CUPRINS

LUCRAREA 1

ANALIZA EXPERIMENTALĂ ENTROPICĂ A SURSELOR DE INFORMAŢIE

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- simularea unei surse de informaţie folosind tehnica de calcul;
- evaluarea entropică a sursei de informaţie simulate prin tehnici software.

2. BREVIAR TEORETIC

O sursă de informaţie discretă este caracterizată de un număr *n* (uzual finit) de stări observabile şi de un vector de probabilităţi asociate p = $[\, p_1\ p_2 \ldots p_n]$; sursa se poate găsi într-o anumită stare k cu o anumită probabilitate p_k. Situarea sursei în una sau alta dintre stări reprezintă, în termeni de teoria probabilităţilor, un sistem complet de evenimente mutual incompatibile (sursa nu se poate afla în acelaşi timp în doua stări) a căror reuniune este evenimentul sigur (sursa se află într-una din stări). În aceste condiţii, suma probabilităţilor asociate celor n stări este egală cu unitatea:

$$\sum_{k=1}^{n} p_k = 1 \ . \qquad (1.1)$$

O sursă de informaţie emite uzual un anumit simbol odată cu ocuparea unei noi stări. Simbolul respectiv este purtător de informaţie. Informaţia asociată fiecărui simbol/stare este dată de relaţia:

$$I(k) = -\log_2(p_k) \qquad (1.2)$$

(i se măsoară în biţi). Informaţia asociată unui simbol este cu atât mai mare cu cât probabilitatea asociată este mai mică.

O sursă pentru care probabilităţile de ocupare a oricăreia dintre stări nu depind de starea (stările) ocupată (ocupate) anterior este o sursă fără memorie. O sursă care nu îndeplineşte această condiţie este, dimpotrivă, o sursă cu memorie. Pentru sursele cu memorie probabilităţile p_k sunt înlocuite în cazul cel mai simplu de probabilităţile condiţionate $p_{k/j}$ în care al doilea indice *j* reprezintă starea anterioară stării *k*. Mai general, probabilitatea ca sursa să fie la un moment dat în starea *k* poate fi condiţionată de mai multe stări ocupate anterior.

O sursă de informaţie fără memorie este caracterizată de informaţia medie generată:

$$H = -\sum_{k=1}^{n} p_k \log_2(p_k) \qquad (1.3)$$

numită şi **entropie** a sursei. În relaţia de calcul a entropiei $x \log(x) = 0$ ori de câte ori $x=0$.

Cazul mai complicat, dar foarte frecvent în realitate, al surselor de informaţie pentru care probabilitatea de producere a unui simbol depinde de secvenţa emisă / produsă anterior este modelat suficient de exact de **secvenţele Markov staţionare** care au următoarele caracteristici:

- sursa se află în una din cele *n* stări posibile *1,2,…n* la începutul fiecărui interval elementar de emitere a unui simbol;
- când sursa trece din starea *i* în starea *j*, se emite un simbol care depinde de starea *i* şi de tranziţia $i \rightarrow j$;
- dacă $s_1, s_2, \ldots, s_m$ sunt simbolurile alfabetului sursei şi $x_1, x_2, \ldots, x_k, \ldots$ este secvenţa variabilelor aleatoare emise de sursă, probabilitatea ca x_k să fie simbolul s_q este condiţionată de cele *k-1* simboluri emise anterior

$$p(x_k = s_q / x_1, x_2, \ldots, x_{k-1}) ;$$

- influenţa reziduală a simbolurilor $x_1, x_2, \ldots, x_{k-1}$ este reprezentată prin starea sistemului la începutul intervalului *k*, notată s_k.

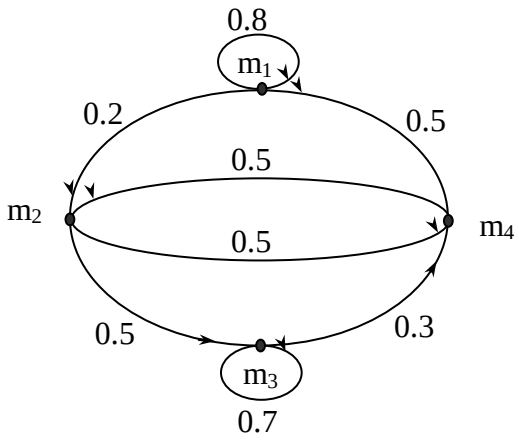
$$p(x_k = s_q / x_1, x_2, \ldots, x_{k-1}) = p(x_k = s_q / s_k);$$

- la începutul primului interval de emisie, sistemul se află în una din cele *n* stări posibile cu probabilităţile $p_1(1)$, $p_2(1)$,…, $p_n(1)$; $\sum_{i=1}^{n} p(1) = 1$;

- dacă $p_j(k)$ este probabilitatea ca sistemul să fie în starea j la începutul intervalului *k*, atunci o tranziţie a sistemului se reprezintă prin:

$$p_j(k+1) = \sum_{i=1}^{n} p_i(k)\, p_i . \qquad (1.4)$$

Sursele Markov discrete se reprezintă prin grafuri orientate cu arce de "capacităţi " egale cu probabilităţile tranziţiilor asociate. Aceste probabilităţi, ca şi graful însuşi, se pot reprezenta şi sub formă matricială. Este dat mai jos cazul relativ simplu al unei surse cu patru stări – figura 1.1.



$$\Phi = \begin{bmatrix} 0.8 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 0.7 & 0.3 \\ 0.5 & 0.5 & 0.0 & 0.0 \end{bmatrix}$$

Fig. 1.1. Sursă Markov cu patru stări

Aplicaţie

Se consideră o sursă de informaţie având ca model un proces Markov aleator, ergodic şi discret, cu graful asociat prezentat în figura 1.2.

Se cere să se calculeze entropia sursei şi informaţia medie pe simbol conţinută în mesaje de 1, 2 şi 3 simboluri.

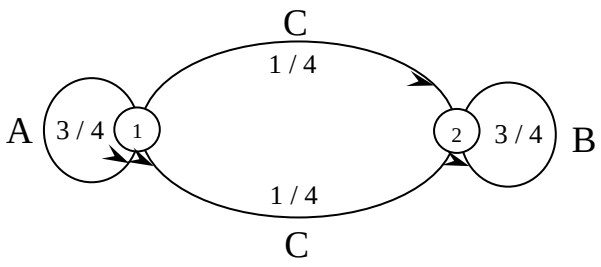


Fig. 1.2. Graful asociat sursei de informaţie

În tabelul 1.1 sunt ilustrate probabilităţile de apariţie ale tuturor mesajelor de lungimi de 1 simbol, 2 simboluri şi 3 simboluri.

Tabelul 1.1.

Mesaje de lungime 1	**Mesaje de lungime 2**	**Mesaje de lungime 3**
A (3/8)	AA (9/32)	AAA (27/128)
B (3/8)	AC (3/32)	AAC (9/128)
C (1/4)	CC (2/32)	ACC (3/128)
	CB (3/32)	ACB (9/128)
	CA (3/32)	CCA (3/128)
	BC (3/32)	CCC (2/128)
	BB (9/32)	CBC (3/128)
		CBB (9/128)
		CAA (9/128)
		CAC (3/128)
		CCB (3/128)
		BCA (9/128)
		BCC (3/128)
		BBC (9/128)
		BBB (27/128)

Se calculează:
$H_1 = H_2 = 1/4 \log 1/4 + 3/4 \log 3/4 = 0{,}8113$ bit / simbol;
$H = 1/2\ H_1 + 1/2\ H_2 = 0{,}8113$ bit / simbol;

Calculând informaţia medie conţinută în cele şapte mesaje de două simboluri, se obţine:
I (AA) = I (BB) = 1,83;
I (BC) = I (AC) = I (CB) = I (CA) = 3,415 biţi.

Ponderând această informaţie cu probabilităţile corespunzătoare, se obţine valoarea de 2,5598 biţi. Rezultă deci informaţiile medii pe simbol, respectiv 1,5612 bit / simbol; 1,2799 bit / simbol; 1,097 bit / simbol.

O sursă de informaţie poate fi creată prin utilizarea funcţiei de generare a numerelor (pseudo)aleatoare uniform repartizate, existentă în biblioteca asociată oricărui limbaj de programare. În particular, pentru limbajul **PASCAL,** funcţia se numeşte **random**. Funcţia fără argument generează numere aleatore de tip *real* cuprinse între 0 şi 1, iar cu argument (întreg de tip *word*) generează numere întregi nenegative, strict mai mici decât argumentul.

Se generează un număr relativ mare de numere (pseudo)aleatoare, de pildă câteva mii sau zeci de mii, şi se studiază frecvenţa de apariţie a valorilor întregi în cazul utilizării funcţiei **random** cu argument, sau frecvenţele asociate unor subintervale ale intervalului (0, 1) de egală întindere în cazul folosirii aceleiaşi funcţii fără argument.

Se calculează frecvenţele relative şi se compară cu probabilităţile teoretice.

Se evaluează entropiile utilizând atât probabilităţile cât şi frecvenţele relative.

Se compară rezultatele.

Se împarte intervalul (0, 1) în subintervale de întindere diferită, de pildă proporţionale cu *n* numere generate cu funcţia **random(10)**.

Se realizează studiul frecvenţelor relative întocmai ca în paragraful precedent şi se compară cu probabilităţile teoretice.

Se calculează entropiile sursei pe baza probabilităţilor teoretice şi utilizând frecvenţele relative.

Se compară rezultatele obţinute, se compară valorile din cazul subintervalelor egale cu cel al subintervalelor inegale.

Se propune pentru acest ultim punct următoarea secvenţă **PASCAL (P1)**:

```
randomize;
sumaa:=0;
for i:=1 to n do begin  a[i]:=random(10);   sumaa:=sumaa+a[i]  end;
c[0]:=0.0;
for i:=1 to n do begin  b[i]:=a[i]/sumaa;   c[i]:=c[i-1]+b[i];  f[i]:=0
end;
for k:=1 to 10000 do begin
r:=random;
for i:=1 to n do if ( r >c[i-1] ) and   ( r <=c[i] ) then  f[i]:= f[i]+1
end;
for i:=1 to n do  fr[i]:=fr[i] /10000;
```

{notaţii principale: a – secvenţă de n numere aleatoare; b – lărgimea subintervalelor intervalului (0, 1); c – coordonatele care marchează diviziunea intervalului (0, 1); f, fr – frecvenţele absolute şi relative asociate subintervalelor

Desigur, secvenţa trebuie completată cu declaraţiile de variabile necesare, etc. Secvenţa poate fi îmbunătăţită, poate fi tradusă în alt limbaj de programare.

Evaluările pentru o **sursă reală** se conduc conform recomandărilor care urmează. Se consideră operaţia de lectură byte cu byte a conţinutului unui fişier la alegere. Fişierul poate fi considerat o sursă discretă de informaţie cu 256 de stări.

Asimilând frecvenţele relative cu probabilităţile de apariţie ale bytes-ilor (ceea ce pentru fişiere voluminoase este aproape adevărat deoarece frecvenţele relative tind "în probabilitate" către probabilităţile de apariţie la lectură a diverşilor bytes-i, pe măsură ce numărul de observaţii asupra sursei creşte), se poate calcula entropia fişierului luat ca sursă de informaţie discretă generatoare de bytes-i.

Repetând operaţia pentru fişiere de diverse tipuri (text, executabile, de date numerice, etc.) se pot face comparaţii între rezultatele obţinute.

Pentru cel mai frecvent byte dintr-un fişier din cele selectate se poate face un studiu al frecvenţelor de apariţie în funcţie de byte-ul anterior. Se apreciază pentru fişierul în cauză calitatea de sursă de informaţie cu sau fără memorie.

Se propune următoarea secvenţă **PASCAL (P2)** pentru evaluarea entropică a sursei – fişier tratată ca sursă fără memorie:

```
assign (fis, fiser);
reset(fis);
for i:=0 to 255 do f[i]:=0;
while not eof(fis) do begin read(fis, b); f[b]:=f[b]+1 end;
close(fis);
sumaf:=0;
for i:=0 to 255 do sumaf:=sumaf+f[i];
for i:=0 to 255 do fr[i]:=f[i]/sumaf;
h:=0.0;
for i:=0 to 255 do h:=h-fr[i]*ln(fr[i])/ln(2.0);
{notaţii principale: b – byte-ul curent citit; f, fr – frecvenţele absolute
şi relative asociate byte-ilor; h - enropia}
```

Pentru aprecierea caracterului de sursă cu sau fără memorie a fişierului în studiu se recomandă următoarea secvenţă **PASCAL(P3)**:

```
assign (fis, fiser);
reset(fis);
for i:=0 to 255 do f[i]:=0;
while not eof(fis) do begin read(fis, b); f[b]:=f[b]+1 end;
close(fis);
maxf:=0;
for i:=0 to 255 do if maxfr <f[i] then begin maxf:=f[i]; k:=I end;
assign (fis, fiser);
reset(fis);
for i:=0 to 255 do f[i]:=0;
read(fis, ba);
while not eof(fis) do begin
read(fis, b); if b=k then f[ba]:=f[ba]+1; ba:=b;
                          end;
close(fis);
sumaf:=0;
for i:=0 to 255 do sumaf:=sumaf+f[i];
for i:=0 to 255 do fr[i]:=f[i]/sumaf;
{notaţii principale: b – byte-ul curent citit; ba – byte-ul citit anterior; k
– byte-ul cel mai frecvent; f, fr – frecvenţele absolute şi relative
asociate bytes-ilor sau tranziţiei de la un byte oarecare la byte-ul k}
```

3. MOD DE LUCRU

- se completează secvenţele de program propuse P1, P2, P3 cu declaraţiile şi celelalte elemente de program necesare;
- se elaborează un program care să calculeze frecvenţele de apariţie absolute şi relative ale celor 256 bytes din sursa reală prezentată la punctul 3 şi se construieşte histograma corespunzătoare;
- se porneşte sistemul de calcul;
- se intră în subdirectorul de lucru al grupei;
- se lansează mediul de programare;
- se introduc secvenţele completate de programe P1, P2 şi P3;
- se compilează, se linkeditează şi se lansează în execuţie;
- se realizează, pentru fiecare din cele trei programe, analiza conform celor prezentate la punctul 3;
- se introduce programul elaborat de studenţi;
- se compilează, se linkeditează şi se lansează în execuţie;

Lucrarea se consideră încheiată când toate programele sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Ce este o sursă de informaţie?
- Care este legătura dintre informaţia asociată unui simbol şi probabilitatea de apariţie a acelui simbol?
- Ce este o sursă fără memorie? Dar o sursă cu memorie? Prezentaţi caracteristicile acestora.
- O sursă emite o frecvenţă independentă de simboluri dintr-un alfabet de şase simboluri M, N, O, P, R, S cu probabilităţile 1/4, 1/4, 1/8, 1/8, 3/16, 1/16. Se cere entropia sursei şi informaţia medie pe simbol conţinută în mesaje de 2 simboluri.

LUCRAREA 2

ANALIZA EXPERIMENTALĂ ENTROPICĂ A SISTEMELOR DE TRANSMISIE DE DATE

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- precizarea modalităţilor de apreciere a caracteristicilor entropice ale sistemelor de transmisie de date;
- simularea canalelor de transmisie de date şi evaluarea descifrabilităţii mesajului la recepţie folosind tehnici software;
- aprecierea capacităţii şi eficienţei canalelor de transmisie cu ajutorul tehnicii de calcul.

2. BREVIAR TEORETIC

Un **sistem de transmisie de date punct la punct** este compus dintr-o sursă, un canal şi un receptor.

Sursa este definită de un număr de stări, uzual finit, şi generează un număr de simboluri x1, x2,..., xn cu probabilităţile p(x1), p(x2),..., p(xn), având suma egală cu unitatea. Acestea sunt de obicei şi simbolurile de la intrarea în canal.

Receptorul este, de asemenea, definit de un număr de stări asociate cu simbolurile y1, y2,..., ym şi cu probabilităţile p(y1), p(y2),..., p(ym). Suma acestor probabilităţi este, de asemenea, 1. Se disting aşadar câmpul de intrare şi câmpul de ieşire, fiecare cu alfabetul său şi cu probabilităţile specifice.

Se pot calcula entropiile la intrarea canalului şi la ieşirea lui cu relaţiile cunoscute:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2.1}$$

$$H(X) = -\sum_{j=1}^{m} p(x_j) \log p(y_j) \tag{2.2}$$

Un sistem de transmitere de date este însă interesant în ansamblul lui. Probabilităţile care intervin în calculul entropiilor care caracterizează un sistem de transmitere a informaţiei se grupează într-o matrice cu numărul de linii egal cu numărul de simboluri utilizate la intrarea canalului de transmisie şi cu numărul de coloane egal cu numărul simbolurilor observate la intrarea aceluiaşi canal

$$P(X,Y) = \begin{bmatrix} p(x_1,y_1)\ldots p(x_1,y_m) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ p(x_n,y_1)\ldots p(x_n,y_m) \end{bmatrix} \tag{2.3}$$

cu

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j) = 1; \tag{2.4}$$

$$\sum_{i=1}^{n} p(x_i,y_j) = p(y_j); \tag{2.5}$$

$$\sum_{j=1}^{m} p(x_i,y_j) = p(x_i); \tag{2.6}$$

în care s-a notat cu p(xi), p(yi), p(xi , yj) probabilităţile asociate producerii simbolului xi la intrarea mediului de transmisie, a simbolului yj la ieşire sau a perechii (xi , yj) la intrarea şi la ieşirea canalului de transmisie. Condiţionarea mutuală a simbolurilor emise şi recepţionate este descrisă probabilistic de matricele:

$$P(X/Y) = \begin{bmatrix} p(x_1/y_1)\ldots p(x_1/y_m) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ p(x_n/y_1)\ldots p(x_n/y_m) \end{bmatrix} \tag{2.7}$$

$$P(Y/X)=\begin{bmatrix} p(y_1/x_1)\ldots p(y_m/x_1) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ p(y_1/x_n)\ldots p(y_m/x_n) \end{bmatrix} \quad (2.8)$$

care au ca elemente probabilităţile simbolurilor de la unul din capetele canalului condiţionate de cele de la cealaltă extremitate şi pentru care:

$$\sum_{j=1}^{m} p(x_i/y_j)=1; \quad (2.9)$$

$$\sum_{i=1}^{n} p(y_j/x_i)=1. \quad (2.10)$$

Cu probabilităţile referitoare la ansamblul sursă - canal - receptor se pot calcula:

- entropia câmpurilor reunite

$$H(X,Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log p(x_i,y_j); \quad (2.11)$$

- echivocaţia

$$H(X/Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log p(x_i/y_j)=H(X,Y)-H(Y); \quad (2.12)$$

- irelevanţa (eroarea medie)

$$H(Y/X)=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log p(y_j/x_i)=H(X,Y)-H(X); \quad (2.13)$$

- transinformaţia

$$I(X,Y)=H(X)+H(Y)H(X,Y)=H(X)H(X/Y)=H(Y)\,H(Y/X) \quad (2.14)$$

Se sugerează construirea unei matricei P(X,Y) cu un număr rezonabil de linii şi de coloane (3 până la 5) prin generarea de numere aleatoare reale în intervalul (0,1) cu funcţia de bibliotecă random, plasarea lor în poziţii succesive (i, j), i=1, 2,...,m şi apoi normalizate pentru a avea suma probabilităţilor egală cu unitatea. Din matricea P(X,Y) se pot obţine matricele P(X / Y) şi P(Y / X) prin simpla divizare a coloanelor, respectiv liniilor cu p(yj), cu p(xi).

