03.02.2023 // NFC Lib - DESFire

LibLogicalAccess

> Müssen wir uns mal anschauen, da die das können, was wir wollen

ISO 7816

Die ISO 7816 spezifiziert in mehreren Teilen die Normen für kontaktbehaftete Chipkarten. Für die DESFire Bibliotheke ist der Teil 4 relevant, da dort die Kommunikation mit der Chipkarte definiert wird.

ISO 7816 Parts

siehe Wikipedia

ISO 7816 Part 4

APDU Command

(wird an PICC gesendet)

1 byte	1 byte	1 byte	1 byte	1* byte	max. 253* bytes	0-3 bytes
siehe Wikipedia						

APDU Response

(kommt als Antwort vom PICC)

max. 253** bytes	2 bytes

siehe Wikipedia

CLA=Instruction Class

> gibt an, ob es sich nachfolgend um standardisierte Befehle oder um proprietäre Befehle handelt. Normalerweise 0x90.

INS=Instruction Code

> jeweiliger Befehl der ausgeführt werden soll.

P1/P2 = Instruction parameter for command

> spezifische Parameter für den Befehl, z.B. ein Offset in der Datie in der geschrieben werden soll.

Lc = commando Length

> nennt die Anzahl der Bytes die in [Data] übergeben werden. Kann 1 Byte (max 255) oder 3 (max 65 535, erstes Byte muss 0 sein) Bytes verwendet werden.

Data = command data

> Daten die im Zuge des Befehls übertragen werden sollen / als Antwort vom PICC zurückgegeben werden.

Le = response Lenght

>

> Anzahl der Bytes die als Antwort erwartet werden. Dabei stellt 0x00 ein "Platzhalter" für "eine undefinierte Länge bis zu 255 Bytes" da.

SW1-SW2 = Response Trailer

> gibt den Anwort-Status wieder, z.B. (Hex): 90 00 für "Command successfully executed (OK)" 91 00 für "OK" siehe Response codes

"*" / "**" wir verwenden maximal 255 bytes, da ansonsten die Datenübertragung mittels MQTT sehr lang werden könnte.

Beispiel

select Application 0xC0FFEE siehe OTA Beispiel

CLA	0x90	Standard Befehlssatz
INS	0x5A	select Application
P1/P2	0x00/0x00	keine zusätzliche Parameter
Lc	0x03	3 Bytes
Data	0xEE, 0xFF, 0xC0	umgekehrte Reihenfolge(?)
Le	0x00	0x00 = maximal 255 bytes

Befehl: "\x90\x5A\x00\x00\x03\xEE\xFF\xC0\x00" bzw. 905A000003EEFFC000

Antwort "\x91\x00" bzw 9100

Data	-	Befehlt erwartet keine Daten (Le=0)
SW1/SW2	0x91, 0x00	"OK"

Befehle

Wrapping von DESFire Native Commands in ISO 7816-4 APDU Frames. ClA(Class) = 0x90 (Bei allen Commands für die Karte) INS(Instrucation) = Command Code des jeweiligen Befehls

Glossar

AID = Application ID

> Jede Application auf der DESFire Karte hat einen eigenen Identifier. Ein AID ist 3 Byte groß.

PICC = Proximity Integrated Circuit Card

> Die Chipkarte selber

IV = Initialisierungsvektor

> Ist ein Begriff aus der Kryptographie und bezeichnet einen Block von Zufallsdaten, der in bestimmten Modi einiger Blockchiffren verwendet wird, wie dem Cipher Block Chaining Mode.

CBC = Cipher Block Chaining

> Ist eine Betriebsart, in der Blockchiffren betrieben werden können. Vor dem Verschlüsseln eines Klartextblocks wird dieser zunächst mit dem im vorhergehenden Schritt erzeugten Geheimtextblock per XOR (exklusives Oder) verknüpft.

Datenstruktur Low Level

0x0A	0x00	n.v.	APDU data	specific
1 byte	1 byte	1 byte	max. 253 bytes	2 bytes

PCB = Protocol Control Byte (low level, prologue field)

> Ist Teil der "Umkapselung" eines APDU Kommandos in der Kommunikation mit dem PICC. Im Normalfall 0x0A (Information Block, no chaining, CID follows, no NAD, BlockNr = 0).

CID = Card ID (low level, prologue field)

> Ist Teil der "Umkapselung" eines APDU Kommandos in der Kommunikation mit dem PICC. Ist der Identifizierer des aktuelle PICCs. Ist nur relevant, wenn mehrere PICCs vom Leser gleichzeitig erkannt wurden und nacheinander selektiert wurden. Bei der Kommunikation mit nur einer Karte immer 0x00.

NAD = Node Address

> Identifizierer für Absender und Empfänger. Wird nicht verwendet.

