Smartcards

Michael Dupre
TroLUG
06.06.2019

Einführung

- Erstes Patent 1969 (Jürgen Dethloff)
  - 1977: Patent der Chipkarte
- Entwicklung:
  - Plastikkarte mit Embossing (erste Kreditkarten)
  - Magnetstreifen
  - Mit Speicher und Prozessor (ab hier: „smart“)
  - Mit Prozessor und Crypto-Coprozessor
  - Kontaktlos (NFC)
- Zur Zeit: Integration in Module/Chips

Einführung (2)

- Anwendungen
  - Telefonkarten (80-er)
  - Payment, e.g. Kreditkarten
  - Mobilfunkkarten (90-er)
  - Öffentlicher Nahverkehr, Maut, ... (NFC)
  - Krankenkassenkarten
  - Identitätskarten (Führerschein, Personalausweis)
  - Zugangskarten (Gebäude, PC, …)
  - ...

Einführung (3)

- Smart Card
  - Speicher, Prozessor, OS mit Kryptofunktionen
    - Authorisierung (PIN Prüfung)
    - Authentikation (Keys)
    - Integrität (MAC)
    - Non-Repudiation (Signatur)

Standardisierung

- Standardisierung als Voraussetzung für Erfolg
  - Maße
  - Kontakte (Lage, Bedeutung)
  - Elektr. Eigenschaften
  - Protokoll (T=0, T=1)
  - Anwendungen (Funktionen, Datenfelder)
  - JAVA-Card API

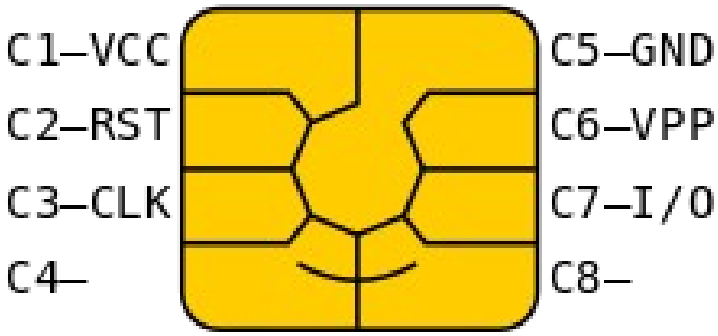
Standardisierung (2)

- Allgemein: ISO 7816-x
  - 7816-3: Protokolle T=0, T=1, APDUs
  - 7814-4: Basic commands (select, read binary, ...)
- GSM (https://www.etsi.org/standards)
  - e.g. 102220, …, 102226 und weitere
- UMTS (https://www.3gpp.org/specifications)
  - e.g. 31102, 31111, ...

Standardisierung (3)

- Kontakte
  - VCC (Voltage Common Connection)

    Power (5, 3, 1.8, 1.2)
  - RST (Reset)
  - CLK (external clock signal)
  - GND: Ground
  - VPP: Programming Voltage Input
  - I/O: half-duplex Input/Output
  - C4, C8: RFU für USB und andere Protokolle



Kartenleser

- Anbindung an PC: USB, Seriell, PCMCIA
  - Senden von APDUs
- Anbindung an Karte: immer seriell
  - T=0, T=1
- mit/ohne Display, Tastatur
  - Verschiedene Sicherheitsklassen
  - Zertifizierung möglich (nur mit eigener Tastatur!)

Frameworks

- Lange Zeit waren Kartenlesegeräte nur über proprietäre Schnittstellen ansprechbar.
- Frameworks
  - CT-API (Card Terminal Application Programming Interface)
  - PC/SC-Standard
  - Secoder-Standard

Frameworks (2)

- CT-API (Card Terminal Application Programming Interface)
  - Deutscher Ansatz für eine erste standardisierte Anwendungsschnittstelle (API).
  - Hat sich international nicht durchgesetzt.
- PC/SC-Standard
  - Internationaler Ansatz
  - Wurde von der PC/SC Workgroup erarbeitet
  - Verfügbar für Linux, macOS und Windows.
  - Der Hersteller muss einen Gerätetreiber bereitstellen.
  - Das OS stellt eine einheitliche API bereit. Funktionen beginnen mit Präfix „SCard“.

Frameworks (3)

- Secoder
  - Vom deutschen Zentralen Kreditausschuss (ZKA) entwickelt
  - Erlaubt das Laden von Geldkarten via Internet
  - Er verfügt über eine eingebaute Firewall
  - Benötigt Lesegerät mit Tastatur und Display
  - Erweiterung für HBCI

PC/SC

- API: https://pcsclite.apdu.fr/api/group__API.html
- Docs: https://docs.rs/pcsc/2.1.1/pcsc
- Implementierung
  - Windows: WinSCard.dll
  - Linux: pcsclite
- Linux-Pakete
  - libpcsclite1, libpcsclite-dev
  - Treiber: libccid, libacscid1, ...