Se propune secvenţa de program **Pascal (P1)** următoare:

```
 randomize;
 sumap : = 0.0;
 for i:=1 to n do
  for j:= 1 to m do begin  p[i, j]:= random;  sumap:=sumap + p[i, j]
end;
 for i:= 1 to n do
  for j:= 1 to m do  p[i, j]:= p[i, j] / sumap;
 for j:= 1 to m do begin
   sumac:= 0.0
   for i:=1 to n do sumac:=sumac + p[i, j];
   for i:=1 to n do pxy[i, j]:=p[i, j] / sumac
                    end;
 for i:=1 to n do begin
   suma1:= 0.0
   for j:=1 to m do suma1:=suma1 + p[i, j];
   for j:=1 to m do pxy[i, j]:=p[i, j] / suma1
                    end;
```

cu notaţii aproape evidente. Este de aşteptat ca un calcul la entropiilor diverse să conducă la concluzia indescifrabilităţii mesajului la recepţie: irelevanţa (eroarea medie) şi echivocaţia sunt foarte mari. Se propune acest calcul.

Realizarea unui canal cu perturbaţii mai reduse, altfel spus realizarea unui canal utilizabil, echivalează de cele mai multe ori cu existenţa / realizarea câtorva perechi (xi, yj) mai probabile decât altele. De exemplu, admiţând o matrice P(X/Y) pătrată, valorile diagonale pot avea probabilităţi mai mari decât celelalte. Într-un program de similare a unui canal cu perturbaţii mai reduse elementele diagonale ale matricei P(X, Y) pot fi majorate toate cu o constantă pozitivă şi apoi repetată operaţia de normalizare. "Noul" canal realizat are alte caracteristici entropice. Se propune efectuarea acestor evaluări pentru mai multe valori ale constantei adunate elementelor diagonale ale matricei P(X,Y).

Este de observat că poziţia diagonală a elementelor care indică perechile (xi, yj) favorizate probabilistic nu este obligatorie, cum nu este obligatorie nici egalitatea numărului de simboluri ale alfabetului de intrare cu cel al alfabetului de ieşire din canalul de transmisie.

O caracteristică importantă a unui canal de transmisie este **capacitatea** lui. Capacitatea unui canal reprezintă transinformaţia maximă. Maximul se referă la setul de probabilităţi asociate simbolurilor de la intrarea canalului:

$$C = \max_{p(x_i)} I(X,Y) \tag{2.15}$$

Raportul

$$\eta = \frac{I(X,Y)}{C} \tag{2.16}$$

poartă numele de **eficienţa canalului**, iar diferenţa până la unitate

$$\rho = 1 - \eta \tag{2.17}$$

se numeşte **redundanţa canalului**.

În echipamentele de transmitere de date, la care în majoritatea cazurilor se transmit simboluri binare, canalul cel mai des întâlnit este **canalul binar simetric** (**CBS**), caracterizat prin reprezentarea din figura 2.1.

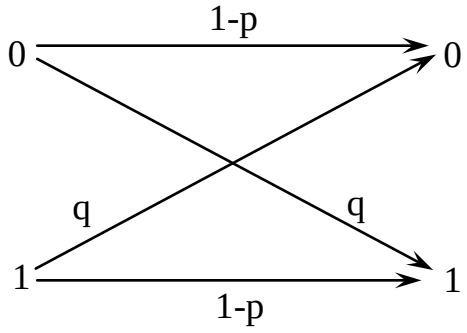


Fig. 2.1. Canalul binar simetric

Legea de tranziţie caracteristică acestui tip de canal este reprezentată de matricea $\pi = [1-p]$, iar capacitatea sa este

$$C_{CBS} = 1 + (1-p)\log(1-p) + p \log p. \tag{2.18}$$

Aplicaţia 1

Să se calculeze capacitatea şi debitul mediu pentru un CBS care emite simboluri echiprobabile cu viteza vs de 1000 simboluri / s, dacă probabilitatea de recepţie eronată este p = 0,1.

Rezolvare
Se calculează succesiv, folosind formulele anterior prezentate.

- entropia sursei: H(x) = ((1/2)(log(1/7)((1/2)(log(1/2) = 1 bit /simbol;
- debitul sursei: Vs = vs(H(x) = 1000(1 = 1000 bit / s;
- echivocaţia: H(X/Y) = ((p log p + (1(p)log(1(p));
- informaţia medie: I(X,Y) = H(X) (H(Y);
- debitul mediu pe canal: D = 531 bit / s;
- capacitatea canalului: C = 0,531 bit.

Aplicaţia 2

Un terminal este utilizat pentru a introduce caractere alfanumerice în calculator, folosind o conectare pe linia telefonică cu B = 3 kHz şi raport semnal-zgomot la ieşire de 10. Ştiind că pot fi transmise 128 de caractere şi că datele se transmit în secvenţe independente echiprobabile, se cer:
- capacitatea canalului;
- viteza maximă (teoretică) de transmisie a datelor fără riscul de a avea erori.

Rezolvare
- capacitatea canalului: C = B log(1+s/z) = 10378 bit /s;
- informaţia medie pe caracter: H = log 128 = 7 bit /caracter;
- viteza maximă de transmisie: V = vs(H = 1482 caractere /s, V<C.

3. MODUL DE LUCRU

- se completează secvenţa de program propusă **P1** cu declaraţiile necesare şi se realizează calculul entropiilor caracteristice pentru aprecierea descifrabilităţii masajului la recepţie;
- se elaborează un program de simulare a unui canal cu perturbaţii mai reduse decât cel descris de **P1** şi se evaluează caracteristicile sursei entropice;

- se consideră cazul simplu al unui canal având atât alfabetul de intrare, cât şi cel de ieşire, formate din două simboluri şi o matrice P(X,Y) asimetrică, generată folosind funcţia **random**. Se vor calcula capacitatea şi eficienţa acestui canal, formulând şi concluziile ce se impun;
- se intră în subdirectorul de lucru al grupei;
- se lansează mediul de programare;
- se introduc programele elaborate;
- se compilează, se linkeditează şi se lansează în execuţie.

Lucrarea se consideră încheiată când toate programele elaborate sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Ce este un sistem de transmisie de date? Definiţi părţile sale componente.

- Ce mărimi intervin la aprecierea descifrabilităţii mesajului la recepţie?

- Care sunt particularităţile canalului binar simetric?

- Se consideră un canal binar simetric caracterizat prin matricea de tranziţie de forma:

$$P(Y/X) = \begin{pmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{pmatrix}$$

având simbolurile de intrare x_1, x_2 şi probabilităţile asociate $p(x_1) = 3/5$, $p(x_2) = 2/5$.
Să se calculeze :
- entropia câmpului de intrare / de ieşire / a câmpurilor reunite;
- transinformaţia;
- capacitatea canalului;
- eficienţa şi redundanţa relativă ale canalului.

- Se consideră un sistem de transmisie de date având

$$P(X,Y) = \begin{pmatrix} 0,25 & 0,0 & 0,0 \\ 0,1 & 0,25 & 0,3 \\ 0,0 & 0,1 & 0,0 \end{pmatrix}$$

Să se calculeze :

- entropia sursei / a receptorului;
- entropia câmpurilor reunite;
- eroarea medie şi echivocaţia;
- transinformaţia;
- capacitatea canalului, eficienţa şi redundanţa.

LUCRAREA 3

STUDIUL EXPERIMENTAL AL PROCESELOR DE MODULAŢIE / DEMODULAŢIE A SEMNALELOR ÎN TRANSMISIA DE DATE

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- recapitularea cunoştinţelor referitoare la modularea / demodularea semnalelor, cu particularităţile specifice transmisiei de date;
- vizualizarea unor semnale modulate cu transportul informaţiei pe purtătoare sinusoidale şi / sau pulsatorii, folosind tehnica de calcul;
- analiza semnalelor modulate pentru diverşi indici de modulaţie în vederea aprecierii descifrabilităţii semnalului la recepţie.

2. BREVIAR TEORETIC

Operaţia de modulare / demodulare face posibilă transmiterea informaţiei prin medii (canale) diverse cu caracteristici diferite. Se disting semnale modulate a căror purtătoare este **sinusoidală** şi semnale a căror purtătoare este o **secvenţă de impulsuri.**

Purtătoarele **sinusoidale** pot fi modulate liniar – este cazul diverselor variante ale **modulaţiei în amplitudine** – sau **exponenţial** – cum se întâmplă în cazul **modulaţiilor de frecvenţă sau de fază**. Se practică uneori **modalităţi mixte de modulare**, adică se modifică simultan în raport cu semnalul – mesaj de transmis mai mult de unul dintre cei trei parametrii ai unui semnal sinusoidal: amplitudine, fază, frecvenţă.

Purtătoarele **pulsatorii**, secvenţe de impulsuri (cvasi)rectangulare, pot fi modulate în **amplitudine**, în **poziţie** sau în **durată.** Secvenţa de

impulsuri modulată poate fi transmisă ca atare sau modulată (a doua oară) pe purtătoare sinusoidale.

O altă modalitate de transmitere a informaţiei, cu anumite avantaje în ceea ce priveşte protecţia la perturbaţii, este **modulaţia după o prealabilă codare a mesajului**. În această variantă, semnalul – mesaj de transmis este eşantionat după regulile date de teorema eşantionării şi eşantioanele sunt cuantificate în raport cu un număr prestabilit de nivele. Rezultă pentru fiecare eşantion o secvenţă binară care se transmite prin mediul de transmisie, sub formă brută sau pe suportul unei purtătoare sinusoidale sau de altă natură. Modalitatea respectivă poartă numele de **modulaţie în cod**.

A. Modulaţia liniară (de amplitudine), în varianta primară, este descrisă de relaţia

$$s(t)= A(1+m\cos \Omega t)\cos \omega t = A\cos \omega t + (A\,m/2)\cos(\omega-\Omega)t + (A\,m/2)\cos(\omega+\Omega)t \qquad (3.1)$$

care se referă la un mesaj sinusoidal de amplitudine Am şi de frecvenţă Ω, modulat pe o purtătoare de amplitudine A şi frecvenţă ω.

Numărul m, numit **grad (indice) de modulaţie**, trebuie să fie subunitar, în caz contrar producându-se aşa-numita supramodulaţie cu consecinţa inadmisibilă a **imposibilităţii recuperării mesajului la recepţie**.

Expresia pune în evidenţă în ultima ei parte trei componente de frecvenţe diferite: purtătoarea în forma ei pură şi celelalte două componente numite benzi laterale. **Purtătoarea nu conţine nimic relativ la mesaj. Toată informaţia conţinută de mesajul modulator (amplitudine, frecvenţă) este conţinută în benzile laterale** şi chiar în manieră redundantă. Prin suprimarea purtătoarei şi / sau a unei benzi laterale, **informaţia transmisă rămâne suficientă pentru reconstituirea mesajului la recepţie**.

B. Modulaţia exponenţială este descrisă matematic de expresia

$$s(t) = A\exp\{ j[\omega_0 t + m(t)]\} \qquad (3.2)$$

sau de expresia

$$s(t) = A \exp\left\{ j\left[\omega_0 t + \int_0^t m(\tau)d\tau \right] \right\}, \qquad (3.3)$$

după cum este vorba de **modulaţia de fază** sau de **modulaţia de frecvenţă**. În primul caz, mesajul *m(t)* modifică faza, în cel de-al doilea, frecvenţa semnalului modulat.

În particular, dacă mesajul este sinusoidal, *m(t)* = M cos Ωt , atunci relaţiile de mai sus devin, în ordine:

$$s(t) = A \exp[\, j(\omega_0 t + \Delta\varphi \cos \Omega t)] = A \exp[\, j(\omega_0 t + \beta \cos \Omega t)]; \qquad (3.4)$$

$$s(t) = A \exp[\, j(\omega_0 t + (\Delta\omega/\Omega) \sin \Omega t)] = A \exp[\, j(\omega_0 t + \beta \sin \Omega t)]; \qquad (3.5)$$

şi pun în evidenţă deviaţii de fază Δφ, deviaţii de frecvenţă Δω şi un indice de modulaţie β.

Secvenţele periodice de impulsuri rectangulare sunt descrise complet de amplitudinea A, de perioada T şi de durata lor τ. Oricare dintre aceste trei caracteristici poate fi modulată urmărind un mesaj sau altul purtător de informaţie. Se obţin astfel **modulaţia în amplitudine**, **în poziţie şi în durată**.

Aşadar, pentru un mesaj sinusoidal, impulsurile modulate în amplitudine au amplitudinea A(1+*m* cos Ωt), impulsurile modulate în poziţie vor avea o periodicitate alterată T(1+*m* cos Ωt) în ritmul mesajului, iar impulsurile modulate în durată vor avea o durată variabilă τ(1+*m* cos Ωt).

Lucrarea are între obiective reprezentarea grafică a semnalelor modulate. De exemplu, pentru modulaţia liniară a purtătoarelor sinusoidale, în cazul menţinerii purtătoarei şi a ambelor benzi laterale (modulaţia de anvelopă), secvenţa de program PASCAL (**P1**) care produce pe ecran graficul semnalului este următoarea:

```
detectgraph (gdriver, gmode);
initgraph (gdriver, gmode, cale);
mx:=getmaxx;
my:=getmaxy;
line(0, my div 2, mx, my div 2);
```

```
        moveto(0, my div 2);
        for x:=0 to mx do begin
              y:=my div 2 –
round(factor*(1.0+m*cos(omega*x))*cos(om*x));
              lineto(x,y)
              end;
       closegraph;
```

{notaţii principale: cale – şir de caractere reprezentând calea către directorul cu bibliotecă grafică; mx, my–dimensiunile ecranului în pixeli; factor–factor numeric care asigură ocuparea verticalei ecranului; omega, om–frecvenţele semnalului modulator şi ale purtătoarei; m–indice (grad) de modulaţie}.

Se recomandă introducerea în program a unui indice de modulaţie variabil între 0 şi 1, valori permise, ca şi prevederea posibilităţii de depăşire cu câteva procente a valorii maxime admise pentru indicele de modulaţie, pentru a observa graficul unui semnal supramodulat. Frecvenţele purtătoarei şi mesajului se aleg astfel încât pe ecran să se reprezinte 1-2 alternanţe complete ale semnalului modulator.

În secvenţa de mai sus se poate înlocui expresia semnalului modulat în amplitudine cu purtătoare şi cu ambele benzi laterale cu un semnal fără purtătoare (purtătoarea suprimată) sau cu partea reală a unui semnal modulat în frecvenţă. Se vor încerca de fiecare dată indici de modulaţie diverşi.

Semnalele modulate pe purtătoare sinusoidale, ca de altfel orice semnal, pot fi caracterizate şi prin **spectrele lor de frecvenţă.**

Un *semnal modulat liniar*, cu una sau două benzi laterale, cu purtătoare prezentă sau suprimată, ocupă o bandă egală cu cea mai mare frecvenţă din spectrul mesajului sau egală cu dublul acesteia.

În cazul unui *mesaj sinusoidal*, în condiţiile menţinerii ambelor benzi laterale şi a purtătoarei, descompunerea din formula dată mai sus pune în evidenţă atât benzile laterale cât şi amplitudinile fiecărei componente ale spectrului de frecvenţe.

Semnalele modulate exponenţial au spectre mult mai bogate chiar atunci când mesajul este sinusoidal, teoretic de întindere nelimitată.

De exemplu, dacă semnalul este modulat în frecvenţă, expresia lui descompusă pe componente este:

$$s(t) = A\sum_{k=-\infty}^{k=\infty} j_k(\beta)\exp[j(\omega_0 + k\Omega)t] \qquad (3.6)$$

în care intervin funcţiile Bessel de speţa I de indice k (întreg) care au ca argument indicele de modulaţie β. Desigur, sub aspect practic, interesează numai o parte (finită) a spectrului, acea parte care cumulează, de exemplu, 99% din puterea semnalului.

Pentru evaluarea amplitudinii componentelor de diverse frecvenţe se propune următoarea secvenţă de program PASCAL (**P2**), destinată evaluării funcţiilor Bessel din relaţia de mai sus:

```
function J(beta: real; k: integer): real;
   { calculul funcţiei Bessel de speţa I, de indice k şi de
argument β
     aplicabilă pentru indici k până la 7, reproducere a relaţiei
     J_k(β) = (β^k / 2^k) Σ_{i=0}^{∞} β^{2i} / (2^{2i} k!(k+i)!)      }
  var
    b1, b2, pas, u: real;
    i, n: integer;
  begin
    b1:=beta /2.0;
    b2:=sqr (b1);
    u:=0.0;
  n:=0;
  repeat
    pas:=1.0;
    if n > 0 then for i:=n downto 1 do pas:=  b2*pas/i/i;
    if k > 0 then for i:=n +k downto n+1 do pas:=  pas/i;
    if 2*(n div 2)<>n then pas:=-pas;
    n:=n+1;
    u:=u+pas;
  until abs(pas)<1.0e-6;
  if k>0 then u:=u*exp(k*ln(b1));
  J:=u;
end;
```

Semnalele secvenţe de impulsuri modulate se reprezintă grafic alături de secvenţa periodică nemodulată. Coeficientul m trebuie stabilit, în fiecare caz, pentru fiecare tip de modulaţie, la o valoare care să evite supramodularea. Se propune introducerea posibilităţii de a modifica respectivul coeficient pentru a observa efectul lui asupra **descifrabilităţii semnalului la recepţie.**

3. MOD DE LUCRU

- se elaborează programul de reprezentare a unui semnal sinusoidal modulat în amplitudine (**P3**) în vederea studierii formei semnalului pentru indici de modulaţie diferiţi;
- se elaborează programul de reprezentare a spectrului unui semnal sinusoidal modulat în frecvenţă (**P4**) în vederea analizei lărgimii benzii ocupate în funcţie de indicele de modulaţie;
- se elaborează programul de reprezentare a impulsurilor modulate în amplitudine, în poziţie, în durată (**P5**);
- se intră în subdirectorul de lucru al grupei;
- se lansează mediul de programare;
- se introduc programele **P3**, **P4** şi **P5**;
- se compilează, se linkeditează şi se lansează în execuţie;
- se studiază forma semnalului pentru diverşi indici de modulaţie având valori admise, între 0 şi 1;
- se atribuie indicelui de modulaţie valori supraunitare, se observă forma semnalului supramodulat şi se apreciază efectul asupra descifrabilităţii mesajului la recepţie.

