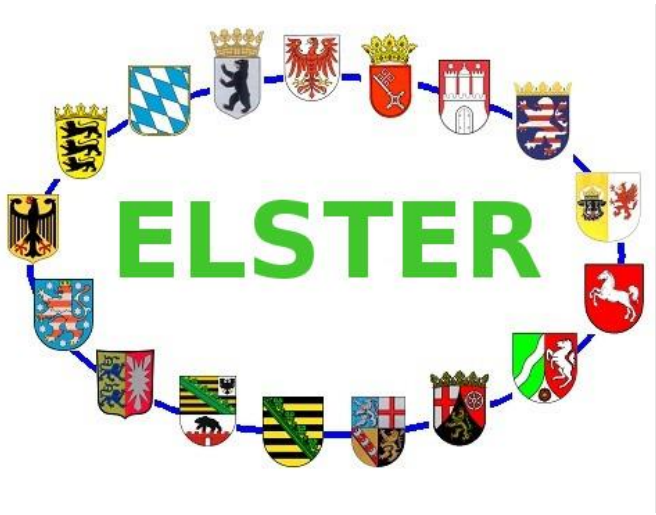


Technische Dokumentation (TDoc)

Sicherheitskomponente Client
Applet für SCE6/7

Bezeichnung	*Verfahren:* **ELSTER** *Projekt:* *Applet* für SCE6/7 Auftragnehmer: secunet Security Networks AG
Verfahrensmanager	Roland Krebs
Projektmanager	Marcus Scherz
Erstellt am	Donnerstag, 16. Juli 2009
Zuletzt geändert	09.11.2017 10:40
Bearbeitungszustand	☐ in Bearbeitung ☐ vorgelegt ☒ fertig gestellt
Dokumentablage	*Bitte einfügen!*

INHALTSVERZEICHNIS

Ansprechpartner

Michael Stoll, SECUNET AG,
Tel. +49 (0) 201 5454-3050, michael.stoll@secunet.com

Abkürzungen

APDU	Application Protocol Data Unit („Datenelement des Anwendungsprotokolls“)
NFC	Near Field Communication („Nahfeldkommunikation“)
USB	Universal Serial Bus („Universeller serieller Bus“)
PIN	Personal identification number (“Persönliche Identifikationsnummer“)
PUK	Personal Unblocking Key
RSA	Rivest, Shamir und Adleman (asymmetrisches kryptographisches Verfahren)

PSS Probabilistic Signature Scheme („probabilistisches Signaturverfahren“)

Historie

Version	Datum	Änderung	Autor
0.1	28.02.2017	Erster Entwurf	Michael Stoll
0.2	13.03.2017	Korrekturen	Michael Stoll
0.3	27.03.2017	Neue APDU (reset token)	Michael Stoll
0.4	02.06.2017	GetResponse APDU ergänzt	Wolfgang Christ
0.5	08.06.2017	Ablaufdiagramme für Schlüsselerzeugung, Entschlüsselung und Signatur	Wolfgang Christ

Referenzierte Dokumente

/1/ GlobalPlatform Card Specification Version 2.2.1, Public Release January 2011 Document Reference: GPC_SPE_034

/2/ Java Card 3 Platform Development Kit User Guide, Java Card™ Platform, Version 3.0.1, Classic Edition, May 2009

1. Einleitung

Dieses Dokument dient als Beschreibung für das secunet Aut-Applet.

1.1 Hardware Unterbau

Das Java-Applet (Applet) soll lauffähig sein auf einem USB-Token vom Typ “Giesecke & Devrient GmbH StarSign CUT S”. Dieser Token basiert auf einer SCE7.0 (SmartCafe Expert 7)

Zwecks weiterer Nutzung alter Sticks, wird auch ein USB-Token vom Typ “Giesecke & Devrient GmbH StarSign CUT” unterstützt. Dieser Token basiert auf einer SCE6.0 (SmartCafe Expert 6).

Das Applet basiert auf der JavaCard Spezifikation 3.0.1. und ist JDK 1.6 kompatibel.

1.2 Anforderungen an das Interface

Der Token soll über den USB-Anschluss das Protokoll “T=1” mit “EXTENDED LENGTH” APDU unterstützen. Aktuell wird das Applet aber so definiert, das nur Command Chaining mit Standard APDUs zum Einsatz kommt. Dies hat den Vorteil, dass das Applet, wenn es auf eine Dual Interface Karte geladen wird, auch mit NFC über das Protokoll “T=CL“ genutzt werden kann.

1.3 Entwicklungsumgebung

Das Applet wird mit der ID Applet Developer Suite v3.0 Stand Juni 2014 entwickelt.

2. Komponenten auf dem Token

2.1 Komponenten des Betriebssystems

2.1.1 Issuer Security Domain

Card Manager Keys: Es werden Standard Cardmanager Keys verwendet: Siehe Anhang

Die zugeh. APDUs sind in der Global Platform Spezifikation beschrieben.

2.2 Komponenten der Applikation

2.2.1 Applet

2.2.1.1 Daten-Container

Es gibt keinen separaten Daten-Container, die Schlüssel, Passwörter und Zertifikate werden im Applet gespeichert. Bei einem Update des Applets werden die Schlüssel und Zertifikate des Anwenders gelöscht. Ein Backup der privaten Schlüssel ist nicht vorgesehen.

2.2.1.2 Kryptographische Komponenten

2.2.1.2.1 Passwörter (PIN/PUK)

- Applikations-PIN
  Für das Ausführen der meisten Funktionen ist die PIN erforderlich, sie muss deshalb bei der Inbetriebnahme (Initialisierung) vergeben werden.

  Die PIN ist genau 6 Zeichen lang und der Fehlbedienungszähler wird mit 3 initialisiert.

- Applikations-PUK

  Die PUK ist optional. Wenn die Verwendung einer PUK gewünscht ist, muss sie vor der PIN gesetzt werden.

  Die PUK ist genau 15 Zeichen lang und der Fehlbedienungszähler ist initial 5.

2.2.1.2.2 Schlüssel, die auf dem Token generiert werden

Diese Schlüsselpaare werden während dem initialen Einrichten des Tokens, nach der Vergabe der PIN, erzeugt. Es sind 2 RSA Schlüssel mit 2048 Bit Länge vorgesehen.

- Schlüssel für Signatur-Operationen
- Schlüssel für Entschlüsselung

2.2.1.2.3 X.509 Zertifikate, welche in das Token importiert werden

Die öffentlichen Schlüssel werden zusätzlich in Form von 2 Zertifikaten gespeichert. Die Zertifikate (auch selbstsignierte) werden extern erzeugt und dann in das Applet abgelegt. Eine Überprüfung, ob die Schlüssel im Zertifikat zu den anderen, im Applet enthaltenen Schlüsseln passen, erfolgt nicht.

Die Zertifikate können beliebig oft überschrieben werden, wenn die PIN Prüfung erfolgreich war.

Die maximale Größe sind 4096 Byte.

2.2.1.2.4 Infodaten, welche in das Token importiert werden

Es besteht die Möglichkeit Infodaten abzulegen. Die Daten können beliebig oft geändert werden, wenn die PIN Prüfung erfolgreich war.

Die maximale Größe sind 2048 Byte.

3. Abläufe für den Token (Use Cases)

3.1 Installation der SW auf dem USB-Token

Das Applet wird als CAP File an den Hersteller geliefert, der das Applet aufbringt (INSTALL for LOAD) und installiert (INSTALL for INSTALL).

Folgende AIDs werden verwendet:

- PackageAid: A0 00 00 00 66 80 01 34 01
- AppletClassAid: A0 00 00 00 66 80 01 34 01 01
- AppletInstanceAid: A0 00 00 00 66 80 01 34 01 01

Das Applet erhält den Versionsnummern:

- Major: 00
- Minor: 01

3.2 Initiale Einrichtung eines neuen Tokens

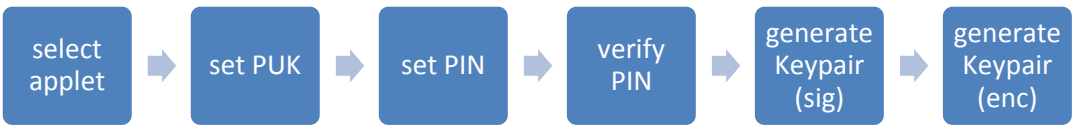
3.2.1 Benötigte Komponenten

- Die Applikations-PIN und die –PUK

3.2.2 Ablauf der Initialen Einrichtung des USB-Token

- Zuerst müssen die 15-stellige PUK und die 6-stellige PIN vergeben werden. Im Gegensatz zur PIN, kann die PUK nachträglich nicht mehr verändert werden.
- Auf dem Token können 2 Schlüsselpaare erzeugt werden. Eines für Signaturen und eines für Entschlüsselungen.