PC/SC (2)

- Wrapper: Einbinden der C-Library in andere Programmiersprachen, falls nicht vorhanden, e.g.
  - go: github.com/ebfe/scard
- Minimum:
  - SCardEstablishContext(), SCardReleaseContext()
  - SCardListReaders()
  - SCardConnect(), SCardDisconnect()
  - SCardStatus()
  - SCardTransmit()

PC/SC (3)

- Prinzipieller Ablauf
  - context = scard.EstablishContext()
  - readerlist = context.ListReaders()
  - reader = readerlist[0]
  - card = context.Connect (reader, …)
  - … // set APDU
  - card.Transmit (apdu)
  - card.Disconnect (…)
  - context.Release()

APDU

- Application Protocol Data Unit
- 2 Typen:
  - Command-APDU
  - Response-APDU

Command-APDU

Aufbau: CLA, INS, P1, P2, Lc, <Data> [,Le]

- CLA (1): Class Byte
- INS (1): Instruction Byte
- P1/P2 (2): Parameters for the Command
- Lc (0|1|3): Length of command (max. 65 535)
  GSM/UMTS: Lc (1), max. 255 Byte
- <Data>: Lc bytes
- Le (0|1): Length Expected

Response-APDU

Aufbau: <>Resp-Data>, SW1, SW2

- <Resp-Data>: Response Daten
- SW1 (1): Status Word 1
- SW2 (1): Status Word 2

Example Status Words:

- 9000: ok
- 9Fxx: Es können xx Byte Antwortdaten mit GetResponse geholt werden
- 6310: More Data available

Class-Byte (ETSI, TS 102 221)

Table 10.3: Coding of class byte for standard logical channels

b8	b7	b6	b5	b4	b3	b2	b1	Value	Meaning
0	0	0	0	-	-	-	-	'0X'	The coding is according to the first interindustry values of CLA byte defined in ISO/IEC 7816-4 [12]
1	0	1	0	-	-	-	-	'AX'	Coded as for '0X' unless stated otherwise
1	0	0	0	-	-	-	-	'8X'	Structured as for '0X', coding and meaning is defined in the present document
-	-	-	-	X	X	-	-	-	Secure Messaging indication (see table 10.4)
-	-	-	-	-	-	X	X	-	Logical channel number from 0 to 3 (see clause 10.3)

Table 10.4: Coding of Secure Messaging Indication for standard logical channels

b4	b3	Meaning
0	0	No SM used between terminal and card
0	1	Proprietary SM format
1	x	Secure messaging according to ISO/IEC 7816-4 [12] used
1	0	Command header not authenticated
1	1	Command header authenticated

...und es gibt noch mehr …
Meistens (ohne SM und Log. Ch): 80 (2G), 00 (3G, ISO), 80 (GP)

Instruction-Byte (ETSI, TS 102 221)

Table 10.5: Coding of Instruction Byte of the Commands for a telecom application

COMMAND	CLA	INS
Command APDUs		
SELECT FILE	'0X' or '4X' or '6X'	'A4'
STATUS	'8X' or 'CX' or 'EX'	'F2'
READ BINARY	'0X' or '4X' or '6X'	'B0'
UPDATE BINARY	'0X' or '4X' or '6X'	'D6'
READ RECORD	'0X' or '4X' or '6X'	'B2'
UPDATE RECORD	'0X' or '4X' or '6X'	'DC'
SEARCH RECORD	'0X' or '4X' or '6X'	'A2'
INCREASE	'8X' or 'CX' or 'EX'	'32'
RETRIEVE DATA	'8X' or 'CX' or 'EX'	'CB'
SET DATA	'8X' or 'CX' or 'EX'	'DB'
VERIFY	'0X' or '4X' or '6X'	'20'
CHANGE PIN	'0X' or '4X' or '6X'	'24'
DISABLE PIN	'0X' or '4X' or '6X'	'26'
ENABLE PIN	'0X' or '4X' or '6X'	'28'
UNBLOCK PIN	'0X' or '4X' or '6X'	'2C'
DEACTIVATE FILE	'0X' or '4X' or '6X'	'04'
ACTIVATE FILE	'0X' or '4X' or '6X'	'44'
AUTHENTICATE	'0X' or '4X' or '6X'	'88', '89'
GET CHALLENGE	'0X' or '4X' or '6X'	'84'
TERMINAL CAPABILITY	'8X' or 'CX' or 'EX'	'AA'
TERMINAL PROFILE	'80'	'10'
ENVELOPE	'80'	'C2'
FETCH	'80'	'12'
TERMINAL RESPONSE	'80'	'14'
MANAGE CHANNEL	'0X' or '4X' or '6X'	'70'
MANAGE SECURE CHANNEL	'0X' or '4X' or '6X'	'73'
TRANSACT DATA	'0X' or '4X' or '6X'	'75'
Transmission oriented APDUs		
GET RESPONSE	'0X' or '4X' or '6X'	'C0'

Software

- Cardpeek
  - Analyse verschiedener Karten
  - Log der APDUs
- Eigene Software entwickeln
  - Support der PCSC-API
  - Spezifikationen → Kenntnis der Files und APDUs
- Beispiel: siehe Demo