STUDIA INFORMATICA 2001
Volume 22 Number 1 (43)

Adam ZIĘBIŃSKI
Politechnika Śląska, Instytut Informatyki

PROGRAMOWALNE KARTY PROCESOROWE

Streszczenie. Programowalne karty procesorowe umożliwiają szybką implementację specjalizowanych aplikacji. Stosowanie ich w dziedzinach wymagających szczególnej ochrony danych wymaga jednak wykorzystania wysokiej jakości zabezpieczeń. Możliwości takie udostępniają wysokiej klasy algorytmy kryptograficzne, które obecnie można bezpośrednio wykorzystywać do projektowania złożonych systemów zabezpieczeń.

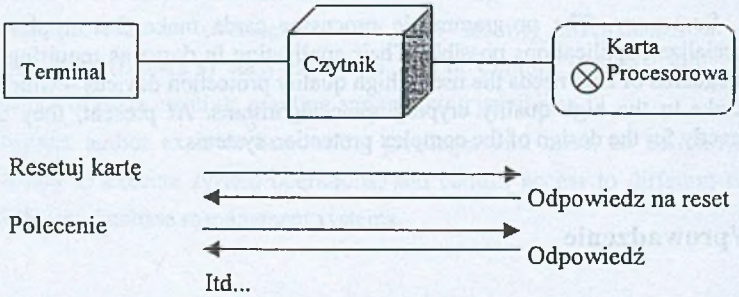
THE PROGRAMMABLE PROCESSOR CARDS

Summary. The programmable processor cards make fast implementation of specialized applications possible. Their application in domains requiring a particular safeguards of data needs the use of high quality protection devices – which is possible thanks to the high quality cryptographic algorithms. At present, they can be used directly for the design of the complex protection systems.

1. Wprowadzenie

Karty inteligentne stosowane są w coraz większej liczbie dziedzin, poczynając od systemów identyfikacyjnych, aż do szyfrowanych systemów baz danych. Wpłynęło to na zwiększenie ich produkcji i znaczne obniżenie cen. Jednak koszty realizacji specjalizowanych aplikacji pozostawały wysokie, czego powodem było drogie środowisko projektowania. Ponadto koszty powiększał długi czas oczekiwania na finalny produkt, co wynikało z programowania systemu operacyjnego karty na poziomie asemblera, sprzętowego zapisu do pamięci EEPROM, czy też implementacji funkcji zabezpieczeń i algorytmów kryptograficznych przez użytkownika. Była to więc podstawowa bariera ograniczająca szybkie wprowadzenie kart inteligentnych do powszechnego użytku.

Nowe możliwości związane z projektowaniem oprogramowania daje zastosowanie języka wyższego poziomu i wykorzystanie narzędzi umożliwiających tworzenie własnych aplikacji w tempie dużo szybszym niż do tej pory. Możliwości takie udostępniają programowalne karty procesorowe, których sercem jest interpreter P-Kodu, pełniący funkcję języka niezależnego od maszyny. Po napisaniu programu w języku JAVA lub Basic wykonywana jest kompilacja do P-Kodu. Następnie wynik kompilacji przesyłany jest do karty, w której wykonuje go interpreter. Przykładem takiego rozwiązania jest system BasicCard firmy ZeitControl, która dostarcza zintegrowany pakiet programowy. Umożliwia on tworzenie oprogramowania w Basicu lub Javie, kompilację do P-Kodu i zapis w EEPROM-ie. Dodatkowo system przechowuje w pamięci EEPROM stałe dane użytkownika oraz pełni funkcję dysku, na co pozwala katalogowy system plików. Natomiast w pamięci RAM znajdują się bieżące dane oraz stos P-Kodu. System umożliwia symulację karty i testowanie stworzonego dla niej oprogramowania zarówno na komputerze PC, jak i bezpośrednio na karcie. Pakiet zawiera również funkcjonalny debugger, który umożliwia równoczesne uruchomienie programu terminalowego i programu karty. Procesor zabezpiecza dostęp do pamięci sprzętowo i programowo, a wysoki poziom bezpieczeństwa zapewniają zaimplementowane algorytmy. Do komunikacji ze światem zewnętrznym karta inteligentna używa dwukierunkowego kontaktu wejścia-wyjścia. Komunikacja ta odbywa się z prędkością 9600 bodów, zgodnie z protokołem polecenie-odpowiedź (rys. 1).