3.2.3 Ablaufdiagramm



Befehl	CLA	INS	P1	P2	Data ([Length]Value)	Le	Rückgabe	Antwort
Select Applet	00	A4	04	00	[0A] AID	3	Version	9000
Set PUK	00	24	01	02	[0F] PUK	-	-	9000
Set PIN	00	24	01	01	[06] PIN	-	-	9000
Verify PIN	00	20	00	01	[06] PIN	-	-	9000
Generate Keypair (sig)	80	46	42	00	-	00	Public Key	9000
Generate Keypair (enc)	80	46	42	01	-	00	Public Key	9000

3.2.4 Zertifikatsrequest erzeugen

- Erzeugung des P10 Requests für Signatur und Entschlüsselungsschlüssel durch externe Anwendung:
    - Die Anwendung liest den Public Schlüssel
    - Die Anwendung erzeugt einen Zertifikatsrequest und errechnet den Hash Wert über den „to be signed“ Bereich
    - Die Anwendung erzeugt das PSS Padding
    - Das Applet signiert mit dem privaten Schlüssel (Raw RSA)
    - Die Anwendung baut die Signatur in den P10 Request ein und schickt ihn an die CA
- Erzeugung des selbstsignierten Zertifikats für Signatur und Entschlüsselungsschlüssel durch externe Anwendung:
    - Die Anwendung erzeugt ein selbstsigniertes Zertifikat und errechnet den Hash Wert über den „to be signed“ Bereich

- Die Anwendung erzeugt das PSS Padding
- Das Applet signiert mit dem privaten Schlüssel (Raw RSA)
- Die Anwendung baut die Signatur in das Zertifikat ein
- Die Anwendung speichert das Zertifikat im Applet

3.3 CA Zertifikat einspielen

Erhält die Anwendung die CA Zertifikate, werden die Benutzerzertifikate (nicht die Zertifikatskette) auf den Token übertragen und gespeichert. Die Anwendung muss vor dem Speichern vergleichen, dass es sich um Zertifikate handelt, die zu den Schlüsseln auf dem Token passen, das Applet prüft dies nicht.

3.3.1 Benötigte Komponenten

- PIN

3.3.2 Ablauf

- Selektion des Applets auf dem Token.
- Autorisierung mit der PIN
- Aufruf des Kommandos speichere Zertifikat 1 (Signatur)
- Aufruf des Kommandos speichere Zertifikat 2 (Entschlüsselung)

3.3.3 Rückgabe

Keine Daten werden zurückgegeben.

3.4 PIN Ändern

Um den Wert der PIN zu ändern darf die PIN nicht durch den Wiederholzähler geblockt sein.

3.4.1 Benötigte Komponenten

- PIN

3.4.2 Ablauf

- Selektion des Applets auf dem USB-Token.
  Der USB-Token antwortet mit der Applet Version.
- Es wird das Kommando CHANGE PIN an den USB-Token geschickt.

- Das Kommando enthält die alte PIN und eine neue PIN in den Kommandodaten.
- Sowohl die alte PIN als auch die neue PIN enthält Passwörter mit genau 6 ASCII Zeichen

3.4.3 Rückgabe

Keine Daten werden zurückgegeben.

3.5 PIN Rücksetzung

Die Funktion PIN zurücksetzen wird benötigt, wenn die PIN geblockt ist, um eine neue PIN zu vergeben und deren Wiederholzähler auf den initialen Wert zu setzen.

3.5.1 Benötigte Komponenten

- PIN und PUK

3.5.2 Ablauf

- Selektion des Applets auf dem USB-Token.
  Der USB-Token antwortet mit der Applet Version.
- Es wird das Kommando RESET PIN an den USB-Token geschickt.
  - Das Kommando enthält die PUK und eine neue PIN in den Kommandodaten.
  - Sowohl die PUK als auch die PIN enthält Passwörter mit exakt 15 (PUK) bzw. 6 (PIN) ASCII Zeichen.

3.5.3 Rückgabe

Keine Daten werden zurückgegeben.

3.6 Entschlüsselung von Daten

Vor der Kartenkommunikation muss die Anwendung das RSA Kryptogramm aus den übergebenen Daten extrahieren. Das Kryptogramm hat die Länge des RSA Schlüssels. Benötigte Komponenten

- PIN

3.6.1 Ablauf

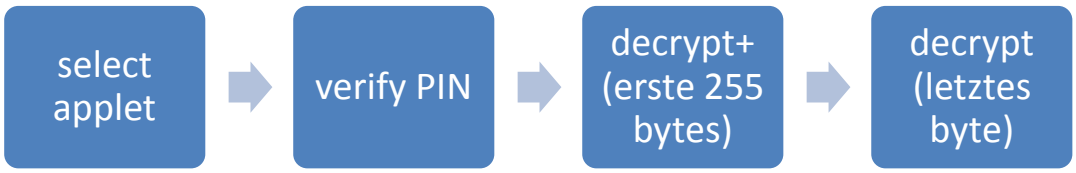
- Selektion des Applets auf dem USB-Token.
  Der USB-Token antwortet mit der Applet Version
- Autorisierung mit der PIN
- Aufruf der Kommandos zum Entschlüsseln von Daten

3.6.2 Rückgabe

- RSA Raw entschlüsselte Daten

  Die Anwendung entfernt das Padding und erhält so üblicherweise den Content Encryption Key (oder ein anderes Geheimnis)

3.6.3 Ablaufdiagramm



Befehl	CLA	INS	P1	P2	Data ([Length]Value)	Le	Rückgabe	Antwort
Select Applet	00	A4	04	00	[0A] AID	03	Version	9000
Verify PIN	00	20	00	01	[06] PIN	-	-	9000
Decrypt (erste 255 Bytes)	90	2A	80	86	[FF] <data>	-	-	9000
Decrypt (verbleibendes Byte)	80	2A	80	86	[01] xx	00	decrypted data	9000

3.7 Signatur von Daten

Vor der Kartenkommunikation muss die Anwendung die zu signierenden Daten hashen und das Padding durchführen.

3.7.1 Benötigte Komponenten

- PIN

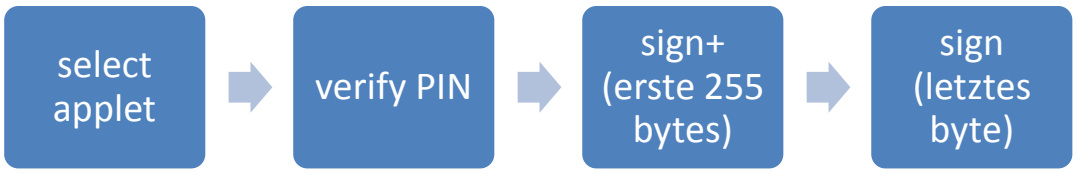
3.7.2 Ablauf

- Selektion des Applets auf dem USB-Token.
  Der USB-Token antwortet mit der Applet Version
- Autorisierung mit der PIN
- Aufruf der Kommandos zum Signieren von Daten

3.7.3 Rückgabe

- RSA Signatur

3.7.4 Ablaufdiagramm



Befehl	CLA	INS	P1	P2	Data ([Length]Value)	Le	Rückgabe	Antwort
Select Applet	00	A4	04	00	[0A] AID	03	Version	9000
Verify PIN	00	20	00	01	[06] PIN	-	-	9000
Sign (erste 255 Bytes)	90	2A	9E	9A	[FF] <data>	-	-	9000
Sign (verbleibendes Byte)	80	2A	9E	9A	[01] xx	00	signed data	9000

3.8 Zufallszahl

Das Applet kann Zufallszahlen generieren.

3.8.1 Benötigte Komponenten

- keine

3.8.2 Ablauf

- Selektion des Applets auf dem USB-Token.
- Aufruf der Kommandos zum Generieren von Zufallszahlen

3.8.3 Rückgabe

- Zufallswert

4. Kommunikation mit dem Token

4.1 APDU Kommandos

4.1.1 Syntax in den APDU Kommandos

‚00'..'FF' Stehen für hexadezimale Daten-Bytes.

‚00 00' .. ‚FF FF' Steht für Datenwort mit 16 Bits.

‚XX' Bedeutet eine Datum mit beliebigem Wert

#(Daten) Steht für eine BER-TLV codierte Länge über die Daten.

