Paweł Twardawa 235072
Aleksandra Wieczorkiewicz 234980

Data: 20.11.2018 r.

Urządzenia peryferyjne

Ćwiczenie 13

Czytnik kart chipowych

Prowadzący:
Dr inż. Jacek Mazurkiewicz

1. Wstęp

Celem zadania było zapoznanie się z obsługą kart chipowych, poznanie sposobu komunikacji oraz odczyty/zapis danych z karty. Program został napisany w języku C# w wersji konsolowej.

2. Zadania do wykonania

2.1. Obsługa kart SIM za pomocą aplikacji SimEditor

2.2. Obsługa kart SIM przy pomocy komend APDU (wykorzystaj zainstalowaną aplikacje)

2.2.1. zapoznać się z komendami APDU do kart SIM (standard GSM 11.11)

2.2.2. zalogować się do karty SIM (podając PIN)

2.2.3. odczytać wybrany plik z karty SIM lub z książki telefonicznej

2.3. Napisać własną aplikację komunikującą się z kartą chipową za pomocą standardu PC/SC

2.3.1. zapoznać się z interfejs programistycznym PC/SC

2.3.2. zapoznać się z przykładowym z książki "Inżynieria programowania kart inteligentnych" (PCSCExample.c)

2.3.3. zbudować własną aplikację realizującą: łączenie się z czytnikiem i wykrywanie karty, odczyt danych z kart z punktu 2, odczyt danych z kart SIM (np. SMS-y czy lista kontaktów)

3. Opis programu

3.1. Nawiązywanie połączenia

Na listingu 1 przedstawiono sposób nawiązania połączenia z kartą chipową. W tym celu wykorzystano dwie klasy SCardContext oraz SCardReader z pakietu PCSC. Ponieważ karty chipowe obsługują różnie standardy komunikacji, podczas otwierania połączenia należy wybrać odpowiedni protokół.

```
public static void connect()
        {
            context = new SCardContext();
            context.Establish(SCardScope.System);

            string[] readerList = context.GetReaders();

            bool noReaders = readerList.Length <= 0;
            if(noReaders)
            {
                Console.WriteLine("error read");
            }

            Console.WriteLine("nazwa czytnika: " + readerList[0]);

            reader = new SCardReader(context);

            error = reader.Connect(readerList[0],
SCardShareMode.Shared, SCardProtocol.T0 | SCardProtocol.T1);


            switch(reader.ActiveProtocol)
            {
                case SCardProtocol.T0:
                    {
                        protocol = SCardPCI.T0;
                        break;
```

```
                }
            case SCardProtocol.T1:
                {
                    protocol = SCardPCI.T1;
                    break;
                }
        }
    }
```

3.2. **Wysyłanie poleceń**

Wysyłanie komend do karty odbywa się za pomocą metody Transmit() klasy SCardReader. Jako parametry przyjmuje ona protokół komunikacji, komendę APDU zapisaną w postaci ciągu bajtów oraz referencje na tablicę bajtów do której zapisuje odpowiedz.

Listing 2.

```
static void sendCommand(byte[] command, String name)
        {
            byte[] receivedBytes = new byte[256];
            error = reader.Transmit(protocol, command, ref
receivedBytes);

            writeResponse(receivedBytes, name);
        }
```

3.3. **Wyświetlanie odpowiedzi**

Po wysłaniu komendy metodą sendCommand() wywoływana jest metoda przedstawiona w listingu 3 która odpowiada za wyświetlanie odpowiedzi na ekranie.

Listing 3.

```
private static void writeResponse(byte[] receivedBytes, string
responseCode)
        {
            Console.Write(responseCode + ": ");
            for(int i = 0; i< receivedBytes.Length; i++)
            {
                Console.Write(" {0:X2}" ,receivedBytes[i]);
            }
            Console.WriteLine();
        }
```

3.4. Główna funkcja programu

Listing 4 przedstawia metodę Main() która wysyła odpowiednie komendy do karty. Sekwencja wysyłania komend nie jest losowa. Pierwsze 3 komendy są potrzebne aby karta z nami współpracowała. Przy użyciu komendy Select SMS przechodzimy do katalogu z zapisanymi wiadomościami tekstowymi, następnie wysyłamy komendę otrzymania odpowiedzi i czytamy rekord. Po wykonaniu wszystkich żądanych poleceń należy zakończyć połączenie. Wszystkie te operacje otoczone są blokiem try catch który odpowiada za obsługę wyjątków PCSCException.

Listing 4.

```
static void Main(string[] args)
        {

            try
            {
                connect();

                byte[] commandBytes = new byte[] { 0xA0, 0xA4, 0x00,
0x00, 0x02, 0x7F, 0x10 };
                sendCommand(commandBytes, "SELECT TELECOM");

                commandBytes = new byte[] { 0xA0, 0xA4, 0x00, 0x00,
0x02, 0x7F, 0x10 };
                sendCommand(commandBytes, "SELECT TELECOM");

                commandBytes = new byte[] { 0xA0, 0xC0, 0x00, 0x00,
0x16};
                sendCommand(commandBytes, "GET RESPONSE");

                commandBytes = new byte[] { 0xA0, 0xA4, 0x00, 0x00,
0x02, 0x6F, 0x3C };
                sendCommand(commandBytes, "SELECT ADN");
                /*
                commandBytes = new byte[] { 0xA0, 0xA4, 0x00, 0x00,
0x02, 0x6F, 0x3C };
                sendCommand(commandBytes, "SELECT SMS");

                */
                commandBytes = new byte[] { 0xA0, 0xC0, 0x00, 0x00,
0x0F};
                sendCommand(commandBytes, "GET RESPONSE");

                commandBytes = new byte[] { 0xA0, 0xB2, 0x02, 0x04,
0xB0};
                sendCommand(commandBytes, "READ RECORD");

                context.Release();

            }
            catch (PCSCException ex)
            {
                Console.WriteLine(ex.Message);
            }

            Console.ReadLine();
        }
```

4. Wnioski

Karty chipowe mają wiele zastosowań. Są wykorzystywane w kartach płatniczych do uwierzytelniania transakcji jak i w telefonach komórkowych do nawiązywania połączenia. Karty chipowe komunikują się za pomocą komend APDU przez odpowiednie protokoły które są uzależnione od producenta oraz przeznaczenia karty.