Karta Sim

Wiktor Wojnar Krzysztof Zalewa

Październik 2024

1 Wstęp

W ramach laboratorium (grupa 3.) mieliśmy wykonać oczytanie danych z karty sim. Jako wyposażenie do laboratorium otrzymaliśmy:

- Karty Sim (3 sztuki)
- Skaner do inteligentych kart z portem usb
- Komputer stacjonarny, wraz z monitorem, myszką i klawiaturą

Do obsługi kart należało użyć interfejsu wbudowanego systemu windows, o nazwie winscard.

2 Karta Sim

Karta SIM, czyli Subscriber Identity Module, to mała plastikowa karta używana w urządzeniach mobilnych zawierająca mikroprocesor, który przechowuje podstawowe informacje, takie jak numer telefonu użytkownika, tożsamość sieciową czy klucze bezpieczeństwa. Umożliwia urządzeniu łączenie się z siecią komórkową, ułatwiając komunikację i usługi danych. Karty SIM występują w różnych rozmiarach: standardowym, mikro oraz nano, i są wymienne, co pozwala użytkownikom na łatwą zmianę karty między urządzeniami. Obsługują również usługi takie jak SMS, przekierowywanie połączeń i mobilne dane szerokopasmowe, co czyni je integralną częścią nowoczesnej telekomunikacji.

2.1 Dane Techniczne

- **Charakterystyka fizyczna**: Karta SIM to mała, prostokątna karta o grubości 0,76 mm i wymiarach 15 mm x 25 mm (mini-SIM) lub 12,3 mm x 8,8 mm (micro-SIM) lub 8,8 mm x 12,3 mm (nano-SIM).
- **Pamięć**: Karta SIM ma zazwyczaj pojemność pamięci od 16 KB do 256 KB, która służy do przechowywania danych, takich jak międzynarodowy numer identyfikacyjny abonenta komórkowego (IMSI), klucze uwierzytelniające i inne informacje związane z siecią.
- **Processor**: Karta SIM ma mały mikroprocesor który wykonuje instrukcje softwarowe i wykonuje operacje kryptograficzne.
- **Communication Interface**: Karta SIM komunikuje się z użądzeniem mobilnym poprzez interfejs interfejs szeregowy, zazwyczaj przy użyciu standardu ISO/IEC 7816.

2.2 Komponenty Funkcjonalne

- **Autentykacja**: Karta SIM uwierzytelnia abonenta w sieci, wykorzystując mechanizm wyzwania-odpowiedzi, który obejmuje wymianę kluczy kryptograficznych i kodów uwierzytelniających.
- **Data Storage**: Na karcie SIM przechowywane są dane takie jak numery telefonów, wiadomości SMS i ustawienia sieciowe.
- **Elementy Bezpieczeństwa**: Karta SIM zapewnia bezpieczne środowisko do przechowywania danych poufnych jak klucze kodujące, czy dane logowania.

2.3 Typy kart SIM

- Pełna Karta, dziś już niemalże nieużywana
- Mini SIM, pomniejszona wersja, używana w starszych urządzeniach mobilnych
- Micro SIM, czyli jeszcze mniejsza wersja karty. Używana w wiekszości urządzeń mobilnych z lat 2010-2015.

· Nano SIM, karta najmniejsza, używana w każdym nowoczesnym urządzeniu mobilnym. Od roku 2018 stała się jedynym interfejsem SIM w nowo produkowanych urządzeniach.

· E-SIM, nowa wersja karty sim. Jest ona wbudowana w płytę głowną nowych urządzeń moblinych. Daje możliwość "utworzenia" nowego numeru w kilka minut.

3 Czytnik Kart

Czytnikiem kart inteligentych, który otrzymaliśmy do użycia i testów, na czas laboratorium jest urządzenie o nazwie "***OMNIKEY CardMan*** $5x21$" Czytnik ten jest podwójnym interfejsem PC-Linked, który odczytuje i zapisuje dane, na bezstykowych kartach 13.56 Mhz, oraz na praktycznie każdej karcie intelegntej, stykowej. Czytnik obsługuje karty z przepustowością do 1mb/s, przy zgodności z iso 14443.