Rys. 1. Protokół polecenie-odpowiedź
Fig. 1. Command-response protocol

Włożenie karty do czytnika rozpoczyna sesję. W czasie wymiany danych stroną pasywną jest karta procesorowa. Po przesłaniu odpowiedzi na reset, otrzymuje jedynie polecenia wydawane z terminala. Po wysłaniu odpowiedzi na polecenie znowu pasywnie oczekuje kolejnego polecenia. Zagadnienia związane z protokołem wymiany danych są niewidoczne dla programisty w Basicu. Należy zdefiniować komendę w karcie i zaprogramować jako zwykłą procedurę. Następnie można wywołać tę komendę z wnętrza programu Basic działającego na PC, przy czym komenda ta jest wywoływana, jakby była normalną

procedurą. System operacyjny BasicCard OS zajmuje się całością komunikacji, jak również umożliwia szyfrowanie i odszyfrowanie poleceń i odpowiedzi.

Firma ZeitControl udostępnia dwa typy kart Compact BasicCard i Enhanced BasicCard.

Compact BasicCard [8] zawiera w sobie 9K kodu w ROM, 1K – EEPROM oraz 256 bajtów RAM-u. System operacyjny wymaga 71 bajtów RAM-u i 35 bajtów EEPROM-u dla swoich potrzeb. Kod zapisany w ROM-ie zawiera:

- pełną implementację protokołu komunikacyjnego T=1 [1],
- ekspedytor poleceń [2],
- wbudowane polecenia umożliwiające programowanie EEPROM-u, szyfrowanie itd.,
- wirtualną maszynę do wykonywania ZC P-Kodu,
- kod do automatycznego szyfrowania i rozszyfrowywania poleceń i odpowiedzi, używający algorytmu Shrinking Generator.

Natomiast Enhanced BasicCard zawiera: 17K kodu w ROM; do 16K EEPROM-u i 256 bajtów RAM-u. System operacyjny wymaga 107 bajtów RAM-u i 405 bajtów EEPROM-u na swój użytek. Oprócz składników wymienionych powyżej dla Compact BasicCard pamięć ROM Enhanced BasicCard zawiera:

- kod umożliwiający szyfrowanie i odszyfrowanie poleceń i odpowiedzi przy użyciu algorytmu DES, (Data Encryption Standard) – pojedynczego lub potrójnego,
- katalogowy system plików (podobny do DOS-owego),
- zmiennoprzecinkową arytmetykę zgodną z normami IEEE.

Dodatkowo dostępne są następujące biblioteki EEPROM dla Enhanced BasicCard:

- EC–161: 161-bit Elliptic Curve Cryptography (proponowany standard IEEE - P1363),
- SHA–1: bezpieczny algorytm mieszania (Secure Hash Algorithm),
- IDEA: algorytm szyfrowania (International Data Encryption Algorithm).

2. Struktura programu BasicCard

Programy dla BasicCard napisane są w języku ZC-Basic, który jest proceduralnym językiem programowania umożliwiającym obsługę środowiska kart procesorowych. Program taki składa się z kodu inicjującego, po którym następują definicje procedur. Programy dla kart Enhanced BasicCard mogą zawierać dodatkowo sekcje definicji plików.

Pierwszy blok kodu nie zawarty w definicji procedur to kod inicjujący. Jest on wykonywany, kiedy pierwsza zdefiniowana przez użytkownika komenda jest wywoływana z terminala. Kod inicjujący nie jest wymagany, ale może być użyteczny na przykład do sprawdzenia, czy karta nie została zablokowana przez wydawcę lub też czy żądane pliki i katalogi są dostępne na danej karcie.

