Capitolul 4

Smart Carduri

4.1 Prezentare generală

Un smartcard este un sistem de calcul portabil (sau detaşabil) utilizat în efectuarea unor tranzacţii, cu competenţe sporite pentru a prezenta sau autentifica o identitate; el încorporează o paletă largă de subansamble care asigură securitatea fizică şi logică.

O serie de aplicaţii utilizează smartcarduri în sisteme de securitate, pentru stocarea de secrete sau utilizarea de algoritmi proprii (care nu influenţează sistemul de securitate).

4.1.1 Istoric

Smartcardurile au fost inventate şi patentate în anii 70. Jürgen Dethloff (Germania), Arimura (Japonia) şi Roland Moreno (Franţa) îşi dispută paternitatea. În mod cert însă, prima utilizare pe scară largă a cardurilor a fost *Télécarte* – cartelă telefonică prepaid folosită în Franţa începând cu 1983.

Roland Moreno este cel care a patentat conceptul de "*card cu memorie*" în 1974. În 1977, Michel Ugon (Honeywell Bull) a inventat primul microprocesor de smartcard. Tot Honeywell Bull patentează în 1978 $SPOM$ (*Self Programmable One-chip Microcomputer*) care defineşte arhitectura unui cip de smartcard. După 3 ani, Motorola produce primul micro-cip "CP8" bazat pe acest patent.

Utilizarea cardurilor explodează în anii 90, odată cu introducerea în telefonia mobilă GSM a smartcardurilor bazate pe SIM.

4.1.2 Clasificări

În general cardurile (cartelele) sunt de trei tipuri:

- *Carduri (cartele) magnetice*:
  Cardurile magnetice au înglobată o unitate de memorie relativ mică (maxim 100

octeţi) sub forma unei benzi magnetice şi sunt destinate unei singure aplicaţii. Cele mai cunoscute sunt cartelele telefonice.

- *Carduri cu procesor (smartcarduri)*:
  Conţin un mini-procesor, ceeea ce le dă posibitatea de a rula mai multe aplicaţii prezente pe acelaşi card. Avantajul unui smartcard faţă de un card magnetic este acela că poate să proceseze datele din memorie.

  Ca o comparaţie între aceste două tipuri, cardurile cu bandă magnetică nu au capacităţi de procesare, sunt semnificativ mai nesigure decât smartcard-urile dar costă mai puţin. În schimb, cititoarele de cartele magnetice sunt mult mai scumpe decât terminalele pentru smartcard-uri, fiind şi mai puţin sigure.

- *Carduri hibride*:
  Sunt o combinaţie între celelalte două tipuri.
  Din 1982, băncile franceze au folosit carduri cu chip şi bandă magnetică pentru operaţii de credit, care permit băncilor să migreze către smartcard-uri. S-au obţinut astfel avantajele unor cartele mai sigure, rezervând în acelaşi timp o perioadă rezonabilă modernizării infrastructurii bazate pe cartele magnetice.

  în prezent există carduri hibride care conţin simultan micro-cip, bandă magnetică, cod de bare, memorie optică, poză şi tabelă de semnătură.

Din punct de vedere al modului de legătură cu exteriorul, smartcardurile se împart în:

- *Cardurile cu contacte.*
  Transferul de date se face prin intermediul unui cititor de card, cu care se stabileşte un contact direct.

- *Cardurile fără contacte* conţin – pe lângă un cip cu micro-procesor – un dispozitiv de emisie-recepţie radio cu antenă. Pot funcţiona numai la distanţe foarte mici de cititorul de carduri.

  Aceste carduri sunt de obicei mai costisitoare şi sunt dedicate unor aplicaţii specifice cum ar fi pentru transporturi sau pentru accesul în clădiri.

Cipul înglobat într-un smartcard este un microprocesor (MCU). Deci un MCU este un sistem de calcul miniatural integrat pe o singură piesă de silicon. Resursele sale sunt separate complet de interfeţele cu utilizatorul sau alt sistem de calcul; lipsesc componente I/O cum ar fi tastatură, monitor, discuri, drivere etc.

Cipul de pe un smartcard este un MCU "securizat": deoarece lucrează doar sub controlul software-ul din ROM, programele sale nu sunt accesibile sub nici o formă unui utilizator extern. Prin construcţia sa, un MCU securizat este capabil să prevină accesul neautorizat la CPU (unitatea centrală de prelucrare), memorie, magistrale sau date stocate/prelucrate în interior.

4.2 Hardware-ul unui smartcard

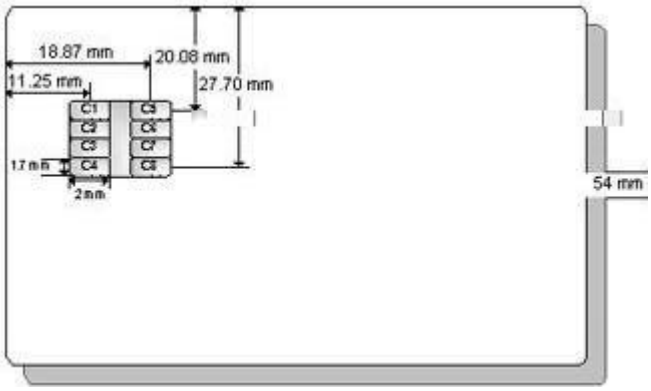
4.2.1 Forma unui smartcard

Un card are o formă dreptunghiulară, ale cărei dimensiuni sunt stabilite de standardul ISO/IEC 7816: 85.60×53.98 mm, iar grosimea este 0.76 mm[1]. De exemplu:

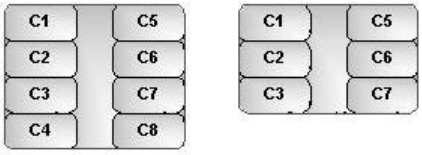
Pe una din feţe este inserat un micro-cip, de o anumită formă. De exemplu:

Oricare ar fi cip-ul inserat, el este poziţionat conform standardului de mai sus, astfel:

[1]Pentru cardurile fără contacte există un alt standard: ISO 14443.



Detaliind, cele opt zone de contact ale unui cip sunt aranjate:



cu semnificaţia şi funcţionarea următoare:

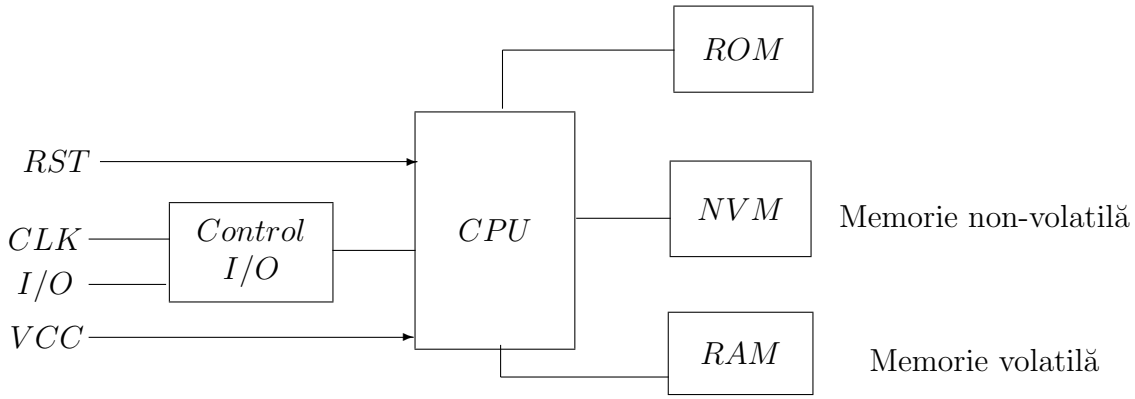
Poziţie	Abreviere	Funcţie
$C1$	VCC	Asigură voltajul
$C2$	RST	Reset
$C3$	CLK	Frecvenţă de ceas
$C4$	RFU	Rezervată pentru utilizare ulterioară
$C5$	GND	Înpământare
$C6$	VPP	Legătură pentru voltaj extern
$C7$	I/O	Legătură serială input/output
$C8$	RFU	Rezervată pentru utilizare ulterioară

Contactul VPP este o reminiscenţă; a fost utilizat pentru a suplimenta voltajul la $EEPROM$ în operaţiile de programare şi ştergere. Din raţiuni de securitate, în prezent se face apel la el extrem de rar.

Contactul VCC asigură un voltaj standard de 5 volţi $\pm 10\%$, dar se prevede construirea unei tehnologii operaţionale la un standard de 3 volţi. Motivul este asigurarea unei compatibilităţi între piaţa smartcardurilor şi cea a telefoniei mobile. Un telefon mobil lucrează pe o configuraţie de 3 volţi, iar cardurile au rămas singura componentă care solicită telefonului mobil să aibă un convertor de încărcare.

4.2.2 Sistemul de calcul al unui smartcard

Sistemul de calcul dintr-un smartcard este format dintr-un singur circuit integrat, care include unitatea centrală de procesare (CPU), memoria şi liniile de Intrare/Ieşire (I/O).



4.2.3 Memoria

Există trei tipuri de memorie într-un smart-card:

1. *Memoria read-only* (ROM).
   Aici este memorat sistemul de operare, precum şi diferite rutine pentru comunicare, pentru administrarea sistemului de fişiere de pe card împreună cu rutine pentru criptare sau funcţii matematice speciale. Datele şi codurile programelor sunt plasate în memoria ROM în faza de fabricare a cardului.

2. *Memoria non-volatilă* (NVM).
   Memoria nonvolatilă (*NonVolatile Memory* –NVM) este o memorie de calculator care poate reţine informaţie chiar dacă sistemul de calcul este debranşat. Exemple de memorii nonvolatile: memorii read-only, memorii flash, tipuri de componente de stocare magnetică (cum sunt hard-discurile, discurile floppy, banda magnetică), discuri optice, cartele perforate. Memoria non-volatilă aflată pe aproape cardurile este $EEPROM$ (*Electrically Erasable Programmable Read-Only Memory*). Aici sunt depuse diverse date variabile (numerele de cont, numărul de puncte de loialitate sau sume de bani) care trebuie reţinute de smartcard.

   O memorie NVM poate fi scrisă şi citită de aplicaţii, dar nu poate fi folosită ca o memorie RAM.

3. O cantitate – relativ minusculă – de memorie cu acces aleator (RAM). Ea este resursa cea mai preţioasă din punctul de vedere al dezvoltatorului de software. Mai mult, RAM - ul este folosit nu numai de aplicaţiile programatorului ci şi de restul rutinelor utilitare.