4.1.2 Select APPLET APDU

Diese Kommando-APDU wird genutzt um eine Instanz eines Applets auf der Smart Card anzuwählen.

Command		
CLA	‚00'	ISO Class unverschlüsselt
INS	‚A4'	SELECT
P1|P2	‚04 00'	Select mit AID
LC	variabel	Länge der Kommandodaten.
DATA	variabel	AID der Instanz.
LE	'00'	Rückgabe aller Daten
Response		
DATA	variabel	Versionsdaten, aktuell ‚00 01'
SW1SW2	'9000'	Erfolgreiche Kommandobearbeitung
	'6700'	Fehler hafte Länge der Kommandodaten
	'6A82'	Fehler, Instanz nicht vorhanden
	'6A86'	Fehler, ungültige Parameter P1, P2

	‘6999’	Fehler, Applet konnte nicht selektiert werden
	‘6A80’	Fehlerhafte Kommandodaten.

Tabelle 1: SELECT APPLET Kommando APDU

4.1.2.1 Voraussetzungen

Keine

4.1.2.2 Effekte

Nach erfolgreichem Selektieren der Instanz gehen alle APDU-Kommandos an das selektierte Applet.

4.1.3 DELETE MF

Diese Kommando-APDU wird genutzt um Benutzerdaten im Applet zu entfernen und das Applet in einen, dem Auslieferungszustand ähnlichen Zustand, zu versetzen.

Command		
CLA	‚00’	ISO Class unverschlüsselt
INS	‚E4’	DELETE
P1|P2	‚00 00’	
LC	‚02’	Länge der Kommandodaten (2 Byte fix).
DATA	‚3F 00 ’	
LE	‘00’	Rückgabe aller Daten
Response		
SW1SW2	’9000’	Erfolgreiche Kommandobearbeitung
	’6700’	Fehler hafte Länge der Kommandodaten
	’6A86’	Fehler, ungültige Parameter P1, P2
	‘6984’	Fehlerhafte Kommandodaten.

Tabelle 2: DELETE MF Kommando APDU

4.1.3.1 Voraussetzungen

Keine

4.1.3.2 Effekte

Nach dem Aufruf dieser Funktion müss das applet neu personalisiert werden, d.h. auch PIN und PUK müssen neu gesetzt werden.

4.1.3.3 GET RANDOM

Diese Kommando-APDU wird genutzt um eine Zufallszahl mit der Länge n anzufordern.

Command		
CLA	‚00'	ISO Class unverschlüsselt
INS	‚84'	GET RANDOM
P1|P2	‚00 00'	
LC	--	N/A
DATA	--	N/A
LE	‚XX'	Länge der Zufallszahl in Bytes
Response		
DATA	variabel	Zufallszahl.
SW1SW2	'9000'	Erfolgreiche Kommandobearbeitung
	'6700'	Fehler hafte Länge der Kommandodaten
	'6A86'	Fehler, ungültige Parameter P1, P2

Tabelle 3: GET RANDOM Kommando APDU

4.1.3.4 Voraussetzungen

Keine

4.1.3.5 Effekte

Die Zufallszahl (Random Bytes) werden bis zur Verarbeitung flüchtig im Applet gespeichert. Bei wiederholtem GET RANDOM wird jeweils die zuletzt ausgegebene Zufallszahl gespeichert.

4.1.4 SET PIN (CHANGE REFERENCE DATA)

Diese Kommando-APDU wird genutzt um PIN und PUK zu vergeben.

Command		
CLA	‚00'	ISO Class unverschlüsselt
INS	‚84'	CHANGE REFERENCE DATA
P1	‚01'	
P2	‚01' | ‚02'	‚01' PIN ‚02' PUK
LC	‚06' | ‚0F'	6 byte für PIN 15 byte für PUK
DATA	‚xx…'	PIN/PUK
LE	N/A	Nicht vorhanden
Response		
SW1SW2	'9000'	Erfolgreiche Kommandobearbeitung
	'6700'	Fehler hafte Länge der Kommandodaten
	'6A86'	Fehler, ungültige Parameter P1, P2

Tabelle 4: CHANGE REFERENCE DATA Kommando APDU

4.1.4.1 Voraussetzungen

Keine

4.1.4.2 Format der PIN

Sinnvolle Werte für die PIN sind alle ASCII-Zeichen von 0x20 bis 0x7E (Zahlen, Buchstaben, Sonderzeichen die in 7 Bit darstellbar sind).

Das PIN-Format der neuen PIN wird im Applet nicht geprüft, es können also auch Werte außerhalb des empfohlenen Bereichs übergeben werden.

4.1.4.3 Effekte

Die PUK muss vor der PIN gesetzt werden. Ein Setzten der PUK nachdem die PIN vergeben wurde, ist nicht möglich (Fehlercode ’6A86’, weil P2 dann kein erlaubter Wert mehr ist).

4.1.5 VERIFY

Diese Kommando-APDU wird benutzt damit sich der Benutzer gegenüber dem Token mit der PIN authentisiert.

Command		
CLA	‚00’	ISO Class unverschlüsselt
INS	‚20’	VERIFY
P1|P2	‚00 01’	Applikations-PIN ‚00‘
LC	‚06’	Länge der Kommandodaten.
DATA	‚xx…‘	ASCII-PIN
LE	N/A	Nicht vorhanden
Response		
DATA	N/A	
SW1SW2	’9000’	Erfolgreiche Kommandobearbeitung
	’6700’	Fehler hafte Länge der Kommandodaten
	’6A86’	Fehler, ungültige Parameter P1, P2
	‘63CX’	Falsche PIN wurde übergeben. ‚X‘ zeigt die Anzahl der noch verfügbaren Versuche an.
	‘6983’ ‚6985‘	Keine weiteren Versuche, die PIN ist gesperrt. Fehler, PIN nicht freigeschaltet.

Tabelle 5: Verify Kommando APDU

4.1.5.1 Voraussetzungen

Keine

4.1.5.2 Effekte

- Ist die PIN noch nicht gesetzt, wird das Kommando mit Fehlercode ‚6985' beendet.
- Wird eine PIN mit falscher Länge eingegeben, wird 6700 zurückgegeben, der Fehlbedienungszähler bleibt unverändert.
- Nach erfolgreichem Überprüfen der eingegebenen PIN wird der Sicherheitszustand für die PIN auf „verified" gesetzt.
- Der Sicherheitszustand wird zurückgesetzt durch
  - Power Off
  - DESELECT oder RESELECT des Applets
  - RESET PIN Kommando

4.1.6 CHANGE PIN (CHANGE REFERENCE DATA)

Diese Kommando-APDU wird vom Applet zur Verfügung gestellt, um einen bereits vorhandenen Wert der PIN gegen einen neuen auszutauschen.

Command		
CLA	‚00'	ISO Class unverschlüsselt
INS	‚24'	CHANGE PIN
P1|P2	‚00 01'	Applikations-PIN ‚01'
LC	‚0C'	Länge der Kommandodaten.
DATA	‚xx…', ‚xx…'	Die aktuelle PIN gefolgt von der neuen PIN. Beide PIN haben eine Länge von 6 Zeichen.
LE	N/A	Nicht vorhanden
Response		
SW1SW2	'9000' '6700' '6A86'	Erfolgreiche Kommandobearbeitung Fehler hafte Länge der Kommandodaten Fehler, ungültige Parameter P1, P2

	’6A80’	Fehlerhafte Kommandodaten (PIN-Format).
	‘63CX’	Falsche PIN wurde übergeben. ‚X‘ zeigt die Anzahl der noch verfügbaren Versuche an.
	‘6983’	Keine weiteren Versuche, die PIN ist gesperrt.
	‚6985‘	Fehler, PIN nicht freigeschaltet.

Tabelle 6: Change PIN Kommando APDU

4.1.6.1 Voraussetzungen

Keine

4.1.6.2 Effekte

- Ist die PIN noch nicht gesetzt, wird das Kommando mit Fehlercode ‚6985‘ beendet.
- Wird eine falsche Länge eingegeben, wird 6700 zurückgegeben, der Fehlbedienungszähler bleibt unverändert.
- Nach erfolgreichem Überprüfen der eingegebenen PIN wird die neue PIN gesetzt.
- Der Fehlbedienungszähler der PIN wird zurückgesetzt.
- Der Sicherheitszustand für die PIN wird auf „nicht verifiziert“ gesetzt.

4.1.7 RESET PIN

Diese Kommando-APDU wird vom Applet zur Verfügung gestellt, um nach Authentisierung mit der PUK einen neuen Wert für die PIN festzulegen.

