POLITECHNIKA WROCŁAWSKA

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Urządzenia peryferyjne

Czytnik kart chipowych

Laboratorium 3

Termin zajęć: Środa TP, 17:30

Autorzy:

Daria Jeżowska, 252731

Prowadzący zajęcia:

dr inż. Jarosław Mierzwa

Wstęp

Karta chipowa to uniwersalny nośnik danych w postaci karty wykonanej z plastiku z umieszczonym na niej (lub wewnątrz niej) jednym lub kilkoma układami scalonymi (chipami), które pozwalają na ochronę procesu logowania użytkownika, kontrolę dostępu i zawartych na niej danych. Może być odczytywana za pomocą urządzeń automatycznych, np. przy zawieraniu i rozliczaniu transakcji finansowych oraz w kasach cyfrowych. Karty chipowe mają skuteczną ochronę danych – odczytanie danych z karty wymaga bardziej skompilowanych urządzeń oraz można na niej szybko i sprawnie modyfikować dane. Mają one także wiele zastosowań – są używane w telefonii, bankowości, jako legitymacje studenckie, do kontroli dostępu do pomieszczeń i zasobów, a w przeszłości jako karty umożliwiające korzystanie z budek telefonicznych. W naszym przypadku będziemy korzystać z karty chipowej mającej zastosowanie jako karta *SIM*.

Podczas pracy z SC poznaliśmy skrót ATR, oznacza *Answer To Reset*. Wykorzystuje się to do komunikacji z kartą *Smart Card* - ATR przekazuje informacje o parametrach komunikacji zaproponowanych przez kartę oraz o stanie karty i jej charakterze. Poznaliśmy także APDU – strukturę danych w protokole komunikacji między czytnikiem, a kartą elektroniczną. Przy APDU rozróżniamy dwa typy – polecenia APDU, które składa się z nagłówka (4 bity na pola CLA, INS, P1, P2) i opcjonalnej liczby danych(od 0 dl 255 bajtów), oraz odpowiedź APDU – 2 bajty statusowe (od 0 do 255 bajtów z danymi).

A command APDU has the following general format:

CLA	INS	P1	P2	P3	Data

The response APDU has the following general format:

Data	SW1	SW2

CLA reprezentuje klasę rozkazów, gdzie w naszym przypadku będzie to *A0*, co oznacza aplikację GSM
INS określa konkretną instrukcję do wykonania na karcie
P1-P3 to parametry instrukcji
SW1 i *SW2* to kody statusu odpowiedzi

Zadania

1. Odczytać dane z kart (lista kontaktów oraz SMS'ów) za pomocą gotowej aplikacji
2. Napisać aplikację (może być wersja konsolowa), która:
   - wyświetli podłączone urządzenia-czytnika do komputera
   - umożliwi wybór czytnika
   - umożliwi wysłanie komendy (w formacie HEX) do czytnika i wyświetli odpowiedź w formacie HEX oraz znakowo
   - odczyta listę kontaktów (książkę telefoniczną)
   - odczyta listę SMS'ów Ostatnie trzy pozycje mają mieć formę menu działającego w kółko (w przypadku aplikacji konsolowej)

Aplikacja

Aplikacja została stworzona jako aplikacja konsolowa w języku C++, było to większe wyzwanie niż pisanie jak dotąd w C#. Użyta do tego została biblioteka *Winscard.*

Kod

Poniższy kod zgodnie z przykładem ze strony Microsoft Docs wywołujemy funkcję *SCardEstablishContext*, która zwraca SCARD_S_SUCCESS jeśli połączenie nie zawiedzie, w przeciwnym wypadku zwróci `error code`

```
printf("SCardEstablishContext : ");
rv = SCardEstablishContext(SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext);

if (rv != SCARD_S_SUCCESS)
{
	printf("failed\n");
	return -1;
}
else printf("success\n");
```

`SCardListReaders` pobiera listę czytników

```
printf("SCardListReaders : ");
rv = SCardListReaders(hContext, mszGroups, mszReaders, &dwReaders);


if (rv != SCARD_S_SUCCESS)
{
	SCardReleaseContext(hContext);
	free(mszReaders);
	printf("failed\n");
	return -1;
}
else printf("success\n");
```

Wypisujemy listy czytników i następnie wczytujemy dane od użytkownika, aby sczytać dane z odpowiedniego czytnika.

```
p = 0;
for (i = 0; i < dwReaders - 1; ++i)
{
	iReaders[++p] = i;
	printf("Reader %02d: %s\n", p, &mszReaders[i]);
	while (mszReaders[++i] != '\0');
}

do
{
	printf("Select reader : ");
	scanf_s("%d", &iReader);
} while (iReader > p || iReader <= 0);
```