EDC / CDC = Error Detection Code / Cyclic Redundancy Check (low level, epilogue field)

> Ist Teil der Umkapselung von jeder Nachricht an einem PICC. Wird forlaufend berechnet. Wird häufig von der Hardware selbst berechnet, muss jedoch manuell vom Programm angehangen werden.

INF = Information Field

> ist der Teil der Nachricht mit der Infromation mit der Karte ausgetauscht werden kann. Hier können einfache Kommandos oder APDUs verwendet

> werden

APDU definition

Get Application IDs

APDU Case: 2

0x6A			

The “Get AID List” command return the Application Identifiers of all active applications on a PICC.

0x000000 is PICC Application ID, is always returned.

> “TODO: 4 or 3 Bytes AID size TODO: MSB or LSB first

Select Application by AID

APDU Case: 3

0x5A			AID (3 Byte, MSB first)

The “Select Application” command allows to select one specific application for further access. If this parameter is 0x000000, the PICC level is selected and any further operations are related to this level. If an application with the specified AID is found in the application directory of the PICC, the subsequent commands interact with this application.

AuthenticateISO

Get Challenge APDU Case: 4

0x1A			key_id (1 Byte)

Send Challenge APDU Case: 4

0x5A			challenge (MSB first)

Authentication Process Authentication Process

Beispiel zum Beispiel

Zur Crypto Es wird CBC verwendet.

Zum Entschlüsslen von rndB wird der IV auf 0 gesetzt.

> The IV of the session key is reset to zero only ONCE when the key is created after authentication. The IV of the authentication key is reset only ONCE when authentication starts.

Bei allen anderen Verschlüsselungen oder Entschlüsselungen wird der letzte verschlüsselte Block als IV verwendet.

Response Codes

> [TODO: Copy from eftlab.com]

Over the Air

Ablauf Kommunikation

Authentication

```
// Reader
1. Authenticate against lokal BFFH
2. Start Proxy Session
```

```
3. Meldet die Präsenz einer Karte



// Client
1. Start Proxy Session

// Lokale BFFH
MifareDESFire mifareDESFire = new MifareDESFire(card);

mifareDESFire.SelectApplication(FabAccessAID);

// FabAccessIdentFileID ist unverschlüsselt
byte[] filedata = mifareDESFire.ReadData(FabAccessIdentFileID, 0x00000000, 0x00000000);

System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();
string userdomain = enc.GetString(filedata);

Console.WriteLine(userdomain);

// Domain BFFH
// Der Eigentümer der Karte
mifareDESFire.SelectApplication(FabAccessAID);

mifareDESFire.Authenticate(0x01, APP_Key_1);
// jetzt wissen wir, das die Karte Authentisch ist

byte[] filedata = mifareDESFire.ReadData(FabAccessIdentFileID, 0x00000000, 0x00000000);

System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();
string userdomain = enc.GetString(filedata);
// jetzt wissen wir, dass die Domain auch wirklich auf die Karte ist

// jetzt müßte man noch einmal die UID der Karte abfragen, um sicher zu stellen, dass der User
auch der ist, für den er sich ursprünglich ausgegeben hat.

connected_successfully = true;

card.Disconnect();
```

Troubleshooting

Error 0x910B

Reinstall NFC Driver on Windows

Quellen

NXP / Mifare

Allgemein

Mifare Application Directory Missing Native Commands

MIFARE DESFire spezifisch

Produktübersicht

EV1 Factsheet EV2 Factsheet EV3 Factsheet

Datasheets

EV1 Datasheet EV2 Datasheet EV3 Datasheet Light Datasheet

Features

Over the Air EV3 Features EV3 Quick Start

andere Links

Generic Access Control Model System Level Security PHILLIPS Presentation PHILLIPS DESFireSAM Authentification Spec ISO/IEC 14443 Type 4 Tag ISO/IEC 14443 Communication Error Code - 0x6A81

ISO

MIFARE ISO/IEC 14443

externe DESFire Spezifikationen oder Dokumentationen

DESFire OTA - Proof of Concept Ridrix DESFire Commands MIFARE DESFire Short Spec DESFire KeySettings Response Codes DESFire Authentification DESFire Command Set AES Authentication PHILLIPS DESFire Specification

ISO Specifications

ISO 7816-4 ISO/IEC7816-4

andere NFC Bibliotheken (mit DESFire)

JavaCardOS Easypay Android Java App RFID Door Lock RFDoorLock DESFire Server libfreefare DesFire for Python libopenkey mfrc522 **LibLogicalAccess**

> „Müssen wir uns mal anschauen, da die das können, was wir wollen

andere Links

DESFire Auth 2K3DES Authentication Question Kommunikationsbeispiele DESFire CRC Question

unsere Mitschnitte der Kommunkation verschiendener Bibliotheken

OTA Proof-of-Concept

Version #2
Erstellt: 2024-10-15 10:13:19 CEST von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 2025-02-25 21:22:13 CET von Mario Voigt (Stadtfabrikanten e.V.)