Command		
CLA	‚00’	ISO Class unverschlüsselt
INS	‚2C’	RESET RETRY COUNTER
P1|P2	‚00 01’	Applikations-PIN ‚00‘
LC	‚15’	Länge der Kommandodaten.
DATA	‚xx…‘, ‚xx…‘	Die PUK gefolgt von der neuen PIN.

LE	--	Nicht vorhanden
Response		
SW1SW2	’9000’	Erfolgreiche Kommandobearbeitung
	’6700’	Fehler hafte Länge der Kommandodaten
	’6A86’	Fehler, ungültige Parameter P1, P2
	’6A80’	Fehlerhafte Kommandodaten (PIN-Format).
	‘63CX’	Falsche PIN wurde übergeben. ‚X‘ zeigt die Anzahl der noch verfügbaren Versuche an.
	‘6983’	Keine weiteren Versuche, die PIN ist gesperrt.
	‘6986’	Kommando nicht erlaubt

Tabelle 7: RESET RETRY COUNTER für PIN Kommando APDU

4.1.7.1 Voraussetzungen

Keine

4.1.7.2 Effekte

- Ist die PIN oder PUK noch nicht gesetzt, wird das Kommando mit Fehlercode ‚6986‘ beendet.
- Wir eine falsche Länge eingegeben, wird 6700 zurückgegeben, der Fehlbedienungszähler bleibt unverändert.
- Nach erfolgreichem Überprüfen der eingegebenen PUK wird die neue PIN gesetzt.
- Der Fehlbedienungszähler von PIN und PUK wird zurückgesetzt.
- Der Sicherheitszustand für die PIN wird auf „nicht verifiziert“ gesetzt.

4.1.8 GET PIN/PUK STATUS

Diese Kommando-APDU wird benutzt um die Anzahl der Fehlversuche für eine PIN/PUK abzufragen.

Command		
CLA	‚00’	ISO Class unverschlüsselt

INS	‚20'	VERIFY
P1	‚00'	
P2	‚0[12]'	PIN ‚01' PUK ‚02'
LC	N/A	Nicht vorhanden
DATA	N/A	Nicht vorhanden
LE	N/A	Nicht vorhanden
Response		
SW1SW2	'9000'	Erfolgreiche Kommandobearbeitung, PIN ist authentisiert.
	'6700'	Fehler hafte Länge der Kommandodaten
	'6A86'	Fehler, ungültige Parameter P1, P2.
	‚63CX'	‚X' zeigt die Anzahl der noch verfügbaren Versuche an.
	‚6983'	Keine weiteren Versuche, die PIN ist gesperrt.

Tabelle 8 GET PIN STATUS Kommando APDU

4.1.8.1 Voraussetzungen

Keine

4.1.8.2 Kommandodaten

Keine

4.1.8.3 Effekte

Keine

4.1.9 DECRYPT

Diese Kommando-APDU wird benutzt um Daten mit dem Entschlüsselungsschlüssel zu entschlüsseln.

Command

CLA	‚80' ‚90'	Proprietary Class unverschlüsselt Proprietary Class unverschlüsselt mit command chaining
INS	‚2A'	PERFORM SECURITY OPERATION
P1	‚80'	Decipher mit SK
P2	‚86' ‚87'	RawRSA OAEP/SHA256 (nur für Performance Tests)
LC	Variabel	Chaining (255+1) notwendig
DATA	Variabel	
LE	‚00'	Alle Daten
Response		
DATA	Variabel	
SW1SW2	'9000' '6700' '6985' '6A86' '6A80' '6982'	Erfolgreiche Kommandobearbeitung Fehler hafte Länge der Kommandodaten Schlüsselobjekte nicht vorhanden oder nicht initialisiert Fehler, ungültige Parameter P1, P2 Fehlerhafte Kommandodaten. Security condition not satisfied

Tabelle 9: DECRYPT Kommando APDU

4.1.9.1 Voraussetzungen

PIN verifiziert, Entschlüsselungsschlüssel vorhanden

4.1.9.2 Kommandodaten

256 Byte RSA Kryptogramm (2048bit Schlüssel)

4.1.9.3 Antwortdaten

Entschlüsselte Daten (256 byte), enthält noch das Padding

4.1.9.4 Effekte

Keine

4.1.10 SIGN

Diese Kommando-APDU wird benutzt um Daten zu signieren.

Command		
CLA	‚80' ‚90'	Proprietary Class unverschlüsselt Proprietary Class unverschlüsselt mit command chaining
INS	‚2A'	PERFORM SECURITY OPERATION
P1	‚9E'	Sign mit privatem Schlüssel
P2	‚9A' ‚9B'	RawRSA mit Signaturschlüssel RawRSA mit Entschlüsslungsschlüssel
LC	Variabel	Chaining (255+1) notwendig
DATA	Variabel	
LE	‚00'	Alle Daten
Response		
DATA	Variabel	
SW1SW2	'9000' '6700' ‚6985' '6A86' '6A80' '6982'	Erfolgreiche Kommandobearbeitung Fehler hafte Länge der Kommandodaten Schlüsselobjekte nicht vorhanden oder nicht initialisiert Fehler, ungültige Parameter P1, P2 Fehlerhafte Kommandodaten. Security condition not satisfied

Tabelle 10: SIGN Kommando APDU

4.1.10.1 Voraussetzungen

PIN verifiziert, Signaturschlüssel vorhanden

4.1.10.2 Kommandodaten

Daten und Padding (256 byte)

4.1.10.3 Antwortdaten

256 Byte RSA Kryptogramm (2048bit Schlüssel)

4.1.10.4 Effekte

Keine

4.1.11 GENERATE KEYPAIR / EXPORT PUBLIC KEY APDU

Dieses APDU-Kommando schickt das Host-System an den USB-Token, um einen öffentlichen Schlüssel vom Token auszulesen.

Command		
CLA	‚80'	Proprietary Class unverschlüsselt
INS	'46'	GENERATE ASYMMETRIC KEY PAIR
P1	'42' ‚43'	Schlüssel generieren und Export des öffentlichen Schlüssels Export des öffentlichen Schlüssel, kein Generieren
P2	'00' '01'	Signaturschlüssel Verschlüsselungsschlüssel
LC	N/A	Nicht vorhanden.
DATA	N/A	Es werden keine Daten übergeben
LE	'00'	Rückgabe aller Daten
Response		
DATA	variabel	Der öffentliche Schlüssel (codiert nach ISO 7816-8) '7F49 …'
SW1SW2	'9000'	Erfolgreiche Kommandobearbeitung

	’6700’	Fehler hafte Länge der Kommandodaten
	’6A86’	Fehler, ungültige Parameter P1, P2
	’6A88’	Fehler, Schlüssel nicht vorhanden oder generiert. Unbekannte Schlüssel-ID.
	‘6A80’	Fehlerhafte Kommandodaten.
	‘6982’	Security condition not satisfied

Tabelle 11: GENERATE KEYPAIR / EXPORT PUBLIC KEY Kommando APDU

4.1.11.1 Voraussetzungen

PIN verifiziert. Bei P1 = ‚43’: Schlüssel vorhanden.

4.1.11.2 Antwortdaten

Der öffentliche Schlüssel wird in nicht komprimierter Form ausgegeben.

4.1.11.3 Effekte

Bei P1 = ‚42’ – permanentes Speichern des erzeugten Schlüssels

4.1.12 PUT DATA Kommando APDU

Diese Kommando-APDU wird genutzt um Daten permanent in das Applets zu schreiben.

Command		
CLA	‚80’ ‚90’	Proprietary Class unverschlüsselt Proprietary Class unverschlüsselt mit command chaining
INS	‚D6’	UPDATE BINARY
P1	‚00’	
P2	‚00’ ‚01’ ‚02’	Signaturzertifikat Verschlüsselungszertifikat Infodaten
LC	Variable	Chaining (255+1) möglich
DATA	Variable	Binärdaten (Byte Array)

LE	N/A	
Response		
SW1SW2	’9000’ ’6700’ ’6A82’ ‚6A84‘ ’6A86’ ‘6A80’ ‘6982’	Erfolgreiche Kommandobearbeitung Fehler hafte Länge der Kommandodaten Fehler, Instanz nicht vorhanden Fehler, kein Speicher mehr vorhanden Fehler, ungültige Parameter P1, P2 Fehlerhafte Kommandodaten. Security condition not satisfied

Tabelle 12: PUT DATA Kommando APDU

4.1.12.1 Voraussetzungen

PIN verifiziert

4.1.12.2 Komponenten

Folgende Komponenten können durch dieses Kommando ausgegeben werden

4.1.12.3 Effekte

Die neuen Daten werden im jeweiligen Container gespeichert.