Lucrarea se consideră încheiată când toate programele elaborate sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Care din metodele de modulaţie studiate în lucrare oferă posibilitatea obţinerii debitului maxim de informaţie?
- Explicaţi modul în care perturbaţiile de tip aditiv, respectiv zgomote, influenţează semnalele modulate prin metodele analizate în lucrare.
- Justificaţi utilizarea structurii de modulare / demodulare în amplitudine a impulsurilor în schemele dispozitivelor de automatizare (exemplu: traductoare de tensiune şi curent).

LUCRAREA 4

TRANSMITEREA SEMNALELOR CODIFICATE PE CANALE FĂRĂ PERTURBAŢII. STUDIUL EXPERIMENTAL AL ALGORITMILOR DE COMPRESIE A DATELOR

1.OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- analiza algoritmilor de compresie a datelor de tip Shannon-Fano şi Huffman;
- implementarea acestor algoritmi folosind tehnica de calcul.

2. BREVIAR TEORETIC

2.1 Definirea codului

Considerând o sursă discretă, fără memorie, având alfabetul

$$S=\{s_1,s_2,\ldots,s_N\} \quad (4.1)$$

cu probabilităţile asociate:

$$p=\{p_1,p_2,\ldots,p_N\};p_i=p(s_i), \quad (4.2)$$

şi ansamblul finit de semne al alfabetului canalului

$$X=\{x_1,x_2,\ldots,x_q\}, \quad (4.3)$$

iar ansamblul de secvenţe finite de litere x_{a1}, x_{a2}, …, x_{an} este reuniunea extensiilor lui X:

$$X^*=\cup X^n;n\geq 1, \quad (4.4)$$

atunci orice aplicaţie $s\rightarrow X^*$ se numeşte **codarea** (**codificarea**) ansamblului S prin alfabetul X. Fiecare element al lui X^*, notat s_i^* , ce corespunde lui s_i , este un cuvânt de cod, caracterizat prin lungimea sa, anume numărul de litere care îl formează:

$$n(s_i)=n_i \quad (4.5)$$

Totalitatea cuvintelor de cod constituie **codul lui S**; în aceste condiţii, un text constituit din secvenţe de mesaj

$$m_j=\{s_{i1},s_{i2},\ldots,s_{in}\}, \tag{4.6}$$

este codificat prin secvenţe de cuvinte de cod:

$$m_j^*=\{s_{i1}^*,s_{i2}^*,\ldots,s_{in}^*\}. \tag{4.7}$$

2.2. Criterii de apreciere a unui cod

Deoarece la transmiterea mesajelor costul explorării unui sistem de transmisie creşte liniar cu timpul, un criteriu convenabil de apreciere a unui cod este lungimea medie a unui cuvânt $n=\sum_{i=1}^{n} n_i p_i$, unde p_i sunt probabilităţile asociate alfabetului sursei iar n_i este numărul de litere din cuvântul de cod cu indicele i; n este un parametru care precizează compactitatea codului, fiind evident că n trebuie să fie cât mai mic. Totodată, n este limitat inferior de condiţia de asigurare a entropiei informaţionale pe simbol al alfabetului de cod:

$$n \geq n_{min}=\frac{H}{\log q} \tag{4.8}$$

unde H reprezintă entropia sursei. În aceste condiţii, eficienţa unui cod este definită de formula

$$\eta=\frac{\overline{n}_{min}}{\overline{n}} \tag{4.9}$$

Aplicaţie

Se consideră pentru sursa prezentată în tabelul 4.1. următoarele probabilităţi de apariţie a mesajelor: p_1=0,5; p_2=0,25; p_3=p_4=0,125. Se cere să se determine eficienţa fiecărui cod.

Tabelul 4.1

Mesaje	A	B	C	D
s_0	00	0	0	0
s_1	01	10	01	10
s_2	10	110	011	110
s_3	11	1110	0111	111

Entropia sursei va fi $H=-\frac{1}{2}\times\log\frac{1}{2}-\frac{1}{4}\times\log\frac{1}{4}-\frac{1}{8}\times\log\frac{1}{8}=\frac{7}{4}$biti.

Pentru codul A, lungimea medie a cuvântului va fi n_A=2, deci:

$$\eta_A=\frac{7/4}{2\log 2}=7/8;\quad \rho_A=1-\eta_A=1/8.$$

Codurile B şi C au aceeaşi lungime medie,
$n_B = n_C = 0,5\times11 + 0,25\times2 + 0,125 \times 4 = 1,875$ şi

$$\eta_B = \eta_C = \frac{1,75}{1,875} = \frac{14}{15}; \quad \rho_B = \rho_C = \frac{1}{15}\ .$$

Pentru codul D avem:
$n_D = 0,5\times1 + 0,25 \times 2 + 0,125 \times 3 = 1,75;$

$$\eta_D = \frac{1,75}{1,75\log 2} = 1; \quad \rho_D = 0.$$

2.3. Metode de elaborare a codurilor compacte

2.3.1. Metoda Shannon – Fano.

Această metodă presupune aranjarea mesajelor în ordinea descrescătoare a probabilităţilor lor de apariţie: $p_1 \geq p_2 \geq \ldots \geq p_n$ şi aşezarea lor în două grupe, având sumele probabilităţilor cât mai apropiate.
Se codifică fiecare grupă cu 0, respectiv cu 1, apoi se repetă procedura în cadrul fiecărei grupe, până când în fiecare rămân doar două mesaje.

În continuare, este exemplificată această metodă prin prezentarea posibilităţii de codare a unei surse de informaţie cu opt mesaje.

Fie următoarele mesaje s_i şi probabilităţile de apariţie asociate p_i:

mesajul s_1: p_1=0,35;	mesajul s_5: p_5=0,06;
mesajul s_2: p_2=0,23;	mesajul s_6: p_6=0,05;
mesajul s_3: p_3=0,14;	mesajul s_7: p_7=0,04;
mesajul s_4: p_4=0,10;	mesajul s_8: p_8=0,03;

Schematic, aplicarea metodei conduce la următoarea diagramă:

0,35						0,35–cod 00
0,23	0,58(0)					0,23–cod 01
0,14		0,14				0,14–cod 100
0,1		0,1	0,24(10)			0,1–cod 101
0,06	0,42(1)	0,06		0,06		0,06–cod 1100
0,05		0,05		0,05	0,11(110)	0,05–cod 1101
0,04		0,04	0,18(11)	0,04		0,04–cod 1110
0,03		0,03		0,03	0,07(111)	0,03–cod 1111

2.3.2. Metoda Huffman (varianta Schvartz).

Metoda se bazează pe proprietatea că într-un cod optimal, la $p_i > p_j$ corespunde relaţia $n_i > n_j$ şi ia în consideraţie, în plus, cerinţa ca cele mai puţin probabile două mesaje să aibă aceeaşi lungime.

Tehnica codării constă în rescrierea tabelei de probabilităţi, intercalând în ordine descrescătoare suma ultimelor două mesaje (cele mai puţin probabile), iteraţia oprindu-se când în tabel rămân două mesaje.
Combinaţia de cod se citeşte urmând sensul invers, de la sfârşit către început.
Exemplificarea foloseşte acelaşi şir de opt mesaje de la 3.3.1.

0,35	0,35	0,35	0,35	0,35	0,4	0,6 | 0
0,23	0,23	0,23	0,23	0,25	0,35 | 0	0,4 | 1
0,14	0,14	0,14	0,17	0,23 | 0	0,25 | 1	
0,1	0,1	0,11	0,14 | 0	0,17 | 1		
0,06	0,07	0,1 | 0	0,11 | 1			
0,05	0,06 | 0	0,07 | 1				
0,04 | 0	0,05 | 1					
0,03 | 1						

Codurile obţinute pentru mesaje sunt următoarele:
0,35 – cod 00
0,23 – cod 10
0,14 – cod 010
0,10 – cod 110
0,06 – cod 0110
0,05 – cod 0111
0,04 – cod 1110
0,03 – cod 1111

2.4. Programe demonstrative

2.4.1. Program demonstrativ scris în C^{++} pentru codificarea de tip Shannon – Fano (P1).

```
#include<iostream.h>
#include<conio.h>
#include<alloc.h>
#include<string.h>
#define false 0
#define true 1
```

```
struct nod
{
float x[30];
int sant1, sant2;
int nodterm;
char cod[20];
struct nod *st, *dr;
} *rad;
float x[30], verif[30];
float s0,s1;
int i, j, n;
char cod[20], c1[20];
struct nod *insert(struct nod *rad, float s, char cod[20], int sant1, int
sant2)
{ static int i;
if(s!=0)
{
if(!rad)
{
rad=(struct nod*)malloc(sizeof(struct nod));
rad->inf=s;
strcpy(rad->cod,"");
strcat(rad->cod, cod);
rad->sant1=sant1;
rad->sant2=sant2;
if(rad->sant2==rad->sant1) rad->nodterm=true;
for(i=rad->sant1; i<=rad->sant2; i++)
{
rad->x[i]=x[i];
verif[i]=rad->x[i];
}
rad->st=rad->dr=NULL;
}
else
{ strcpy(cod, c1);
if (s>rad->inf/2)
{
srtcat(cod,"0");
rad->st=insert(rad->st, s, cod, sant1, sant2);
}
else
```

```
{
if (s<=rad->inf/2)
{
strcat(cod, "1");
rad->dr=insert(rad->dr, s, cod, sant1, sant2);
};
return(rad);
};
void suma(struct nod *rad,float x[30],int sant1,int sant2,float s0,float
s1)
{
float tol;
int st1, st2, st3, st4;
int i,j;
st1=sant1; st4=sant2; s0=s1=0;
tol=rad->inf/10;
if(rad)
{
if(rad->sant1!=rad->sant2)
{
if(rad->sant2==rad->sant1+1)
{
s0=rad->x[rad->sant1];
s1=rad->x[rad->sant2];
st1=st2=rad->sant1;
st3=st4=rad->sant2;
}
        else
        {
        s0=rad->x[st1];
        st2=st1;
        while(s0<=rad->inf/2-tol)
        {
        s0+=rad->x[st2+1];
        st2++;
        };
        st3=st2+1;
        j=st3;
        for(i=st3; i<=st4; i++)
        {
        s1+=rad->x[j];
```

```
        j++;
        };
        };
strcpy(c1, rad->cod);
rad=insert(rad, s0, cod, st1, st2);
rad=insert(rad, s1, cod, st3, st4);
suma(rad->st, x, rad->st->sant1, rad->st->sant2, s0, s1);
suma(rad->dr, x, rad->dr->sant1, rad->dr->sant2, s0, s1);
}
void del(struct nod*rad)
{
if(rad)
{
del(rad->st);
del(rad->dr);
free(rad);
};
void rsd(struct nod*rad)
{
if(rad)
{ cout.width(8);
cout.precision(3);
if(rad->nodterm==true)
cout<<"Nodul"<<rad->inf<<"are codul:"<<rad->cod<<endl;
rds(rad->dr);
rds(rad->st);
}
int main()
{
int i; char c[5];
clrscr();
cout<<"Introduceti n=";
cin>>n;
for(i=0; i<n; i++)
{
cout<<"x["<<i<<"]=";
cin>>x[i];
}
for(i=0; i<n; i++)
cout<<" "<<x[i];
cout<<endl;
```

```
rad=insert(rad, 1, "", 0, n-1);
suma(rad, x, rad->sant1, rad->sant2, s0, s1);
rds(rad);
del(rad);
cout<<"Gata"<<endl;
getch();
return 0;
}
```

2.4.2. Program de compresie a datelor folosind codificarea Huffman

Este prezentată în continuare o secvenţă de program pentru compactarea datelor folosind arbori **Huffman** (**P2**).

Algoritmul care stă la baza acestui program are în vedere construirea unui arbore care în nodurile terminale reprezintă caracterele ce apar în fişier (max. 256), astfel organizat încât caracterele care apar mai frecvent să fie la o distanţă mai mică de rădăcina arborelui. Practic, dacă fiecărui nod terminal îi ataşăm o valoare f_i (i=0…255) care reprezintă frecvenţa de apariţie a caracterului i în fişierul original şi o valoare n_i (i=0…255) care reprezintă lungimea drumului de la rădăcină până la nodul terminal, corespunzător caracterului i, poate fi construit un arbore a cărui proprietate este că suma produselor $f_i \times n_i$; i=0…255, este minimă.

În fişierul compactat va fi păstrată numai imaginea arborelui şi drumurile de la rădăcină spre nodurile terminale corespunzătoare caracterelor din fişier. Condiţia de minimalitate a sumei de mai sus asigură obţinerea unui fişier mai scurt, cu condiţia ca arborele de codificare memorat la început să nu fie mai lung decât spaţiul câştigat prin compactare.

Secvenţa de program prezentată în continuare pleacă de la o mulţime de mesaje, împreună cu probabilităţile lor de apariţie în şirul respectiv şi construieşte arborele Huffman corespunzător.

```
#include<stdio.h>
#include<malloc.h>
#define NUMAR MAGIC 0X1234      /*pentru recunoaşterea arhivei */
#define OCTET 8
typedef unsigned char byte;
```

```
typedef unsigned short intword;
typedef struct ar{
struct ar *sptr;                /*legătura stânga */
struct ar *dptr;                /*legătura dreapta */
byte sval;                      /*caracterul din stânga */
byte dval;                      /*caracterul din dreapta */
} ARBORE;                       /*structura unui nod de arbore */

ARBORE *arbore[256];            /*arbore de caractere */
long frecvente[256];            /*tabloul frecventelor de apariţie
                                a caracterelor in fişier */
byte *codificari[256];          /*codificările caracterelor */
word magic = NUMAR MAGIC;
long lungimeFisier = 0L;
void *radacina;                 /*rădăcina arborelui de refacere
*/
void scrieBit(byte bit, FILE *iesire)
 /*scrie un bit 0 sau 1 în fişierul arhiva */
static byte biti = 0;
static byte contor=0;
biti=(biti<<1)|((bit)?:0);
if(++contor==OCTET){
putc(biti, iesire);
biti=contor=0;
 }
}
void scrieOctet(byte biti, FILE *iesire)   /*scrie opt biţi
                                consecutivi  în fişierul arhiva */
{
byte masca = 0x80;
while(masca) {
scrieBit(biti & masca, iesire);
masca>>=1;
 }
}
int citesteBit(FILE *iesire)        /*citeşte următorul bit din
                                        fişierul de intrare */
static word biti = 0;
static byte contor=0;
if(contor==0){
biti=getc(intrare);
```

```
contor=OCTET;
}
contor--;
biti<<=1;
return (biti&0x100)?1:0;
}
byte citesteOctet(FILE *intrare)          /*citeşte următorii opt biţi
din fişierul de intrare si ii asamblează intr-un octet */
{
byte rezultat=0;
int contor=OCTET;
while(contor--){
rezultat<<=1;
rezultat = citesteBit(intrare);
}
return rezultat;
}
void memoreazaDrum(int caracter, byte pozitie, byte *drum)
/*memorează drumul în arborele de codificare de la rădăcina
până la caracter */
{
int I;
codificari[caracter]=(byte*)malloc(pzitie+1);
codificari[caracter][0]=pozitie;
for(I=0,i<pozitie;i++)
codificari[caracter][i+1]=drum[i];
}
void decodificaArbore(ARBORE *adresa)
/*memorează structura arborelui de codificare în tabloul de
codificări ale caracterelor */
{
static byte drum[256];
static byte pozitie=0 ;
drum[pzitie++]=0;
if(adresa->sptr)
decodificaArbore(adresa->sptr);
  else
    memoreazaDrum(adresa->sval, pozitie, drum);
    drum[pozitie-1]=1;
if(adresa->dptr)
decodificaArbore(adresa->dptr);
```

```
  else
    memoreazaDrum(adresa->dval, pozitie, drum);
    pozitie--;
  }
int construiesteArbore(FILE *intrare)
/*construieşte arbore codificare bazat pe algoritmul Huffman /
{
ARBORE *temporar;
unsigned long lmin1, lmin2;
short min1, min2;
int i, c;
while((c=getc(intrare))!=EOF) {
frecvente[c]++;
lungimeFisier++;
}
```

3. MOD DE LUCRU

- se porneşte sistemul de calcul;
- se intră în subdirectorul de lucru al grupei;
- se lansează mediul de programare;
- se introduce programul demonstrativ **P1**;
- se compilează, se linkeditează şi se lansează în execuţie programul **P1**;
- se generează codul Shannon-Fano pentru lista de mesaje considerată la 3.2.1.;
- se completează secvenţa de program propusă **P2** în aşa fel încât să calculeze pe baza arborelui Huffman construit cuvintele de cod;
- se compilează, se linkeditează şi se lansează în execuţie programul **P2**;
- se generează codul Huffman pentru lista de mesaje considerată la 3.2.2.

Lucrarea se consideră încheiată când ambele programe P1 şi P2 sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Definiţi operaţia de codificare a unui ansamblu de simboluri.
- Comparaţi cele două metode prezentate din punct de vedere al eficienţei lor.

- Care este principiul metodei Shannon-Fano? Dar al codării de tip Huffman?
- Se consideră o sursă având alfabetul X = (x_1, x_2, x_3, x_4, x_5, x_6), cu probabilităţile asociate p_1=0,4; p_2=0,2; p_3=0,15; p_4=0,1; p_5=0,1; p_6=0,05.

