernt werden.  Keines der bisher vom CMS bereitgestellten Szenarien ist für die Smartcard akzeptabel. |

## Kryptographie

Zur Erfüllung der in F.3 genannten kryptographischen Anforderungen werden symmetrische und asymmetrische Authentisierungsschlüssel verwendet.

### CMS Aktivitäten

Auf Seiten des CMS ist der Ablauf wie folgt:

1. Erstellen des gesicherten Szenarios, hierfür sind keine kartenindividuellen Informationen erforderlich:
   1. Zu Zwecken des Kartenmanagements wird eine Sequenz von ungesicherten Kommando-APDU erstellt.
   2. Das CMS erzeugt einen Seed zur Ableitung von AES-Sessionkeys und entsprechenden Werten für die SendSequenceCounter für die CMAC-Berechnung.
   3. Die ungesicherten Kommando-APDU werden mittels der Sessionkeys und SendSequenceCounter konform zu Kapitel 13 gesichert.
2. Kartenindividuelles Szenario:
   1. Das CMS besorgt sich (aus einer Datenbank oder ähnlichem) kartenspezifische Werte für ICCSN, NumberPrecondition, NumberScenario und das Schlüsselmaterial:
      1. symmetrische Administration: SK.CMS\_ENC,   SK.CMS\_MAC.
      2. asymmetrische Administration: PuK.SmartCard.AUT,   PrK.CMS, PuK.CMS.
   2. Der Seed aus Schritt 16)b) wird verschlüsselt:
      1. symmetrische Administration:   
         cipherText = AES\_CBC\_ENC( SK.CMS\_ENC, ICV=´00…00´, Seed ).
      2. asymmetrische Administration:
         1. ( *POA*, *C*, *T* ) = ELC\_ENC( Seed, PuK.SmartCard.AUT, PuK.CMS.*dP* )
         2. Setze   *oidDO* = ´06–L06–*affectedObject*.*domainParameter*.OID´.
         3. Setze   *keyDO* = ´7F49–L7F49–( 86–L86–*POA* )´.
         4. Setze   *cipherDO* = ´86–L86–( 02   ||   *C* )´.
         5. Setze   *macDO* = ´8E–L8E–*T*´.
         6. Setze   cipherText = ´A6–LA6–(*oidDO* || *keyDO* || *cipherDO* || *macDO* )´.
   3. Es wird eine Nachricht M wie folgt zusammengestellt:   
      M = NumberPrecondition || NumberScenario || cipherText.
   4. Für die Nachricht M werden Authentisierungsdaten berechnet:
      1. symmetrische Administration: Es wird ein CMAC für M berechnet:   
         CMAC = CalculateCMAC\_IsoPadding( SK.CMS\_MAC, M ).
      2. asymmetrische Administration: Es wird eine Signatur für M berechnet:
         1. Falls PrK.CMS.*dP* = brainpool256   =>   *hash* = SHA\_256( *M* ).
         2. Falls PrK.CMS.*dP* = brainpool384   =>   *hash* = SHA\_384( *M* ).
         3. Falls PrK.CMS.*dP* = brainpool512   =>   *hash* = SHA\_512( *M* ).
         4. ( *R*, *S* ) = ELC\_SIG( PrK.CMS, *hash* )
         5. SIG = *R* || *S*.
   5. Eine Authentisierungssequenz mit M und CMAC bzw. SIG wird dem Szenario aus Schritt 16)c) vorangestellt.

### Puffer-Aktivitäten

Auf Seiten des Puffers ist der Ablauf im Gutfall wie folgt:

1. Der Puffer ermittelt von einer Smartcard die Werte ICCSN und NumberScenario.
2. Der Puffer wählt ein passendes Szenario aus.
3. Der Puffer schickt das ausgewählte Szenario an die Smartcard.
4. Der Puffer schickt die gesicherten Antwort-APDU zusammen mit ICCSN und NumberScenario als Quittung an das CMS.