4.1.13 GET DATA Kommando APDU

Diese Kommando-APDU wird genutzt, um Daten, die permanent im Applet gespeichert wurden, zu lesen.

Command		
CLA	‚80’ ‚90’	Proprietary Class unverschlüsselt Proprietary Class unverschlüsselt mit command chaining
INS	‚B0’	READ BINARY
P1	‚00’	
P2	‚00’ ‚01’	Signaturzertifikat Verschlüsselungszertifikat

	‚02'	Infodaten
LC	N/A	Nicht vorhanden
DATA	N/A	Nicht vorhanden
LE	‚00'	Alle Daten
Response		
SW1SW2	'9000' '61xx' '6700' '6A82' ‚6A84' '6A86' '6A80' '6982'	Erfolgreiche Kommandobearbeitung Weitere Daten (Länge ‚xx') verfügbar Fehler hafte Länge der Kommandodaten Fehler, Instanz nicht vorhanden Fehler, kein Speicher mehr vorhanden Fehler, ungültige Parameter P1, P2 Fehlerhafte Kommandodaten. Security condition not satisfied

Tabelle 13: GET DATA Kommando APDU

4.1.13.1 Voraussetzungen

Daten vorhanden

4.1.13.2 Komponenten

Folgende Komponenten können durch dieses Kommando ausgegeben werden:

- Signaturzertifikat
- Verschlüsselungszertifikat
- Infodaten

4.1.13.3 Effekte

Keine.

4.1.14 GET RESPONSE Kommando APDU

Diese Kommando-APDU wird genutzt, um von Daten, die eine größere Länge als 256 Bytes haben, die verbleibenden Bytes zu lesen.

Command		
CLA	‚00'	
INS	‚C0'	GET RESPONSE
P1	‚00'	
P2	‚00'	
LC	N/A	Nicht vorhanden
DATA	N/A	Nicht vorhanden
LE	‚xx'	Muss der Länge der verbeibenden Daten entsprechen
Response		
SW1SW2	'9000' '61xx'	Erfolgreiche Kommandobearbeitung Weitere Daten verfügbar, mit der Länge ‚xx'.

Tabelle 14: GET RESPONSE Kommando APDU

4.1.14.1 Voraussetzungen

Daten vorhanden

4.1.14.2 Komponenten

Folgende Komponenten können durch dieses Kommando ausgegeben werden:

- Signaturzertifikat
- Verschlüsselungszertifikat
- Infodaten

4.1.14.3 Effekte

Keine.

5. Anhang

5.1 Cardmanger Schlüssel

```
<GpSecureChannelKeySet Identifier="00" Version="00">
 <SecureChannelEncKey>
  <DesEdeKey>40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F</DesEdeKey>
 </SecureChannelEncKey>
 <SecureChannelMacKey>
  <DesEdeKey>40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F</DesEdeKey>
 </SecureChannelMacKey>
 <DataEncryptionKey>
  <DesEdeKey>40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F</DesEdeKey>
 </DataEncryptionKey>
</GpSecureChannelKeySet>
```

5.2 Beispiel APDUs

5.2.1 Select ISD

AID der ISD: "A000000003000000"

```
Select apdu: 00 a4 04 00 08 a0 00 00 00 03 00 00 00 00
Select apdu: CommmandAPDU: 14 bytes, nc=8, ne=256
Select status: 9000
Select answer: 6f 10 84 08 a0 00 00 00 03 00 00 00 a5 04 9f 65 01 ff
```

5.2.2 Select Applet

AID des Applets: " A00000006680013401 01"

```
Select apdu: 00 A4 04 00 0A A0 00 00 00 66 80 01 34 01 01
Select apdu: CommmandAPDU: 15 bytes, nc=9, ne=256
Select status: 9000
Select answer: 00 01
```

5.2.3 Reset Token (delete masterfile)

Löscht die PIN, PUK, Schlüssel und Zertifikate, setzt alle Fehlbedienungszähler zurück.

```
DeleteMF apdu: 00 e4 00 00 02 3f 00 00
DeleteMF apdu: CommmandAPDU: 8 bytes, nc=2, ne=256
DeleteMF status: 9000
```

5.2.4 Set PUK

PUK: „123456789012345“

```
SetPUK apdu: 00 24 01 02 0f 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35
00
SetPUK apdu: CommmandAPDU: 21 bytes, nc=15, ne=256
SetPUK status: 9000
```

5.2.5 Set PIN

PIN: „123456“

```
SetPIN apdu: 00 24 01 01 06 31 32 33 34 35 36 00
SetPIN apdu: CommmandAPDU: 12 bytes, nc=6, ne=256
SetPIN status: 9000
```

5.2.6 Verify PIN

PIN: „123456“

```
VerifyPIN apdu: 00 20 00 01 06 31 32 33 34 35 36 00
VerifyPIN apdu: CommmandAPDU: 12 bytes, nc=6, ne=256
VerifyPIN status: 9000
```

PIN: „“ (gibt den Wert des Fehlbedienungszählers aus – im Beispiel 3)

```
VerifyPIN apdu: 00 20 00 01 00
VerifyPIN apdu: CommmandAPDU: 5 bytes, nc=0, ne=256
VerifyPIN status: 63c3
```

5.2.7 Change PIN

Alte PIN: „123456“, neue PIN: „111111“

```
ChangePIN apdu: 00 24 00 01 0c 31 32 33 34 35 36 31 31 31 31 31 31 00
ChangePIN apdu: CommmandAPDU: 18 bytes, nc=12, ne=256
ChangePIN status: 9000
```

5.2.8 Reset PIN

PUK: „123456789012345“, neue PIN: „123456“

```
ResetPIN apdu: 00 2c 00 01 15 31 32 33 34 35 36 37 38 39 30 31 32 33 34
35 31 32 33 34 35 36 00
ResetPIN apdu: CommmandAPDU: 27 bytes, nc=21, ne=256
ResetPIN status: 9000
```

5.2.9 Get Random

Zufallswert anfordern – hier 8 Byte

```
GetRandom apdu: 00 84 00 00 08

GetRandom apdu: CommmandAPDU: 5 bytes, nc=0, ne=8

GetRandom status: 9000

GetRandom data: 4b 6d a4 ac 7d 24 48 bd
```

5.2.10 Generate SIGN Key

```
Key apdu: 80 46 42 00 00

Key apdu: CommmandAPDU: 5 bytes, nc=0, ne=256

Key status: 9000

Key data:
7F:49:82:01:09:81:82:01:00:9F:CE:CC:C9:43:F2:88:8F:E3:B5:A4:53:1F:C5:5B:E
D:AB:E7:81:FC:BF:3B:C8:4C:61:56:18:BA:07:9E:75:6A:D5:F4:D2:5D:CF:D8:E3:5B
:AD:E0:AD:39:67:7F:5D:36:50:25:4F:47:A2:90:0B:0B:E4:1C:57:86:F9:13:5E:E7:
B6:03:9C:67:1A:7D:76:05:26:01:44:9D:6E:27:A8:11:64:1C:BA:D9:37:86:97:96:2
3:9C:42:8C:57:61:89:D5:B5:00:F9:AD:DE:1A:94:84:5A:36:0D:B1:30:66:DC:D6:46
:7F:06:E8:C4:A5:C7:DE:48:83:25:D0:ED:CB:53:0E:21:34:67:B3:EB:34:70:0B:CF:
73:DF:A3:65:C3:3B:3F:2C:5A:74:D6:64:D9:EA:CE:61:0B:78:86:03:34:6A:B7:5F:D
C:EC:54:FC:DE:3A:B6:CC:E5:BA:11:24:47:2F:F8:49:29:17:2C:1A:8D:15:EE:ED:19
:3F:F7:89:44:7F:0A:01:80:7B:3A:64:8B:1A:7D:3B:EE:7E:EA:04:C9:31:11:66:B9:
2B:34:1C:63:62:A1:C4:43:57:46:01:09:F0:82:88:30:C6:0C:2A:E3:2D:D7:AE:E9:3
D:D3:59:0F:89:E5:06:88:1B:29:89:F9:19:F3:C3:92:9E:F3:42:B1:D5:CD:82:03:01
:00:01
```