Nawiązanie połączenia z kartą przy pomocy funkcji SCardConnect – w przypadku niepowodzenia wyświetlamy *failed,* a w przypadku powodzenia zwracamy *success.*

```
printf("SCardConnect : ");
rv = SCardConnect(hContext, &mszReaders[iReaders[iReader]],
	SCARD_SHARE_SHARED, SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
	&hCard, &dwPref);

if (rv != SCARD_S_SUCCESS)
{
	SCardReleaseContext(hContext);
	free(mszReaders);
	printf("failed\n");
	return -1;
}
else printf("success\n");
```

Rozpoczynanie transakcji z kartą, jednocześnie uniemożliwia to korzystanie z karty innym programom i funkcjom. Umożliwia to nam funkcja SCardBeginTransaction.

```
printf("SCardBeginTransaction : ");
	rv = SCardBeginTransaction(hCard);
	if (rv != SCARD_S_SUCCESS)
	{
		SCardDisconnect(hCard, SCARD_RESET_CARD);
		SCardReleaseContext(hContext);
		printf("failed\n");
		free(mszReaders);
		return -1;
	}
	else                                                    printf("success\n");
```

Przesyłanie danych odbywa się za pomocą funkcji SCardTransmit. W jest zawarte polecenie APDU (w tym CLA, INS, P1, P2, P3). 0XA0 oznacza, że korzystamy z aplikacji GSM, 0xA4 (INS), 0x00 (P1), 0x00 (P2), 0x02 (P3) oznaczają komendę *SELECT.*

```
	BYTE SELECT_TELECOM[] = { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x7F, 0x10 };
	printf("SCardTransmit : ");
	dwRespLen = 30;
	rv = SCardTransmit(hCard, SCARD_PCI_T0, SELECT_TELECOM,
		7, NULL, pbResp1, &dwRespLen);

	if (rv != SCARD_S_SUCCESS)
	{
		SCardDisconnect(hCard, SCARD_RESET_CARD);
		SCardReleaseContext(hContext);
		printf("failed\n");
		free(mszReaders);
		return -1;
	}
	else printf("success\n");
	printf("Response APDU : ");

	for (i = 0; i < dwRespLen; i++)
	{
		printf("%02X ", pbResp1[i]);
	}
	printf("\n");
```

Czytanie SMSÓW umożliwia nam funkcja, której używaliśmy także w kawałku kodu wyżej, `SCardTransmit`. Odpowiednie wartości komendy APDU umożliwiają nam odczytanie pierwszego sms'a z karty SIM. Jak w przykładzie wyżej została wybrana funkcja SELECT (`0xA0, 0xA4, 0x00, 0x00, 0x02`), natomiast `0x6F, 0x3C` umożliwiają nam wybór pliku, z którego mamy odczytać dane.

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

GSM file type identifiers (first byte).

```
BYTE SELECT_SMS[] = { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x6F, 0x3C };
	printf("SCardTransmit : ");
	dwRespLen = 30;
	rv = SCardTransmit(hCard, SCARD_PCI_T0, SELECT_SMS,
		7, NULL, pbResp3, &dwRespLen);

	if (rv != SCARD_S_SUCCESS)
	{
		SCardDisconnect(hCard, SCARD_RESET_CARD);
		SCardReleaseContext(hContext);
		printf("failed\n");
		free(mszReaders);
		return -1;
	}
	else printf("success\n");
	printf("Response APDU : ");

	for (i = 0; i < dwRespLen; i++)
	{
		printf("%02X ", pbResp3[i]);
	}
	printf("\n");
```

Wnioski

Program działał prawidłowo – nawiązywał połączenie z czytnikiem i z karty odczytywał SMSy w formie zakodowanej, lecz nie udało nam się stworzyć funkcji, która by je odszyfrowywała. Niestety na zajęciach z pośpiechu nie zdążyliśmy zrobić zrzutów ekranu naszej aplikacji. Została ona jednak przedstawiona prowadzącemu i uzyskaliśmy pozytywną ocenę za wykonanie programu. Nie było to najprostsze zadanie biorąc pod uwagę ilość różnych funkcji oraz poleceń i odpowiedzi APDU z którymi musieliśmy się zapoznać przed pisaniem kodu. To laboratorium pokazuje, jak skomplikowana i niekoniecznie intuicyjna jest obsługa kart SIM z których korzystamy na co dzień nie zastanawiając się nad mechanizmami, które zachodzą.