ZC-Basic posiada trzy typy procedur: podprogramy, funkcje oraz polecenia. Każda z procedur jest samodzielna – nie są dozwolone zagnieżdżone definicje procedur. Polecenia GoTo i GoSub mogą przekazywać sterowanie tylko w obrębie bieżącej procedury. Podprogramy są blokami kodu, które mogą być wywoływane z innych procedur, funkcje są to podprogramy zwracające wartość. Polecenie natomiast jest mechanizmem charakterystycznym dla ZC-Basic – dzięki niemu program terminalowy komunikuje się z programem karty BasicCard. Zgodnie z dokumentem standaryzującym ISO/IEC 7816-4: Interindustry commands for interchange każde polecenie ma przypisany unikalny, dwubajtowy identyfikator (ten dwubajtowy identyfikator to CLA oraz INS - Class oraz Instruction). Musi on pojawić się pomiędzy słowem kluczowym Command i nazwą polecenia. Oto przykład (&H jest prefiksem szesnastkowym):

```
Command &H80 &H10 GetCustomerName (Name$)
Name$ = CustomerName$
End Command
```

Kiedy BasicCard odbiera polecenie z terminala z CLA=&H80 i INS = &H10, system operacyjny karty automatycznie wykonuje polecenie GetCustomerName. Polecenie jest definiowane tak jak podprogram, ale wywoływane jak funkcja. System operacyjny BasicCard wypełnia wartość powrotną, która jest przekazywana z powrotem do programu terminala. Ta wartość składa się z dwóch bajtów stanu SW1 i SW2 zdefiniowanych w ISO/IEC 7816-4. Powinna ona zostać zawsze sprawdzona, przykładowo karta mogła zostać usunięta z czytnika lub czytnik mógł zostać pozbawiony zasilania. Jeśli SW1=&H90 i SW2=&H00, lub jeśli SW1=&H61, wówczas polecenie zostało wykonane prawidłowo. W przeciwnym przypadku wystąpiły problemy uniemożliwiające poprawne wykonanie polecenia. Te dwa bajty stanu są dostępne jako predefiniowane zmienne w karcie BasicCard, możliwe jest więc zdefiniowanie własnych kodów błędów.

Enhanced BasicCard zawiera system plików podobny do DOS-owego, z katalogami zorganizowanymi w strukturę drzewiastą. Istnieje kilka sposobów dostępu do plików i katalogów. Wewnątrz karty pliki mogą być tworzone, odczytywane jak również zapisywane tak samo jak pod DOS'em czy Windows'em. Są również specjalne wyrażenia służące ustalaniu praw dostępu do plików i katalogów, aby ograniczyć do nich dostęp z poziomu programów terminalowych. Prawa dostępu mogą zależeć od kluczy kryptograficznych, haseł użytkowników itd. Z poziomu programu terminalowego Enhanced BasicCard wygląda jak dyskietka, ze specjalną nazwą napędu "@:". W Enhanced BasicCard można przechowywać dane stałe w plikach, natomiast dla Compact BasicCard dane te muszą być przechowywane jako dane Eeprom np.:

```
Eeprom Balance As Long : Rem deklaracja stałej w pamięci Eeprom
```

Ciągi znaków (string) jak i zmienne tablicowe mogą być również zapamiętywane w EEPROM-ie. Zapisywanie do pamięci EEPROM może zająć do 6 ms, więc istnieje

możliwość utraty zasilania przez kartę w czasie zapisywania danych. Enhanced BasicCard automatycznie zapamiętuje wszystkie żądania zapisu do EEPROM-u, aby umożliwić odzyskanie danych w przypadku utraty zasilania. Karta Compact BasicCard nie posiada mechanizmu odzyskiwania danych, więc po awarii dane w EEPROM-ie mogą być niespójne. Z tego powodu w Compact BasicCard należy powielić dane EEPROM na czas ich modyfikacji, aby chronić je przed przypadkową utratą zasilania przez kartę podczas operacji zapisu do pamięci EEPROM.