5.2.11 Get SIGN Cert

```
Read cert apdu: 80 b0 00 00 00

Read cert apdu: CommmandAPDU: 5 bytes, nc=0, ne=256

Read cert status: 9000

Read cert data:
30:82:02:CC:30:82:01:B4:A0:03:02:01:02:02:06:01:5A:AD:32:6E:F0:30:0D:06:0
9:2A:86:48:86:F7:0D:01:01:0A:05:00:30:19:31:17:30:15:06:03:55:04:03:13:0E
:53:65:6C:66:53:69:67:6E:65:64:43:65:72:74:30:1E:17:0D:31:37:30:33:30:38:
30:39:31:34:30:37:5A:17:0D:32:32:30:33:30:37:30:39:31:34:30:37:5A:30:19:3
1:17:30:15:06:03:55:04:03:13:0E:53:65:6C:66:53:69:67:6E:65:64:43:65:72:74
:30:82:01:22:30:0D:06:09:2A:86:48:86:F7:0D:01:01:01:05:00:03:82:01:0F:00:
30:82:01:0A:02:82:01:01:00:9F:CE:CC:C9:43:F2:88:8F:E3:B5:A4:53:1F:C5:5B:E
D:AB:E7:81:FC:BF:3B:C8:4C:61:56:18:BA:07:9E:75:6A:D5:F4:D2:5D:CF:D8:E3:5B
:AD:E0:AD:39:67:7F:5D:36:50:25:4F:47:A2:90:0B:0B:E4:1C:57:86:F9:13:5E:E7:
B6:03:9C:67:1A:7D:76:05:26:01:44:9D:6E:27:A8:11:64:1C:BA:D9:37:86:97:96:2
3:9C:42:8C:57:61:89:D5:B5:00:F9:AD:DE:1A:94:84:5A:36:0D:B1:30:66:DC:D6:46
:7F:06:E8:C4:A5:C7:DE:48:83:25:D0:ED:CB:53:0E:21:34:67:B3:EB:34:70:0B:CF:
73:DF:A3:65:C3:3B:3F:2C:5A:74:D6:64:D9:EA:CE:61:0B:78:86:03:34:6A:B7:5F:D
C:EC:54:FC:DE:3A:B6:CC:E5:BA:11:24:47:2F:F8:49:29:17:2C:1A:8D:15:EE:ED:19
:3F:F7:89:44:7F:0A:01:80:7B:3A:64:8B:1A:7D:3B:EE:7E:EA:04:C9:31:11:66:B9:
2B:34:1C:63:62:A1:C4:43:57:46:01:09:F0:82:88:30:C6:0C:2A:E3:2D:D7:AE:E9:3
D:D3:59:0F:89:E5:06:88:1B:29:89:F9:19:F3:C3:92:9E:F3:42:B1:D5:CD:02:03:01
:00:01:A3:1A:30:18:30:0B:06:03:55:1D:0F:04:04:03:02:05:A0:30:09:06:03:55:
1D:13:04:02:30:00:30:0D:06:09:2A:86:48:86:F7:0D:01:01:0A:05:00:03:82:01:0
1:00:79:87:6B:0C:83:60:FB:81:31:22:D9:5C:F8:FE:B4:3D:0E:6F:38:29:B4:9D:C0
:50:F1:B3:54:1E:DA:84:2E:B0:2C:DC:41:3B:17:99:23:2D:2A:FA:DA:1E:C6:BA:7C:
```

```
05:91:A5:7E:CF:67:33:58:98:F2:C9:A8:9C:F1:EE:F0:CF:36:A3:39:20:E5:F4:54:9
F:69:4F:00:1B:92:3D:B8:7A:01:B8:C7:47:03:7A:2B:7F:85:B3:22:00:C2:F4:D9:6F
:54:45:14:9D:29:C2:95:DA:22:EB:6A:4A:7A:DD:21:59:7E:1A:68:E0:31:68:25:F4:
F8:DD:76:BA:36:01:72:AA:CE:E6:EB:9A:2F:7A:B5:CC:33:8C:3C:B7:7E:08:70:0F:9
3:82:3D:DE:08:25:6B:B7:55:E4:41:0E:D7:0B:4B:0B:34:18:17:57:3D:1A:EB:C4:C4
:A6:1D:88:40:26:B3:BD:FA:0B:66:8E:C3:EA:34:A3:50:88:68:69:7B:B9:74:97:E1:
64:38:FD:5F:2E:CE:AD:53:46:34:27:6B:F0:00:48:2A:32:15:18:33:EF:A8:C1:17:5
F:78:CE:72:BE:8C:32:AB:B4:04:93:88:36:A5:88:F6:7B:A0:A4:01:5F:EA:0E:F4:EA
:29:46:06:74:0F:93:59:3E:95:49:86:8F:DE:FA
```

5.2.12 Generate ENCR Key

```
Key apdu: 80 46 42 01 00

Key apdu: CommmandAPDU: 5 bytes, nc=0, ne=256

Key status: 9000

Key data:
7F:49:82:01:09:81:82:01:00:C7:58:3A:2A:C6:E4:44:CD:9A:71:38:E9:E2:24:B3:6
5:79:24:31:D7:19:61:31:C9:3A:E8:89:85:FF:AA:DF:C0:91:85:DD:36:0B:A8:71:ED
:85:21:F4:0F:AF:8A:EB:44:CE:48:5D:C0:BC:0A:38:A0:30:E9:E4:BE:B5:D3:27:21:
FA:38:3B:7E:DF:9F:13:4C:C9:21:25:8F:A6:8F:E9:81:3C:14:0F:8F:47:60:75:73:9
8:18:73:2D:E1:9E:79:0F:A7:9A:DB:36:7F:E6:7A:29:B9:19:58:1A:14:40:AD:14:5D
:D8:E1:5E:19:15:75:F9:DC:AB:0D:D6:05:42:4F:B9:C6:74:03:ED:1A:AB:F7:5D:9A:
C6:37:80:FF:96:20:8D:28:D9:A9:63:99:46:3E:89:D1:28:15:01:4B:2A:7A:74:DC:1
C:0E:C4:E0:BD:E6:D7:5C:C7:19:70:2B:D6:0E:CB:A0:75:EB:D0:AB:5F:C5:00:9B:F2
:B5:2B:FF:93:80:D6:18:AA:C6:22:EF:52:EC:44:FD:EB:D3:72:EE:1E:5D:4D:D7:57:
CD:95:D7:50:22:33:38:E0:D6:CA:E6:BD:19:26:32:00:9D:EB:22:21:54:D4:4E:30:F
A:97:5C:91:C4:53:9B:23:AF:07:0C:FB:91:7C:16:CD:F2:C7:2C:33:0F:69:82:03:01
:00:01
```

5.2.13 Get ENCR Key

```
Key apdu: 80 46 43 01 00

Key apdu: CommmandAPDU: 5 bytes, nc=0, ne=256

Key status: 9000

Key data:
7F:49:82:01:09:81:82:01:00:C7:58:3A:2A:C6:E4:44:CD:9A:71:38:E9:E2:24:B3:6
5:79:24:31:D7:19:61:31:C9:3A:E8:89:85:FF:AA:DF:C0:91:85:DD:36:0B:A8:71:ED
:85:21:F4:0F:AF:8A:EB:44:CE:48:5D:C0:BC:0A:38:A0:30:E9:E4:BE:B5:D3:27:21:
FA:38:3B:7E:DF:9F:13:4C:C9:21:25:8F:A6:8F:E9:81:3C:14:0F:8F:47:60:75:73:9
8:18:73:2D:E1:9E:79:0F:A7:9A:DB:36:7F:E6:7A:29:B9:19:58:1A:14:40:AD:14:5D
:D8:E1:5E:19:15:75:F9:DC:AB:0D:D6:05:42:4F:B9:C6:74:03:ED:1A:AB:F7:5D:9A:
C6:37:80:FF:96:20:8D:28:D9:A9:63:99:46:3E:89:D1:28:15:01:4B:2A:7A:74:DC:1
C:0E:C4:E0:BD:E6:D7:5C:C7:19:70:2B:D6:0E:CB:A0:75:EB:D0:AB:5F:C5:00:9B:F2
:B5:2B:FF:93:80:D6:18:AA:C6:22:EF:52:EC:44:FD:EB:D3:72:EE:1E:5D:4D:D7:57:
CD:95:D7:50:22:33:38:E0:D6:CA:E6:BD:19:26:32:00:9D:EB:22:21:54:D4:4E:30:F
A:97:5C:91:C4:53:9B:23:AF:07:0C:FB:91:7C:16:CD:F2:C7:2C:33:0F:69:82:03:01
:00:01
```

5.2.14 Write Certificate ENCR

Es wird Command Chaining verwendet.

