Oskar Szubert

Paweł Biel

Poniedziałek TP, 10:15

Grupa D

Prowadzący: dr inż. Jan Nikodem

Urządzenia Peryferyjne

Ćwiczenie 13- Czytnik kart chipowych.

Celem drugich ćwiczeń z Urządzeń Peryferyjnych było zapoznanie się z budową oraz funkcjonowaniem czytnik kart GSM(Ćwiczenie 13- Czytnik Kart Chipowych) . Zadaniem do wykonania w trakcie zajęć było napisanie programu komunikującego się z kartą chipową za pomocą standardu PC/SU. A następnie obsługującego kartę SIM przy pomocy komend APDU.

1. Zagadnienia teoretyczne:

Karty chipowe są nośnikiem danych, które charakteryzują się wielokrotnością usług oraz skuteczniejszą ochroną danych w stosunku do kart z paskiem magnetycznym. Opisywana karta jest najogólniej plastikiem wraz z wbudowanym układem scalonym- tzw. Chipem. Styki pokryte są złotem- w standardzie opisywanych jest osiem rodzajów – jednakże producent nie musi zamieszać wszystkich- a jedynie tych z których będzie korzystał. Mikroprocesor zapewnia kontrolę odczytu i zapisu danych, które umieszczone są w pamięci. Mikroprocesor ma możliwość kontrolowania nieudanych prób wprowadzania, po ustalonej wcześniej liczbie złych logowań karta może być zablokowana. Najczęściej stosowanymi mikroprocesorami w kartach są 8-bitowe moduły z pamięcią do ponownego zapisu EEPROM (ang. electrically erasable programmable read-only memory). Wyróżniamy 3 obszary pamięci karty:

- Obszar swobodnego odczytu- zazwyczaj znajdują się tu powszechne informacje o karcie i/lub użytkowniku.
- Obszar Poufny- dostęp do tego rodzaju pamięci dostępny jest jedynie po podaniu kodu PIN
- Obszar roboczy- przechowuje dane które ulegaj ciągłej modyfikacji

Karty chipowe możemy też podzielić ze względu interfejsy komunikacyjne czyli na sposób przesyłania danych-:

- Karty Stykowe* – zgodne ze standardem z ISO-7816. W tego rodzaju kartach rozróżniamy różne protokoły komunikacyjne: T=0 (jednokierunkowa transmisja bajtów) lub T=1 (jednokierunkowa transmisja bloków) oraz inne.

** Tego rodzajem kart będziemy się zajmować na zajęciach laboratoryjnych.*

- Karty bezstykowe- zgodne ze standardem z ISO14443. Komunikują się z czytnikiem za pomocą fal elektromagnetycznych na różnych częstotliwościach np. 13.56 Mhz.

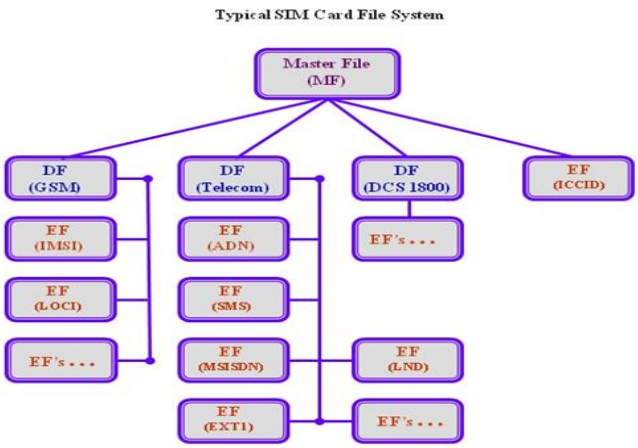
Standard GSM 11.11 jest oryginalnym Standarem dla kart SIM, pokrywa się on z formatem plików i formatem komunikatów APDU, które są wysyłane do kart SIM. GSM 11.11 jest oparty na standardzie ISO 7816-4

System plików na karcie SIM możemy przedstawić w postaci drzewa.

Rozróżniane będą trzy typy plików głównych:

- MF (master file) – Root
- DF (dedicated file) – odpowiednik folderów
- EF (elementary file) – plik z danymi

Nazwy plików są długości 2 bajtów.