4.2.4 Unitatea centrală de procesare (CPU)

Pentru micro-controlerele pe 8 biţi, CPU-ul unui smartcard foloseşte de obicei instrucţiuni din seturile Motorola 6805 sau Intel 8051.

Aceste seturi conţin instrucţiunile uzuale de manipulare a memoriei şi regiştrilor, de adresare şi de intrare/ieşire. Unitatea centrală de prelucrare execută instrucţiunile la o

rată de 400.000 de instrucţiuni pe secundă (400 KIP), deşi viteze de până la 1 milion de instrucţiuni pe secundă (1 MIP) devin disponibile la cipurile de ultimă generaţie.

Evoluţia smartcard-urilor este extrem de rapidă. Astfel, dimensiunea memoriei creşte iar arhitecturile procesoarelor se mută de pe configuraţii pe 8 biţi, pe unele de 16 sau chiar 32 biţi.

4.2.5 Ieşirile/intrările unui smartcard

Canalul de intrare/ieşire al unui smartcard este unul serial uni-direcţional. Hardware-ul smartcardului poate suporta transfer de date de până la 115.200 biţi/secundă, dar cititoarele de carduri comunică de obicei la viteze mult mai mici.
Comunicarea dintre terminal şi card se bazează pe o relaţie master (terminal) - slave (smartcard): terminalul trimite comenzi către card şi aşteaptă răspunsul. Smartcard-ul nu trimite niciodată date către terminal – decât sub forma unui răspuns la o comandă dată de terminal.

4.3 Software-ul pentru smartcard-uri

Există două tipuri fundamentale de software pentru smartcard-uri:

1. Software-ul pentru terminal (gazdă); aplicaţiile înglobate în el rulează pe un computer conectat la un smartcard.

2. Software-ul instalat pe card.

Este uneori util să clasificăm software-ul de smartcard în:

- *Software aplicaţie*:
  Foloseşte capacităţile computaţionale şi de memorare ale smartcard-ului, ca şi cum ar fi cele ale unui computer oarecare; el face abstracţie de integritatea şi securitatea datelor prezente pe smartcard.

- *Software sistem*:
  Foloseşte explicit aceste cerinţe şi – deci – poate contribui la sporirea integrităţii şi securităţii datelor.

4.3.1 Comparaţie între software-ul gazdă şi software-ul de card

Cea mai mare parte a softului pentru smartcarduri constă în programele scrise pentru terminale. Aceste programe accesează smartcard-urile existente integrându-le în sisteme mai mari. Software-ul gazdă include de obicei:

- aplicaţii pentru utilizatori,

- programe la nivel de sistem care acceptă smartcard-uri specifice, folosite pentru a ajuta aplicaţiile pentru utilizatori.
- aplicaţii utilitare necesare administrării infrastructurii smartcard-urilor.

Programele gazdă sunt scrise de obicei în limbaje de programare de nivel înalt aflate pe computere personale şi staţii de lucru: C, C++, Java, FORTRAN; ele sunt legate de biblioteci de funcţii şi drivere pentru a accesa cititoarele de smartcard-uri şi cartele introduse în ele.

Software-ul de card este scris de obicei în limbaje sigure precum Java sau limbaje apropiate de cod - maşină (exemplu: Forth sau limbaje de asamblare).

Aplicaţiile gazdă substituie uneori smartcard-ul cu o implementare alternativă având aceeaşi funcţionalitate. Astfel, o cheie de criptare sau o informaţie medicală este păstrată pe un smartcard şi nu într-un fişier pe hard disk sau într-o bază de date pe un server.

Software-ul gazdă manipulează capacităţile unice de calcul şi stocare ale cardului trimiţând comenzi acestuia şi obţinând rezultatele generate.

Software-ul de card este clasificat în:

- sisteme de operare,
- utilitare,
- aplicaţii software.

Aplicaţiile de card sunt de obicei folosite pentru a personaliza un smartcard existent pentru o utilizare dedicată şi pentru a realiza un transfer de funcţionalitate de la aplicaţia gazdă la card. Acest lucru poate fi făcut din considerente de eficienţă sau de securitate (pentru a proteja o anumită parte din sistem).

Software-ul de card este scris în general în limbaje de nivel inferior, având ca ţintă un anumit tip de card şi este folosit pentru a extinde sau înlocui funcţii de bază ale smartcard-ului.

Aplicaţiile gazdă şi cele de card diferă în orientare şi înfăţisare. Software-ul de card se concentrează pe conţinutul unui anumit card. El oferă servicii computaţionale pentru unele aplicaţii care accesează acest conţinut şi pe care îl protejează de celelalte aplicaţii care ar putea încerca să îl acceseze incorect.

Software-ul gazdă se poate folosi însă de diferite carduri. El este de obicei compatibil cu multe terminale şi – posibil – mai mulţi emiţători de carduri precum şi diferite tipuri de cartele.

Software-ul de card implementează particularităţile de procesare, precum şi cerinţele de securitate ale unei cartele specifice. De exemplu, un program care rulează pe un card poate să nu dezvăluie numărul de cont memorat decât dacă i se prezintă un PIN corect sau poate aplica semnătura digitală folosind o cheie privată memorată pe card. Software-ul care rulează pe un smartcard oferă acces autorizat securizat la datele memorate pe

card. El este "conştient" doar de conţinutul unui anume smartcard şi de entităţi (precum oamenii, computerele, terminalele, consolele de jocuri etc.) care încearcă să-l acceseze.

Software-ul gazdă conectează smartcard-urile şi posesorii lor la sisteme mai mari. De exemplu, software-ul dintr-un bancomat foloseşte smartcard-urile introduse de clienţii băncii pentru a-i identifica şi a-i conecta la conturile bancare pe care le deţin.
Software-ul gazdă este compatibil cu multe smartcard-uri şi îşi va adapta răspunsul în funcţie de smartcard-ul prezent.

Spre deosebire de software-ul obişnuit, care depinde de suportul serviciilor din contextul care-l înconjoară, software-ul de card porneşte de la premiza că se află într-un context ostil, în care nu poate avea încredere.

Până când nu li se prezintă o dovadă convingătoare a contrariului, smartcard-urile nu au încredere în terminalele unde sunt introduse, şi reciproc: terminalele nu au încredere în cardurile prezente. Un program de smartcard are încredere numai în el însuşi. Orice se află înafara programului trebuie să se autentifice înainte ca programul să interacţioneze cu el.

4.4 Funcţiile unui smartcard

Conform celor prezentate anterior, un smartcard asigură cinci operaţii de bază:

1. Introduce date;
2. Scoate date;
3. Citeşte date din memoria nonvolatilă (NVM);
4. Scrie sau şterge date din NVM;
5. Calculează o funcţie criptografică.

Deşi toate aceste operaţii au legături cu componenta de securitate a cardului, gradul major de risc aparţine operaţiilor (2) şi (4). Operaţia (2) oferă în exterior date şi rezultate, iar (4) modifică conţinutul NVM.
De exemplu:

- Nici o informaţie referitoare la cheile secrete folosite de microprocesor nu trebuie oferită mediului exterior.
- Unele date din memoria nonvolatilă pot da drept de acces la anumite resurse; deci trebuie luate măsuri speciale de protecţie înainte de ştergerea sau scrierea datelor.
- Rezultatul unui calcul criptografic poate fi un cuvânt de control livrat exteriorului (exemplu: pentru a decodifica un semnal TV); deci şi aici trebuie asigurate măsuri de protecţie înainte de efectuarea calculului şi externarea rezultatului.

Cardul trebuie să fie sigur că operaţiile sunt efectuate de proprietarul de card, sau că anumite comenzi primite au fost formulate de emiţătorul de card. O serie de mecanisme şi tehnici – de la PIN sau MAC până la scheme sofisticate de semnătură electronică – sunt utilizate de card pentru a controla aceste lucruri.
Aceste mecanisme sunt bazate atât pe tehnici criptografice cât folosind şi tehnici empirice.

Cardul poate reacţiona la unele tipuri de încercări de fraudă. De exemplu, trei PIN-uri greşite vor duce la blocarea cardului

Creşterea dimensiunii memoriei şi a vitezei de calcul asociate cu proceduri sofisticate de securitate fac posibilă elaborarea unui set sporit de mecanisme care să asigure securitatea logică[2] a smartcardului.

Exemplul 4.1. *Smartcardurile emise de unele unităţi bancare au memoria nonvolatilă segmentată (fizic) în mai multe zone:*
- O zonă deschisă, accesibilă fără restricţii;
- O zonă protejată unde citirea este liberă, dar pentru scriere este necesară cunoaşterea unei parole;
- O zonă confidenţială, unde accesul la citire este permis doar cu parolă;
- O zonă secretă care conţine PIN-uri, parole şi chei criptografice.

Cardurile actuale au memoria nonvolatilă organizată diferit la nivel logic faţă de nivelul fizic. În acest fel, zonele sunt mult mai puţin vizibile: locaţia fizică exactă a unui fişier în NVM poate diferi pe diverse carduri şi deci este mult mai greu de depistat.

4.5 Sistemul de operare pentru smartcard

Sistemul de operare al cipului de pe smartcard (numit frecvent COS sau – uneori – $Mask$) este o secvenţă de instrucţiuni stocată în memoria ROM a smartcard-ului. Similar sistemelor de operare aflate pe PC-uri (cum ar fi Unix sau Windows), instrucţiunile COS nu depind de o aplicaţie anume, dar sunt frecvent folosite de majoritatea aplicaţiilor.

Sistemele de operare de pe un cip sunt împărţite în:

1. COS pentru scopuri generale; conţin un set de instrucţiuni generale care acoperă majoritatea aplicaţiilor.

2. COS-uri dedicate; conţin comenzi speciale pentru unele aplicaţii – eventual însăşi aplicaţia. Este cazul cardurilor special concepute în acest scop (mai ales în zona de comerţ electronic).

Funcţiile de bază ale unui COS (comune tuturor smartcard-urilor) cuprind:

[2]Spre deosebire de securitatea fizică, securitatea logică a unui card are ca scop protejarea informaţiei aflate în microcomputerul de pe card.

- Administrarea inter-schimbului de informaţii dintre card şi exterior, în primul rând în termeni de protocol de comunicare.

- Administrarea fişierelor şi datelor aflate în memorie.

- Controlul accesului la informaţii şi funcţii (de exemplu, selecţie de fişiere, citirea, scrierea şi actualizarea datelor).

- Administrarea securităţii cardului şi a algoritmilor criptografici.

- Menţinerea fiabilităţii, în special în cazul consistenţei datelor, secvenţelor de întrerupere şi revenirea după o eroare.