APDU 1:

```
Write Certificate apdu: 90 d6 00 01 ff 30 82 02 cc 30 82 01 b4 a0 03 02
01 02 02 06 01 5a ad 32 70 c9 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0a 05
```

```
00 30 19 31 17 30 15 06 03 55 04 03 13 0e 53 65 6c 66 53 69 67 6e 65 64
43 65 72 74 30 1e 17 0d 31 37 30 33 30 38 30 39 31 34 30 38 5a 17 0d 32
32 30 33 30 37 30 39 31 34 30 38 5a 30 19 31 17 30 15 06 03 55 04 03 13
0e 53 65 6c 66 53 69 67 6e 65 64 43 65 72 74 30 82 01 22 30 0d 06 09 2a
86 48 86 f7 0d 01 01 01 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01 00
c7 58 3a 2a c6 e4 44 cd 9a 71 38 e9 e2 24 b3 65 79 24 31 d7 19 61 31 c9
3a e8 89 85 ff aa df c0 91 85 dd 36 0b a8 71 ed 85 21 f4 0f af 8a eb 44
ce 48 5d c0 bc 0a 38 a0 30 e9 e4 be b5 d3 27 21 fa 38 3b 7e df 9f 13 4c
c9 21 25 8f a6 8f e9 81 3c 14 0f 8f 47 60 75 73 98 18 73 2d e1 9e 79 0f
a7 9a db 36 00

Write Certificate apdu: CommmandAPDU: 261 bytes, nc=255, ne=256

Write Certificate status: 9000
```

APDU 2:

```
Write Certificate apdu: 90 d6 00 01 ff 7f e6 7a 29 b9 19 58 1a 14 40 ad
14 5d d8 e1 5e 19 15 75 f9 dc ab 0d d6 05 42 4f b9 c6 74 03 ed 1a ab f7
5d 9a c6 37 80 ff 96 20 8d 28 d9 a9 63 99 46 3e 89 d1 28 15 01 4b 2a 7a
74 dc 1c 0e c4 e0 bd e6 d7 5c c7 19 70 2b d6 0e cb a0 75 eb d0 ab 5f c5
00 9b f2 b5 2b ff 93 80 d6 18 aa c6 22 ef 52 ec 44 fd eb d3 72 ee 1e 5d
4d d7 57 cd 95 d7 50 22 33 38 e0 d6 ca e6 bd 19 26 32 00 9d eb 22 21 54
d4 4e 30 fa 97 5c 91 c4 53 9b 23 af 07 0c fb 91 7c 16 cd f2 c7 2c 33 0f
69 02 03 01 00 01 a3 1a 30 18 30 0b 06 03 55 1d 0f 04 04 03 02 05 a0 30
09 06 03 55 1d 13 04 02 30 00 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0a 05
00 03 82 01 01 00 1d 1e ba 5b eb 6d ce 2b 93 e1 7e 78 f3 9e 28 f4 d5 ca
ef bb 2a 18 5d db 6b 5a d5 0c a6 cd 74 20 d3 10 31 0f b6 44 9a 91 8f 22
2d d3 4c a7 00

Write Certificate apdu: CommmandAPDU: 261 bytes, nc=255, ne=256

Write Certificate status: 9000
```

APDU 3:

```
Write Certificate apdu: 80 d6 00 01 d2 ed 01 f5 7d fb 8b 2a c3 3a 7c 3c
86 00 65 30 f0 7b de df 63 ce cd bd 4a 5a d7 14 22 6a c8 d9 09 2c cb 5b
c4 36 85 e4 b0 8a 38 46 93 c0 65 e5 6e 43 50 f7 4d e4 62 90 74 c8 0d 06
a3 fb b2 51 3c d4 5f 2a 2a 86 3f 00 11 d0 d2 45 6c 75 f6 63 98 0e 89 7c
5d ad bd f2 61 e0 60 0d e2 91 fb 63 08 f8 34 f2 05 63 91 15 91 79 8d 96
40 32 0a 34 36 26 87 e4 cf e5 71 e6 51 81 9b 2e fb 39 ab 03 a4 02 a9 73
e3 ed 81 bf 07 e5 e2 18 7b de 72 d5 b3 a7 07 63 a9 e3 f7 47 cc 9e da 9c
9b f6 af a0 dc fa 10 cc 6b 09 34 c4 55 24 7c 0a 41 c4 14 e1 3c 6e f5 59
e2 54 3c 86 e0 41 98 30 66 06 a5 d2 84 f1 a2 af 17 91 93 94 7b b7 47 b7
70 b9 b4 84 bf 26 bf 00

Write Certificate apdu: CommmandAPDU: 216 bytes, nc=210, ne=256

Write Certificate status: 9000
```

5.2.15 Write Certificate SIGN

Es wird Command Chaining verwendet.

APDU 1:

```
Write Certificate apdu: 90 d6 00 00 ff 30 82 02 cc 30 82 01 b4 a0 03 02
01 02 02 06 01 5a ad 32 6e f0 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0a 05
00 30 19 31 17 30 15 06 03 55 04 03 13 0e 53 65 6c 66 53 69 67 6e 65 64
43 65 72 74 30 1e 17 0d 31 37 30 33 30 38 30 39 31 34 30 37 5a 17 0d 32
32 30 33 30 37 30 39 31 34 30 37 5a 30 19 31 17 30 15 06 03 55 04 03 13
```

```
0e 53 65 6c 66 53 69 67 6e 65 64 43 65 72 74 30 82 01 22 30 0d 06 09 2a
86 48 86 f7 0d 01 01 01 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01 00
9f ce cc c9 43 f2 88 8f e3 b5 a4 53 1f c5 5b ed ab e7 81 fc bf 3b c8 4c
61 56 18 ba 07 9e 75 6a d5 f4 d2 5d cf d8 e3 5b ad e0 ad 39 67 7f 5d 36
50 25 4f 47 a2 90 0b 0b e4 1c 57 86 f9 13 5e e7 b6 03 9c 67 1a 7d 76 05
26 01 44 9d 6e 27 a8 11 64 1c ba d9 37 86 97 96 23 9c 42 8c 57 61 89 d5
b5 00 f9 ad 00

Write Certificate apdu: CommmandAPDU: 261 bytes, nc=255, ne=256

Write Certificate status: 9000
```

APDU 2:

```
Write Certificate apdu: 90 d6 00 00 ff de 1a 94 84 5a 36 0d b1 30 66 dc
d6 46 7f 06 e8 c4 a5 c7 de 48 83 25 d0 ed cb 53 0e 21 34 67 b3 eb 34 70
0b cf 73 df a3 65 c3 3b 3f 2c 5a 74 d6 64 d9 ea ce 61 0b 78 86 03 34 6a
b7 5f dc ec 54 fc de 3a b6 cc e5 ba 11 24 47 2f f8 49 29 17 2c 1a 8d 15
ee ed 19 3f f7 89 44 7f 0a 01 80 7b 3a 64 8b 1a 7d 3b ee 7e ea 04 c9 31
11 66 b9 2b 34 1c 63 62 a1 c4 43 57 46 01 09 f0 82 88 30 c6 0c 2a e3 2d
d7 ae e9 3d d3 59 0f 89 e5 06 88 1b 29 89 f9 19 f3 c3 92 9e f3 42 b1 d5
cd 02 03 01 00 01 a3 1a 30 18 30 0b 06 03 55 1d 0f 04 04 03 02 05 a0 30
09 06 03 55 1d 13 04 02 30 00 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0a 05
00 03 82 01 01 00 79 87 6b 0c 83 60 fb 81 31 22 d9 5c f8 fe b4 3d 0e 6f
38 29 b4 9d c0 50 f1 b3 54 1e da 84 2e b0 2c dc 41 3b 17 99 23 2d 2a fa
da 1e c6 ba 00

Write Certificate apdu: CommmandAPDU: 261 bytes, nc=255, ne=256

Write Certificate status: 9000
```

APDU 3:

```
Write Certificate apdu: 80 d6 00 00 d2 7c 05 91 a5 7e cf 67 33 58 98 f2
c9 a8 9c f1 ee f0 cf 36 a3 39 20 e5 f4 54 9f 69 4f 00 1b 92 3d b8 7a 01
b8 c7 47 03 7a 2b 7f 85 b3 22 00 c2 f4 d9 6f 54 45 14 9d 29 c2 95 da 22
eb 6a 4a 7a dd 21 59 7e 1a 68 e0 31 68 25 f4 f8 dd 76 ba 36 01 72 aa ce
e6 eb 9a 2f 7a b5 cc 33 8c 3c b7 7e 08 70 0f 93 82 3d de 08 25 6b b7 55
e4 41 0e d7 0b 4b 0b 34 18 17 57 3d 1a eb c4 c4 a6 1d 88 40 26 b3 bd fa
0b 66 8e c3 ea 34 a3 50 88 68 69 7b b9 74 97 e1 64 38 fd 5f 2e ce ad 53
46 34 27 6b f0 00 48 2a 32 15 18 33 ef a8 c1 17 5f 78 ce 72 be 8c 32 ab
b4 04 93 88 36 a5 88 f6 7b a0 a4 01 5f ea 0e f4 ea 29 46 06 74 0f 93 59
3e 95 49 86 8f de fa 00

Write Certificate apdu: CommmandAPDU: 216 bytes, nc=210, ne=256

Write Certificate status: 9000
```

5.2.16 Sign

Es wird RSA RAW gerechnet. Die Daten enthalten das Padding und sind 256 Byte lang (2048 Bit Schlüssel). Der Rückgabewert sind ebenfalls 256 Byte

Es wird Command Chaining verwendet.