### Smartcard-Aktivitäten

Auf Seiten der Smartcard ist der Ablauf wie folgt:

1. Schlüsselauswahl: Zu Beginn des Szenarios werden ein passendes Schlüsselobjekt und ein passender Algorithmus ausgewählt (etwa durch ein MSE-Set-Kommando).
2. Authentisierung: Die Authentisierungssequenz aus Schritt 17)e) folgt im Szenario als nächstes. Dabei werden M und CMAC bzw. SIG übertragen und in der Smartcard wie folgt verarbeitet:
   1. Der CMAC bzw. SIG wird überprüft. Ein unpassender CMAC oder eine fehlerhafte Signatur führt zum Kommandoabbruch.
   2. Der Nachricht M werden NumberPrecondition, NumberScenario und cipherText entnommen.
      1. Falls NumberPrecondition in M größer als SK.CMS🡪*numberScenario* bzw. PrK.SmartCard.AUT🡪*numberScenario* ist, führt dies zum Kommandoabbruch.
      2. Falls NumberScenario in M kleiner oder gleich SK.CMS🡪*numberScenario* bzw. PrK.SmartCard.AUT🡪*numberScenario* ist, führt dies zum Kommandoabbruch.
   3. SK.CMS🡪*numberScenario* bzw. PrK.SmartCard.AUT🡪*numberScenario* wird mit Transaktionsschutz auf den Wert von NumberScenario aus M gesetzt.
   4. cipherText wird mittels SK.CMS\_ENC bzw. PrK.SmartCard.AUT entschlüsselt und Seed wird zum Berechnen von Sessionkeymaterial inklusive SendSequenceCountern verwendet.
3. Kartenmanagementkommanods: Die übrigen, Secure Messaging gesicherten Kommando-APDU werden an die Smartcard geschickt. Secure-Messaging-Fehler führen (wie üblich) zum Kommandoabbruch.

## Auswirkungen auf die Kommandoschnittstelle

Das hier vorgestellte Konzept der asynchronen Kommunikation zwischen CMS und Smartcard wirkt sich an zwei Stellen auf die Smartcard aus: Zum einen muss es dem Puffer möglich sein NumberScenario aus der Smartcard auszulesen, damit das passende Szenario ausgewählt werden kann. Zum anderen gilt es den Sessionkeykontext in der Smartcard zu etablieren.

1. RESET der Smartcard
2. MSE Set P1P2=´81A4´, keyReference=SK.CMS, algID=asynchroneCMS
3. GENERAL AUTHENTICATE Datenfeld so, dass NumberScenario von der Smartcard angefordert wird.
4. GENERAL AUTHENTICATE Datenfeld mit M und CMAC (oder M und Signatur), ein passender CMAC (bzw. eine gültige Signatur) setzt einen Sicherheitszustand. Mit den Informationen aus M werden Sessionkeys und SendSequenceCounter eingerichtet.
5. Der Rest des Szenarios wird mit Secure Messaging geschützt an die Smartcard geschickt.

# Anhang G – CV-Zertifikate für RSA-Schlüssel, Option\_RSA\_CVC

Der Inhalt dieses Anhangs wurde nach [gemSpec\_PKI] verschoben.

# Anhang F – CV-Zertifikate für ELC-Schlüssel (normativ)

Der Inhalt dieses Anhangs wurde nach [gemSpec\_PKI] verschoben.

# Anhang F – Speichern öffentlicher Schlüssel (informativ)

Dieser Anhang beschreibt, wie öffentliche Schlüsselobjekte (siehe 8.6.4) bezüglich ihrer Einordnung in ein Objektsystem in diesem Dokument betrachtet werden. Einerseits werden öffentliche Schlüsselobjekte genau so wie andere Objekte auch einem Ordner zugeordent und damit in die Objektsystemhierarchie eingeordnet wie die übrigen Objekte. Zudem werden sie aus funktionaler Sicht analog zu anderen Schlüsselobjekten mittles Manage Security Environment selektiert und im Rahmen einer Kommandobearbeitung gesucht und angesprochen. Andererseits werden bestimmte öffentliche Schlüsselobjekte mittels CV-Zertifikaten dynamisch zur Laufzeit in eine Smartcard importiert und es wäre unpassend, wenn dadurch der gesamte freie Speicher verbraucht würde. Deshalb ist das Importieren mittels CV-Zertifikaten aus Sicht des Speichermanagements anders zu betrachten, als etwa das Einbringen neuer Objekte mittels Load Application.