Se cere să se analizeze eficienţa codurilor Shannon-Fano şi Huffman ale acestui alfabet.

LUCRAREA 5

STUDIUL EXPERIMENTAL AL ALGORITMILOR DE CODIFICARE ARITMETICĂ

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- studiul algoritmilor de codificare aritmetică a secvenţelor de simboluri;
- studiul posibilităţilor de implementare a acestor algoritmi folosind tehnica de calcul.

2. BREVIAR TEORETIC

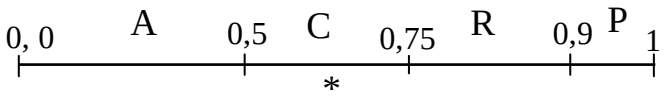
2.1.Algoritmul de codificare aritmetică

Caracteristica esenţială a codificării aritmetice este aceea că realizează o codificare a secvenţelor de simboluri şi nu a fiecărui simbol individual. Ea asociază fiecărei secvenţe de simboluri un subinterval al numerelor reale cuprinse între 0 şi 1 şi face codificarea secvenţei printr-un număr oarecare aparţinând acestui subinterval.

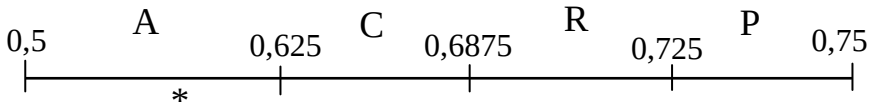
Se prezintă, în continuare, o exemplificare a modului de codificare aritmetică.

Fie caracterele **A, C, R, P** cu probabilităţile de apariţie 0,5; 0,25; 0,15; 0,1. Din aceste caractere se formează mesajul **CAP** pe care urmează să-l codificăm.
Algoritmul de codificare aritmetică presupune parcurgerea următorilor paşi:

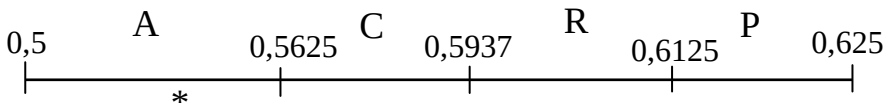
- se împarte subintervalul (0,1) proporţional cu probabilităţile simbolurilor din setul de litere şi se reţine intervalul (0,5; 0,75), corespunzător primului simbol din mesaj –C



- se împarte apoi subintervalul reţinut proporţional cu probabilităţile de apariţie a simbolurilor din setul de litere şi se reţine (0,5; 0,625) care corespunde simbolului A



- se procedează analog pentru cel de-al treilea simbol, P, obţinând intervalul (0,6125; 0,625).



- pentru codificarea mesajului poate fi aleasă o valoare oarecare din acest subinterval, de exemplu 0,62.

După cum se observă, fiecare nou simbol al mesajului îngustează intervalul de alegere a cuvântului de cod. Un subinterval de mărime s poate fi reprezentat prin –**logs** biţi.

Deoarece dimensiunea intervalului final este produsul probabilităţilor simbolurilor sursă, se poate scrie:

$$-\log s = -\sum_{j=1}^{n} \log p(mesajsursaj) = -\sum_{i=1}^{m} p(s_i) \log p(s_i) \qquad (5.1)$$

unde:

n = lungimea secvenţei de mesaje;
m = numărul de mesaje distincte.

Se subliniază în acest mod faptul că numărul de biţi generaţi de codificarea aritmetică este egal cu entropia H.

Pentru refacerea mesajului la recepţie (decodificare), se determină succesiv subintervalele în care se încadrează codul, deducându-se succesiunea de simboluri.

Pentru terminarea procesului de decodificare este însă necesară cunoaşterea lungimii mesajului. În acest scop, se include de regulă în mesaj un simbol special, care marchează sfârşitul lui.

2.2. Soluţii de implementare

În ipotezele:
- întregul şir de date este considerat un mesaj pentru care se va calcula un cuvânt de cod corespunzător;
- simbolurile sursei sunt notate cu 1, 2, 3, …, fiecare cu o probabilitate de apariţie **prob[i]**;
- mesajul se încheie cu un simbol terminator special,

se calculează probabilităţile cumulate. Acestea vor fi păstrate într-un vector, denumit **prob cum [număr simboluri +1]**, astfel încât simbolului i să-i corespundă domeniul de probabilităţi între **prob cum [i]** şi **prob cum [i-1]**.

Vectorul **prob cum [ ]** are elementele în ordine descrescătoare, cu proprietatea prob cum [0] = 1 (acumularea se face de la dreapta spre stânga).

Intervalul curent va fi dat de [inf, sup), iniţializat la [0,1) atât la codificare, cât şi la decodificare.

Se propun următoarele frecvenţe simplificate de program pentru algoritmii de codificare şi decodificare:

```
    void codifică simbol (simbol, prob cum) {
          domeniu = inf–sup;

          sup = inf+domeniu*prob cum[simbol-1];
          inf = inf+domeniu*prob cum[simbol];
     }
    int decodifică simbol (prob cum) {
    găseşte simbol astfel ca prob cum [simbol]<=(valoare-inf)/(inf-
sup)<
          prob cum [simbol-1];
    domeniu=sup-inf;
```

```
sup = inf+domeniu*prob cum[simbol-1];
inf = inf+domeniu*prob cum[simbol];
return simbol;
}
```

Funcţiile **codifică simbol** şi **decodifică simbol** sunt apelate pentru fiecare simbol al mesajului, inclusiv pentru terminator. Iteraţia ia sfârşit când simbolul returnat este terminatorul mesajului, codul său aflându-se în **valoare**.

În elaborarea programului ce implementează codificarea aritmetică este necesară luarea în consideraţie a mai multor aspecte specifice.

Primul se referă la modelul datelor, în număr de 256, reprezentabile prin valorile tipului **char**, de la 0 la 255; simbolul terminator va avea rezervat indexul 257. Pentru transmitere şi recepţie, pe parcursul obţinerii cuvântului cod, mărimile inf, sup ce delimitează domeniul curent se păstrează ca întregi. Pe măsură ce domeniul codului se îngustează, biţii superiori de sup şi inf devin coincidenţi şi pot fi transmişi imediat, nemaifiind schimbaţi de o ulterioară îngustare a domeniului. Considerând **val max** cea mai mare valoare de cod posibilă şi **mediu** jumătatea sa, secvenţa de program corespunzătoare va fi:

```
for (;;) {
    if(sup<mediu) {                 /* inf şi sup, sub mediu */
       ieşire bit(0);               /* bitul comun are valoarea 0 */
       inf=2*inf;
       sup=2*sup+1;
        }
     else if(inf>=mediu){           /* inf şi sup, peste mediu*/
        ieşire bit(1);              /* bitul comun are valoarea 1*/
        inf=2*(inf-mediu);
        sup=2*(sup-mediu)+1;
            }
      else break;
    }
```

Condiţia de terminare a ciclului este **inf**<**mediu**<=**sup**.

În ceea ce priveşte probabilităţile acumulate, este necesară scalarea acestora în intervalul [inf, sup] pentru fiecare caracter transmis. Este important ca intervalul [inf, sup] să fie suficient de larg, astfel încât simboli diferiţi să nu poată conduce la acelaşi întreg din acest interval. Rezultă, deci, că intervalul trebuie să fie cel puţin egal cu **prob max** (valoarea maximă a probabilităţii cumulate).

O altă problemă este cea legată de depăşirile inferioară şi superioară ale valorilor probabilităţilor cumulate. Cât timp domeniul acoperit de **prob cum** este sub un sfert din cel prevăzut de valoarea codului, nu poate apare depăşire inferioară, ceea ce corespunde condiţiei

prob max<=(val max+1)/4+1.

Problema depăşirii superioare nu se ia, practic, în consideraţie, deoarece probabilitatea cumulată nu depăşeşte **prob max**.

Transmisia trebuie încheiată cu simbolul terminal, urmat de un număr suficient de biţi pentru a asigura încadrarea şirului codificat în domeniul final.

Sunt prezentate în continuare două secvenţe de program scrise în limbajul C^{++} care reprezintă interfaţa cu modelul şi programul corespunzător codificării unui simbol.

```
/*INTERFAŢA CU MODELUL*/
/*mulţimea simbolurilor ce pot fi codificate */

#define no car 256
#define simbol eof (no car + 1)
#define no simboli (no car + 1)
/* tabele de translatare caractere-index*/
extern int car index [no car];
extern unsigned char index car [no simboli + 1];
/* tabela de probabilităţi cumulate */
#define prob max16383
extern int prob cum [no simboli + 1];
void start model ();
void start iesire biti ();
void start codificare ();
void codifica simbol (int, int [ ]);
```

```
void termina codificare ();
void termina iesire biti ();

/* PROGRAM DE CODIFICARE A UNUI SIMBOL*/

void codifica simbol (int simbol, int prob cum[ ]) {
long domeniu;                          /* dimensiunea domeniului
curent */
domeniu = (long) (sup – inf) + 1;      /* îngustează domeniul conform
                                       simbolului curent */
sup = inf – (domeniu *prob cum [simbol – 1] )/prob cum [0] – 1;
inf = inf – (domeniu *prob cum [simbol ] )/prob cum [0];
for( ; ; ){

if(sup<mediu)
        {
        fprintf(temp, ″iesire 0″);
        bit plus următori (0);
        }
        else if (inf>=mediu) {      /*atribuie 1 pentru jumătatea
superioară */
        fprintf(temp, ″iesire 1″);
        bit plus urmatori (1);
        inf  - = mediu;             /* deplasează capătul stâng la zero */

        sup - =mediu;
        }
        else if(inf>=sfert&&sus<treisf) {
        fprintf(temp,  ″ un bit opus ″);
        biti urmatori + = 1;
        ine - = sfert;
        sup - = sfert;
        }
        else break;
        inf = 2* inf;                  /* scalează domeniul codului */
        sup = 2* sup + 1;
        }
}
void termina codificare(){
biti urmatori + = 1;                   /atribuie doi biţi corespunzător */
```

```
   if (inf<sfert) bit plus urmatori (0);        /*sfertului conţinut de
domeniul */
        else bit plus urmatori (1);             /*curent */
}
static void bit plus urmatori (int bit) {
iesire bit(bit);
        while (biti urmatori>0) {
        iesire bit(!bit);
        biti urmatori - = 1;
        }
}
```

3. MOD DE LUCRU

- se completează cele două secvenţe de programe propuse cu părţile de program referitoare la:
  - declaraţiile necesare la codificare;
  - algoritmul de codificare numerică;
  - ieşire rezultate,

reunind toate aceste secvenţe într-un program de codificare **PCOD**;

- se elaborează un program de decodificare **PDECOD** care să conţină:
  - declaraţiile necesare de decodificare;
  - o secvenţă de program de decodificare simbol;
  - o secvenţă de intrări de date codificate.
- se asamblează cele două programe elaborate într-unul singur, de codificare /decodificare **PROG**;
- se porneşte sistemul;
- se intră în subdirectorul de lucru al grupei;
- se lansează mediul de programare;
- se introduce programul **PROG**;
- se compilează, se linkeditează şi se lansează în execuţie;
- se generează o codificare aritmetică pentru câteva secvenţe de simboluri, urmată de decodificare şi se apreciază corectitudinea codificării.

Lucrarea se consideră încheiată când programul **PROG** este funcţional.

4. CHESTIUNI DE STUDIAT

- Ce particularitate prezintă codificarea aritmetică, în comparaţie cu celelalte metode de codificare?
- Descrieţi modalitatea de generare a codului prin metoda de codificare aritmetică.
- Care este rolul simbolului terminal special folosit la codificarea aritmetică?
- Cu cât este egal numărul de biţi generaţi de codificarea aritmetică?
- Cum se procedează pentru refacerea mesajului la recepţie?

LUCRAREA 6

TRANSMITEREA DATELOR PE CANALE CU PERTURBAŢII FOLOSIND CODURI DETECTOARE ŞI CORECTOARE DE ERORI

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- analiza modalităţilor de detectare si corectare a erorilor care apar la transmisia datelor;
- generarea codurilor de tip Hamming şi a codurilor ciclice folosind tehnica de calcul;
- aprecierea corectitudinii transmisiei pentru aceste tipuri de codificare.

2. BREVIAR TEORETIC

2.1. Coduri Hamming

Codurile Hamming sunt coduri de grup la care fiecare cuvânt de cod de **n** biţi conţine **m** biţi informaţionali şi **k** = n-m biţi de control, eficienţa maximă a acestor coduri fiind obţinuta în situaţia în care $n = 2^k-1$.

Biţii de control sunt determinaţi in funcţie de biţii informaţionali prin relaţii de condiţie care asigură paritatea prin suma modulo 2.

Codurile Hamming pot fi:
- *sistematice*, situaţie în care primii *m* biţi sunt informaţionali, iar următorii *k=n-m* sunt biţi de control;
- *ponderate*, în cazul în care biţii de control apar pe poziţii care reprezintă puteri ale lui 2: 1,2,4…

Generarea codului se poate face prin două metode:
- direct din relaţiile de condiţie care furnizează biţii de control şi apoi îi inserează în poziţiile corespunzătoare din cuvântul de cod;
- prin metode matriceale, mai ales pentru coduri cu cuvinte de cod lungi.

Se propune studiul detaliat al codului Hamming de tip (7, 4) (n=7, m=3).

Se consideră cuvântul de 7 biţi $u = a_1a_2a_3a_4a_5a_7$ şi cuvântul recepţionat $u' = a_1'a_2'a_3'a_4'a_5'a_7'$.

Erorile singulare care pot să apară la recepţie, în care e_1, e_2, e_3 sunt simbolurile pentru cei trei biţi de test (corespunzători celor 8 erori posibile), sunt prezentate în tabelul 6.1.

Tabelul 6.1

eroare asupra	**e_3**	**e_2**	**e_1**
nici unei cifre	0	0	0
a_1'	0	0	1
a_2'	0	1	0
a_3'	0	1	1
a_4'	1	0	0
a_5'	1	0	1
a_6'	1	1	0
a_7'	1	1	1

Din examinarea tabelului rezultă condiţiile ca e_1, e_2, e_3 să aibă valoarea 1:

$$\begin{aligned} e_1 &= a_1' + a_3' + a_5' + a_7' \\ e_2 &= a_2' + a_3' + a_6' + a_7' \\ e_3 &= a_4' + a_5' + a_6' + a_7' \end{aligned} \qquad (6.1)$$

Pentru a determina biţii de control a_5, a_6, a_7 în funcţie de biţii informaţionali, este suficient să punem condiţia de nonexistenţă a erorii, anume:

$$a_1' = a_i,\ i = 1\ldots7 \qquad (6.2)$$

şi deci $e_1 = e_2$, astfel încât se obţin următoarele relaţii ;

$$\begin{aligned} a_1 + a_3 + a_5 + a_7 &= 0 \\ a_2 + a_3 + a_6 + a_7 &= 0 \\ a_4 + a_5 + a_6 + a_7 &= 0 \end{aligned} \qquad (6.3)$$

care generează condiţiile:

$$\begin{aligned} a_5 &= a_2 + a_3 + a_4 \\ a_6 &= a_3 + a_6 + a_7 \\ a_7 &= a_1 + a_2 + a_4 \end{aligned} \qquad (6.4)$$

În cazul în care codurile Hamming se scriu în formă ponderată, adică în forma în care biţii de control ocupă poziţiile corespunzătoare puterilor crescătoare ale lui 2: a_1, a_2, a_4,…, relaţiile de control devin:

$$\begin{aligned} a_1 &= a_3 + a_5 + a_7 \\ a_2 &= a_3 + a_6 + a_7 \\ a_4 &= a_5 + a_6 + a_7 \end{aligned} \qquad (6.5)$$

Condiţiile de control pot fi scrise şi sub formă matriceală şi anume:

$$[a_5\ a_6\ a_7] = [a_1\ a_2\ a_3\ a_4] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad (6.6)$$

sau $< t > = < s > \Gamma_{m.k}$, (6.7)

cu $< t >$: matricea de test

$< s >$: matricea biţilor infomaţionali.

De regulă, se generează direct din combinaţiile de cod:

$$[a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7] = [a_1\ a_2\ a_3\ a_4] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad (6.8)$$

sau $< u > = < s > G_{4\ x\ 7} = < s > [\ I_{4\ x\ 4}\ \Gamma_{4\ x\ 3}]$, (6.9)

cu $G_{4 x 7}$: matrice generatoare de cod

Se propune programul demonstrativ **P1** pentru ilustrarea modalităţii de generare a codului sistematic Hamming de tip (7,4).

```
/* codarea hamming 7/4 */
#include<conio.h>
#include <stdio.h>
#define nmax 15
typedef enum {false, true}  boolean;
void citeste(int, int*);
void gen_cod_control(int, int*);
void meniu(void);
void main()
{
  char  car;
  int nr_test=0;
  //clrscr();
  do{
        if(!nr_test) meniu();
                else if(car==0x0D)  meniu();
                nr_test = 1;
  } while((car=getch())!=0x1B);
}
void meniu()
{
  clrscr();                          /* m-numărul de biţi informaţionali */
  int cc,m, cuv_cod[nmax];           /*cuvântul ce conţine biţii
                                     informaţionali */
  printf("Codarea hamming 7/4:\n");
  printf("Introduceti bitii informationali: \n");
  citeste(4,cuv_cod);
  printf("\n");
  gen_cod_control(4,cuv_cod); puts("\n");
  printf("\n");
  printf("\nReluare... Press ENTER..");
  printf("\nExit....PressESC...");
}
void citeste(int p, int cuv[])
{
  char  c; int i;
  boolean valid;
  fflush(stdin);
  do{
        valid = true;
        i=1;
```

```
        while(i<=p){
          c= getche();
          switch(c) {
                case '0':cuv[i]=0;
                                                break;
                case '1':cuv[i]=1;
                                                break;
                case 0x0D:if(i<=p)
                          { valid=false;
                                i=p+1;
                          }
                                                break;
                default: valid=false;
                                                break;
          }
          i++;
        }
        if(!valid) printf("\nGresit...Reintroduceti:");
  } while(!valid);
}
void gen_cod_control(int m, int cuv[]){
  int t, k=3, n;
  n=m+k;                                /* lungimea cuvantului de cod */
  cuv[5]=cuv[2]^cuv[3]^cuv[4];
  cuv[6]=cuv[1]^cuv[3]^cuv[4];
  cuv[7]=cuv[1]^cuv[2]^cuv[4];
  printf("\nBitii de control sunt:");
  for(t=5;t<=n;t++) printf("%d ",cuv[t]); // repl
  printf("\Intregul cuvant de cod este:");
  for(t=1;t<=n;t++) printf("%i ",cuv[t]);
}
```

2.2 Coduri ciclice

Codurile ciclice sunt coduri liniare - deci de grup - închise pentru permutarea circulară a cifrelor si care conţin simultan cuvântul de cod,

$$u = a_1a_2\ldots a_{n-1}a_n \qquad (6.10)$$

şi permutările succesive:

$$\begin{aligned} u^{(1)} &= a_2a_3\ldots a_na_1 \\ u^{(2)} &= a_3a_4\ldots a_1a_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ u^{(n)} &= a_1a_2\ldots a_{n-1}a_n = u. \end{aligned} \qquad (6.11)$$

O primă metodă de construcţie a unui cod ciclic este **metoda directă,** cu cele două variante ale sale:
- prin înmulţire, la care cuvântul de cod u(x) se obţine ca produs între partea semnificativă s(x) si polinomul generator g(x);
- prin împărţire, la care partea de test reprezintă restul împărţirii parţii semnificative s(x) la polinomul generator g(x);

A doua modalitate de construcţie a codurilor ciclice este **metoda matricii generatoare** care generează codul plecând de la o bază de cuvinte de dimensiune egală cu dimensiunea codului.