Działanie czytnika kart jest intuicyjne. Po podłączaniu karty chipowej do czytnika następuje zasilenie karty przez odpowiedni styk. Następnie zostanie z karty zostanie wysłana wiadomość ATR(Answer to Reset), która informuje czytnik o:

- Rodzaju karty
- Sposobie kodowania bitów 0 i 1
- Wspieranych protokołach komunikacji.

Po połączeniu będziemy mogli się komunikować z kartą za pomocą protokołu APDU. Opisuje on logiczną warstwę interfejsu komunikacyjnego- co oznacza, że jest niezależny od fizycznego interfejsu.

Warto zaznaczyć, że ATR oraz APDU jest ściśle ustandaryzowane .

2. *Format komend APDU:*

Command ADPU						
CLA	INS	P1	P2	L_c	DATA FIELD	L_e

CLA - klasa komendy – 1byte (czy jest to komenda bezpieczeństwa itp.).

Określa klasę instrukcji:

- 0XA0 – karty GSM

- 0x80 – karty pamięciowe
- 0x00 – karty bankowe (ISO 7816)

INS – instrukcja, którą ma wykonać karta- 1byte

- Wartość zawsze parzysta
- 256 wartości

Podstawowe Komedy:

- -0x20 – VERIFY
- -0X84 - GET CHELLENGE
- -0XA4 -SELECT FILE
- -0XB0/0XB2 – READ BINARY/READ RECORD
- -0XD0/0XD2 – WRITE BINARY/READ RECORD
- 0XC0 – GET RESPONSE
    - P1 - parametr 1 instrukcji - 1byte
    - P2 - parametr 2 instrukcji - 1byte
  - L_c – Długość komendy – 0-3byte
  - DATA – zapytanie ADPU- $L_c$byte
  - L_e – Długość oczekiwanej odpowiedzi – 0-3byte

ADPU request		
DATA	SW1	SW2

- DATA –informacje będące odpowiedzią na komendę
- SWx:
    - - 0x9000 – komenda wykonana poprawnie
    - - 0x61zz - komenda wykonana poprawnie, **zz** byte do odczytu
    - -0x69zz – komenda niedozwolona, **zz** byte koduje przyczynę
    - -0x6AZZ- niepoprawne parametry P1 lub/i P2

3. *Kod programu:*

Metody, typy zmiennych oraz wyjątki zostały pobrane z githuba twórcy pakietu PCSC.

https://github.com/danm-de/pcsc-sharp

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using PCSC;         //bibloteka obsługująca API dla czytnika kart SIM

namespace sim_card
{
    class Program
    {
        private static SCardError error;
        private static SCardReader reader;
        private static System.IntPtr intptr;
        private static SCardContext context;

        static void Main(string[] args)
        {
            //odpowiednie polecenia będą przesyłane do czytnika w formie bitowej.
            byte[] commandB;
            try
            {
                connection();

                //wysłanie kolejnych komend do czytnika, celem programu jest odczytanie SMSa
                //poruszanie sie po poszczególnych poziomach:

                // wejscie w galaz telecom:
                // adres złożony  z 2 znaków 107F
                commandB = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x7F, 0x10 };
                sendCommand(commandB, "SELECT(TELECOM)");
                Console.WriteLine("Odpowiedz TELECOMU\n");

                //oczekiwanie odpowiedzi o długości 22(10)
                commandB = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x16 };
                sendCommand(commandB, "GET RESPONSE");

                // wejscie w galaz SMS
                commandB = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x6F, 0x3C };
                sendCommand(commandB, "SELECT SMS");
                Console.WriteLine("Odpowiedz gałęzi SMS\n");
                commandB = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x16 };

                // odczyt smsa
                commandB = new byte[] { 0xA0, 0xB2, 0x01, 0x04, 0xB0 };
```

```
                sendCommand(commandB, "READ RECORD");
                Console.WriteLine("Odczyt SMS");
                commandB = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x16 };
                sendCommand(commandB, "GET RESPONSE");