APDU 1:

```
Sign apdu: 90 2a 9e 9a ff 2f b7 19 2a 12 8f c3 91 24 33 5d c1 21 6f ff 10
56 3c d5 cb 4a 4d 56 9b b5 e1 a6 74 d8 87 79 bf cc 2e d4 85 2e e3 ea 36
9d f6 1f 33 59 d0 c3 4a 30 e8 1d 6e 2c 10 9d 25 5f 46 5d c0 4b aa 1b 32
```

```
d8 50 fa a4 9c 9d e7 e6 71 4e 59 06 6f 56 50 59 7b c3 bc f0 71 2e c0 74
58 42 c5 38 e6 22 21 3b af be cc 38 6c 9c 33 18 11 bc 31 a8 63 60 4c c1
de dd c1 ad c4 07 9b 11 fd 31 07 00 d2 13 f0 84 f6 e7 68 f0 3b c5 b4 4c
23 94 f2 fa 5e 5f dc ed 9b 66 53 73 40 0f f3 40 46 c6 a9 b3 d4 09 9b e0
46 72 d5 d3 0a b8 aa 65 ea 5e e0 13 06 4e 46 7e 94 10 0d 3c 34 2c fa 4c
83 50 69 4e f3 b6 87 81 e1 68 07 6b c3 81 b5 3b 4d 3e 1d 97 44 89 99 5d
76 30 97 17 8b 6d 74 cb 42 96 7e 7c 86 d4 7c 1e 81 16 30 b2 a9 d5 15 18
1b 45 2c e2 0a 41 d8 4f 8e dc c8 19 8f e1 84 17 87 a7 91 cd 69 91 d4 00

Sign apdu: CommmandAPDU: 261 bytes, nc=255, ne=256

Sign status: 9000
```

APDU 2:

```
Sign apdu: 80 2a 9e 9a 01 bc 00

Sign apdu: CommmandAPDU: 7 bytes, nc=1, ne=256

Signn status: 9000

Sign data: 79 87 6b 0c 83 60 fb 81 31 22 d9 5c f8 fe b4 3d 0e 6f 38 29 b4
9d c0 50 f1 b3 54 1e da 84 2e b0 2c dc 41 3b 17 99 23 2d 2a fa da 1e c6
ba 7c 05 91 a5 7e cf 67 33 58 98 f2 c9 a8 9c f1 ee f0 cf 36 a3 39 20 e5
f4 54 9f 69 4f 00 1b 92 3d b8 7a 01 b8 c7 47 03 7a 2b 7f 85 b3 22 00 c2
f4 d9 6f 54 45 14 9d 29 c2 95 da 22 eb 6a 4a 7a dd 21 59 7e 1a 68 e0 31
68 25 f4 f8 dd 76 ba 36 01 72 aa ce e6 eb 9a 2f 7a b5 cc 33 8c 3c b7 7e
08 70 0f 93 82 3d de 08 25 6b b7 55 e4 41 0e d7 0b 4b 0b 34 18 17 57 3d
1a eb c4 c4 a6 1d 88 40 26 b3 bd fa 0b 66 8e c3 ea 34 a3 50 88 68 69 7b
b9 74 97 e1 64 38 fd 5f 2e ce ad 53 46 34 27 6b f0 00 48 2a 32 15 18 33
ef a8 c1 17 5f 78 ce 72 be 8c 32 ab b4 04 93 88 36 a5 88 f6 7b a0 a4 01
5f ea 0e f4 ea 29 46 06 74 0f 93 59 3e 95 49 86 8f de fa
```

5.2.17 Decrypt

Es wird RSA RAW gerechnet. Die Daten sind 256 Byte lang (2048 Bit Schlüssel). Der Rückgabewert sind ebenfalls 256 Byte und enthalten die entschlüsselten Daten und das Padding.

Es wird Command Chaining verwendet.

APDU 1:

```
Decrypt apdu: 90 2a 80 86 ff 11 84 d7 48 8b ca ad ab 46 f1 00 9b b4 d1 90
c7 23 8e 9e 27 38 f2 0e 31 fe 68 b8 a7 80 59 04 b5 ab 7e a2 a9 ea 06 6f
99 73 90 bb b2 97 4a c8 4f a6 f0 10 5c bc 76 84 89 a2 fd bf 9d 5f 80 18
0f c2 d4 37 d5 be fa f7 1d 00 f5 96 1f e0 78 bb 3b af b6 93 b0 20 31 98
b4 8f 16 e9 3a 6e 34 7a 7c 91 2c 9a 03 9e 27 a3 a2 5e 43 62 02 de bd a1
2e bd 88 d0 bf 35 c1 ae 17 58 59 7d 07 cb 46 b6 09 6c f8 62 00 33 9d b8
04 83 47 0a 8d 82 cc 71 2c 8e 69 38 b8 11 ce 24 e2 09 f3 ce 1d 5d c7 32
c2 73 62 e1 d9 cf 27 72 0b 34 7c 81 ee 8f e5 42 e4 1d 65 ca d0 ad f6 26
c4 35 f1 44 a9 fa 00 8c 82 ef 8f 55 f3 43 bd d1 eb 40 f3 8f 84 25 06 0c
4a f0 2e 8b 63 35 26 bd a8 ed 3e 38 8b a0 6b 7b 40 b5 cd 26 6f 45 bf f8
b4 91 a9 2c 3f ea 7e d3 d0 36 1a 0a 3c 08 ea c8 af 2f 25 c7 84 b8 00 56
00

Decrypt apdu: CommmandAPDU: 261 bytes, nc=255, ne=256

Decrypt status: 9000
```

APDU 2:

```
Decrypt apdu: 80 2a 80 86 01 4f 00

Decrypt apdu: CommmandAPDU: 7 bytes, nc=1, ne=256

Decrypt status: 9000

Decrypt data: 00 e9 3e 6a 8a bf 62 89 c0 61 0b b0 22 3e 33 28 cd 08 46 3b
42 1f 75 96 c2 34 08 5e 8a 56 fc 8c 92 ad 88 6a 5a ab 01 10 5b 1a b8 b6
26 62 75 3c 91 d9 9d 02 91 9b 78 db a9 1a 2e b1 f0 65 c6 e9 9a 25 a7 fd
60 96 eb 50 4c cb a2 ce a5 4b 7a 00 7f 38 f6 b7 04 db 59 8a 50 eb 96 7e
a0 1a 51 a2 cb 49 db e5 76 9b f1 fa 91 80 6a bd be f2 b9 76 64 20 24 1f
28 cb c2 5c f8 bf 85 c6 dc 54 51 53 d1 f4 80 b0 5a 87 3a d9 6f 88 e3 d8
29 18 41 18 3f 72 ba 2f 27 7d 11 9b 47 b1 34 8e fc 2d be f6 9b 26 30 df
96 d6 7b aa 4d 53 bc 6c 37 f5 65 1d f1 05 23 6b 9b ba 32 35 a9 c8 cc 95
22 0a e8 15 7d ae 1b 96 3f 22 f4 49 a1 7e d6 6d 3d cd 4d 7e a7 55 fa f0
76 3b 73 19 11 e5 ce ab 23 2b 4d 33 d9 17 b0 a1 fd 07 e3 6c 3f 99 8d b4
32 10 a7 e1 b2 97 66 45 b2 98 e8 dc 5f 4e ad e2 9d 87 3e 1d
```