Czytnik kart chipowych (karty SIM)

Zasady działania kart chipowych:

Karty chipowe to stykowe karty plastikowe z wbudowanym układem elektronicznym. Chip czyli scalony układ elektroniczny jest w tym przypadku nośnikiem danych i zastępuje tradycyjne kody kreskowe czy paski magnetyczne.

Wszystkie karty chipowe są zgodne ze standardem EMV, który odpowiada za kompatybilność mikroprocesorów i terminali niezależnie od tego, jaka firma wydała kartę i gdzie się jej używa. Chip to po prostu mały układ scalony, który zapisuje informacje na karcie. Przesyłanie informacji do komputera następuje wtedy, gdy odpowiednie styki w czytniku połączą się z jego powierzchnią. Mikroprocesor, kontroluje zapis i odczyt informacji wtedy, gdy w pamięci zapisywane są dane. Chip wyposażony jest też w pamięć ROM (read-only memory), która dzieli się na trzy obszary. Pierwszy z nich to odczyt swobodny, w którym zawarte jest imię i nazwisko posiadacza karty, jej numer oraz data ważności. Drugi to obszar poufny, który zawiera poufne informacje o użytkowniku i dane producenta karty. Natomiast w trzecim obszarze zwanym roboczym przechowywane są informacje, które stale i dynamicznie się zmieniają. Są to np. saldo rachunku czy lista operacji i transakcji.

Działanie czytnika kart chipowych:

Czytnik kart inteligentnych podłączony do komputera hosta, komputera w chmurze lub dowolnego terminala sterującego zbiera informacje przechowywane na chipie mikroprocesorowym karty inteligentnej. Następnie wysyła takie informacje otrzymane z karty inteligentnej z powrotem do terminala sterującego w celu natychmiastowego przetworzenia.

Karta inteligentna łączy się z czytnikiem kart inteligentnych za pomocą unikalnego identyfikatora częstotliwości radiowej (**RFID**) lub za pośrednictwem protokołu systemu wykrywania kolizji nośnika (**CSCD**).

Tagi RFID znajdują zastosowanie w wielu różnych zastosowaniach, takich jak - śledzenie towarów w łańcuchu dostaw, śledzenie zasobów, kontrola dostępu do budynków i inne podobne zastosowania. Protokoły CSCD wykrywają częstotliwość nośną karty inteligentnej, aby dopasować jej prędkość. Komunikacja dwukierunkowa jest stosowana w przypadku wykrycia kolizji, a retransmisja opiera się na priorytecie wykrywania.

Standard PC/SC wersja 1.0:

Standardowy interfejs dostępu do kart elektronicznych. Głównym celem jest ustandaryzowanie, ułatwienie i popularyzacja użycia kart elektronicznych w komputerach osobistych. Korzystanie ze standardu odsuwa od programisty aplikacji wykorzystującej karty szczegóły związane z obsługą konkretnego typu czytnika czy protokołów komunikacji czytnik - karta.

Komendy APDU PC/SC, Funkcje API PC/S.C.

Struktura komend:

APDU Command						
Mandatory head				Optional body		
CLA	INS	P1	P2	Lc	Data field	Le

- CLA (1 byte): instruction class --- dedicated for an application domain
- INS (1 byte): defines the instruction of the command
- P1 (1 byte) and P2 (1 byte): the parameters of the instruction
- Lc (1 byte): data length
- With Le=0, - if a writing command => no useful data
- - if reading command => the command must return 256 bytes of data
- Data field (bytes whose length is the Lc value): a sequence of bytes.

APDU Response		
Optional body	Mandatory part	
Data field	SW1	SW2

- Data field (with a variable length): byte sequence
- SW1 (1 byte) and SW2 (1 byte): Status words sent by the card.

Fields of APDU command	Values
CLA	BC = french credit cards, vitale cards, A0 = SIM cards
INS	20 =PIN code verification, B0 = Binary read B2 = Read record D0 = Binary write DC = Write record A4 = Directory selection C0 = get an anwer
P1, P2	parameters
LEN	Length of the data sent by the command
ARG	contains LC bytes (PIN code to check)

Komendy:

APDU COMMAND	INS	P1	P2	P3
SELECT	A4	00	00	02
STATUS	F2	00	00	Length
READ BINARY	B0	Offset high	Offset low	Length
UPDATE BINARY	D6	Offset high	Offset low	Length
READ RECORD	B2	Record number	Mode	Length
UPDATE RECORD	DC	Record number	Mode	Length
SEEK	A2	00	Type/mode	Length
INCREASE	32	00	00	03
VERIFY CHV	20	00	CHV number	08
CHANGE CHV	24	00	CHV number	10
DISABLE CHV	26	00	01	08
ENABLE CHV	28	00	01	08
UNBLOCK CHV	2C	00		10
INVALIDATE	04	00	00	00
REHABILITATE	44	00	00	00
RUN GSM ALGORITHM	88	00	00	10
SLEEP	FA	00	00	00
GET RESPONSE	C0	00	00	Length
TERMINAL PROFILE	10	00	00	Length
ENVELOPE	C2	00	00	Length
FETCH	12	00	00	Length
TERMINAL RESPONSE	14	00	00	Length

Identyfikacja plików:

First Byte	GSM file type
3F	Master File
7F	Dedicated File
2F	Elementary File under the Master File
6F	Elementary File under a Dedicated File

Standard GSM 11.11

Ta specyfikacja ETSI określa elektroniczny i logiczny interfejs z kartą GSM SIM, który zawiera wszystkie informacje o tożsamości abonenta, klucze szyfrowania i różne inne pliki, takie jak książki telefoniczne i aplety. Niektóre elementy są oparte na normie ISO 7816, ale inne są specyficzne dla aplikacji telefonii komórkowej GSM.