3. Terminal

Język ZC-Basic został zaprojektowany z myślą o BasicCard. Ale może również działać na komputerze PC, z czytnikiem kart przyłączonym do portu szeregowego lub bez czytnika. Można więc napisać dowolny program w tym języku. Program ZC-Basic, który działa na PC, jest programem terminalowym. Kompilator WZCBASIC.EXE może tworzyć pliki wykonywalne i pliki obrazu z kodu źródłowego programu. Tworzy on standardowe pliki wykonywalne o rozszerzeniu .EXE, które będą działały jak programy w środowisku DOS czy w oknie DOS-owym pod Windows'em. Programy te nie mogą komunikować się z symulatorem karty BasicCard, jeśli więc wywołują komendy BasicCard, wówczas karta ta musi być obecna w czytniku. Ponieważ programy te nie mają możliwości samomodyfikacji, nie mogą wywołać wyrażeń takich, jak Write Eeprom. Parametry linii poleceń przekazywane do pliku wykonywalnego mogą być dostępne z poziomu języka ZC-Basic w predefiniowanej tablicy Param$ (1 do nParams). Dla większej elastyczności w czasie tworzenia oprogramowania, kompilator może tworzyć pliki obrazu (ang. Image File) z rozszerzeniem IMG z kodu źródłowego programu terminalowego. Interpreter P-Kodu WZCDOS może wówczas uruchomić taki program razem z programem karty (uruchomionym w karcie lub jej symulatorze). Debugger dla Windows - ZCDD - pracuje z plikami ZC Debug Files (z rozszerzeniem DBG), są to pliki obrazu z włączonymi w nie informacjami dla debuggera.

Program terminalowy składa się z procedury głównej oraz z definicji procedur. Komendy BasicCard są deklarowane w sekcji deklaracji poleceń, po czym mogą być wywoływane jak funkcje. Program terminalowy jest wykonywany przez interpreter P-Kodu WZCDOS.EXE, który może uruchomić program karty w symulatorze jak również komunikować się z rzeczywistą kartą za pomocą czytnika.

Procedura główna zaczyna się pierwszym wyrażeniem znajdującym się poza definicjami procedur i kończy się na początku definicji następnej procedury lub trwa do końca pliku źródłowego. Program terminalowy rozpoczyna wykonanie od pierwszego wyrażenia w procedurze głównej i działa dopóty, dopóki nie znajdzie jej końca lub do polecenia Exit.

Definicje procedur w programie terminalowym składają się z funkcji i podprogramów tak jak w zwykłym programie Basic. Każda z procedur jest samodzielna – nie są dopuszczane zagnieżdżone definicje procedur, a wyrażenia GoTo i GoSub mogą przekazywać sterowanie tylko w obrębie danej procedury. Zanim będzie możliwe wywołanie polecenia BasicCard, musi zostać ono zadeklarowane, tak aby kompilator ZC-Basic znał dwa bajty identyfikatora i typ parametrów komendy. Poza tym deklaracja polecenia wygląda jak deklaracja zwykłego podprogramu, np.:

```
Declare Command &H80 &H10 GetCustomerName (Name$)
```

Wywołanie jest takie jak wywoływanie funkcji.

```
Status = GetCustomerName (Name$)
If Status <> &H9000 And (Status And &HFF00) <> &H6100 Then
Print "GetCustomerName: Status = &H"; Hex$ (Status)
GoTo Retry
End If
```

Zawsze powinno nastąpić sprawdzenie wartości powrotu – nawet gdy polecenie nie wnosi błędu, błąd może wyniknąć z problemów w czasie komunikacji – usunięcie karty z czytnika w czasie odczytu. Można użyć predefiniowanych zmiennych SW1, SW2 oraz SW1SW2, które przechowują bajty stanu ostatnio wywołanego polecenia:

```
Call GetCustomerName (Name$)
If SW1SW2 <> &H9000 And SW1 <> &H61 Then
Print "GetCustomerName: Status = &H"; Hex$ (SW1SW2)
GoTo Retry
End If
```

Plik COMMANDS.DEF definiuje kody stanu w postaci deklaracji wyrażeń stałych (Const), więc można odwołać się do &H9000 i &H61 jako swCommandOK i sw1LeWarning odpowiednio, oczywiście wcześniej należy włączyć ten plik do programu. Alternatywnie można wywołać podprogram CheckSW1SW2(), który jest zdefiniowany w pliku COMMERR.DEF. Jeśli wystąpi błąd komunikacji, podprogram ten wypisuje powód błędu i kończy się.