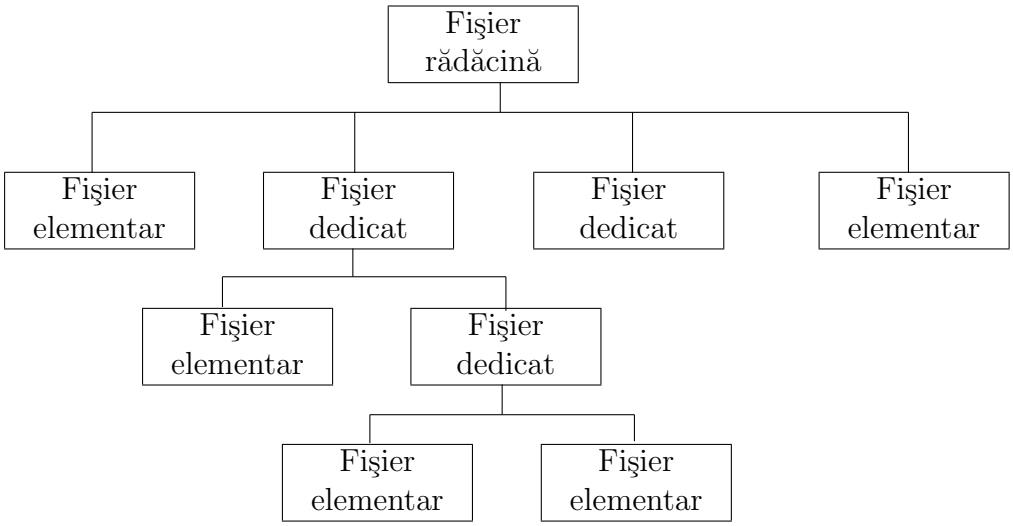
- Administrarea diverselor faze din ciclul de viaţă al cardului (fabricarea micro-cipului, personalizarea, perioada activă, distrugerea).

După cum s-a precizat, relaţia de bază dintre terminal şi smartcard este de tip master - slave: terminalul trimite o comandă smartcard-ului, acesta o execută, întoarce rezultatul către terminal şi apoi aşteaptă altă comandă.

În plus, standardele referitoare la smartcarduri (ISO 7816 şi CEN 726) descriu o gamă largă de comenzi pe care smartcardurile le pot implementa. Majoritatea fabricanţilor de smartcard-uri oferă produse cu sisteme de operare ce implementează aceste comenzi standard (sau o parte din ele) împreună cu extensii şi îmbunătăţiri specifice producătorului.

4.5.1 Sistemul de fişiere al smartcardurilor

Majoritatea sistemelor de operare pentru smartcarduri suportă un sistem redus de fişiere, bazat pe standardul ISO 7816. Deoarece smartcardurile nu au periferice, un sistem de fişiere smartcard are o singură rădăcină şi o ierarhie bazată pe directoare, în care fişierele pot avea nume alfanumerice lungi, denumiri scurte numerice şi nume relative.



Sistemele de operare de pe smartcard suportă setul uzual de operaţii cu fişiere: crearea, ştergerea, citirea, scrierea şi actualizarea.
În plus, sunt suportate operaţii pe tipuri particulare de fişiere.

Exemplul 4.2. *Fişierele liniare sunt formate dintr-o serie de înregistrări de lungime fixă, care pot fi accesate după numărul înregistrării, sau citite secvenţial folosind operaţiile de citit următoarea sau precedenta înregistrare. Mai mult, unele sisteme de operare suportă o formă limitată de căutare în astfel de fişiere.*

Fişierele ciclice sunt fişiere liniare care ciclează înapoi la prima înregistrare atunci când după ultima înregistrare este citită sau scrisă încă o inregistrare.

Fişierele "portofel" formează un tip de fişier specific (folosit în aplicaţii de comerţ electronic), suportat de sistemele de operare smartcard.
Aceste fişiere sunt ciclice, fiecare înregistrare conţinând informaţie despre o tranzacţie a "portofelului" electronic.

În fine, fişierele transparente sunt blocuri nediferenţiate de memorie, care pot fi structurate de aplicaţii în funcţie de necesităţi.

Fiecărui fişier de pe card îi este asociată o listă pentru controlul accesului. Această listă reţine operaţiile pe care le poate efectua fiecare din identităţile prezente pe card.

De exemplu, entitatea A poate avea dreptul de a citi un anumit fişier, dar nu şi de a-l actualiza, în timp ce B poate să citească, scrie şi chiar modifica drepturile de acces pentru acel fişier.

4.5.2 Structurile de date ale protocolului de aplicaţie ($APDU$)

Protocolul de bază în stabilirea unei linii de comunicaţie între smartcard şi terminal este pachetul *"Application Protocol Data Units"* ($APDU$).

Un $APDU$ poate fi considerat un pachet de date care conţine o cerere completă sau un răspuns complet dat de un smartcard.

$ISO\ 7816-4$ defineşte două tipuri de APDU - uri:

- *De comandă*: trimise de aplicaţia gazdă.

  Formatul unui astfel de fişier este:

CLA	INS	$P1$	$P2$	Lc	Date opţionale	Le

  unde:

  1. CLA: Identifică clasa instrucţiunii; de exemplu, instrucţiunea poate fi standard ori particulară.
  2. INS: Octet care determină comanda respectivă.
  3. $P1$, $P2$: Doi octeţi folosiţi pentru a transfera parametri specifici comenzii.
  4. Lc: Octet care codifică lungimea datelor opţionale transmise card-ului cu acest $APDU$.
  5. Date opţionale: şirul de octeţi – cu lungimea Lc – care poate conţine datele propriu-zise ce vor fi procesate de card.
  6. Le: Octet care specifică lungimea datelor ce se aşteaptă a fi returnate de către $APDU$-ul de răspuns următor.
     Dacă $Le = 0x00$, gazda se aşteaptă ca toate datele disponibile să fie trimise în răspunsul la comandă.

- *De răspuns*: trimise înapoi de smartcard, ca reacţie la comenzi.

  Formatul $APDU$-urilor de răspuns este:

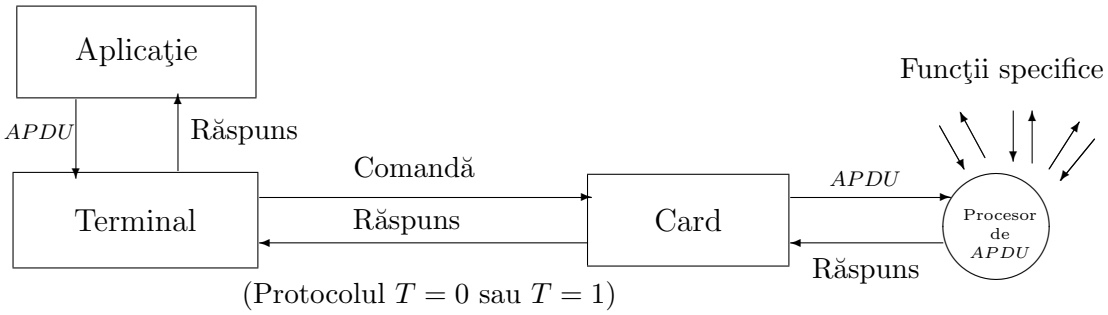
Date opţionale	$SW1$	$SW2$

  unde:

  1. Date opţionale: un şir de Le octeţi – dacă acest $APDU$ este răspunsul unui $APDU$ comandă ce avea octetul Le setat pe o valoare nenulă.
     Altfel numărul octeţilor este variabil.
  2. $SW1$, $SW2$: Doi octeţi de stare, care conţin informaţii de stare definite conform ISO 7816-4.

Aplicaţia software foloseşte un protocol bazat pe $APDU$-uri pentru schimbul de control şi informaţie între card şi cititorul de carduri. Aceste $APDU$-uri sunt schimbate folosindu-se protocoalele la nivel de conexiune $T = 0$ şi $T = 1$. Protocolul $T = 0$ este orientat pe octeţi, iar $T = 1$ este orientat pe blocuri de octeţi. Mai există şi alte protocoale alternative pentru smartcardurile fără contacte (bazate pe tehnologia $JavaCard$) cum ar fi $T = USB$ sau $T = RF$.

O componentă software de pe card interpretează aceste $APDU$-uri şi execută operaţiile specificate; această arhitectură este ilustrată mai jos:



4.6 Entităţi implicate în construcţia/utilizarea unui smartcard

Există mai multe entităţi implicate potenţial într-un sistem bazat pe smartcard.

1. **Proprietarul cardului** este persoana care posedă cardul în mod uzual; acesta decide dacă şi când să-l folosească.

   Proprietarul poate controla datele de pe card oferite de sistem, dar extrem de rar are şi controlul protocoalelor, softului sau hardului încorporate în card.

2. **Proprietarul datelor** este componenta care deţine controlul datelor aflate pe card.

   În anumite cazuri, el coincide cu proprietarul cardului.
   Pentru cardurile de plăţi, proprietarul datelor este cel emite plăţile cash; această diferenţiere între cele două componente deschide anumite oportunităţi de atac.

3. **Terminalul** este componenta care asigură interacţiunea smartcardului cu exteriorul.

   Terminalul controlează toate intrările şi ieşirile spre şi dinspre smartcard: tastatura prin care datele sunt încorporate în smartcard şi monitorul pe care este vizualizat smartcardul.
   La un card telefonic, această componentă este chiar aparatul telefonic. Pentru un card de plăţi, terminalul este ATM-ul la care este făcută conexiunea.

4. **Emiţătorul de card** este componenta care emite smartcardul.

   El controlează sistemul de operare al smartcardului, precum şi datele stocate iniţial. Exemple: pentru un card telefonic, această componentă este compania de telefoane. Pentru un card de serviciu, emiţătorul este instituţia la care este angajat proprietarul cardului.

   În anumite cazuri, emiţătorul doar emite cardul, după care dispare din sistem; alteori, el rămâne implicat în funcţionarea cardului.

   În unele carduri multifuncţionale, emiţătorul de card nu are nici o legătură cu aplicaţiile care rulează pe smartcard; în altele, el are controlul tuturor aplicaţiilor.

Din punct de vedere al securităţii este mai simplu de considerat că emiţătorul de card, fabricantul şi programatorul de software formează o singură componentă. În realitate, acest lucru se întâmplă foarte rar.

5. **Fabricantul de card**: este componenta care realizează smartcardul.