Przykłady:

To access the Master File, we perform two steps. First, issue a "SELECT" instruction. If it is successful, we issue a "GET RESPONSE" instruction to retrieve the response data from the "SELECT" instruction.

1. Send the APDU to "SELECT" the Master File (which has the identifier `3F 00`):

   To Towitoko (9 bytes): `6F 05 05 62 A0 A4 00 00 02`
   From Towitoko (1 byte): `A4`
   To Towitoko (6 bytes): `6F 02 05 7E 3F 00`
   From Towitoko (2 bytes): `9F 1A`

2. Now send the APDU to GET RESPONSE from the previous SELECT instruction, using the length information (`1A` from the `9F` status). Note that we actually get back three additional bytes on top of the indicated 26 bytes:

   To Towitoko (9 bytes): `6F 05 05 62 A0 C0 00 00 1A`
   From Towitoko (29): `C0 00 00 AC FF 3F 00 01 00 00 00 00 00 0D 13 02 04 04 00 83 8A 83 8A 00 01 AC FF 90 00`

This 29-byte response has three parts:

- The first byte, `C0`, is an acknowledgment of the original APDU (The `C0` is the instruction code for "GET RESPONSE").
- The last two bytes (`90 00`) are the two status words indicating a successful "GET RESPONSE" command.
- The 26 bytes in between are the actual response data, which is broken out and annotated below:

```
[MF/DF] RFU: 00 00
  Free Memory: AC FF
  File ID:     3F 00 (MF)
  File Type:   01 (Master File)
  RFU:         00 00 00 00 00
  Length Following: 0D
  File characteristics:    13
    Clock stop:     Allowed, no preferred level
    Required speed: 13/4
    CHV:            Enabled
  Child DFs:               02
  Child EFs:               04
  CHVs, Unblock CHVs, etc: 04
  RFU:                     00
  CHV1 Status:             83 (Initialized, 3 remaining)
  Unblock CHV1 Status:     8A (Initialized, 10 remaining)
  CHV2 Status:             83 (Initialized, 3 remaining)
  Unblock CHV2 Status:     8A (Initialized, 10 remaining)
  RFU:                     00
```

To read an Elementary File, perform the following steps:

1. Send the APDU to SELECT the desired Elementary File. The proper Master File or Directory File must have been selected previously. In this example we read the ICCID file (identifier `2F E2`).

   To Towitoko (9 bytes): `6F 05 05 62 A0 A4 00 00 02`
   From Towitoko (1 byte): `A4`
   To Towitoko (6 bytes): `6F 02 05 7E 2D E2`
   From Towitoko (2 bytes): `9F 0F`

2. Now send the APDU to GET RESPONSE from the previous SELECT instruction, using the length information (`0F` from the `9F` status):

   To Towitoko (9 bytes): `6F 05 05 62 A0 C0 00 00 0F`
   From Towitoko (18): `C0 00 00 00 0A 2F E2 04 00 04 00 44 01 01 00 00 90 00`

```
[EF] RFU:    00 00
  File Size:   00 0A
  File ID:     2F E2 (EF-ICCID)
  File Type:   04 (Elementary File)
  RFU:         00
  Access:      04 00 44
    Read/Seek:    Always
    Update:       Admin 4
    Increase:     Always
    RFU:          Always
    Rehabilitate: Admin 4
    Invalidate:   Admin 4
  Status:      01 (Not Invalidated)
  Length:      01
  EF Structure:  00 (Transparent)
```

3. Now we read the contents of the file. This is a *transparent* type of data file, so we can perform a binary read. We request the 10 bytes (the `00 0A` file size) as given in the file information:

   To Towitoko (9 bytes): `6F 05 05 62 A0 B0 00 00 0A`
   From Towitoko (13 bytes): `B0 98 88 61 39 52 00 10 81 51 F4 90 00`

   Again we discard the first byte (`B0`, the original "READ BINARY" instruction) and the last two bytes (`90 00`, the two status words that indicate success). The remaining ten bytes are interpreted this way:

```
Identification Number: 8 9 8 8 1 6 9 3 2 5 0 0 0 1 1 8 1 5 4
Luhn is 4, computed is 4.
MII: 89 (Telecommunications), Country ID: 881 (Global Mobile Satellite System), Issuer ID: 6 (Iridium)
```

http://www.decodesystems.com/smartcards.html?fbclid=IwAR1nreeYRArrzg_XoYASqlaIKFeqwRseJfnb3wWlWB-YyCCLrMuPFMcSVXw

https://cedric.cnam.fr/~bouzefra/cours/TP1_SmartCard_SIM_Anglais.pdf

https://github.com/w4-pwr/Urzadzenia-Peryferyjne/blob/master/ChipReader/ConsoleApplication1/Program.cs

https://github.com/matjan98/Peryferia/blob/master/Czytnik%20kart%20chipowych/Program.cs