A treia metodă de construcţie este cea a **matricei de control,** construită pe baza condiţiilor de control de paritate aplicate polinomului ortogonal codului.

Cea de-a patra metodă este cea a rădăcinilor **polinomului generator** g(x) bazată pe îndeplinirea condiţiei ca aceste rădăcini să fie şi rădăcinile polinomului asociat cuvântului de cod u(x).

Se propune analiza în vederea implementării software a metodei directe, prin împărţire. Algoritmul de generare a codului este următorul:
- se separă fiecare cuvânt de cod în partea semnificativă s(x) şi partea de test t(x):

$$s(x) = a_1x^{m-1}+a_2x^{m-2}+\ldots+a_mx^k; \quad (6.12)$$

$$t(x) = a_{m-1}x^{k-1}+a_{m-2}x^{k-2}+\ldots+a_n;$$

$$u(x) = s(x) + t(x); \quad (6.13)$$

cu îndeplinirea condiţiei ca

$$s(x) + t(x) = g(x)Q(x), \quad (6.14)$$

g(x) fiind polinomul generator (polinomul unic care divide pe x^n-1);

- partea de test se obţine ca restul împărţirii părţii semnificative la polinomul generator.

Aplicaţie

Se consideră un cod ciclic de tip Γ(7,4) cu polinomul generator $g(x) = x^3+x+1$.

Pentru combinaţiile posibile de biţi ce definesc partea semnificativă *s(x),* partea de test *t(x)* şi întregul cuvânt de cod *u* obţinut sunt prezentate în tabelul 6.2.

Cele două subansamble obţinute au patru simboluri pentru partea semnificativă *s(x)* şi respectiv trei simboluri pentru partea de test *t(x).*

Coloana finală indică forma cuvântului de cod *u* rezultat prin concatenarea biţilor informaţionali cu cei de test.
Se verifică astfel şi dimensiunea codului, în relaţie directă cu numărul m de biţi informaţionali $N = 2^m$; în cazul de faţa, $N = 2^4 = 16$.

Tabelul 6.2.

s	s(x)	t(x)	U
0000	0	0	0000000
1111	$x^6+x^5+x^4+x^3$	x^2+x+1	1111111
0001	x^3	$x+1$	0001011
0010	x^4	x^2+x	0010110
0101	x^5+x^3	x^2	0101100
1011	$x^6+x^4+x^3$	0	1011000
0110	x^5+x^4	1	0110001
1100	x^6+x^5	x	1100010
1000	x^6	x^2+1	1000101
0011	x^4+x^3	x^2+1	0011101
0111	$x^5+x^4+x^3$	x	0111010
1110	$x^6+x^5+x^4$	x^2	1110100
1101	$x^6+x^5+x^3$	1	1101001
1010	x^6+x^4	$x+1$	1010011
0100	x^5	x^2+x+1	0100111
1001	x^6+x^3	x^2+x	1001110

În vederea generării software a codurilor ciclice prin metoda prezentată, se propune următoarea secvenţă de program (**P2**):

```
#include<stdlib.h>
#include<stdlib.h>
#include<conio.h>
#include <stdio.h>
#include <string.h>
#define true 1
#define false 0
#define bool int
char  *sb, *s;
int scoef[4], tcoef[4];
int checksb(char  *sb, int scoef[4])
{  static int i, len;
  static bool find;
  len=strlen(sb);
  find=f        alse;
  for(i=0;i<len;i++){
```

```
        if((sb[i]!='0')&&(sb[i]!='1'))
                find = true;
///Determinarea coeficientilor polinomului s(x)
  if(!find)
        for(i=0;i<len;i++)
          if(sb[i]=='0') scoef[i] =0;
          else scoef[I]=1;
return find;
```

3. MOD DE LUCRU

- pe baza programului demonstrativ **P1** se elaborează un program (**PG**) de generare a codului Hamming pentru cazul general (n, m);
- se completează secvenţa de program **P2** cu partea corespunzătoare determinării coeficienţilor polinomului de test *t(x),* pentru generarea unui cod ciclic;
- se generează codul Hamming (7, 4);
- se generează codul Hamming în variantele (3,1); (15,11); (31,26);
- se generează codul ciclic pentru câteva din exemplele prezentate în tabelul 6.2;

Lucrarea se considera încheiată când toate programele sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Ce este un cod Hamming? Cum se pot clasifica aceste coduri?
- Cum se generează un cod de tip Hamming?
- Definiţi codurile ciclice. În ce constă algoritmul de generare a acestor coduri?
- Apreciaţi eficienţa codului ciclic comparativ cu cea a codului Hamming.
- Biţii de control ai unui bloc (8, 4) sunt generaţi de relaţiile :

$$c_5 = i_1 + i_2 + i_4$$
$$c_6 = i_1 + i_2 + i_3$$
$$c_7 = i_1 + i_3 + i_4$$
$$c_8 = i_1 + i_3 + i_4,$$

unde i_i sunt biţii informaţionali. Să se determine matricea generatoare de cod şi matricea de test.

LUCRAREA 7

IMPLEMENTAREA UNUI PROTOCOL DE COMUNICAŢIE SERIALĂ PENTRU INTERFAŢA RS 232-C

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- recapitularea noţiunilor de bază privind comunicaţia serială;
- însuşirea modului de programare a interfeţei seriale **RS 232-C**;
- utilizarea programului **Comander Link** pentru gestionarea resurselor de memorie pe suport magnetic pentru două sisteme **PC/AT**.

2. BREVIAR TEORETIC

2.1. Comunicaţia pe linii seriale. Aspecte de principiu

Realizarea serviciilor pe care sistemele de transmisie le oferă utilizatorilor presupune transmisia datelor între oricare două noduri dintr-o reţea.

Comunicarea între programe sau echipamente din noduri diferite implică existenţa unei conexiuni fizice care să permită transmisia serială a datelor în formă digitală, ca o succesiune de biţi. La acest nivel se realizează codificarea semnalului, stabilirea şi desfiinţarea conexiunilor, conform modului de transmisie (duplex / semiduplex).

Până în prezent s-a impus utilizarea ca mediu de comunicaţie a reţelei telefonice analogice (clasice). în figura 7.1 este prezentată structura principială a unei conexiuni "punct la punct" pe linia telefonică.

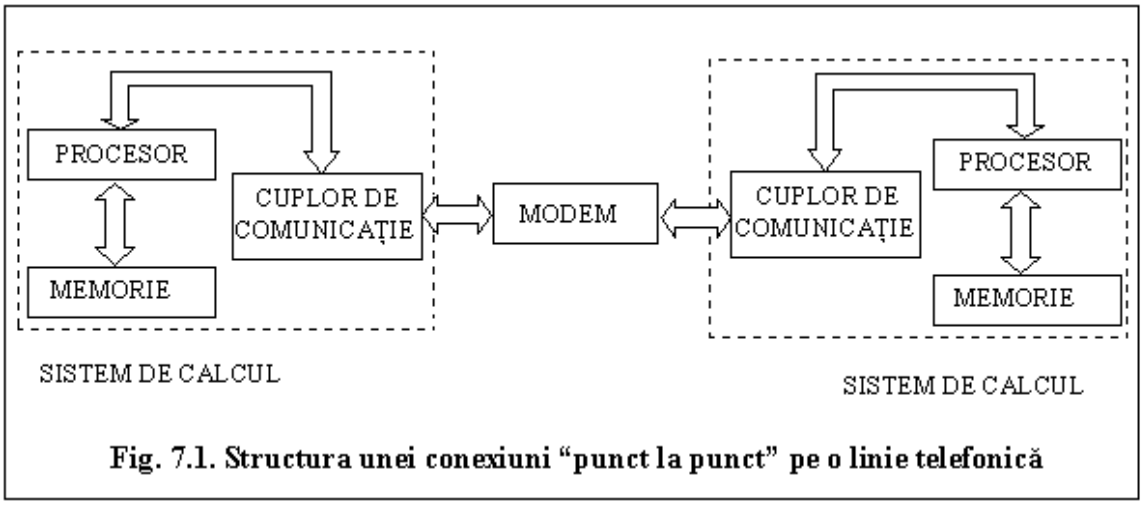


Fig. 7.1. Structura unei conexiuni "punct la punct" pe o linie telefonică

Datorită caracteristicilor mediului de comunicaţie, se impune transmisia în curent alternativ, prin modularea unui semnal sinusoidal; dispozitivul care realizează conversia de la forma digitală a semnalului la cea analogică şi invers este **modem-ul.** Legătura cu modemul este asigurata de un cuplor de comunicaţie care realizează serializarea datelor, precum şi controlul funcţionarii modem-ului.

2.2. Interfaţa serială standard RS 232-C. Caracteristici.

Legătura dintre calculator şi modem se conformează unui protocol de comunicaţii, ceea ce presupune **un ansamblu de reguli, proceduri şi funcţii** îndeplinite pentru asigurarea unei totale compatibilităţi între cele două sisteme.

Cel mai răspândit standard relativ la interfaţa calculator-modem este **EIA RS 232-C** ("Reference Standard 232 version C"). Pentru aplicaţii specifice industriale se utilizează si alte standarde: **RS 422, RS 485**.

Standardul **RS 232-C** prevede legarea calculatorului la modem printr-un conector cu 25 pini, tip DB 25 Canon. Viteza de transmisie este limitată la 20000 bps; cablul de legătură trebuie sa aibă o lungime de 3m şi o capacitate totală pe linie sub 2500pF. Nivelele logice "0" si "1" sunt reprezentate prin tensiuni diferite ale semnalelor electrice, cuprinse între –15V si +15V.

Caracteristicile funcţionale ale interfeţei **RS 232-C** se referă la rolul diferitelor linii de legătură intre calculatoare şi modem. În figura 7.2 sunt reprezentate liniile folosite în cazul cuplării microcalculatoarelor la mediile de comunicaţie seriale.

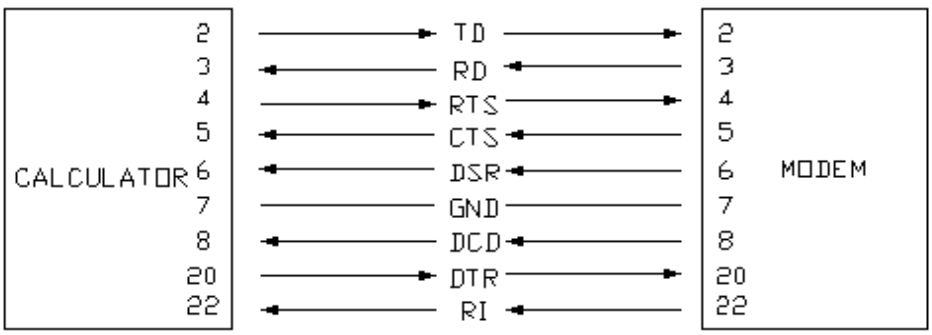


Fig. 7.2. Conectarea calculator – modem prin cablu " one to one "

Semnificaţia notaţiilor liniilor de semnal este următoarea:

TD – Transmitted Data (emisie de date)
RD – Received Data (recepţie de date)
RTS – Request to Send (cerere de emisie)
CTS – Clear to Send (gata de emisie)
DSR – Data Set Ready (modem pregătit)
DTR –Data Terminal Ready (calculator pregătiţi)
DCD – Data Carrier Detect (detecţie purtătoare)
RI – Ring Indicator (indicator pregătit)
GND – Ground (masa)

Două calculatoare aflate la mică distanţă unul faţă de celălalt pot fi conectate direct, fără modem-uri şi fără linia telefonică dintre ele. Se utilizează în acest scop un cablu special tip **NMC** ("Null Modem Cabel") care realizează conexiunile indicate în figura 7.3.

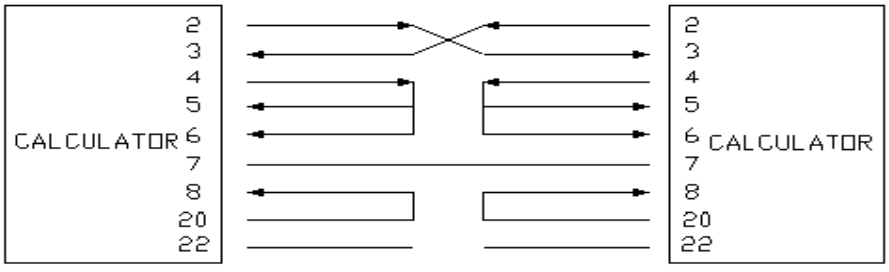


Fig. 7.3. Conectarea a două calculatoare prin cablu NMC

Datele sunt transmise serial utilizând semnale **TD** şi **RD** sub forma unor pachete de 5 . . . 8 biţi informaţionali împreună cu 2-3 biţi de

sincronizare (**START, STOP**) şi, eventual, un bit de paritate, ca în diagrama din figura 7.4.

Prezenţa în structura pachetului a bitului de paritate reprezintă o încercare de realizare a unui protocol liber de erori (în cadrul nivelului fizic); totuşi, controlul corectitudinii comunicaţiei se face la un nivel funcţional superior, cu asigurarea sincronizării calculator-modem. Această sincronizare este realizată fie pe cale software, prin transmiterea periodică a unor caractere speciale de control (protocol **Xon-Xoff**)**,** fie pe cale hardware, prin intermediul semnalelor **RTS** si **CTS** (protocol **RTS/CTS**).

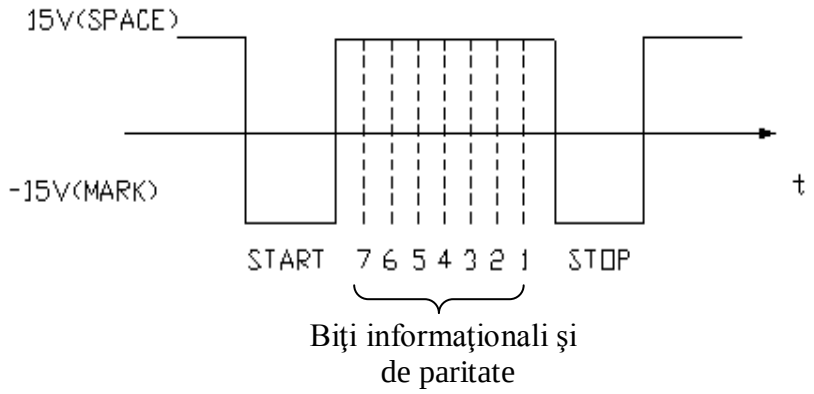


Fig. 7.4. Structura unui pachet de date transmis pe o linie serială

2.3. Programarea interfeţei seriale

Realizarea cuplorului de comunicaţie se bazează pe circuite standard **UART** ("Universal Asynchronous Receiver Transmitter") care asigură transmiterea independentă a fiecărui caracter pe linie, cea ce conferă caracterul de asincronism al comunicaţiei. În prezent, echipamentele de calcul utilizează circuite **UART tip 8250, 16450 si 16550** care uşurează mult controlul comunicaţiei prin program, reducându-l la citirea / scrierea unor porturi la intrare / ieşire sau a unor locaţii de memorie. Structura generală a unui astfel de circuit este prezentată in figura 7.5.

Circuitul **UART** are un număr de registre interne de date, de control şi de stare adresabile separat de microprocesorul sistemului de calcul. Prin înscrierea şi citirea acestora se realizează **programul interfeţei seriale**.

La nivelul procesorului central, controlul comunicaţiei prin **RS 232-C** constă în stabilirea parametrilor de comunicaţie, transmiterea si recepţionarea caracterelor si citirea stării prin apelul direct **BIOS** sau prin utilizarea unor funcţii de bibliotecă.