4 WINSCARD

Do obsługi czytnika kart mieliśmy użyć wbudowanej biblioteki windowsa, o nazwie winscard. Zawiera ona komendy pozwalające na znalezienie czytnika kart inteligentych, oraz jego obsługę. Poniżej opisujemy użyte komendy, jedna po drugiej, wraz z ich działaniem.

5 Komendy APDU

5.1 Struktura komend APDU

Tabela 1: Przykład tablicy w LaTeX

Nazwa pola	**Długość**	**Opis**
CLA	1	Klasa instrukcji - wskazuje na typ polecenia np. interindustry or proprietary.
INS	1	Kod instrukcji - wskazuje na konkretne polecenie np. "select", "write data".
P1-P2	2	Parametry instrukcji dla komendy np. ,przesunięcie w pliku do którego zapisujemy dane
Lc	0, 1 or 3	Koduje liczbę (Nc) bajtów danych poleceń, które mają nastąpić. 0 bajtów oznacza Nc=0. 1 bajt o wartości od 1 do 255 oznacza Nc o tej samej długości. 3 bytes, the first of which must be 0, denotes Nc in the range 1 to 65 535 (all three bytes may not be zero).
Command data	Nc	Nc bitów danych.
Le	0, 1, 2 or 3	Koduje maksymalną liczbę (Ne) oczekiwanych bajtów odpowiedzi. 0 bajtów oznacza Ne=0. 1 bajt w zakresie od 1 do 255 oznacza wartość Ne, lub 0 oznacza Ne=256. 2 bajty (jeśli w poleceniu była obecna rozszerzona Lc) w zakresie od 1 do 65 535 oznaczają Ne o tej wartości, lub dwa bajty zerowe oznaczają 65 536. 3 bajty (jeśli Lc nie była obecna w poleceniu), z których pierwszy musi być 0, oznaczają Ne w ten sam sposób jak dwubajtowe Le.

5.2 Struktura dpowiedzi APDU

Tabela 2: Odpowiedzi APDU

Nazwa pola	**Długość**	**Opis**
Response data	Nr (at most Ne)	Dane odpowiedzi
SW1-SW2(Response trailer)	2	Status przetwarzania komendy np 90 00 (hexadecymalny) wskazuje na sukces

6 Kod programu

```
#include <iostream>
#include <winscard.h>
#include <cstdint>

int32_t main()
{
    SCARDCONTEXT hContext = NULL;
    SCARDHANDLE hCard = NULL;
    DWORD dwActiveProtocol = 0;
    LPTSTR mszReader = NULL;
    DWORD cchReader = SCARD_AUTOALLOCATE;

    LONG lReturn = SCardEstablishContext(
        SCARD_SCOPE_USER, NULL, NULL, &hContext);
    if (lReturn != SCARD_S_SUCCESS)
    {
        printf("Failed to establish context");
        exit(1);
    }

    lReturn = SCardListReaders(hContext, NULL, (LPTSTR)
        &mszReader, &cchReader);
    if (lReturn != SCARD_S_SUCCESS) {
        if (lReturn == SCARD_E_NO_READERS_AVAILABLE)
            printf("NO reader");
        else
            printf("Failed to get List");
        SCardReleaseContext(hContext);
        exit(1);
    }