   Evident, aceasta este o imagine simplificată; în realitate, fabricantul de card poate deţine sau nu facilităţile de fabricare ale cipului încorporat.
   De exemplu, există funcţii subcontractate care folosesc componente externe sistemului (cum ar fi compilatoare $VHDL$); deşi nu sunt în proprietatea fabricantului de card, ele sunt introduse pe card.

6. **Programatorul de software** este componenta care realizează programele rezidente pe smartcard.

   Evident, şi aici problemele sunt mult simplificate: în realitate sunt mai mulţi programatori care scriu codul pentru compilatoare, programe utilitare, componente de securitate etc.

4.7 Exemple de partajare a securităţii sistemelor de carduri

Să trecem în revistă câteva din cele mai utilizate sisteme de smartcard, privite din punct de vedere al controlului efectuat de componentele sale.

- **Cardul de plăţi** (Digital Stored Value Card): Este un card folosit pentru înlocuirea modului de plată cash. Exemplu: Both Mondex, VisaCash.

  Aici comerciantul este proprietarul terminalului, iar cumpărătorul este proprietarul de card. Atât propietarul datelor cât şi emiţătorul de card sunt instituţia financiară (de obicei o bancă) care sprijină sistemul.

- **Cardul digital** (Digital Check Card): Este similar cardului de plăţi, cu diferenţa că proprietarul cardului este acelaşi cu proprietarul datelor.

- **Cartelă telefonică Prepaid**: Este un card de plăţi cu utilizare specificată. Proprietarul cardului este utilizatorul. Terminalul, datele şi emiţătorul cardului sunt asigurate de o singură entitate: compania de telefoane.

- **Cartelă telefonică**: În acest sistem, smartcardul stochează doar un număr de cont – un pointer la o bază de date. Proprietarul cardului şi al datelor este utilizatorul, iar cel care emite cardul, deţinând şi terminalul, este compania de telefoane.

- **Card de acces** (Access Token): În această aplicaţie, smartcardul stochează o cheie care este utilizată într-un login sau protocol de autentificare. Pentru o instituţie, proprietarul cardului este angajatul, iar proprietarul datelor, al terminalului şi emiţătorul de card este compania.

  În cazul unui card de acces multifuncţional, proprietarul cardului şi al datelor poate fi eventual una şi aceeaşi persoană, în timp ce terminalul este deţinut de un organism comercial, iar emiţătorul de card – de o instituţie financiară.

- **Card de navigare Web** (Web Browsing Card): Utilizatorul poate folosi cardul pe propriul PC pentru a cumpăra diverse produse oferite prin Internet.
  Este o variantă a cardului de plăţi, cu diferenţa că cel care face cumpărături deţine atât cardul cât şi terminalul.

- **Card de certificare** (Digital Credential Device): Aici smartcardul stochează certificate digitale sau alte credenţiale necesare autentificării faţă de parteneri. Proprietarul cardului este şi proprietarul datelor. Terminalul este deţinut de către partener (într-o aplicaţie in-store de exemplu), sau tot de proprietarul cardului (care efectuează căutarea pe WWW). Emiţătorul cardului este unitatea de certificare (CA) care emite credenţialele, sau o companie care gestionează aceste certificate.

- **Cartelă de stocare a cheilor** (Key Storage Card): În această aplicaţie, utilizatorul stochează diverse chei publice (eventual verificate) pe un smartcard, pentru a le păstra într-un mediu mai sigur decât propriul PC.

  El este proprietarul cardului, al datelor precum şi al terminalului.

4.8 Componente criptografice

4.8.1 Exemple de componente criptografice integrate pe un smartcard

Smartcard-urile conţin suficient de multe componente criptografice pe care se dezvoltă aplicaţii şi protocoale de securitate cunoscute.

- *Semnătura electronică:*

  Cele mai folosite protocoale de semnătură sunt bazate pe RSA; pe ele sunt implementate chei de 512, 768 sau 1024 biţi. Timpul necesar unei semnături este de obicei sub o secundă.

  De cele mai multe ori fişierul $EEPROM$ care conţine cheia privată este conceput astfel încât informaţiile sensibile despre cheie să nu părăsească niciodată cip-ul. Nici cititorul de carduri nu poate accesa acest fişier. Folosirea cheii private este protejată

de PIN-ul utilizatorului, astfel încât posesia cardului nu implică şi posibilitatea semnării digitale.
Unele carduri au implementat padding-ul $PKCS\#1$ corespunzător lui RSA.

Desi cardurile au abilitatea de a genera perechi de chei RSA, acest proces poate dura foarte mult. Timpii uzuali necesari pentru generearea unei perechi de chei RSA de 1024 biţi poate ajunge pâna la 3 minute, ceea ce depăşeşte specificaţiile ISO pentru timpul afectat unei comunicaţii; din acest motiv este necesară uneori existenţa unui hardware specializat sau a unui co-procesor dedicat.
De asemenea, calitatea perechii de chei nu este foarte mare. Lipsa puterii de procesare implică implementarea de generatori relativ slabi de numere pseudo-aleatoare.

Algoritmul de Semnătură Digitală (DSA) este implementat mai puţin frecvent decât RSA. Atunci când este disponibil, el foloseşte chei de numai 512 biţi.

- *Algoritmi de criptare*:

  DES şi $3DES$ sunt întâlnite frecvent în smartcard-urile performante. De obicei există şi opţiunea de a fi folosiţi în calculul unui MAC. Totuşi, datorită faptului că interfaţa serială a smartcard-ului are o lăţime mică de bandă, criptarea cu un sistem simetric a unei cantităţi mari de informaţie este foarte lentă.

- *Alte aplicaţii*:

  - Funcţia de portofel electronic este prezentă în multe aplicaţii; ea se bazează de obicei pe criptări cu cheie simetrică, cum ar fi DES sau $3DES$.
  - Algoritmii de dispersie cuprind de obicei $SHA-1$ şi $MD5$; dar – din nou – lăţimea scăzută a benzii seriale de comunicare împiedică folosirea efectivă a acestor funcţii pentru dispersia unui volum mare de informaţii.
  - Generatoarele de numere pseudo-aleatoare (RNG) diferă destul de mult de la un tip de card la altul. Unii fabricanţi implementează un pseudo-RNG în care fiecare card are propria sa iniţializare.
    În acest caz, numerele aleatoare ciclează în funcţie de algoritm şi iniţializare.
    Alte carduri au un RNG bazat pe hardware, care foloseşte unele aspecte fizice ale cipului de siliciu.

    Dacă un card va fi folosit într-un context criptografic sensibil, este recomandat să se cunoască cât mai multe detalii referitoare la RNG.

De asemenea – ca şi în cazul oricărei tehnologii – există aspecte legale care trebuie luate în considerare în lucrul cu smartcard-urile. În mod normal, un smartcard are capacitatea de a rula algoritmi cu licenţă (de exemplu sistemul de criptare RSA). Taxele de licenţă se regăsesc de cele mai multe ori în preţul smartcard-ului.

4.8.2 Aspecte legate de securitate

Unul din conceptele de bază pe care se bazează arhitectura securităţii smartcard-urilor este acela că extragerea de informaţii despre sistemul de operare şi cel de fişiere sunt operaţii extrem de dificile – dacă aceste procese nu sunt controlate de cip împreună cu sistemul de operare al card-ului.

Pentru a realiza acest lucru, în smartcard-urile actuale sunt prezente diferite metode de monitorizare a securităţii hardware.

- O măsură de siguranţă (ireversibilă) dezactivează orice cod de test din $EEPROM$. Ea se poate folosi o singură dată în perioada de viaţă a unui smartcard.
- Pentru a evita clonarea cardurilor, o secvenţă nealterabilă specifică este parcursă obligatoriu la orice utilizare a memoriei ROM.
- Când detectează fluctuaţii de voltaj, temperatură sau frecvenţă de ceas, cardurile sunt concepute să se reseteze automat într-o stare iniţială.
- Operaţia de citire din exterior a memoriei ROM este de obicei dezactivată.

Totuşi, deoarece fiecare dezvoltator de carduri are propria sa schemă de măsuri, este bine să se cerceteze rezultatele unor teste provenite de la mai multe laboratoare independente.

Protocoalele de comunicare ale smartcard-urilor la nivel de comandă pot avea de asemenea inclus un protocol de securitate. Acesta este bazat de obicei pe tehnologii cu cheie simetrică şi oferă smartcard-ului posibilitatea să autentifice terminalul cu care intră în contact direct.

Totuşi, sistemele de criptare şi algoritmii utilizaţi pentru aceste protocoale sunt – în marea lor majoritate – specifice unor anumite aplicaţii şi unor anumite terminale.

Smartcard-urile au abilitatea de a configura mai multe PIN-uri cu scopuri diferite. Aplicaţiile pot configura un PIN pentru supervizor, care să poată debloca PIN-ul unui utilizator – după un număr de încercări eşuate – sau să reiniţializeze cardul.
Alte PIN-uri pot fi configurate pentru a controla accesul la fişiere sensibile sau funcţii ale portofelului electronic.

4.8.3 Codul PIN

Cea mai simplă metodă de identificare a unui utilizator este de a-i ataşa un număr secret, numit *Personal Identification Number*[3] (PIN). Un PIN este un număr de 4 cifre zecimale[4] El este introdus folosind un terminal sau o tastatură de calculator şi direcţionat

[3]Unele surse denumesc acest număr CHV - Cardholder Verification.

[4]Conform standardului $ISO-9546-1$, codul PIN trebuie să conţină între 4 şi 12 caractere alfanumerice (pentru a minimiza efectul unui atac prin forţă brută), lucru imposibil practic deoarece terminalele de card au doar taste numerice, iar codul trebuie să fie uşor de memorat.

spre smartcard. Cardul compară valoarea primită cu cea stocată în interior şi reportează spre terminal rezultatul comparării.

Un PIN poate fi de două feluri: static sau modificabil.

- Un PIN *static* nu poate fi schimbat de utilizator; deci el trebuie memorat. Dacă devine public, utilizatorul trebuie să distrugă cardul şi să obţină un card nou, cu alt PIN.

- Un PIN *modificabil* poate fi modificat la solicitarea utilizatorului sau schimbat de acesta cu un număr mai uşor de memorat. Acest lucru comportă un anumit risc, deoarece utilizatorul va prefera un PIN de forma $'1234'$, sau legat de date personale (data naşterii, căsătoriei etc). Smartcardul nu poate controla alegerea numerelor triviale, deoarece nu are suficientă memorie pentru a construi o tabelă cu ele. În schimb un terminal poate interzice introducerea unor numere de forma $'1234'$, $'9999'$,$'1111'$, $'0000'$ etc.