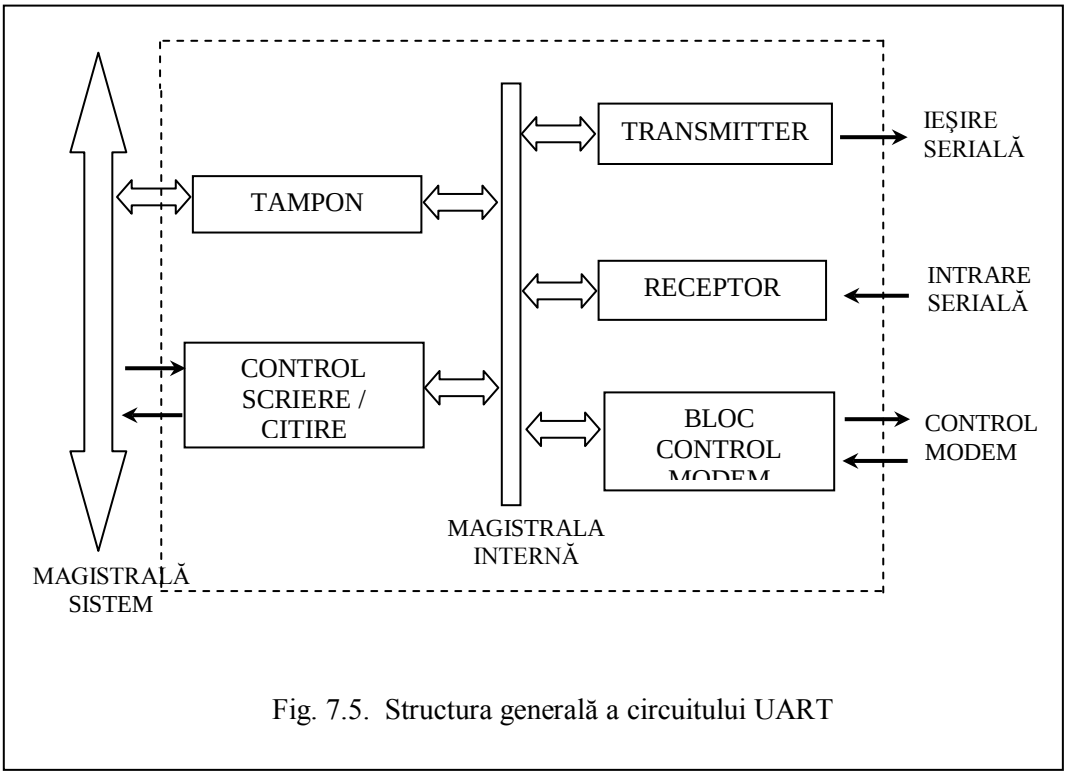


Fig. 7.5. Structura generală a circuitului UART

Funcţia **bioscom** este o funcţie de bibliotecă a limbajul C++ ce realizează comunicarea printr-un anumit port de intrare / ieşire. Prototipul funcţiei este declarat in fişierul header **bios.h** de tipul:

int bioscom (**int cmd, char byte, int port**);

unde:

- **port** este numărul portului serial (0 pentru **COM1**, 1 pentru **COM2**, etc.);
- **cmd** este comanda, având valorile posibile:
  - 0- stabilirea parametrilor de comunicaţie la valoarea din **byte**;
  - 1- transmitea caracterelor din **byte**;
  - 2- recepţia unui caracter;
  - 3- citirea stării.

Pentru comanda de stabilire a parametrilor de comunicaţie, **byte** este o combinaţie de valori selectate din următoarele grupuri:

1. Număr de biţi de paritate:
   - 0x02 - 7 biţi;
   - 0x02 - 8 biţi;

2. Paritate:
   0x00 - fără paritate;
   0x08 - paritate impară;
   0x18- paritate pară.
3. Număr de biţi de stop:
   0x00 - 1 bit;
   0x04 - 2biţi.
4. Viteza de transmisie:
   0x00 - 110 bauds;
   0x20 - 150 bauds;
   0x40 - 300 bauds;
   0x60 - 600 bauds;
   0x80 - 1200 bauds;
   0xA0 - 2400 bauds;
   0xC0 - 4800 bauds;
   0xE0 - 9600 bauds.

Valorile 0xab sunt, conform convenţiei din limbajul C++, valori hexazecimale. Structura unui cuvânt de control se obţine prin efectuarea operaţiei SAU pe biţi între valorile corespunzătoare fiecărui parametru.

Exemplu.
Pentru o comunicaţie la 9600 bauds (0xE0), cu 8 biţi de date (0x03), fără controlul parităţii (0x00) şi un bit de stop (0x00), valoarea lui **byte** în program se calculează **byte = (0xE0/0x03/0x00/0x00)**.
Pentru a stabili aceşti parametri de comunicaţie pentru portul serial COM1 se apelează funcţia **bioscom** astfel: **bioscom (0, byte, 0)**.
Pentru a trimite un caracter (ex. litera A) pe linia serială, după stabilirea parametrilor se apelează din nou funcţia **bioscom** pentru transmisie:

```
byte = ´A´;
bioscom (1, byte, 0);
```

Citirea stării circuitului 8250 se realizează prin apelul

```
stare = bioscom (3, 0, 0);
```

Valoarea variabilei **stare** depinde de evenimentele diferite care au avut loc la emisie /recepţie (de exemplu, recepţionarea corectă a unui caracter este semnalată de valoarea hexazecimală 0x00).
Dacă după citirea stării circuitului 8250 se indică recepţionarea unui caracter, atunci citirea sa efectivă se face prin:

```
out = bioscom (2, 0, 0) & 0xFF
```

şi variabila **out** va conţine caracterul recepţionat.

Folosirea adecvată a funcţiei **bioscom ()** permite astfel un control total al comunicaţiei seriale, una dintre aplicaţii fiind emisia /recepţia caracterelor în paralel cu supravegherea liniei de comunicaţie serială.

2.4. Programul ″Commander Link ″

Programul ″**Commander Link** ″ din pachetul de programe **Norton Commander** realizează o legătură de tip **Master /Slave** între două sisteme de calcul, permiţând pentru calculatorul **Master** accesul complet la resursele de memorare pe suport magnetic (hard-disk, floppy-disk) aferente calculatorului **Slave**.

Programul se apelează – pentru fiecare calculator – prin selectarea opţiunii **linK** din meniul **Left** sau **Right**. În fereastra de dialog care apare se cer informaţii utilizatorului asupra portului folosit (**COM1** sau **COM2**) şi a regimului de lucru (**Master /Slave**) pentru fiecare calculator. Selectarea opţiunii **linK** din cadrul acestei ferestre are semnificaţia indicării conectării pentru calculatorul respectiv; pe ecran este afişată o casetă de dialog ce anunţă utilizatorul ce regim de lucru trebuie să aleagă pentru celălalt calculator, oferind în acelaşi timp posibilitatea întreruperii legăturii.

După efectuarea aceleiaşi succesiuni de operaţii şi pentru cel de-al doilea sistem de calcul, se realizează conectarea efectivă, în urma căreia calculatorul **Slave** se blochează, toate comenzile fiind date numai de la calculatorul **Master**, acesta având acces total la fişierele sistemului **Slave**.

În panelul **linK** este vizualizată structura pe directoare şi subdirectoare corespunzătoare calculatorului **Slave**, utilizatorul având libertatea de a crea /şterge fişiere şi /sau directoare, conform metodologiei cunoscute la **Norton Commander**.
Desfiinţarea conexiunii se poate face numai de la calculatorul **Master** prin apelarea din nou a opţiunii **linK** şi indicarea în fereastra de dialog a opţiunii închiderii sesiunii de lucru.

3. MOD DE LUCRU

3.1. Utilizând limbajul **Borland C++** se elaborează un program pentru interfaţa serială RS 232-C având următoarele funcţii:
- stabilirea parametrilor de comunicaţie;

- transmiterea caracterelor introduse de la tastatură;
- afişarea caracterelor recepţionate.

3.2. Se cuplează cablul serial NMC la conectorii porturilor seriale ale celor două calculatoare.

3.3. Se cuplează la reţeaua 220V /50Hz cele două calculatoare.

Obs. Conectarea trebuie realizată la două prize situate pe aceeaşi fază a tensiunii de alimentare pentru a se evita distrugerea interfeţelor seriale.

3.4. Pentru lucrul cu programul **Commander Link** se execută următoarele operaţii:

- se lansează mediul de programare **Norton Commander**;
- se selectează opţiunea **linK** din meniul **Left /Right** pentru fiecare din cele două calculatoare;
- se alege pentru calculatorul **Master** portul **COM2**, iar pentru calculatorul **Slave** portul **COM1**;
- comenzile de lucru vor fi date numai calculatorului **Master**, calculatorul **Slave** fiind blocat;
- se desfiinţează conexiunea numai de la calculatorul **Master** prin apelarea opţiunii **linK** şi indicarea opţiunii de închidere a sesiunii de lucru.

3.5. Pentru programarea interfeţei seriale **RS 232-C** se execută următoarea succesiune de operaţii:

- se lansează mediul de programare **Borland C++**;
- se introduce programul elaborat la 4.1.;
- se compilează, se linkeditează şi se lansează în execuţie.

4. CHESTIUNI DE STUDIAT

- Descrieţi structura unei conexiuni punct la punct pe o linie telefonică analogică.
- Ce este protocolul de comunicaţie? Daţi exemple de standarde de interfaţă calculator – modem.
- Care sunt liniile de semnal folosite la cuplarea microcalculatoarelor la mediile de comunicaţie serială?
- Care este rolul bitului de paritate în pachetul de date transmis pe linia serială?
- Caracterizaţi circuitul UART.
- Care este utilitatea programului Commander Link din pachetul de programe Norton Commander?

LUCRAREA 8

STUDIUL EXPERIMENTAL AL ALGORITMILOR DE CRIPTARE A DATELOR CU CHEIE ASIMETRICĂ

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- analiza algoritmilor de criptare a datelor cu cheie asimetrică de tip R.S.A.
- implementarea acestor algoritmi folosind tehnica de calcul.

2. BREVIAR TEORETIC

2.1.Algoritmi de criptare cu cheie publică. Algoritmul R.S.A

Deoarece toţi criptologii au considerat întotdeauna ca de la sine înţeles faptul că atât pentru criptare cât şi pentru decriptare se foloseşte aceeaşi cheie şi că aceasta trebuie distribuită tuturor utilizatorilor sistemului, părea a exista întotdeauna aceeaşi problemă inerentă: cheile trebuiau protejate împotriva furtului dar, în acelaşi timp, ele trebuiau să fie distribuite, astfel încât nu puteau fi sechestrate într-un seif de bancă.

În 1976, doi cercetători, Diffie şi Hellman, au propus un tip radical nou de criptosistem în care cheile de criptare şi decriptare sunt diferite, iar cheia de decriptare nu poate fi dedusă din cheia de criptare. În propunerea lor, algoritmul (cheia) de criptare `E` şi algoritmul (cheia) de decriptare `D`, trebuiau să satisfacă trei cerinţe. Aceste trei cerinţe pot fi exprimate simplificat după cum urmează:

a) `D(E(P)) = P;`
b) Este mai mult decât dificil să se deducă `D` din `E;`
c) `E` nu poate fi spart printr-un atac cu text clar ales.

Respectându-se aceste trei condiţii, nu există nici un motiv pentru ca E, respectiv cheia de criptare, să nu poată fi făcută publică, din contră, toţi utilizatorii ce au adoptat acest model de criptosistem trebuie să-şi facă cunoscute cheile publice.

Plecând de la aceste trei condiţii, în anul 1978 a fost inventat criptosistemul **RSA**. Denumirea lui provine de la numele celor trei inventatori ai acestui mod de codificare a informaţiei: Ron **R**ivest, Adi **S**hamir şi Leonard **A**delman.

Acest criptosistem stă şi astăzi în diverse variante, la baza sistemelor de protecţie a datelor şi transmisiilor de informaţii.

Pentru obţinerea cheilor (*cheia privată* şi *cheia publică*), se procedează astfel:

1. Se aleg două numere prime p şi q;
2. Se calculează $n = p \times q$ şi $z = (p-1) \times (q-1)$;
3. Se alege un număr e relativ prim cu z, astfel încât $1 < e < z$;
4. Se găseşte un număr d, astfel încât $(e \times d) \bmod z = 1$ şi $1 < d < z$.

Numărul e se numeşte exponent public iar d exponent privat.

În urma operaţiilor de mai sus obţinem două perechi de numere (n,e) şi (n,d) ce reprezintă cheia publică, respectiv cheia privată.

Pentru a obţine mesajul criptat c, mesajul clar m (privit ca şir de biţi), se împarte în k blocuri de text clar. Fiecărui bloc m_i, $(i = \overline{0, k-1})$ i se aplică funcţia:

$$c_i(n, e) = m^e \bmod n, \text{ unde } i = \overline{0,k-1} \qquad (8.1)$$

Astfel şirul c obţinut reprezintă mesajul criptat.

Pentru decriptare (obţinerea mesajului clar m), criptogramei c i se aplică funcţia:

$$m_i(n, d) = c_i^d \bmod n, \text{ unde } i = \overline{0,k-1} \qquad (8.2)$$

Din motive de securitate numerele p şi q se şterg, după generarea cheilor publice şi private.

Securitatea metodei se bazează pe dificultatea factorizării numerelor mari. Dacă un criptanalist ar putea factoriza numărul n (public cunoscut), atunci el ar putea obţine p şi q, iar din acestea pe z. Cu acesta din urmă aflat, se restrâng şi variantele pentru e, respectiv d. Din fericire, matematicienii încearcă de peste 300 de ani să factorizeze numere mari şi experienţa acumulată sugerează că aceasta este o problemă mai mult decât dificilă.

În figura 8.1 este ilustrat modul de funcţionarea al algoritmului R.S.A.

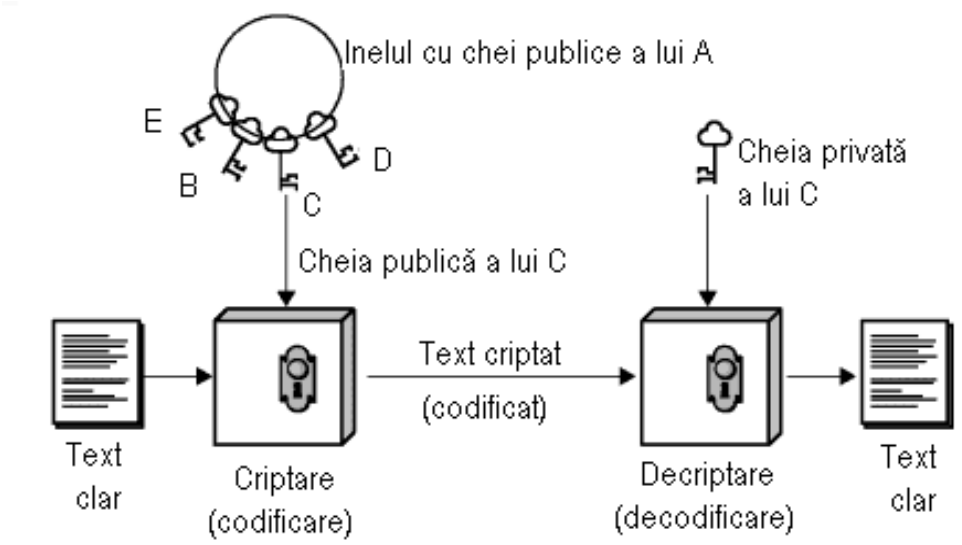


Fig. 8.1. Schema bloc a funcţionării criptosistemului R.S.A.

Persoana A deţine un grup (inel) de chei publice ale persoanelor B, C, D şi E. Pentru a transmite un mesaj criptat persoanei C, criptează mesajul (textul clar) cu cheia publică a lui C. Persoana C primeşte mesajul criptat de la A şi îl decodifică cu cheia sa privată, obţinând astfel textul clar original.

În cadrul grupului de persoane A, B, C, D, E, fiecare deţine cheile publice ale celuilalt şi le utilizează pentru transmiterea mesajelor. De asemenea, fiecare persoană îşi utilizează cheia privată (personală) pentru a decripta mesajele primite, astfel numai destinatarul mesajului poate citi mesajul.

Această modalitate de criptare este utilizată atunci când expeditorul este interesat ca nimeni (nici măcar cei din grup) nu va putea citi mesajul clar. Dezavantajul acestei metode este că oricine din grup poate trimite mesaje, iar destinatarul nu poate fi 100% sigur de identitatea expeditorului.

Un exemplu de calcul, pur didactic, este prezentat în continuare.

Se aleg două numere prime:
p = 61 şi
q = 53
(Pentru o criptare eficientă p şi q se aleg mai mari de 10^{100})

Se calculează:
$n = p \cdot q = 61 \cdot 53 = 3233$ şi
$z = (p-1) \cdot (q-1) = 60 \cdot 52 = 3120$

Conform algorimului se alege e = 17
Tot conform algoritmului se alege d = 2753

Cheia publică (n,e)=(3233,17)
Cheia privată (n,d)=(3233,2753)

Se alege mesajul clar (de criptat) m=123.

Codificarea este
$c = m^e \bmod n = 123^{17} \bmod 3233 =$
$= 337587917446653715596592958817679803 \bmod 3233 = 855$

Decodificarea este
$m = c^d \bmod n = 855^{2753} \bmod 3233 =$ `123`

3.2. Semnătura digitală R.S.A.

Avantajul algoritmului R.S.A. este că poate fi utilizat şi pentru semnarea mesajelor expediate. Acest tip de semnătură este cunoscut sub numele de *semnătură digitală*.

Semnătura digitală este folosită pentru a identifica autorul unui mesaj.

Schema bloc a sistemului de transmitere a mesajelor semnate digital este reprezentată în figura 8.2.

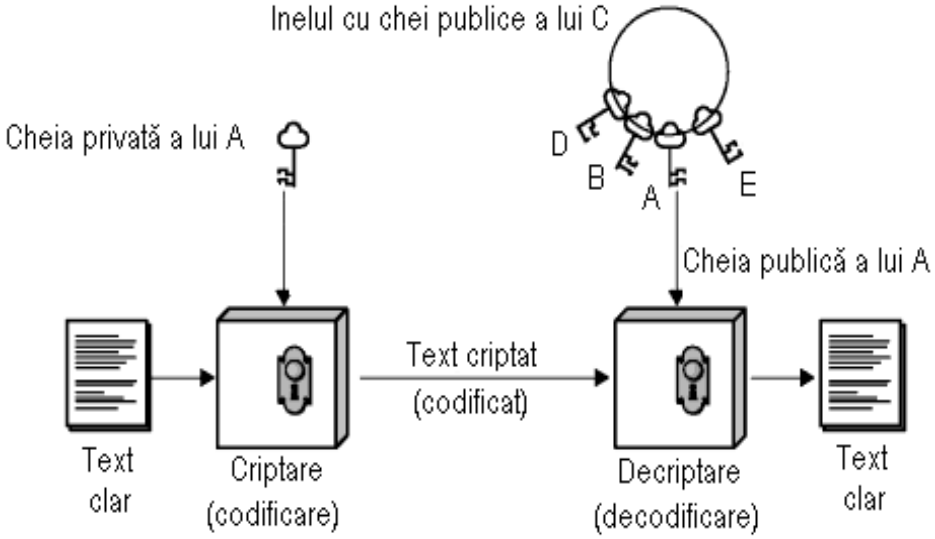


Fig. 8.2. Schema bloc a sistemului de transmitere a mesajelor semnate digital

După cum este cunoscut, fiecare membru al unui grup deţine cheia publică a celorlalţi membri ai grupului şi cheia sa privată. Pentru a transmite persoanei C un mesaj criptat şi semnat, persoana A criptează mesajul (textul clar) cu cheia sa privată (personală). Persoana C primeşte mesajul criptat de la A şi îl decodifică cu cheia publică a acesteia (cheia publică a lui A), obţinând astfel textul clar original.