```

```
            int j = 1;
            printf("Available Readers:\n%d. ", j);
            for (int32_t i = 0; i < cchReader; i++) {
                if (mszReader[i] == '\000') {
                    j++;
                    printf("\n%d. ", j);
                }
                else
                    printf("%c", mszReader[i]);
            }
            printf("\n");

            int dest;
            printf("Wybierz urzadzenie: ");
            scanf_s("%d", &dest);

            wchar_t buf[256] = { 0 };
            uint32_t bufIt = 0;
            j = 1;
            for (int32_t i = 0; i < cchReader; i++) {
                if (mszReader[i] == '\000') {
                    if (j == dest) {
                        buf[bufIt] = '\0';
                        break;
                    }
                    j++;
                }
                else {
                    if (j == dest) {
                        buf[bufIt] = mszReader[i];
                        bufIt++;
                    }
                }
            }

            if (bufIt == 0) {
                printf("No valid reader selected.\n");
                SCardFreeMemory(hContext, mszReader);
                SCardReleaseContext(hContext);
                exit(1);
            }

            static wchar_t readerName[] = L"OMNIKEY CardMan 5
                x21 0";
            while (true) {
                lReturn = SCardConnect(
                    hContext,
                    readerName,
                    SCARD_SHARE_SHARED,
```

```
                    SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                    &hCard,
                    &dwActiveProtocol
                );
                if (lReturn == SCARD_S_SUCCESS)
                    break;
                Sleep(100);
            }
            BYTE mfResponse[256];
            DWORD mfResponseLength = 256;
            BYTE mfCommand[] = { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x3F, 0x00 };
            LONG result = SCardTransmit(hCard, SCARD_PCI_T0, mfCommand, sizeof(mfCommand), NULL, mfResponse, &mfResponseLength);
            if (result != SCARD_S_SUCCESS) {
                // Handle error
            }
            // Print the response
            printf("Response: ");
            for (DWORD i = 0; i < mfResponseLength; i++) {
                printf("%02X ", mfResponse[i]);
            }
            printf("\n");

            BYTE dfResponse[256];
            DWORD dfResponseLength = 256;
            BYTE dfCommand[] = { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x2F, 0x00 };
            result = SCardTransmit(hCard, SCARD_PCI_T0, dfCommand, sizeof(dfCommand), NULL, dfResponse, &dfResponseLength);
            if (result != SCARD_S_SUCCESS) {

            }
            for (DWORD i = 0; i < mfResponseLength; i++) {
                printf("%02X ", dfResponse[i]);
            }
            printf("\n");

            BYTE adnResponse[256];
            DWORD adnResponseLength = 256;
            BYTE adnCommand[] = { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x6F, 0x07 };
            result = SCardTransmit(hCard, SCARD_PCI_T0, adnCommand, sizeof(adnCommand), NULL, adnResponse, &adnResponseLength);
            if (result != SCARD_S_SUCCESS) {

```

```
118             }
119             for (DWORD i = 0; i < mfResponseLength; i++) {
120                 printf("%02X ", adnResponse[i]);
121             }
122             printf("\n");
123 
124             BYTE kontakty[] = { 0xA0, 0xB2, 0x2C, 0x14, 0x00 };
125 
126             BYTE responseBuffer[256];
127             DWORD responseLength = 256;
128 
129             if (lReturn == SCARD_S_SUCCESS) {
130 
131                 lReturn = SCardTransmit(hCard, SCARD_PCI_T0,
                        kontakty, sizeof(kontakty), NULL,
                        responseBuffer, &responseLength);
132 
133                 if (lReturn == SCARD_S_SUCCESS) {
134                     // Print the response
135                     printf("Response: ");
136                     for (DWORD i = 0; i < responseLength; i++)
                            {
137                         printf("%02X ", responseBuffer[i]);
138                     }
139                     printf("\n");
140                 }
141                 else {
142                     printf("Failed to send APDU command\n");
143                 }
144             }
145             else {
146                 printf("Failed to connect to smart card\n");
147             }
148 
149 
150             BYTE numRecords = responseBuffer[1];
151 
152             // Loop through each record
153             for (DWORD i = 0; i < numRecords; i++) {
154                 // Get the record
155                 BYTE record[14];
156                 memcpy(record, responseBuffer + 2 + (i * 14),
                        14);
157 
158                 // Extract the phone number
159                 char phoneNumber[13];
160                 for (DWORD j = 0; j < 12; j++) {
161                     phoneNumber[j] = record[j];
162                 }
```

```
163                 phoneNumber[12] = '\0'; // Null-terminate the
                        string
164
165                 // Extract the name length and name
166                 BYTE nameLength = record[12];
167                 char name[15];
168                 for (DWORD j = 0; j < nameLength; j++) {
169                     name[j] = record[13 + j];
170                 }
171                 name[nameLength] = '\0'; // Null-terminate the
                        string
172
173                 // Print the contact information
174                 printf("Phone number: %s, Name: %s\n",
                        phoneNumber, name);
175             }
176             SCardDisconnect(hCard, SCARD_LEAVE_CARD);
177             SCardFreeMemory(hContext, mszReader);
178             SCardReleaseContext(hContext);
179
180             printf("SUCCESS");
181
182             return 0;
183         }
```