Există şi chei personale de deblocare (PUK), numite şi *super-*PIN. Ele au mai mult de 4 cifre (6 cifre de obicei) şi sunt folosite pentru resetarea counterului unui smartcard atunci numărul de încercări a fost depăşit. Odată cu utilizarea unui PUK se va defini un nou PIN, deoarece un număr mare de încercări cu PIN-uri greşite este un indiciu – în variantă optimistă – că utilizatorul a uitat propriul PIN.

Alte aplicaţii folosesc PIN-uri "de transport". Smartcardul este personalizat cu un PIN generat aleator, pe care proprietarul de card îl primeşte într-o scrisoare trimisă prin poştă. În momentul folosirii pentru prima oară a cardului, el înlocuieşte acest PIN cu unul propriu.
O metodă similară este PIN*-zero*: cardul este preîncărcat cu un PIN banal, cum ar fi $'0000'$, valoare care este înlocuită obligatoriu la prima utilizare a cardului.

4.9 Criptanaliza smartcardurilor

Nu vom discuta aici metodele de atac legate de softul introdus pe smartcard (atacurile asupra RSA, DES sau asupra $SHA-1$ etc.). Acestea au fost studiate odată cu prezentarea sistemelor respective. Vom trece în revistă o serie de atacuri specifice structurii smartcardului.

4.9.1 Tipuri de atac contra smartcardurilor

Definiţia 4.1. *Un atac este o încercare a uneia sau mai multor părţi – implicate într-o tranzacţie bazată pe smartcard – de a trişa.*

Sunt două tipuri de atac:

- Atacuri venite din partea componentelor sistemului de smartcard. De exemplu: încercări ale proprietarului de card de a înşela terminalul; sau – ale fabricantului de card de a înşela proprietarul de card.

- Atacuri venite dinafara sistemului. Sunt generate de obicei de persoane care fură un card de la proprietar sau care înlocuiesc softul/hardul terminalului.

Datorită numărului mare de părţi implicate într-un sistem de smartcard, apar mai multe clase de atacuri; majoritatea lor sunt specifice smartcardurilor (nu apar în sistemele de calcul obişnuite).

Atacuri ale terminalului contra proprietarului de card sau de date

Este un atac realizat de obicei cu ATM-uri contrafăcute. Când un card este introdus într-un terminal, proprietarul său are încredere că acesta va lucra corect.
Astfel, dacă proprietarul cardului cere să se scadă (pentru o plată) 100 lei, mesajul trimis mai departe de terminal constă în scoaterea din contul proprietarului a sumei de 100 lei, şi nu 1000 lei (de exemplu). Sau, dacă un card trimite spre proprietar un mesaj de tipul "Contul conţine 100 lei", terminalul trebuie să transmită corect acest mesaj pe ecran.
În contextul unui singur terminal, proprietarul cardului nu poate detecta acest tip de atac.

Mecanismele de eliminare a acestui atac se bazează general pe faptul că pentru un interval scurt de timp, doar terminalul are acces la card. De aceea, multe softuri de pe card limitează suma ce poate fi retrasă la o cerere, sau într-o perioadă de timp. Astfel, în general nu se poate efectua retragerea de pe card a unei sume mai mari de 2.000 lei în 24 ore. De remarcat că aceste mecanisme nu semnalează nici o anomalie a terminalului; încearcă doar limitarea unor eventuale pagube.
Există şi mecanisme de prevenire împotriva acestui tip de atac, care nu au nici o legătură cu schimbul de informaţii card - terminal; sunt sisteme centrale care monitorizează toate cardurile şi terminalele şi semnalează orice comportament care provoacă suspiciuni.

Atacuri ale proprietarului de card contra terminalului

Aceste atacuri folosesc carduri false sau modificate, bazate pe softuri pirat, cu intenţia de a păcăli protocolul de autentificare card - terminal.

De obicei un protocol bun reduce la minimum acest tip de atac. În plus, pot fi utilizate pentru autentificare anumite aspecte fizice ale cardului, greu de imitat (de exemplu hologramele utilizate de sistemele Visa şi Mastercard) şi care sunt controlate separat de terminal.

De remarcat că o semnătură digitală controlată de software nu este sigură, deoarece un card fals poate contraface (adesea) o semnătură, iar terminalul nu are nici o posibilitate de a inspecta interiorul cardului.

Apărarea contra acestui tip de atac constă în definirea unor funcţii care să nu permită accesul proprietarului de card la datele din interiorul cardului.

Atacuri ale proprietarului de card contra proprietarului de date

În multe smartcarduri (mai ales cele folosite în sisteme comerciale), datele stocate pe card trebuie protejate faţă de proprietarul de card. De obicei acestuia nu îi este permis să aibă acces la datele respective.

De exemplu, o cartelă de acces într-o clădire poate conţine o valoare secretă; cunoaşterea ei va permite proprietarului să construiască un acces multiplu (pe mai multe zone ale clădirii). Sau – cunoaşterea unei chei secrete de pe un card de comerţ electronic poate permite proprietarului să efectueze tranzacţii ilegale.

În alte cazuri, se permite cunoaşterea valorii de către proprietarul cardului, dar nu şi modificarea ei. Este cazul unui card de debit: dacă proprietarul poate modifica valoarea, el poate scoate bani pe care nu îi are în cont.

Aceste atacuri au două proprietăţi majore:

- Cardul acţionează ca un perimetru de securitate, limitând strict accesul proprietarului la datele aflate în interiorul cardului.
- Atacatorul are acces la card în condiţii stabilite de el: poate să-l supună oricăror verificări şi experimente doreşte, pentru a avea acces la informaţiile stocate. În particular, poate distruge cardul, pentru a vedea cum este construit.

Există multe tipuri reuşite de atac contra datelor stocate pe card; cea mai mare parte a lotr vor fi prezentate în secţiunile următoare.

Multe astfel de atacuri s-au materializat efectiv contra cardurilor de acces pay-TV şi a cartelelor de telefon. De asemenea, sunt unele încercări reuşite de sustrageri de date din carduri folosite în comerţul electronic.

Atacuri ale proprietarului de card contra emiţătorului de card

Sunt cunoscute mai multe atacuri de acest fel; toate sunt de fapt atacuri care au ca ţintă integritatea şi autenticitatea informaţiilor stocate pe card. Multe din aceste informaţii sunt cunoscute de proprietarul de card; acesta doreşte să le modifice fără acordul emiţătorului de card (şi – implicit – al fabricantului de card).
De exemplu, la o cartelă de telefon prepaid, proprietarul (nu neapărat cel legal) poate încerca să modifice contul, cu scopul de a putea vorbi mai mult cu aceeaşi sumă. În acest caz, securitatea poate fi asigurată adăugând un mecanism de tip provocare/răspuns (utilizat frecvent la protocoalele de comerţ electronic) sau un lanţ de funcţii de dispersie inverse.

Dacă emiţătorul de card preferă să pună pe card biţi care să permită utilizarea sistemului, evident că principalul atac va fi construit pentru a-i afla. În general aceşti biţi

autentifică contul, sau formează un sistem bazat pe o cheie stocată pe card, cheie care nu poate fi extrasă fără a distruge fizic cardul.

Evident, toate aceste precauţii se bazează pe presupunerea (discutabilă) că perimetrul de securitate al cardului este suficient de sigur.

Atacuri ale proprietarului de card contra programatorului de software

Precauţiile ce se pot lua în această situaţie folosesc diverse protocoale preliminare de recunoaştere, bazate pe transformări neinversabile, pentru a avea siguranţa că nu este posibil accesul la softul de pe card. Ele folosesc prezumţia că nu există nici o legătură între proprietarul cardului şi proprietarul softului.

Totuşi, în acest caz, intruşii dovedesc o abilitate deosebită în construcţia unor structuri hard adecvate lansării unor astfel de atacuri, structuri oferite adesea gratis pe Internet.

Atacuri ale proprietarului de terminale contra emiţătorului de carduri

Într-un sistem închis la exterior (cum este cel al cartelelor de telefon prepaid), proprietarul de terminale este de asemenea şi emiţătorul de carduri (compania de telefoane îndeplineşte ambele funcţii).

La sistemele deschise (când cele două entităţi sunt distincte), terminalul controlează toate comunicaţiile între card şi emiţătorul de carduri (aflat în general la capătul sistemului). Deci el poate totdeauna să falsifice mesajele care nu au legătură cu conţinutul smartcardului; de exemplu să refuze înregistrarea tranzacţiei, să lanseze comenzi false etc. De asemenea, terminalul poate omite intenţionat completarea unor paşi dintr-o tranzacţie, facilitând astfel o fraudă sau creind dificultăţi de management emiţătorului de card.
Astfel, dacă nu modifică suma de pe card (scăzând suma cheltuită), terminalul poate genera o pagubă; sau – dacă completează o tranzacţie dar nu oferă serviciul respectiv (de exemplu, scoate banii, dar nu oferă marfa) – crează prejudicii atât emiţătorului cât şi proprietarului de card.

Aceste atacuri nu sunt legate de natura sistemului de utilizare a smartcardurilor; sunt atacuri contra relaţiei dintre proprietarul de terminale şi emiţătorul de carduri. De obicei ele sunt prevenite printr-o monitorizare permanentă la ambele capete a protocoalelor de comunicare.

Atacuri ale emiţătorului de carduri contra proprietarului de card

Cele mai multe sisteme de smartcard se bazează pe supoziţia că emiţătorul de carduri slujeşte interesele proprietarilor de card. Acest lucru nu este totdeauna adevărat şi un emiţător de carduri are posibilitatea de a lansa mai multe tipuri de atac împotriva proprietarilor de carduri.

Majoritatea lor încalcă sub o formă oarecare conceptul de confidenţialitate (privacy). Astfel smartcardurile care efectuează plăţi trebuie să asigure menţinerea proprietăţilor de

anonimitate şi fără legături (unlinkability)[5] a tranzacţiilor. Acestea pot fi compromise de emiţătorul de carduri.

Este posibil ca un sistem să fie oferit cu asigurarea că asigură mai multă confidenţialitate decât are de fapt; proprietarul de card nu poate verifica dacă datele pe care le are nu sunt alterate în interiorul cardului. Aceste posibile atacuri (deşi unele sunt doar slăbiciuni ale sistemului smartcard) se pot efectua fără ca proprietarul de card să fie conştient de existenţa lor, sau – eventual – fără să fie avertizat asupra lor.
Astfel, schimbarea unui sistem efectuată de emiţătorul de carduri este obligatorie pentru proprietarul de card şi acesta nu este consultat sau conştient de eventualele falii de securitate pe care le deschide modificarea.