4. Algorytmy szyfrujące w kartach BasicCard

4.1. Elliptic Curve

Algorytm szyfrujący Elliptic Curve wykorzystywany jest jedynie w kartach Enhanced BasicCard. W kartach tych zaimplementowano dwie odmiany tego algorytmu: Elliptic Curve i Elliptic Curve Fast Signature. Algorytm ten może być łatwo wykorzystywany do wymiany klucza. Obie strony przesyłające wspólnie tworzą klucz przesyłu na podstawie publicznie

znanej krzywej eliptycznej E i punktu na tej krzywej P. Pierwsza strona generuje losową liczbę k1, wylicza punkt k1*P i wysyła wynik do drugiej strony, analogicznie postępuje druga strona. Wspólny klucz przesyłu powstaje poprzez wymnożenie otrzymanego punktu i własnej wygenerowanej liczby, czyli K=k1*k2*P. Podsłuchujący, aby uzyskać ten klucz, musi go wyznaczyć na podstawie P, k1*P, k2*P, ale bez znajomości samych k1 i k2. Problem wyliczenia tego zadania jest nazywany „problemem Diffiego-Hellmana dla krzywych eliptycznych" i jest on porównywalny do złamania „problemu logarytmu dyskretnego".

Kolejnym zastosowaniem krzywych eliptycznych może być przesyłanie szyfrowanych wiadomości. Załóżmy, że uzgodniliśmy pewien sposób zanurzenia zbioru jednostek wiadomości M w krzywą eliptyczną E i klucz przesyłania K=k1*k2*P. Wysyłający wyliczając losowy punkt L może przesłać wiadomość zaszyfrowaną w parze punktów (L*P,M+L*(k1*P)). Aby zdeszyfrować taką wiadomość należy wymnożyć pierwszy punkt przez k1, a następnie wynik odjąć od drugiego punktu w parze.

Algorytm dla BasicCard napisany został w C++ dla terminala, natomiast dla karty w asemblerze mikroprocesora 8051. Wszystkie parametry i klucze używane są dokładnie w ten sam sposób jak w Basicu. Biblioteka zawierająca definicje funkcji Elliptic Curve nie została zawarta w pamięci ROM, dlatego aby korzystać z tego algorytmu należy tę bibliotekę załadować do pamięci EEPROM dyrektywą Basic #include EC-161.DEF.

W kartach Enhanced BasicCard dostępne są następujące operacje związane z szyfrowaniem Elliptic Curve:

- Generowanie kluczy prywatnych i publicznych.
- Generowanie kluczy sesyjnych.
- Generowanie elektronicznego podpisu.
- Weryfikacja elektronicznego podpisu (tylko na terminalu).

4.2. Standard szyfrowania danych DES

Algorytm DES [3] wykorzystywany jest jedynie w kartach Enhanced BasicCard. W kartach tych zaimplementowano dwie wersje tego algorytmu: pojedynczy DES i potrójny DES. Użycie tego algorytmu zaimplementowanego w BasicCard polega na dokonaniu następujących operacji:

1. Wywołanie programu KEYGEN do generacji pliku klucza, zawierającego klucze kryptograficzne.
2. Dołączenie wygenerowanego pliku klucza do programu terminala i karty.
3. Dołączenie pliku COMMAND.DEF do programu terminala.

4. W programie terminala włączenie i wyłączenie automatycznego szyfrowania steruje się za pomocą procedur:

Call StartEncryption(P1=Algorytm,P2=numer_klucza)

Call StopEncryption()

Wiadomość P, która ma być przesyłana, zostaje podzielona na 8-bajtowe bloki Pi. Jeśli po podziale ostatni blok Pn jest niepełny, to blok ten zostaje uzupełniany do końca wartościami równymi zeru. Zaszyfrowaną wiadomość C uzyskuje się poprzez złożenie bloków Ci uzyskanych dzięki następującym obliczeniom:

Ci=EK(Ci-1 Xor Pi) dla całkowitego i ∈ <1,n>

Przy pierwszym uruchomieniu komendy StartEncryption zainicjalizowany musi być wektor C0. Dokonuje się to poprzez złożenie dwóch losowych liczb 4-bajtowych wygenerowanych przez system terminala (RA) i system karty (RB). Zainicjowany wektor ma następującą postać: pierwsze dwa bajty pochodzą od RA, następne cztery od RB, ostatnie dwa od RA.