Spre deosebire de prima schemă de criptare, în acest caz, toate persoanele din grup pot decodifica mesajul dar nu există nici un dubiu în privinţa identităţii expeditorului.

A, B, C, D şi E pot fi atât persoane cât şi programe, ceea ce înseamnă că acest sistem de criptare poate fi folosit atât de:
- persoane în vederea transmiterii de maseje (de exemplu transmiterea de e-mail-uri, fişiere în orice format),
cât şi de
- programe (pachete de programe client/server) în vederea transmiterii de informaţii de la aplicaţia server la aplicaţia client şi/sau invers, în cadrul reţelelor de tip LAN (Local Area Network) sau WAN (World Area Network).

Folosind notaţiile de la metoda de criptare, pentru a obţine mesajul criptat c (criptogramă semnată digital), mesajul clar m (privit ca şir de biţi), se împarte în k blocuri de text clar. Fiecărui bloc $m_i, (i = \overline{0, k-1})$ i se aplică funcţia

$$c_i(n, d) = m^d \bmod n, \text{ unde } i = \overline{0,k-1} \tag{8.3}$$

Astfel şirul c obţinut reprezintă mesajul criptat semnat.

Pentru decriptare (obţinerea mesajului clar m), criptogramei c i se aplică funcţia:

$$m_i(n, e) = c_i^e \bmod n, \text{ unde } i = \overline{0,k-1} \tag{8.4}$$

3.3. Programe pentru criptarea datelor

Programul „Algoritmul de criptare R.S.A” este structurat în trei module.

Primul modul este destinat generării perechilor de chei (cheie publică – cheie privată).

La execuţia butonului GENERAREA CHEILOR, programul generează automat o listă de numere prime cuprinse în intervalul [3, 255]. Din această listă alege aleatoriu două numere. Pe baza celor două numere prime alese, se calculează exponentul public şi cel privat. Perechea de chei creată, este salvată în două fişiere ce reprezintă fişierul cheie publică (cu extensie *.kpb) şi cu fişierul cheie privată (cu extensie *.kpv). Atât fişierul cheie publică, cât şi cel cheie privată au o lungime de 8 octeţi (64 biţi).

```
unsigned char NrPrime[255], Prim, i, j, k, p, q;
 unsigned int n, z, e, d, i1, m, mc, md;
 div_t x;

 if((Edit1->Text.Length())&&(Edit2->Text.Length()))
   {
    BitBtn3->Enabled=false; Form1->Enabled=false;
    // Stabilirea numerelor prime de la 3 la 255
    k=0;
    for(i=3;i<255;i+=2)
       {
```

```
     Prim=1;
        for(j=2;j<=(i/2+1);j++)
           if(!(i%j))
             Prim=0;
        if(Prim)
          {
           NrPrime[k]=i;
             k++;
          }
       }
    // Alegerea a doua numere prime p si q, din cele
stabilite
    do
      {
       p=GeneratorPQ(NrPrime, k);
       q=GeneratorPQ(NrPrime, k);
      }while((p==q)||(p*q<=255));
    // Se calculeaza n=p*q
    n=p*q;
    //Se calculeaza z=(p-1)*(q-1)
    z=(p-1)*(q-1);
    // Alegerea lui e
    e=AlegereE(z);
    // Alegerea lui d
    i1=0;
    do{
       i1++;
       if(i1>z)
         {
          //Realegerea lui p si q
          do{
             p=GeneratorPQ(NrPrime, k);
             q=GeneratorPQ(NrPrime, k);
            }while((p==q)||(p*q<=255));
          // Se recalculeaza n si z
          n=p*q;
          z=(p-1)*(q-1);
          //Realegerea lui e
          e=AlegereE(z);
          i1=0;
         }
       i=1;
       d=random(z-2)+2;
       if((e*d)%z==1)
          i=0;
       if(d==e) i=1;

      }while(i);
```

```
 ScrieCheile(Edit1->Text.c_str(), Edit2->Text.c_str(),
n, e, d);
    Form1->Enabled=true; BitBtn3->Enabled=true;

 ShowMessage("Fisierele cheie publica si cheie privata
au fost create.");
  }else
     ShowMessage("Trebuie specificare: \n-Numele
fisierului cheie publica si \n-Numele fisierului cheie
privata.");

unsigned char GeneratorPQ(unsigned char NrPrime[255],
unsigned char k)
{
 randomize();
 return NrPrime[random(k)];
}

unsigned int AlegereE(unsigned int z)
{
 unsigned char Prim;
 unsigned int e, i;

 do
   {
    e=random(z-3)+3;
    Prim=1;
    for(i=2;i<=(e/2+1);i++)
        if(!(e%i))
          Prim=0;
   }while(!Prim);
 return e;
}

void ScrieCheile(char *FisierCheiePublica, char
*FisierCheiePrivata, unsigned int Modulator, unsigned
int ExponentPublic, unsigned int ExponentPrivat)
{
 unsigned char OctetScris;
 int i;
 ofstream FisierScris;
 // Scriere fisier cheie publica
 FisierScris.open(FisierCheiePublica, ios::binary |
ios::trunc);
 FisierScris.write((unsigned char *) &Modulator,
sizeof(Modulator));
 FisierScris.write((unsigned char *) &ExponentPublic,
sizeof(ExponentPublic));
 FisierScris.close();
```

```
// Scriere fisier cheie privata
 FisierScris.open(FisierCheiePrivata, ios::binary |
ios::trunc);
 FisierScris.write((unsigned char *) &Modulator,
sizeof(Modulator));
 FisierScris.write((unsigned char *) &ExponentPrivat,
sizeof(ExponentPrivat));
 FisierScris.close();
}
```

În al doilea modul se poate realiza operaţia de criptare. În căsuţele de editare se specifică fişierul sursă, numele fişierului destinaţie şi numele fişierului cheie publică. La apăsarea butonului CODIFICĂ se realizează următoarea secvenţă de cod:

```
 ifstream FisierSursa, FisierCheie;
 ofstream FisierDestinatie;
 unsigned char OctetCitit;
 unsigned int OctetScris, n, e;

if(!(Edit3->Text.Length()))
   ShowMessage("A T E N T I E ! ! !\nNu a fost
specificat fisierul sursa (de codificat)!");
  else
    if(!(Edit4->Text.Length()))
       ShowMessage("A T E N T I E ! ! !\nNu a fost
specificat fisierul destinatie.");
      else
        if(!(Edit5->Text.Length()))
           ShowMessage("A T E N T I E ! ! !\nNu a fost
specificat fisierul cheie.");
         else
           {
            BitBtn7->Enabled=false; Form1-
>Enabled=false; _sleep(1);
            FisierCheie.open(Edit5->Text.c_str(),
ios::binary | ios::nocreate);
            FisierCheie.read((unsigned char *) &n,
sizeof(n));
            FisierCheie.read((unsigned char *) &e,
sizeof(e));
            FisierCheie.close();
            FisierSursa.open(Edit3->Text.c_str(),
ios::binary | ios::nocreate);
            FisierDestinatie.open(Edit4->Text.c_str(),
ios::binary | ios::trunc);
            do
```

```
              {

FisierSursa.read((unsigned char *) &OctetCitit,
sizeof(OctetCitit));
               if(FisierSursa.eof())

       break;

OctetScris=CriptareDecriptare(OctetCitit, e, n);
                FisierDestinatie.write((unsigned char
*) &OctetScris, sizeof(OctetScris));
              }while(!FisierSursa.eof());
            FisierDestinatie.close();
            FisierSursa.close();
            Form1->Enabled=true; BitBtn7->Enabled=true;
            ShowMessage("Codificarea a fost
executata!");
           }

unsigned int CriptareDecriptare(unsigned int Baza,
unsigned int Exponent, unsigned int Modulator)
{
 unsigned int i, Mesaj;

 Mesaj=1;
 for(i=0; i<Exponent; i++)
   {
    Mesaj*=Baza%Modulator;
    Mesaj%=Modulator;
   }
// Mesaj%=Modulator;
 return Mesaj;
}
```

Al treilea modul este rezervat operaţiilor de decriptare. Pentru această operaţie, este nesesar a se specifica fişierul criptogramă, calea şi numele fişierului destinaţie şi fişierul cheie privată cu extensia *.kpv.

La apărarea butonului DECODIFICĂ se execută următoarea secvenţă de coduri:

```
 ifstream FisierSursa, FisierCheie;
 ofstream FisierDestinatie;
 unsigned char OctetScris;
 unsigned int OctetCitit, n, d, Octet;

 if(!(Edit6->Text.Length()))
```

```
ShowMessage("A T E N T I E ! ! !\nNu a fost specificat
fisierul sursa (de decodificat)!");
  else
    if(!(Edit7->Text.Length()))

       ShowMessage("A T E N T I E ! ! !\nNu a fost
specificat fisierul destinatie.");
      else
        if(!(Edit8->Text.Length()))
           ShowMessage("A T E N T I E ! ! !\nNu a fost
specificat fisierul cheie.");

       else
           {
            BitBtn11->Enabled=false; Form1-
>Enabled=false; _sleep(1);
            FisierCheie.open(Edit8->Text.c_str(),
ios::binary | ios::nocreate);
            FisierCheie.read((unsigned char *) &n,
sizeof(n));
            FisierCheie.read((unsigned char *) &d,
sizeof(d));
            FisierCheie.close();
            FisierSursa.open(Edit6->Text.c_str(),
ios::binary | ios::nocreate);
            FisierDestinatie.open(Edit7->Text.c_str(),
ios::binary | ios::trunc);
            do
               {
                FisierSursa.read((unsigned char *)
&OctetCitit, sizeof(OctetCitit));
                Octet=CriptareDecriptare(OctetCitit, d,
n);
                if(FisierSursa.eof())
                  break;
                OctetScris=(unsigned char)Octet;
                FisierDestinatie.write((unsigned char
*) &OctetScris, sizeof(OctetScris));
               }while(!FisierSursa.eof());
            FisierDestinatie.close();
            FisierSursa.close();
            Form1->Enabled=true; BitBtn11-
>Enabled=true;

            ShowMessage("Decodificarea a fost
executata!");
           }
```

3. MOD DE LUCRU

- se completează secvenţele de program propuse P1, P2, P3 cu declaraţiile şi celelalte elemente de program necesare;
- se porneşte sistemul de calcul;
- se intră în subdirectorul de lucru al grupei;
- se lansează mediul de programare;
- se introduc secvenţele completate de programe P1, P2, P3;
- se compilează, se linkeditează şi se lansează în execuţie;

Lucrarea se consideră încheiată când toate programele sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Care sunt cele 3 cerinţe propuse pentru criptare şi decriptare de Diffie şi Hellman?
- Cine sunt inventatorii criptosistemului RSA?
- Descrieţi principiul de funcţionare al algoritmului RSA.
- Pentru ce se foloseşte cheia publică şi cheia privată?
- La ce se foloseşte sistemul de criptare RSA?

LUCRAREA 9

STUDIUL EXPERIMENTAL AL ALGORITMILOR DE CRIPTARE A DATELOR CU CHEIE SIMETRICĂ

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:

- analiza algoritmilor de criptare a datelor cu cheie simetrică de tip cifrul lui Caesar, substituţie monoalfabetică, transpoziţie pe coloane, metoda cheilor acoperitoare;
- implementarea acestor algoritmi folosind tehnica de calcul.

2. BREVIAR TEORETIC

2.1. Introducere în criptografie

Criptografia descrie un câmp larg al comunicaţiilor secrete şi se identifică prin totalitatea mijloacelor şi metodelor utilizate pentru protecţia interceptării pasive (înregistrarea mesajului transmis) sau/şi active (modificarea informaţiei sau introducerea de mesaje false pe canalul transmisiunii, între emiţătorul şi receptorul legali).

Mesajele ce trebuie criptate, cunoscute sub numelele de *text clar* sunt transformate printr-o funcţie parametrizată de o *cheie* (*key*). Cheia constă dintr-un şir (relativ) scurt care selectează una dintre mai multe criptări potenţiale.

Ieşirea procesului de criptare cunoscută sub numele de *text cifrat* sau *criptogramă* este apoi transmisă adeseori prin curier sau radio.

Arta de a sparge cifruri se numeşte *criptanaliză*. Arta de a concepe cifruri (*criptografia*) şi cea de a le sparge *(criptanaliza*) sunt cunoscute sub numele colectiv de *criptologie*.

Din punctul de vedere al criptanalistului, problema sa are trei variante principale:
- Când are la dispoziţie o cantitate de text cifrat şi nici un fel de text clar este confruntat cu *problema textului cifrat*;
- Când are la dispoziţie ceva text clar şi textul criptat corespunzător, problema este cunoscută sub numele de *problema textului clar cunoscut*;
- Când criptanalistul poate cripta bucăţi de text clar la propria sa alegere, avem de-a face cu *problema textului clar ales*

În criptografia tradiţională, metodele de criptare au fost împărţite în două categorii:
- cifruri cu substituţie;
- cifruri cu transpoziţie.

În criptografia modernă, clasificarea sistemelor de criptare se realizează în funcţie de cheie. Astfel, acestea se pot clasifica în:
- *Sisteme criptografice cu chei secrete*, cunoscute şi sub numele de *sisteme criptografice simetrice*, care necesită dezvoltarea unor servicii suplimentare de management al cheilor secrete;
- *Sisteme criptografice cu chei publice*, cunoscute şi sub numele de sisteme criptografice asimetrice, care furnizează servicii specializate de autentificare dar sunt în general ineficiente pentru criptările de date corespunzătoare unor mesaje scurte.

Sistemul criptografic este sigur dacă criptanaliza nu îşi atinge obiectivul de determinare a mesajului.
Un *sistem* secret care rezistă la orice atac criptanalitic, indiferent de volumul calculelor care se cer, se numeşte *sigur necondiţionat*.

Sistemul este *sigur computaţional* când se recunoaşte posibilitatea criptanalistului de a intra în posesia mesajului după o cantitate finită de calcule care ocupă un volum de calcul foarte mare, neacoperit din punct de vedere economic.

2.2. Algoritmi de criptare cu cheie simetrică

2.2.1. Cifrul lui Caesar generalizat

Cifrul lui Caesar este un cifru cu substituţie în care fiecare literă din grup este înlocuită pentru deghizare cu o altă literă.

Acest algoritm este unul dintre cele mai vechi cifruri cunoscute şi este atribuit lui Julius Caesar. În această metodă, *A* devine *D*, *B* devine *E*, *C* devine *F*, …, *X* devine *A*, *Y* devine *B*, *Z* devine *C*.

Alfabet mesaj	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabet criptogramă	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

De exemplu, mesajul

ACESTA ESTE UN TEXT CODIFICAT

devine

DFHVWD HVWH XQ WHAW FRGLILFDW.

O mică *generalizare a cifrului lui Caesar* permite alfabetului textului cifrat să fie deplasat cu k litere, în loc de a fi deplasat întotdeauna cu 3 . În acest caz, k devine o cheie pentru metoda generală a alfabetelor deplasate circular.

Matematic, cifrul lui Caesar generalizat se exprimă astfel:

$$C = E_k(M), \tag{9.1}$$

unde:

$M = m_1, m_2, \ldots, m_n$ este mesajul de criptat;

$C = c_1, c_2, \ldots, c_n$ este criptograma rezultată aplicării lui M a funcţiei $E_k(M)$;

$i = \overline{1, n}$, n este lungimea mesajului.

Printr-o transformare liniară a funcţiei de mai sus, se obţine

$$c_i = (m_i + k) \bmod p, \text{ pentru } k \in [1, p - 1] \tag{9.2}$$

unde:

$i = \overline{1, n}$ şi n este lungimea mesajului,

p este lungimea alfabetului,

k este cheia.

Pentru o cheie $k = 3$, se obţine cifrul lui Caesar
Pentru o cheie $k = 9$, alfabetul de 26 de litere ($p = 26$) şi criptograma corespunzătoare lui arată astfel:

Alfabet mesaj	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabet criptogramă	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I

Dacă mesajului

ACESTA ESTE UN TEXT CODIFICAT

i se aplică corespondenţa din tabelul de mai sus, rezultă următorul mesaj criptat (criptogramă):

JLNBCJ NBCN DW CNGC LXMRORLJC

2.2.2.Criptarea prin substituţie monoalfabetică

Principala îmbunătăţire a acestui tip de criptare o reprezintă stabilirea pentru fiecare simbol din textul clar, să spunem pentru simplitate cele 26 de litere de mai sus, o corespondenţă cu o altă literă.

Matematic, dacă există o singură lege de corespondenţă notată cu f (între elementele alfabetului mesajului şi elementele alfabetului criptogramei), substituţia este monoalfabetică.

Pentru mesajul $M = m_1, m_2, \dots, m_n$, se obţine criptograma $C = c_1, c_2, \dots, c_n$:

$$C = E_k(M) = f(m_1), f(m_2), \dots, f(m_n), \tag{9.3}$$

printr-o transformare liniară de forma:

$$c_i = (a \cdot m_i + b) \bmod p, \tag{9.4}$$

unde:

$M = m_1, m_2, \dots, m_n$ este mesajul de criptat

$C = c_1, c_2, \dots, c_n$ este criptograma rezultată aplicării lui M a funcţiei $E_k(M)$,

$i = \overline{1, n}$, n este lungimea mesajului,

p este lungimea alfabetului,

a şi b sunt două numere de tip întreg.

cheia k este dată de ansamblul (a, b)

Criptarea care foloseşte substituţia monoalfabetică este slabă la atacuri criptanalitice (în principal cu text criptat), pentru că identificarea cheii conduce la obţinerea întregului mesaj.

Ca un caz particular, este prezentat **cifrul aleator de substituţie**. Cheia este constituită din 26 de perechi de numere echivalente de forma (a, b), cu $a, b \in \{1, 2, 3, \ldots, 26\}$. Într-un mod pseudoaleator, fierărei litere a alfabetului primar îi corespunde o literă a alfabetului secundar. Literele alfabetului de substituire sunt static independente dar există dezavantaje legate de generarea, transmiterea şi păstrarea cheii.

De exemplu

Alfabet mesaj	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabet criptogramă	Q	W	E	R	T	Y	U	I	O	P	A	S	D	F	G	H	J	K	L	Z	X	C	V	B	N	M

Cheia acestui tip de codificare o reprezintă şirul de 26 de litere corespunzând întregului alfabet.