Majoritatea acestor atacuri pot fi realizate de emiţătorul de carduri doar în colaborare cu fabricantul de carduri şi cu programatorul de software. De aceea este bine ca – din punct de vedere al proprietarului de carduri – aceste entităţi să fie complet distincte.

Atacuri ale fabricantului de carduri contra proprietarului de date

Anumite designuri de fabricare pot avea efecte negative destul de importante privind securitatea datelor de pe card.

Un sistem de operare care permite mai multor utilizatori să ruleze programe pe acelaşi smartcard ridică numeroase probleme de securitate.

Prima problemă este versiunea sistemului de operare – şi deci şi a programelor care se bazează pe el. Opţiunea de a construi sisteme de operare noi, specifice smartcardurilor, nu au convins din punct de vedere al securităţii.

Dar, chiar şi cu un sistem de operare (teoretic) sigur, securitatea interfeţei cu utilizatorul constituie permanent un risc de securitate.

1. Cum poate şti utilizatorul (sau designerul de card) că programul rulează atunci când cardul este inserat într-un terminal ?

2. Cum poate fi el sigur că programul interacţionează cu acel terminal şi nu cu un alt program ?

3. Cum poate un program, care ajunge la concluzia că este compromis, să termine în siguranţă şi să semnaleze utilizatorului că trebuie înlocuit ?

4. Ce atacuri devin posibile atunci când un card anunţă că nu mai lucrează deoarece nu mai este sigur ? Poate să asigure un astfel de card – odată ce acest mesaj este trimis – că memoria sa este într-adevăr distrusă şi nu este sub controlul unui intrus?

Mai puţin evidente sunt implementări intenţionate de generatori slabi de numere pseudo-aleatoare, sau alte module criptografice de calitate inferioară privind securitatea.

[5] A se vedea [1], capitolul dedicat sistemelor electronice de plată.

Fabricantul de carduri are numeroase posibilităţi de a-şi asigura baza unor atacuri ulterioare asupra informaţiilor de pe card. Nici un fabricant de carduri nu a asigurat până acum un sistem de operare sigur, fără vulnerabilităţi (mai mult sau mai puţin evidente). În plus, prin implementarea diverselor protocoale, el poate aranja unele canale subliminale care să permită aflarea cheilor sau a altor date de pe card.

Mai mult, este posibil ca pentru o aplicaţie pe smartcard să existe o altă aplicaţie care să ruleze simultan pe acelaşi smartcard. S-a demonstrat că dacă avem un protocol sigur, se poate crea alt protocol – tot sigur – care să-l spargă pe primul, dacă ambele rulează pe acelaşi echipament şi folosesc aceleaşi chei.

4.9.2 Tehnici invazive de atac

Definiţia 4.2. *Un atac asupra unui smartcard este "invaziv" dacă implică o modificare vizibilă a dispozitivului.*

De obicei tehnicile invazive presupun distrugerea totală a cipului smartcard-ului.

În plus, în timp ce atacurile non-invazive pot fi efectuate prin sustragerea pentru o scurtă perioadă de timp a unui dispozitiv smartcard, atacurile invazive pot necesita ore de muncă specializată în laboratoare, fiind accesibile doar atacatorilor foarte experimentaţi şi cu suport financiar puternic. Există, astfel, o probabilitate neglijabilă ca un atac invaziv să se realizeze fără cunoştinţa utilizatorului (care va realiza mai devreme sau mai târziu că nu mai posedă cardul), ceea ce face ca astfel de atacuri împotriva dispozitivelor de semnătură şi autentificare să nu fie viabile (certificatele pot fi revocate când s-a descoperit pierderea). Dacă însă smartcard-ul este proiectat să stocheze informaţii foarte valoroase, aceste atacuri constituie o problemă importantă.

Toate atacurile invazive încep prin scoaterea cip-ului din suportul de plastic. Este foarte simplu şi nu necesită unelte speciale. Eventual se fac demersuri pentru îndepărtarea stratului de pasivizare (răşina epoxidică) care înfăşoară cip-ul. Acest pas constituie principala diferenţă între atacurile invazive şi cele semi-invazive.

- Dacă cipul este proiectat să efectueze funcţii speciale, există o anumită şansă ca cei ce l-au proiectat să nu fi ţinut cont de principiul lui Kerckhoff[6]. Securitatea prin obscuritate nu a funcţionat niciodată, inclusiv în cazul smartcard-urilor; astfel, componentele cipului pot fi analizate iar proiectul iniţial poate fi descoperit ("reverse engineering").

- *Microsondarea* reprezintă o metodă de atac strâns legată de "reverse engineering"-ul cip-ului, care presupune interacţiunea directă cu componentele acestuia. Prin această tehnică se încalcă prezumţia conform căreia smartcard-ul poate fi accesat

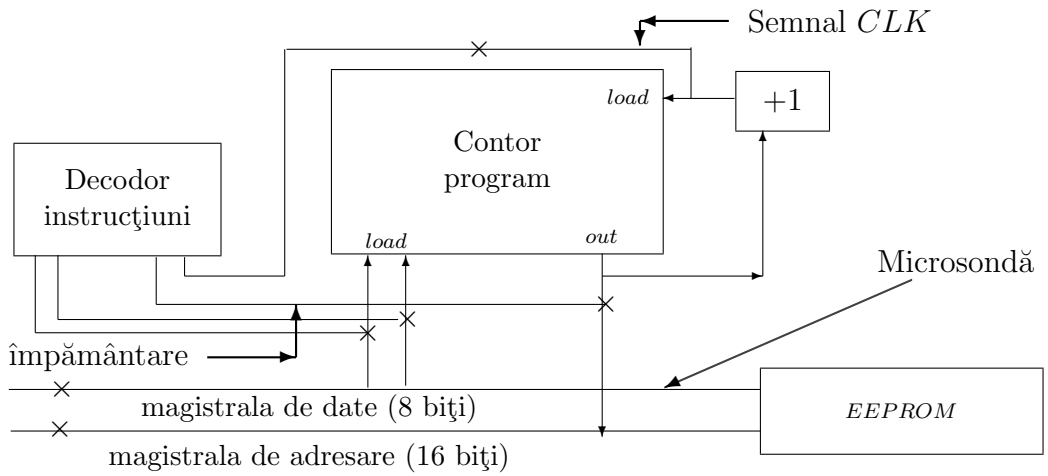
[6]Reamintim, acest principiu spune că puterea unui sistem de criptare şi – prin extensie – a unui dispozitiv de criptare, trebuie să se bazeze numai pe cheie, nu pe structura şi algoritmii sistemului. A se vedea şi [1], pag 11.

numai prin intermediul unui cititor de carduri. De exemplu, cu două microsonde, orice bit din $EEPROM$ poate fi setat sau resetat (aceasta face banală, de exemplu, extragerea unei chei a cărei paritate este o precondiţie pentru algoritm); similar, cu un microscop şi un laser-cutter, orice bit din ROM poate fi modificat.

- Un atac construit de Biham şi Shamir împotriva unei implementări hardware a DES a fost prezentat în 1997; el a folosit un laser-cutter pentru a distruge o poartă din implementarea hardware a unui bloc de criptare cunoscut, anume ultimul bit din registrul ce duce ieşirea unei runde în intrarea rundei următoare. Blocarea bit-ului cel mai nesemnificativ are ca efect blocarea pe 0 a ultimului bit întors de funcţia de rundă. Prin compararea celor mai nesemnificativi 6 biţi din jumătatea stângă şi cea dreaptă, pot fi descoperiţi câţiva biţi din cheie. Folosind 10 texte criptate de un astfel de cip şi cu o tehnică bazată pe criptanaliza diferenţială, s-au obţinut informaţii despre cheia de rundă a rundelor precedente. Se pot afla astfel destule informaţii despre cheie pentru a facilita căutarea. Acesta este de fapt primul atac eficient împotriva sistemelor de criptare bazate pe reţele Feistel, în cazul când nu există nici o informaţie despre textul criptat. Este cazul multor aplicaţii smartcard în care cardul foloseşte tranzacţii succesive pentru a raporta starea internă emitentului.

  Există o masură simplă de protecţie împotriva unui astfel de atac: un cip modificat în felul acesta are proprietatea că funcţiile de criptare şi decriptare nu mai sunt inverse. Aşa că se poate ataşa sistemului o auto-testare simplă, care foloseşte o intrare arbitrară, criptează şi decriptează folosind o cheie arbitrară, iar apoi compară rezultatul cu blocul original.

- Un alt atac tipic presupune deconectarea aproape în întregime a Unităţii Centrale de Procesare (CPU) de magistrala de date, lăsând conectate numai $EEPROM$-ul şi o componentă a CPU care să asigure accesul la citire. Pentru a citi toate celulele de memorie fără ajutorul software-ului card-ului, trebuie folosită o componentă a CPU-ului drept contor de adrese pentru a accesa toate celulele de memorie. Contorul de program este implicit incrementat automat în timpul fiecărui ciclu de instrucţiuni şi folosit la citirea următoarei adrese, ceea ce îl face foarte potrivit pentru rolul de generator de secvenţe de adrese. Totuşi, procesorul trebuie împiedicat să execute instrucţiuni cum ar fi *jump, call* sau *return*, care ar deturna contorul de program de la citirea secvenţială. Mici modificări ale decodorului de instrucţiuni sau ale circuitelor contorului de program (care se pot realiza prin deschiderea unui anumit conector metalic) pot avea efectul dorit.



Odată ce aceste măsuri au fost luate, *Oscar* are nevoie de un singur ac pentru microsondare, sau o sondă electro-optică pentru a citi întregul conţinut al $EEPROM$-ului. Această abordare face programul mult mai uşor de analizat decât în cazul atacurilor pasive, care produc doar o urmărire a execuţiei; de asemenea înlătură dificultăţile mecanice de a avea simultan mai multe sonde inserate pe circuite (cu o lăţime de un micron).

Pentru unele configuraţii ale cipului, acest atac nu funcţionează. De exemplu, cardurile bancare citesc date critice din memorie doar după ce se asigură că se află într-o tranzacţie cu un cititor autorizat. Se pot adăuga contoare auxiliare care să reseteze procesorul dacă nu se execută instrucţiuni *jump, call* sau *return* într-un anumit interval de timp.
Totuşi astfel de dispozitive pot fi dezactivate prin modificări minore ale circuitelor, aşa că de obicei protecţia este incorporată în structura cip-ului; de exemplu prin folosirea unui contor de program care să nu poată acoperi întregul spaţiu de adrese. Un contor de program pe 16 biţi poate fi înlocuit uşor cu o combinaţie dintr-un contor O de deplasare pe 7 biţi şi un registru de segment R pe 16 biţi, astfel încât adresa accesată să fie $S+O$. Contorul de deplasare resetează procesorul dacă ajunge la valoarea maximă. Procesorului îi va fi astfel imposibil să execute mai mult de 127 octeţi de cod-maşină fără *jump*, şi acest lucru nu poate fi schimbat de către un atacator prin mijloace simple. Un post-procesor de cod - maşină poate fi folosit ulterior de programator pentru a insera *jump*-uri la următoarea adresă atunci când ramurile necondiţionate depăşesc lungimea de 127 octeţi.