4.3. SG-LFSR

Opisywany algorytm SG-LFSR (ang. Shrinking Generator – Linear Feedback Shift Register with Cyclic Redundancy Check) jest zaimplementowany, w odróżnieniu od poprzednio opisywanych algorytmów, jedynie w kartach Compact BasicCard. W kartach tych zaimplementowane są dwie wersje tego algorytmu: SG-LFSR i SG-LFSR z CRC (umożliwia sprawdzanie autentyczności przesyłanych danych).

Sposób użycia algorytmu SG-LFSR zaimplementowanego w BasicCard jest niemalże identyczny jak algorytmu DES. Dla celów bezpieczeństwa takie elementy, jak: początkowe „polynomials" PolyA, PolyS oraz klucz szyfrujący K muszą być znane jedynie stronom komunikującym się. Przy wywołaniu polecenia StartEncryption inicjowane są wartości dwóch rejestrów A i S. Do inicjacji wymagany jest klucz K oraz dwie losowo wygenerowane liczby: RA – 4-bajtowa liczba wygenerowana przez terminal i RB – 4-bajtowa liczba wygenerowana przez kartę. Dla ułatwienia podzielmy każdą z tych liczby na liczby dwubajtowe: RA(0):RA(1), RB(0):RB(1) oraz K(0): K(1): K(2): K(3). Na podstawie tych danych można wyliczyć wartości A i S:

A(0)=(RA(0) Xor K(0)) And &H7FFF

A(1)=RB(0) Xor K(1)

S(0)=RB(1) Xor K(2)

S(1)=RB(1) Xor K(3)

Szyfrowanie zaczyna się zaraz po wywołaniu polecenia StartEncryption i kończy się po wywołaniu polecenia EndEncryption.

5. Wnioski

Programowalne karty procesorowe są zaawansowanym technologicznie produktem umożliwiającym szybką implementację specjalizowanych aplikacji. Pozwala na to programowanie z wykorzystaniem języków wysokiego poziomu. Cechy te sprawiają, że doskonale nadają się do dydaktyki. Wprowadzenie do powszechnego użytku systemów z wykorzystaniem kart jest często ograniczone jakością stosowanych zabezpieczeń. Umieszczenie w kartach na stałe (w pamięci ROM) lub poprzez wykorzystanie bibliotek systemowych złożonych funkcji zabezpieczeń pozwala na szybką ich implementację. Natomiast przeźroczysta komunikacja i możliwość stosowania systemu plików pozwalają na realizacje coraz bardziej złożonych funkcji i aplikacji. W efekcie karta inteligentna może stanowić autonomiczny system komputerowy o wysokim standardzie bezpieczeństwa.

LITERATURA

1. ISO/IEC 7816-3: Electronic signals and transmission protocols.
2. ISO/IEC 7816-4;
3. Schneier B.: Kryptografia dla praktyków. Protokoły, algorytmy źródłowe w języku C. Wydawnictwa Naukowo-Techniczne, Warszawa 1995.
4. Schneier B.: Ochrona poczty elektronicznej. Jak chronić prywatność korespondencji w sieci Internet? Wydawnictwa Naukowo-Techniczne, Warszawa 1996.
5. M. Molski, M. Glinkowska „Karta elektroniczna bezpieczny nośnik informacji"
6. J.L.Zoreda, J.M.Oton „Smart Card".
7. M.Hendry "Smart Card security and Applications".
8. www.ZeitContol.de
9. www.BasicCard.com

Recenzent: Dr inż. Andrzej Białas

Wpłynęło do Redakcji 31 marca 2001 r.

Abstract

The characteristics of the programmable processor cards are presented in the article. The P-Code interpreter make available a new possibility. It is like a language independent on a computer. The program coded in Java or Basic was compiled to P-Code and sent to the card. The processor card is the passive partner in the communication. It receives only a command from the terminal (Fig.1). The BasicCard operating system manage all communications and make available the cipher and decipher of commands and answers. The complex functions of protection are placed in the card or available in the system library. It enables fast implementation. The characteristics of the programmable processor cards can be used directly for the design of the complex protection systems.