Listing 1: Kod programu

7 Typy Danych

- SCARDCONTEXT: Jest to wskaźnik kontekstu urządzenia docelowego. Jest to zarazem wersja klasycznego windowsowego hContextu.
- SCARDHANDLE: Jest to handler kontekstu, służy bezpośrednio do obsługi urządzenia.
- DWORD: Jest to uint32_t (Liczba całkowita 32 bitowa, oznaczona)
- LONG: Jest to int32_t (Liczba całkowita 32 bitowa, nieoznaczona)
- LPTSTR: Jest to typ danych literycznych (string), służacy jako interfejs, dla LPWSTR, w przypdaku użycia unicode, albo LPSTR, gdy unicode nie jest używany.
- BYTE: Jest to uint8_t (Liczba całkowita 8 bitowa, oznaczona)

8 Komendy użyte

- SCardEstablishContext:
  [IN] DWORD ... dwScope: Przyjmuje jedną z wartości: SCARD_SCOPE_USER (Operacje dostępu realizowane są w zakresie użytkownika), SCARD_SCOPE_SYSTEM (Operacje realizowane są w zakresie systemu. Aplikacje wywołujące muszą mieć odpowiednie uprawnienia, aby móc wykonać jakiekolwiek akcje na danych).
  [IN] LPCVOID ... pvReserved1: Miejsce zarezerwowane na przyszłe możliwości użycia.
  [IN] LPCVOID ... pvReserved2: Miejsce zarezerwowane na przyszłe możliwości użycia.
  [OUT] LPSCARDCONTEXT ... phContext: Handler do instancji menadżera zasobów.
  Wartość Zwracana: Jeżeli funkcja zakończy się z sukcesem zwracany jest kod : SCARD_S_SUCCESS, w przeciwnym wypadku funkcja napotkała błąd. Wartość błedu można sprawdzić przy pomocy strony microsoftu [1]

- SCardListReaders:

  [IN] SCARDCONTEXT ... hContext: Handler menadżera zasobów, identyfikujący jego kontekst, dla zapytania, jego wartość może być ustawiona tylko i wyłącznie poprzez uprzednie zapytanie SCardEstablishContext.
  [IN,OUT] LPCSTR ... mszGroups: Opcjonalny parametr umożliwiający ogarniczenie typów zdobytych czytników. Wartość bazowa to null, może ona także wynosić : SCARD_ALL_READERS, SCARD_DEFAULT_READERS, SCARD_LOCAL_READERS, SCARD_SYSTEM_READERS.
  [OUT] LPCSTR ... mszReaders: Jest to multi-string zawierający w sobie nazwy kolejncych znalezionych czytników. Jeżeli ta wartość jest NULL, zmienna długości bufora jest ignorowana.
  [IN,OUT] LPDWORD ... pcchReaders: Jest to zmienna trzymająca wartość dłguości bufora. Zawiera w sobie także wszystkie znaki null. Blok ten musi zostać Dealokowany Ręcznie za Pomocą SCardFreeMemory()
  Wartość Zwracana: Jeżeli funkcja zakończy się z sukcesem zwracany jest kod : SCARD_S_SUCCESS, w przeciwnym wypadku funkcja napotkała błąd. Wartość błedu można sprawdzić przy pomocy strony microsoftu [1]