Dieser Anhang und die normativen Teile dieses Dokumentes beschreiben öffentliche Schlüsselobjekte aus diversen Perspektiven mit dem Ziel die Funktionalität an der äußeren Kartenschnittstelle zu verdeutlichen und normativ festzuschreiben. Hier, wie auch im Rest des Dokumentes, ist es nicht das Ziel eine bestimmte Implementierung zu fordern oder zu präferieren.

## Definitionen

*allPublicKeyList*: Ist die Vereinigungsmenge von *applicationPublicKeyList* und Cache.

*applicationPublicKeyList*, (N019.900)d: Liste zur persistenten Speicherung von öffentlichen Schlüsselobjekten der Anwendungen. Im Rahmen dieser Betrachtung sind das öffentliche Schlüsselobjekte, die im Rahmen einer Initialisierung oder als Teil einer Ordnerstruktur mittels Load Application in die Smartcard geladen werden.

Cache: Bereich zur temporären, möglicherweise persistenten Speicherung von Schlüsselobjekten, die mittels CV-Zertifikaten importiert wurden. Vereinigungsmenge von *persistentCache* und *volatileCache*.

CA-Schlüssel: Öffentlicher Signaturprüfschlüssel einer CVC-Sub-CA, welcher mittels CV-Zertifikat in eine Karte importiert wird.

EE-Schlüssel: Öffentlicher Authentisierungsschlüssel eines Endnutzers (**E**nd-**E**ntity), welcher mittels CV-Zertifikat in eine Karte importiert wird.

*persistentCache*, (N019.900)e: Teil des Cache der persistent gespeichert ist.

*persistentPublicKeyList*: Vereinigungsmenge von *applicationPublicKeyList* und *persistentCache*.

Sicherheitsanker: Öffentliches Signaturprüfobjekt, welches den öffentlichen Schlüssel einer CVC-Root-CA enthält. Typischerweise werden Sicherheitsanker im Rahmen einer Initialisierung oder Personalisierung in eine Smartcard geladen. Zusätzlich ist es möglich Sicherheitsanker mittels Link-Zertifikaten in die Karte zu importieren.

*volatileCache*, (N019.900)g: Teil des Cache der volatil gespeichert ist.



Abbildung 15: Zusammenhang Schlüsselspeicher

## Perspektiven

Aus der Perspektive der Kommandobearbeitung wird in diversen Kommandos zunächst nach einem Objekt gesucht, auf welches sich das Kommando bezieht. Bezogen auf öffenliche Schlüsselobjekte fasst dieses Dokument alle öffentlichen Schlüsselobjekte in einer Liste *allPublicKeyList* zusammen.

Aus der Perspektive des Nachladens von Ordnern mittels Load Application werden öffentliche Schlüsselobjekte, die im Ordner enthalten sind, in der Sichtweise dieses Dokumentes *applicationPublicKeyList* hinzugefügt. Es wird davon ausgegangen, dass *applicationPublicKeyList* nur eine begrenzte Kapazität hat. Deshalb ist es denkbar, dass ein Load Application aus Platzmangel scheitert, wenn die Menge der nachgeladenen Schlüsselobjekte die Kapazität von *applicationPublicKeyList* übersteigt.

Aus der Perspektive des Importes mittels CV-Zertifikaten werden importiere Schlüssel dem Cache hinzugefügt. Es wird davon ausgegangen, dass der Cache nur eine begrenzte Kapazität hat. Falls die Menge an importierten Schlüsseln die Kapazität des Cache übersteigt, werden „unwichtige“ Einträge aus dem Cache entfernt um Platz für den importierten Schlüssel zu schaffen.