                Console.WriteLine("Wcisnij dowolny klawisz by kontynowac...");
                Console.ReadKey();
            }
            catch (Exception e)
            {
                Console.WriteLine("Podczas uruchamiana programu wystąpił blad.");
                Console.ReadKey();
            }
        }
        public static void connection()
        {
            context = new SCardContext(); //nawiązanie połączenia z czytnikiem

            string[] readerList = context.GetReaders(); // wczytanie dostępnych czytników do listy
            Boolean noReaders = readerList.Length <= 0;
            if (noReaders)
            {
                throw new PCSCException(SCardError.NoReadersAvailable,
                                                                "Czytnik nie zostal odnaleziony");
            }

            int counter = 1;
            Console.WriteLine("Wybierz czytnik: ");
            foreach (string element in readerList)
            {
                Console.WriteLine("F" + counter + " -> " + element);
                counter++;
            }
            var input = Console.ReadKey();
            string tmp = readerList[0];
            switch (input.Key)
            {

                case ConsoleKey.F1:
                    tmp = readerList[0];
                    break;

                case ConsoleKey.F2:
                    tmp = readerList[1];
                    break;
            }

            Console.WriteLine("Wcisnij dowolny klawisz by kontynowac...");
            Console.ReadKey();


            //w zależności od wybranego czytniku wybrany zostanie odpowiedni protokół T0 lub T1.
             W przypadku pozostałych zostanie wyrzucony wyjątek
            reader = new SCardReader(context);
            error = reader.Connect(tmp, SCardShareMode.Shared, SCardProtocol.T0 | SCardProtocol.T1);
            checkError(error);
            if (reader.ActiveProtocol == SCardProtocol.T0)
            {
                intptr = SCardPCI.T0;

            }
            else if (reader.ActiveProtocol == SCardProtocol.T1)
            {
```

```
                intptr = SCardPCI.T1;
            }



            else
            {
                Console.WriteLine("Protokol nie jest obslugiwany");
                Console.WriteLine("Wcisnij dowolny klawisz by kontynowac...");
                Console.ReadKey();
            }
        }

        public static void sendCommand(byte[] command, String name) // przesyłanie komend do karty
        {
            byte[] recivedBytes = new byte[256];
            error = reader.Transmit(intptr, command, ref recivedBytes);
            checkError(error);
            writeResponse(recivedBytes, name);
        }
        //odczytanie odpowiedzi z karty
        public static void writeResponse(byte[] recivedBytes, String responseCode)
        {
            Console.Write(responseCode + ": ");
            for (int i = 0; i < recivedBytes.Length; i++)
                Console.Write("{0:X2} ", recivedBytes[i]); // wypisanie odpowiedzi binarnie
            Console.WriteLine();
        }

        static void checkError(SCardError error) //sprawdzenie czy włożona została karta
        {
            if (error != SCardError.Success)
            {
                throw new PCSCException(error, SCardHelper.StringifyError(error));
            }
        }
    }
}
```

4. *Wynik działania programu:*

Napisany przez nas program został przetestowany na karcie SIM dostępnej w zestawie laboratoryjnym. Z racji przetestowania kart za pomocą gotowych programów wiedzieliśmy, że karta SIM jest pusta, co zostało potwierdzone za pomocą naszego programu.

```
Wybierz czytnik:
F1 -> OMNIKEY CardMan 5x21 0
F2 -> OMNIKEY CardMan 5x21-CL 0
 Wcisnij dowolny klawisz by kontynowac...
SELECT(TELECOM): 9F 1B
Odpowiedz TELECOMU

GET RESPONSE: 00 00 0A 05 7F 10 02 FA FF 66 FF 01 0E 37 00 0C 04 00 80 8A 80 8A
90 00
SELECT SMS: 9F 0F
Odpowiedz wejscia w galaz SMS

GET RESPONSE: 67 0F
READ RECORD: 98 04
Odczyt SMS
GET RESPONSE: 6F 00
Wcisnij dowolny klawisz by kontynowac...
```

Z powyższego zrzutu ekranu, możemy odczytać, iż podłączone zostały dwa czytniki kart SIM.

Odkodowanie niektórych odpowiedzi:

- Wartość na końcu odpowiedzi, **90 00** oznacza, że polecenie zostało wykonane poprawnie
- Odpowiedz **9F 1B** – oznacza wykonanie poprawne komendy, a **1B** są bitami dostępnymi, które można odczytać.
- Odpowiedz **67 0F** – oznacza błędna długość oczekiwanej odpowiedzi- w tym przypadku, brak wiadomości do odczytu
- Odpowiedz **98 04** – Nie są spełnione warunki dostępu, uwierzytelnianie nie powidło się.
- Odpowiedz **6F 00** – oznacza nie wykonie się polecenia. W tym przypadku nie będzie możliwa dogłębniejsza diagnoza przyczyna porzucenia wykonania polecenia.