- SCardReleaseContext:

  [**IN**] **SCARDCONTEXT ... hContext**: Handler identyfikujący konteks menadżera zasobów.
  Wartość Zwracana: Jeżeli funkcja zakończy się z sukcesem zwracany jest kod : SCARD_S_SUCCESS, w przeciwnym wypadku funkcja napotkała błąd. Wartość błedu można sprawdzić przy pomocy strony microsoftu [1]

- SCardConnectA:

  [**IN**] **SCARDCONTEXT ... hContext**: Handler identyfikujący konteks menadżera zasobów.
  [**IN**] **LPCSTR ... szReader**: Nazwa czytnika kart, w który włożona jest karta.
  [**IN**] **DWORD ... dwShareMode**: Tryb udostępniania czytnika, wraz z kartą, może być ustawiony w tryb: SCARD_SHAR_SHARED, SCARD_SHAR_EXCLUSIVE, SCARD_SHARE_DIRECT
  [**IN**] **DWORD ... dwPreferredProtocols**: Tryb akceptowanych protokołów połączenia : SCARD_PROTOCOL_T0, SCARD_PROTOCOL_T1
  [**OUT**] **DWORD ... dwPreferredProtocols**: Handler identyfikujący aktualnie obsługiwaną kartę
  [**OUT**] **DWORD ... pdwActiveProtocol**: Aktualny protokół połączenia : SCARD_PROTOCOL_T0, SCARD_PROTOCOL_T1
  Wartość Zwracana: Jeżeli funkcja zakończy się z sukcesem zwracany jest kod : SCARD_S_SUCCESS, w przeciwnym wypadku funkcja napotkała błąd. Wartość błedu można sprawdzić przy pomocy strony microsoftu [1]

- SCardTransmit:

  [**IN**] **SCARDHANDLE ... hCard**: Wartość referencji, pozyskana z SCardConnect.
  [**IN**] **LPCSCARD_IO_REQUEST ... pioSendPci**: Wskaźnik do struktury nagłówka protokołu dla instrukcji. Ten bufor ma format struktury SCARD_IO_REQUEST, po której następują określone informacje o sterowaniu protokołem (PCI).

  W przypadku protokołów T=0, T=1 i Raw struktura PCI jest stała. Podsystem karty inteligentnej dostarcza globalną strukturę T=0, T=1 lub Raw PCI, do której można się odwołać, używając odpowiednio sym-

boli SCARD_PCI_T0, SCARD_PCI_T1 i SCARD_PCI_RAW.
[IN] LPCBYTE ... pbSendBuffer: Wskaźnik na strukturę przekazywaną do karty, dla t0 Wykorzystana jest ta struktura danych
[IN] DWORD ... cbSendLength: Wartośc długości bufora wysyłanego na kartę, podana w bajtach
[IN,OUT] LPSCARD_IO_REQUEST ... pioRecvPci: Wskaźnik do struktury nagłówka protokołu dla instrukcji, po którym następuje bufor, w którym można odbierać wszelkie zwrócone informacje o sterowaniu protokołem (PCI) specyficzne dla używanego protokołu. Ten parametr może być NULL, jeśli nie zwrócono PCI.
[OUT] LPBYTE ... pbRecvBuffer: Bufor otrzymany spowrotem z karty, w postaci ciągu bajtów.
[IN,OUT] LPDWORD ... pcbRecvLength: Podaje długość parametru pbRecvBuffer w bajtach i odbiera rzeczywistą liczbę bajtów odebranych z karty inteligentnej.

Literatura

[1] Microsoft Error Codes

[2] Omnikey 5321 documentation

[3] Microsoft Winscard Documentation