- Dacă nu mai poate folosi de contorul de program, *Oscar* poate încerca să crească numărul de iteraţii din cicluri software care citesc şiruri de octeţi din memorie, pentru a avea acces la toţi octeţii. Acest deziderat se poate obţine folosind o microsondă care să producă o perturbaţie direct pe magistrala de date. Programatorii care vor să utilizeze contoare pe 16 biţi trebuie să ţină cont de acest lucru. Ca o primă linie de apărare împotriva acestui tip de atac, cele mai multe sisteme de operare pentru smartcard-uri criptează datele importante de pe $EEPROM$; astfel este dificil pentru un intrus să obţină text clar direct din $EEPROM$.

Aceasta nu este de fapt o soluţie, decât dacă criptarea depinde de un secret (cum ar fi PIN-ul introdus de utilizator).

- Proiectanţii de dispozitive bazate pe $EEPROM$ mai au de înfruntat următoarea problemă: ştergerea unei celule de memorie necesită un voltaj relativ ridicat. Dacă *Oscar* poate preveni acest lucru, atunci informaţia va fi imposibil de şters. Primele smartcard-uri primeau voltajul necesar programării printr-o conexiune dedicată cu gazda, folosind contactul $C6$. Acest mod de alimentare extern a dus la atacuri împotriva sistemelor de televiziune cu plată, în care cardurile erau iniţial activate pentru toate canalele, iar acele canale pe care abonatul nu le plătea erau dezactivate prin semnale difuzate. Acoperind contactul cardurilor cu bandă izolantă abonaţii puteau preveni ca aceste semnale să reprogrameze cardul. Ei puteau apoi să-şi anuleze abonamentul fără ca prestatorul să-i poată anula serviciul.

  Cardurile folosite astăzi sunt capabile să genereze cei 12 volţi necesari funcţionării din tensiunea de 5 volţi obţinută de la cititor. Aceasta ridică sensibil costul unui atac dar nu-l face imposibil: condensatorii mari necesari transformării pot fi identificaţi uşor la microscop şi distruşi folosind laser, ultrasunete sau fascicule concentrate de ioni.

  Acest tip de atac nu este caracteristic cartelelor telefonice sau TV. Un cip modificat în acest fel poate fi investigat fără riscul de a şterge din greşeală $EEPROM$-ul, sau din cauza unei funcţii de protecţie incorporate.

4.9.3 Tehnici non-invazive de atac

Un atac non-invaziv asupra unui dispozitiv smartcard este oarecum mai limitat, dar prezintă un pericol major: din moment ce nu face nici o modificare asupra smartcardului, este foarte greu de descoperit. De exemplu, dacă s-ar găsi cheia privată a unui dispozitiv de semnătură fără ca *Alice* să observe, se pot falsifica documente în numele ei mult timp înainte de a se descoperi infracţiunea.

Unele atacuri non-invazive pot decurge într-un timp scurt şi folosind resurse hardware uzuale (eventual modificate puţin). Dar aceste tehnici înfruntă limitări puternice: deoarece ele trebuie să aibă loc în timp ce cardul operează în condiţii normale, orice manipulare va avea ca obiect condiţiile de mediu sau octeţii care intră şi ies din smartcard.

În general, în cazul atacurilor non-invazive, este necesar ca *Oscar* să cunoască structura software şi hardware a smartcard-ului.

- **Atacul prin cronometrare**:

  În acest atac – descris de Paul Kocher în [3] – diferite secvenţe de octeţi sunt trimise card-ului pentru a fi semnate cu cheia privată. Informaţii cum ar fi timpul necesar efectuării unei operaţii şi numărul de 0 şi 1 din şirul de intrare sunt folosite ulterior de *Oscar* pentru a obţine cheia privată.

Fiind un atac cu text clar ales, este necesar ca *Oscar* să cunoască PIN-ul cardului sau să-l păcălească pe utilizator să semneze şirurile de biti pe care acesta i le furnizează.

Există măsuri de apărare la nivel hard împotriva acestui tip de atac dar nu toţi producătorii de smartcard-uri le implementează.

Unele variante de atac prin cronometrare pot fi executate via software. De exemplu, ar putea fi folosită o aplicaţie "cal troian", înfiltrată – printr-un procedeu oarecare – pe o staţie de lucru a lui *Alice*. "Calul troian" aşteaptă până când *Alice* introduce un PIN valid dintr-o aplicaţie de încredere, activând astfel folosirea cheii private; apoi cere smartcard-ului să semneze digital nişte date (de exemplu un contract).
Operaţia se încheie după acest schimb de mesaje, dar utilizatorul nu ştie că cheia sa privată a fost folosită fără consimţământul său.

Măsura de protecţie împotriva acestui tip de atac este folosirea arhitecturii "single - access device driver". Pe acest tip de arhitectură, sistemul de operare se asigură că, la orice moment, doar o singură aplicaţie poate avea acces la un dispozitiv (smartcardul – în cazul nostru).
Acest lucru împiedică atacul dar va afecta folosirea smartcard-ului, deoarece limitează accesul la serviciile smartcardului.

O altă cale de a preveni astfel de atacuri este folosirea unui smartcard care solicită introducerea PIN-ului pentru fiecare utilizare a cheii private. În acest model, *Alice* trebuie să introducă PIN-ul de fiecare dată când cheia privată trebuie folosită, şi deci "calul troian" nu va avea acces automat la cheie. Acest lucru este însă rareori convenabil pentru utilizatorii comozi.

- **Atacuri prin analiza consumului de energie**:

  Se bazează pe măsurarea fluctuaţiilor în curentul consumat de dispozitiv. Diferite instrucţiuni provoacă nivele diferite de activitate în decodorul de instrucţiuni şi unităţile de calcul aritmetic; ele pot fi adesea distinse clar, iar pe baza lor pot fi reconstituite părţi din algoritmi.

  Aceste tehnici intră in categoria monitorizării informaţiei şi constituie o ameninţare reală datorită faptului că pe piaţă există un număr mare de produse vulnerabile. Atacurile sunt uşor de implementat, pot fi automatizate, au un cost scăzut şi sunt non-invazive.

  Analiza simplă a consumului de curent (*Supply Power Analysis* – SPA) presupune observarea directă a consumului de energie al sistemului pentru obţinerea de informaţii despre secvenţa de instrucţiuni executată. Dacă *Oscar* are acces doar la o singură tranzacţie, poate fi decelat un număr limitat de indicii.
  Atacatorii cu acces la tranzacţii multiple şi care posedă informaţii despre mecanismele interne specifice arhitecturii cip-ului folosit, pot fi periculoşi. Kocher arată

cum poate fi spart sistemul de criptare DES în cazul unei implementări hardware slabe, dar şi cum poate fi evitat acest atac dacă sunt îndeplinite câteva condiţii, cum ar fi de exemplu nelăsarea biţilor din cheie să fie folosiţi în alegerea dintre două ramuri ale unui *jump*.

Analiza diferenţială a consumului (*Differential Power Analysis* - DPA) se bazează pe faptul că memorarea unui bit 1 într-un flip - flop necesită de obicei mai multă energie decât memorarea unui bit 0. De asemenea, schimbarea de stare cauzează un consum sporit de energie. În plus, faţă de variaţiile la scară mare datorate secvenţei de instrucţiuni, există efecte corelate cu valorile datelor manipulate. Aceste valori tind să fie mai mici şi sunt uneori mascate de erori de măsurare şi alte interferenţe, dar există tehnici pentru rezolvarea acestor probleme.

Diferite instrucţiuni produc nivele variate de activitate în decodorul de instrucţiuni şi în unităţile aritmetice; adesea ele pot fi diferenţiate clar, şi pe baza lor pot fi reconstruite părţi din algoritm. Componentele procesorului îşi schimbă valorile temporare stocate în momente diferite relativ la intervalul de oscilaţie al ceasului şi pot fi separate prin măsurători de înaltă frecvenţă.

Algoritmii cu cheie publică pot fi analizaţi folosind DPA prin corelarea valorilor candidat pentru calculele intermediare cu măsurătorile consumului de energie ([4]). Astfel, pentru operaţiile de exponenţiere modulară, este posibil să se testeze biţi ghiciţi prin verificarea dacă valorile intermediare prezise sunt corelate cu calculele reale.

Prin această metodă este posibil un atac "reverse engineering" pentru un protocol sau un algoritm necunoscut. În general, semnalele care se scurg în timpul unei operaţii asimetrice tind să fie mult mai puternice decât acelea provenite de la algoritmii simetrici – de exemplu, din cauza complexităţii computaţionale relativ ridicate a operaţiilor de multiplicare.
De aceea implementarea măsurilor de apărare împotriva SPA şi DPA poate fi dificilă.

DPA poate fi folosită pentru a sparge implementări ale aproximativ tuturor algoritmilor de criptare. De exemplu, o cheie secretă $Twofish$ de 128 biţi (lungime considerată sigură) a fost recuperată dintr-un smartcard după ce s-au monitorizat 100 criptări independente. În acest caz se poate observa că DPA descoperă $1 - 2$ biţi de informaţie per criptare.
Singura soluţie de încredere presupune proiectarea de criptosisteme având o imagine clară a hardware-ului aflat la bază.

Există tehnici de prevenire a DPA şi a atacurilor înrudite. Ele se pot împărţi în trei categorii:

– Se poate reduce dimensiunea semnalului folosind căi de execuţie constantă a

codului, alegând operaţii care permit scurgerea a cât mai puţină informaţie sau adăugând porţi suplimentare care să compenseze consumul.
Din nefericire semnalul nu se poate reduce la zero, iar un atacator cu un număr (teoretic) infinit de mostre va fi capabil să folosească cu succes un atac bazat pe DPA.

- Se pot introduce interferenţe în măsurarea consumului de energie; ca şi în cazul precedent, un număr foarte mare de mostre va ajuta analiza statistică.
- Folosirea procedurilor neliniare de actualizare a cheii. De exemplu, aplicarea funcţiei $SHA-1$ unei chei de 160 biţi ar trebui să ascundă efectiv toate informaţiile parţiale pe care un atacator le-a putut afla despre cheie. Similar, pot fi folosite modificări ad-hoc în procesele de utilizare a exponenţilor şi de modificare a modulelor în cadrul sistemelor cu cheie publică (pentru a împiedica atacatorii să adune date în timpul unui număr mare de operaţii). De asemenea se pot utiliza contori ai folosirii cheilor, pentru a preveni obţinerea unui număr mare de mostre criptate.

Un atac similar cu DPA este analiza electromagnetică (EMA – *ElectroMagnetic Analysis*). Atacul constă în măsurarea câmpului radiat de procesor şi corelarea rezultatelor cu activitatea procesorului respectiv.
Un astfel de atac poate obţine cel puţin acelaşi rezultat ca cel prin analiza consumului de energie.

- **Atacuri prin generarea de erori**:

  Atacurile prin generare de erori se bazează pe exercitarea de presiuni asupra procesorului cardului, pentru a-l forţa să execute operaţii ilegale sau să dea rezultate greşite.

  Există o mare varietate de forme pe care le pot lua astfel de atacuri.

  O modalitate este aceea de a modifica tensiunea de alimentare şi semnalul ceasului. Subtensionarea şi supratensionarea pot fi folosite pentru a dezactiva circuitele de protecţie sau pentru a forţa procesoarele să efectueze operaţii greşite. În alte condiţii, unii parametri fizici (cum ar fi temperatura) sunt modificaţi pentru a obţine acces la informaţii sensibile de pe smartcard.

  Un alt atac fizic posibil presupune o fluctuaţie intensă a unor parametri fizici, exact în momentul în care se efectuează verificarea PIN-ului. Astfel, funcţii sensibile ale cardului pot fi folosite chiar dacă PIN-ul este necunoscut.
  Deşi nu se poate descrie exact cum este posibil ca o perturbare să cauzeze executarea de către procesor a unor instrucţiuni greşite, perturbarea se poate afla relativ uşor printr-o căutare sistematică.

  Exemplul 4.3. *O subrutină des întâlnită în procesoarele de securitate este un ciclu care scrie conţinutul unei zone limitate de memorie către un port serial:*

```
1 b = answer_address
2 a = answer_length
3 if (a == 0) goto 8
4 transmit(*b)
5 b = b + 1
6 a = a - 1
7 goto 3
8
```

Se poate căuta o perturbare care să incrementeze normal contorul de program, dar care să transforme în altceva saltul condiţionat de la linia 3 ori decrementarea variabilei de la linia 6.
A găsi acea perturbare înseamnă a activa cardul într-un mod repetabil. Toate semnalele trimise către card trebuie să ajungă exact la acelaşi interval de timp după reset, pentru fiecare rundă de testare.

Pentru fiecare ciclu de ceas pot fi testate mai multe perturbări, până când una din ele are ca efect transmiterea către portul serial a unui octet suplimentar. Repetând acea perturbare, se obţine o citire a memoriei rămase, care – cu ceva şansă – poate conţine cheile căutate.

Salturile condiţionate creează ferestre de vulnerabilitate în stadiile de procesare a multor aplicaţii de securitate, care permit ocolirea barierelor criptografice sofisticate prin simpla împiedicare a execuţiei codului care verifică dacă o autentificare are succes sau nu.

Contramăsurile hardware includ generatori interni independenţi de ceas, sincronizaţi doar cu frecvenţa externă de referinţă. O altă abordare, mai radicală dar potenţial mai benefică, este să se elimine complet ceasul, transformând procesoarele de card în circuite autocronometrate asincrone.

Ceasul extern ar fi folosit doar ca referinţă pentru comunicare şi – prin urmare – perturbările de ceas ar cauza doar o corupere a datelor.

4.9.4 Analiza diferenţială a erorilor

Analiza diferenţială a erorilor (*Differential Fault Analysis – DFA*) este o tehnică de atac detaliată de Biham şi Shamir ([2]) împotriva sistemelor de criptare înglobate în dispozitive precum smartcard-urile.
Dacă un dispozitiv aflat sub stress (căldură, vibraţii, presiune, radiaţii etc.) poate fi făcut să returneze ieşiri eronate, atunci un criptanalist care compară ieşirile corecte cu cele eronate, are un punct de pornire în atac.

DFA poate sparge DES folosindu-se de 200 texte criptate în care s-au introdus erori de un bit, un număr infim comparativ cu numărul de texte solicitat de atacurile standard (criptanaliza diferenţială sau liniară).

Modelul folosit a fost propus de Boneh şi dezvoltat ulterior de Biham şi Shamir. Se presupune că prin expunerea unui procesor la o sursă slabă de radiaţie ionizantă, acele erori de un bit pot fi induse în datele folosite şi – în special – în corpul sub-cheii generate în rundele succesive DES. Metoda poate fi extinsă pentru a realiza "reverse engineering" pentru algoritmi a căror structură este necunoscută.

1. Cu această metodă a perturbaţiilor se poate efectua un atac direct (propus de Lenstra) asupra algoritmului RSA (sau DSA):

   Presupunem că un smartcard calculează o semnătură RSA, S pe un mesaj M, modulo $n = p*q$, calculând întâi modulo p şi q separat şi apoi combinând rezultatele cu Teorema Chineză a Resturilor. Dacă poate fi indusă o eroare în oricare din aceste calcule, atunci se poate factoriza n. Mai exact, dacă e este exponentul public, iar semnatura $S = M^d \ (mod\ p*q)$ este corectă modulo p dar incorectă modulo q, atunci $p = cmmdc(n, S^e - M)$.

2. Şi atacurile asupra DES sunt destul de directe, dacă $Oscar$ poate alege ce instrucţiune va avea de suferit în urma unei perturbaţii.
   De exemplu el poate opta să înlăture una din operaţiile XOR pe 8 biţi folosite pentru a combina cheia de rundă cu intrările în $S - boxurile$ provenite de la ultimele două runde ale textului, şi repetă această procedură pe rând pentru fiecare octet din cheie. Fiecare din textele criptate alterate obţinute ca rezultat al acestui atac va diferi de textul criptat autentic în ieşirile a două sau trei $S - boxuri$.

   Folosind tehnicile criptanalizei diferenţiale, intrusul poate obţine – ca rezultat al erorii induse – în jur de 5 biţi de informaţie despre cei 8 biţi ai cheii care nu au fost XOR-aţi.

   Astfel, de exemplu, 6 texte criptate folosind runde finale greşite pot demasca aproximativ 30 biţi din cheie, permiţând o căutare exhaustivă destul de uşoară a cheii.

3. Un atac mai rapid se obţine dacă se reduce numărul de runde din DES la maxim 2, prin coruperea unei anumite variabile sau a unui salt condiţionat. Astfel, cheia poate fi aflată printr-un atac simplu. Aplicabilitatea acestui atac constă în detaliile de implementare.

4. DFA poate fi folosit chiar şi pentru un sistem necunoscut de criptare. Biham şi Shamir arată că – în anumite condiţii – structura unui sistem de criptare similar cu DES poate fi schiţată folosind aproximativ 500 texte criptate; mai mult – dacă numărul lor ajunge la 10.000, atunci se pot reconstitui toate detaliile legate de S-boxuri.

O metodă de apărare împotriva DFA constă în adăugarea unor rutine pentru detectarea erorilor.
Dacă ieşirile nu conţin niciodată erori, DFA devine imposibil de aplicat, iar dacă ieşirile eronate sunt destul de rare, DFA devine nepractic.

Există numeroase propuneri pentru construirea rutinelor de detecţie a erorilor.

1. O primă abordare este ca textul să fie criptat de două sau mai multe ori, iar rezultatele obţinute să fie comparate – şi respinse dacă nu coincid. Este o metodă destul de ineficientă practic.

2. *Verificarea parităţii*:

   Paritatea biţilor dintr-o secvenţă nu se schimbă dacă biţii sunt permutaţi. Verificarea parităţii poate proteja deci orice operaţie criptografică care presupune permutare de biţi. Se acoperă astfel câteva operaţii cum ar fi permutările la nivel de octeţi sau blocuri, deplasări circulare, programarea cheilor pentru $IDEA$ sau pentru câteva versiuni de $CAST$. Sunt acoperite de asemenea toate operaţiile echivalente cu permutarea nulă, cum ar fi stocare şi recuperare sau transmisie şi recepţie a unei valori.

   Bitul de paritate al concatenarii a două sau mai multe şiruri de biţi este XOR-ul biţilor de paritate ai secvenţelor iniţiale. Verificarea parităţii poate proteja astfel orice operaţie criptografică bazată pe concatenare de biţi sau – invers – pe spargere în subblocuri.

   Chiar şi operaţia XOR poate beneficia de pe urma verificării parităţii, deoarece bitul de paritate al unui XOR este egal cu XOR-ul biţilor de paritate ai intrării.

3. *Verificare prin aritmetica modulară:*

   Orice operaţie aritmetică pe $\mathcal{Z}$ rămâne adevărată modulo orice bază (convenabilă). Folosirea unei baze mari, sau verificarea calculelor în mai multe baze duce la creşterea şanselor găsirii unei erori.

   O tehnică bună de verificare a calculelor constă în adăugarea de grupuri de n cifre binare, pentru a obţine valori modulo $2^n - 1$.

Bibliografie

[1] Atanasiu, A. – *Securitatea Informaţiei, vol 1 (Criptografie)*, Ed. InfoData, Cluj, 2007.

[2] Biham, E., Shamir. A. – *Differential Fault Analysis of Secret Key Cryptosystems*. In Burton S. Kaliski Jr. ed., Advances in Cryptology - CRYPTO 97, LNCS vol. 1294, Springer - Verlag, 1997, pp. 513 - 525.

[3] Kocher, P. – *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Related Attacks*, Proceedings of Advances in Cryptology - CRYPTO96, Springer-Verlag, 1996, pp. 104-130

[4] Kocher, P., Jaffe, J., Jun, B. – *Differential Power Analysis*, Advances in Cryptology - Crypto 99 Proceedings, LNCS vol. 1666, M. Wiener, ed., Springer-Verlag, 1999

[5] Schneier, B, Shostack, A - *Breaking Up Is Hard to Do: Modeling Security Threats for Smart Cards*, USENIX Workshop on Smartcard Technology Chicago, Illinois, USA, May 1011, 1999

[6] Shamir, A. – *Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies*, LNCS, Springer - Verlag, 2000

[7] Simmons, G. J. – *Contemporary Cryptology*, IEEE Press, 1992