Ţinând cont de cheia prezentată în tabelul anterior, mesajul

ACESTA ESTE UN TEXT CODIFICAT

devine:

QETLZQ TLZT XF ZTBZ EGROYOEQZ.

2.2.3.Criptarea prin transpoziţie pe coloane

Spre deosebire de cifrurile cu substituţie, cifrurile cu transpoziţie reordonează caracterele, dar nu le deghizează.

În exemplul următor este arătat modul de realizare a criptogramelor folosind un cifru cu transpoziţie pe coloane. Cifrul are drept cheie un cuvânt sau o expresie ce nu conţine caractere repetate. În următorul exemplu cheia este universală.

Textul clar (necodificat) este scris orizontal, pe rânduri. Scopul cheii este să stabilească numărul de coloane şi să ordoneze caracterele, coloana 1 fiind sub litera din cheie cea mai apropiată de începutul alfabetului.

Text clar (mesaj): Acest algoritm de codificare este foarte bun.

u	**n**	**i**	**v**	**e**	**r**	**s**	**a**	**l**
8	**5**	**3**	**9**	**2**	**6**	**7**	**1**	**4**
A	c	e	s	t		a	l	g
o	r	i	t	m		d	e	
c	o	d	i	f	i	c	a	r
e		e	s	t	e		f	o
a	r	t	e		b	u	n	.

Text criptat (criptogramă):

leafntmft eidetg ro.cro r iebsdc uAoceastise

2.2.4.Metoda cheilor acoperitoare

Construirea unui cifru imposibil de spart este actualmente destul de simplă. Tehnica este cunoscută de decenii, având următoarele etape:
- se alege un şir aleatoriu de biţi pe post de cheie;
- se converteşte textul clar într-un şir de biţi;
- pentru a obţine codificarea textului clar, se calculează *xor* între cheie şi textul clar, bit cu bit;
- pentru a realiza decodificarea se calculează tot *xor* între aceeaşi cheie şi textul codificat, tot bit cu bit.

Tabela de adevăr a funcţiei XOR este

X	**Y**	**X xor Y**
0	0	0
0	1	1
1	0	1
1	1	0

Această metodă, cunoscută sub numele de *metoda cheilor acoperitoare (one-time pad)* are următoarele avantaje:
- textul codificat nu oferă criptanalistului nici o informaţie,
- se poate codifica orice tip fişier (text, imagine, sunet, video, bază de date, executabil),
- chiar şi atunci când află metoda de codificare utilizată, criptanalistul nu are şanse să deducă cheia, deoarece, şirul de biţi al cheii poate avea orice lungime în raport cu şirul de biţi corespunzător textului clar.

Cu toată siguranţa pe care o oferă această metodă, ea prezintă şi dezavantaje practice importante:
- cheia nu poate fi memorată, necesitând un suport de preferintă electronic, o copie a ei putând ajunge oricând în posesia unei persoane neautorizate;
- indiferent că este consultată parţial sau în întregime informaţia codificată trebuie decodificată în înregime.
Acesta este un foarte mare dezavantaj în cazul bazelor de date, utilizatorul fiind obligat să o decodifice şi codifice în întregime, indiferent de numărul de înregistrări pe care le consultă/modifică;
- cu cât fişierul codificat este mai mare, cu atât maniputarea lui (consultare/modificare) este mai greoaie, încetinind operarea cu produse software în care este implementată acestă metodă de codificate.

Exemplu

Mesaj: 00101010 10101000 10101010 01111100 01011101 11101111
Cheie: 10010111 00111011 00001111
Criptogramă: 10111101 10010011 10100101 11101011 01100110 11100000

2.3. Programe pentru criptarea datelor

2.3.1.Programul „Cifrul lui Ceasar generalizat“ (P1)

La baza acestui program sunt cele două funcţii de criptare şi respectiv decriptarea caracterelor.

Funcţia *Codificare()* returnează caracterul codificat şi are două argumente:
- unul de tip caracter (el ia valuarea caracterului ce trebuie criptat şi
- unul de tip întreg şi reprezintă cheia după care se realizează criptarea.

Funcţia *DeCodificare()* returnează caracterul decodificat şi are tot două argumente:
- unul de tip caracter (el ia valuarea caracterului ce trebuie decriptat şi
- unul de tip întreg şi reprezintă cheia după care se realizează decriptarea.

Funcţia de codificare în limbajul C++

```
unsigned char Codificare(unsigned char CaracterC, unsigned char
Cheie)
{ unsigned char CaracterR;

 if((CaracterC>=(unsigned char)'A')&&(CaracterC<=(unsigned
char)'Z'))
   { CaracterR=CaracterC+Cheie;
     if(CaracterR>(unsigned char)'Z')
       CaracterR-=(unsigned char)26;   }
  else
    if((CaracterC>=(unsigned char)'a')&&(CaracterC<=(unsigned
char)'z'))
     {CaracterR=CaracterC+Cheie;

       if(CaracterR>(unsigned char)'z')
      CaracterR-=(unsigned char)26;   }
     else CaracterR=CaracterC;
 return CaracterR;
}
```

Funcţia de decodificare în limbajul C++

```
unsigned char DeCodificare(unsigned char CaracterC, unsigned
char Cheie)
{ unsigned char CaracterR;
 if((CaracterC>=(unsigned char)'A')&&(CaracterC<=(unsigned
char)'Z'))
   { CaracterR=CaracterC-Cheie;
     if(CaracterR<(unsigned char)'A')
       CaracterR+=(unsigned char)26;   }
  else
    if((CaracterC>=(unsigned char)'a')&&(CaracterC<=(unsigned
char)'z'))
    { CaracterR=CaracterC-Cheie;
       if(CaracterR<(unsigned char)'a')
         CaracterR+=(unsigned char)26;  }
     else CaracterR=CaracterC;
 return CaracterR;
}
```

2.3.2.Programul „Substituţia monoalfabetică“ (P2)

Funcţia *CodificareMonoalfabetica()* returnează caracterul codificat şi are drept argumente două şiruri de caractere corespunzătoare alfabeului şi cheii, caracterul de codificat şi un număr întreg corespunzător lungimii alfabetului/cheii.

Funcţia de codificare în limbajul C++

```
unsigned char CodificareMonoalfabetica(char Alfabet[251], char
Cheie[251],
                        char OctetNecodificat, int LungimeAlfabet)
{ unsigned char i;
 char OctetCodificat;
 OctetCodificat=OctetNecodificat;
 for(i=0; i<LungimeAlfabet; i++)
   if(OctetNecodificat==Alfabet[i])
     OctetCodificat=Cheie[i];
 return OctetCodificat;
}
```

Funcţia *DecodificareMonoalfabetica()* returnează caracterul decodificat şi are drept argumente două şiruri de caractere corespunzătoare alfabeului şi cheii, caracterul codificat şi un număr întreg corespunzător lungimii alfabetului/cheii.

Funcţia de decodificare în limbajul C++

```
unsigned char DecodificareMonoalfabetica(char Alfabet[251],
            char Cheie[251], char OctetCodificat, int LungimeCheie)

{unsigned char i;
 char OctetDecodificat;
 OctetDecodificat=OctetCodificat;
 for(i=0; i<LungimeCheie; i++)
   if(OctetCodificat==Cheie[i])
     OctetDecodificat=Alfabet[i];
 return OctetDecodificat;
}
```

2.3.3.Programul „Transpozitia pe coloane“ (P3)

Funcţiile *CodificarePrinTranspozitie()* şi *DecodificarePrinTranspozitie()* se utilizează la criptarea respectiv decriptarea unor fişiere de tip text. Ele returnează valoarea 0 dacă s-a executat codificarea sau valoarea 1 dacă fişierul sursă nu există. Ele au argumentele: numele fişierului sursă, numele fişierului destinaţie şi cheia după care se realizează codificarea.

Funcţia de codificare în limbajul C++

```
unsigned char CodificarePrinTranspozitie(char *FisierSursa,
                    char *FisierDestinatie, char *Cheie)
{ unsigned char Eroare=0, OctetCititScris;
 int i, j, k, PozitiiCaractere[251], LungimeCheie;
 long LungimeFisier, PozitieInFisier;
 fstream FisierCitit;
 ofstream FisierScris;
 FisierCitit.open(FisierSursa, ios::binary | ios::in | ios::ate |
ios::nocreate);
 if(!FisierCitit.rdstate())
   {LungimeCheie=strlen(Cheie);
    for(i=0; i<LungimeCheie; i++)
      {j=0;
       for(k=0; k<LungimeCheie; k++)
         if(Cheie[i]>Cheie[k]) j++;
       PozitiiCaractere[j]=i+1;    }
    FisierScris.open(FisierDestinatie, ios::binary | ios::trunc);
    LungimeFisier=FisierCitit.tellp();
    for(i=0; i<LungimeCheie; i++)
      {PozitieInFisier=PozitiiCaractere[i]-1;
       while(PozitieInFisier<LungimeFisier)
         {FisierCitit.seekp(PozitieInFisier);
          FisierCitit.read((unsigned char *) &OctetCititScris,
sizeof(OctetCititScris));
          FisierScris.write((unsigned char *) &OctetCititScris,
sizeof(OctetCititScris));
          PozitieInFisier+=LungimeCheie;    }
       }
    FisierCitit.close();
    FisierScris.close();
```

```
   }
   else Eroare=1;
 return Eroare;
}
```

Funcţia de decodificare în limbajul C++

```
unsigned char DecodificarePrinTranspozitie(char *FisierSursa,
                                char *FisierDestinatie, char *Cheie)
{unsigned char Eroare=0, OctetCititScris;
 int i, j, k, PozitiiCaractere[251], LungimeCheie;
 long LungimeFisier, m, LocatieGrupe[251], PozitieInFisier;
 div_t LungimeGrupe;
 fstream FisierCitit;
 ofstream FisierScris;
 FisierCitit.open(FisierSursa, ios::binary | ios::in | ios::ate |
ios::nocreate);
 if(!FisierCitit.rdstate())
   {LungimeCheie=strlen(Cheie);
    for(i=0; i<LungimeCheie; i++)
      {j=0;
       for(k=0; k<LungimeCheie; k++)
         if(Cheie[i]>Cheie[k]) j++;
       PozitiiCaractere[j]=i+1; }
    LungimeFisier=FisierCitit.tellp();
    LungimeGrupe=div(LungimeFisier, LungimeCheie);
    for(i=0; i<LungimeCheie; i++)
      {LocatieGrupe[i]=0;
       for(j=0; j<LungimeCheie; j++)
         if(PozitiiCaractere[j]<PozitiiCaractere[i])
           {LocatieGrupe[i]+=LungimeGrupe.quot;
            if((j+1)<=LungimeGrupe.rem)
              LocatieGrupe[i]+=1;  }
      }
    FisierScris.open(FisierDestinatie, ios::binary | ios::trunc);
    for(m=0; m<=LungimeGrupe.quot; m++)
      for(i=0; i<LungimeCheie; i++)
        { if((m==LungimeGrupe.quot)&&(i>=LungimeGrupe.rem))
          PozitieInFisier=LungimeFisier;
         else
```

```
            PozitieInFisier=LocatieGrupe[i]+m;
          if(PozitieInFisier<LungimeFisier)
            {FisierCitit.seekp(PozitieInFisier);
             FisierCitit.read((unsigned char *) &OctetCititScris,
sizeof(OctetCititScris));
             FisierScris.write((unsigned char *) &OctetCititScris,
sizeof(OctetCititScris));    }
         }
    FisierScris.close();
    FisierCitit.close();
   }
   else Eroare=1;
 return Eroare;
}
```

2.3.4. Programul „Metoda cheilor acoperitoare“ (P4)

Funcţia *MetodaCheilorAcoperitoare()* este utilizată atât pentru operaţia de codificare cât şi pentru operaţia de decodificare.

În cazul codificării argumentele ei sunt: numele fişierului ce conţine informaţia de codificat, numele fişierului de conţine cheia şi numele fişierului ce va conţine informaţia codificată.

În cazul decodificării argumentele ei sunt: numele fişierului ce conţine informaţia codificată, numele fişierului de conţine cheia şi numele fişierului ce va conţine informaţia decodificată.

Funcţia de codificare/decodificare în limbajul C++

```
unsigned char MetodaCheilorAcoperitoare(char *FisierSursa,
                                        char *FisierDestinatie, char *Cheie)
{unsigned char OctetSursa, OctetCheie, OctetDestinatie;
 ifstream FisierCitit, FisierCheie;
 ofstream FisierScris;
 FisierCitit.open(FisierSursa, ios::binary | ios::nocreate);
 FisierCheie.open(Cheie, ios::binary | ios::nocreate);

 FisierScris.open(FisierDestinatie, ios::binary | ios::trunc);
 FisierCitit.read((unsigned char *) &OctetSursa, sizeof(OctetSursa));
 while(!FisierCitit.eof())
```

```
   {
    FisierCheie.read((unsigned char *) &OctetCheie,
sizeof(OctetCheie));
    if(FisierCheie.eof())
      {
       FisierCheie.close();
       FisierCheie.open(Cheie, ios::binary | ios::nocreate);
       FisierCheie.read((unsigned char *) &OctetCheie,
sizeof(OctetCheie));
      }
    OctetDestinatie=OctetSursa^OctetCheie;
    FisierScris.write((unsigned char *) &OctetDestinatie,
sizeof(OctetDestinatie));
    FisierCitit.read((unsigned char *) &OctetSursa,
sizeof(OctetSursa));
   }
 FisierCitit.close();
 FisierCheie.close();
 FisierScris.close();
 return 0;
}
```

3. MOD DE LUCRU

- se completează secvenţele de program propuse P1, P2, P3, P4 cu declaraţiile şi celelalte elemente de program necesare;
- se compilează, se linkeditează şi se lansează în execuţie;

Lucrarea se consideră încheiată când toate programele sunt funcţionale.

4. CHESTIUNI DE STUDIAT

- Ce avantaje şi dezavantaje are criptosistemul „Cifrul lui Caesar generalizat”?
- Ce proprietăţi se folosesc pentru a sparge cu uşurinţă criptosistemul "mono-substituţie alfabetică"?
- Cum se poate deduce că un text este codificat cu transpoziţie pe coloane?
- Cum se poate sparge ipotetic un criptosistem cu transpoziţie pe coloane?

- Enunţaţi principalele avantaje de criptare prin "metoda cheilor acoperitoare"?
- Care este dezavantajul major al criptării prin "metoda cheilor acoperitoare"?

Să se calculeze:
- numărul de chei posibile ale criptosistemul „cifrul lui Caesar generalizat”;
- numărul de chei posibile ale criptosistemul "monosubstituţie alfabetică";
- numărul de chei posibile ale criptosistemul " transpoziţie pe coloane";
- lungimea minimă şi maximă a cheii prin "metoda cheilor acoperitoare".

LUCRAREA 10

PROTECŢIA DIGITALĂ A DREPTULUI DE PROPRIETATE ÎN INTERNET FOLOSIND TEHNICI DE MARCARE ŞI VERIFICARE WATERMARK

1. OBIECTIVELE LUCRĂRII

Obiectivele lucrării sunt următoarele:
- familiarizarea cu tehnicile de marcare şi verificare online a mai multor tipuri de documente în scopul realizării unei protecţii a dreptului de proprietate în Internet;
- implementarea software a sistemului de marcare şi verificare Watermark.

2. BREVIAR TEORETIC

2.1. Introducere

Digital Watermarking este asociat cu vechea tehnică a ascunderii informaţiei şi anume steganografia. Watermarking ascunde un mesaj secret şi personal pentru a proteja dreptul de proprietate sau pentru a demonstra autenticitatea, originalitatea conţinutului sau verificarea conţinutului, integritatea datelor sau detectarea falsificării.

Marcarea imaginilor digitale, audio şi video sau a produselor multimedia în general are ca scop rezolvarea problemei dreptului de proprietate şi verificarea conţinutului original. Facilităţile digitale pentru crearea, procesarea şi memorarea produselor multimedia au fost considerate foarte convenabile de creatori, producători, editori şi clienţi. În acelaşi timp, comunicaţiile digitale prin reţea au crescut

rapid. Într-un astfel de mediu, produsele digitale pot fi foarte uşor copiate, procesate pentru variate scopuri sau expuse public.

Un *watermark digital* este un semnal inserat într-o imagine, un semnal audio sau semnal video sau în general într-un document digital. Watermark poate fi un şir de numere, numele unei companii, semnătura unei persoane, etc.

Principala caracteristică a unui watermark este că nu poate fi detectat de către ochii sau auzul nostru. Deci vizionând sau ascultând obiectul multimedia care conţine un watermark, nu vom sesiza existenţa semnalului watermark. In schimb, watermark-ul poate fi detectat şi extras cu ajutorul computerului după un anume algoritm.

Aceste proprietăţi evidenţiază cea mai importantă aplicaţie a watermark-ului digital (fig.10.1), detaliată în continuare.

Watermark împotriva utilizării de către persoane neautorizate.
Principala aplicaţie a watermark-ului digital este protecţia copyright-ului. Neputând fi detectat şi totodată nici şters, watermark-ul poate stabili adevăratul proprietar al unui document, poate identifica sursa şi utilizatorul de drept al obiectului multimedia păstrând totodată şi calitatea lui. Cele trei etape ale protejării copyright-ului prin tehnologia oferită de watermark-ul digital sunt descrise în cele ce urmează.

-declaraţia dreptului de proprietate
Uşurinţa cu care se pot duplica datele digitale face declaraţia dreptului de proprietate mai dificilă. Pentru a determina cu exactitate autorul sau proprietarul conţinutului digital, acesta din urmă poate insera un watermark în fiecare copie a documentului. Prezenţa semnului proprietarului într-un obiect multimedia din posesia unei persoane neautorizate poate dovedi hoţia acestuia din urmă.
*- depistarea distribuitorului ilegal (traitor tracing***)**
Watermark-ul digital inserat poate conţine informaţii despre cumpărătorii legitimi ai obiectului media. Înainte de a transmite obiectul media doritorului, date personale despre acesta cum ar fi numărul cărţii de credit, pot fi ascunse în respectivul obiect prin inserarea watermark-ului. Odată ce proprietarul găseşte o

copie ilegală, el poate găsi distribuitorul ilegal prin extragerea numărului de identificare din produsul respectiv.

- *controlul dreptului de folosinţă*

Când obiectul multimedia necesită utilităţi speciale de multiplicare sau vizionare, un watermark digital poate fi inserat pentru a indica numărul de copii permis sau pentru a da dreptul beneficiarului legitim de a accesa data.

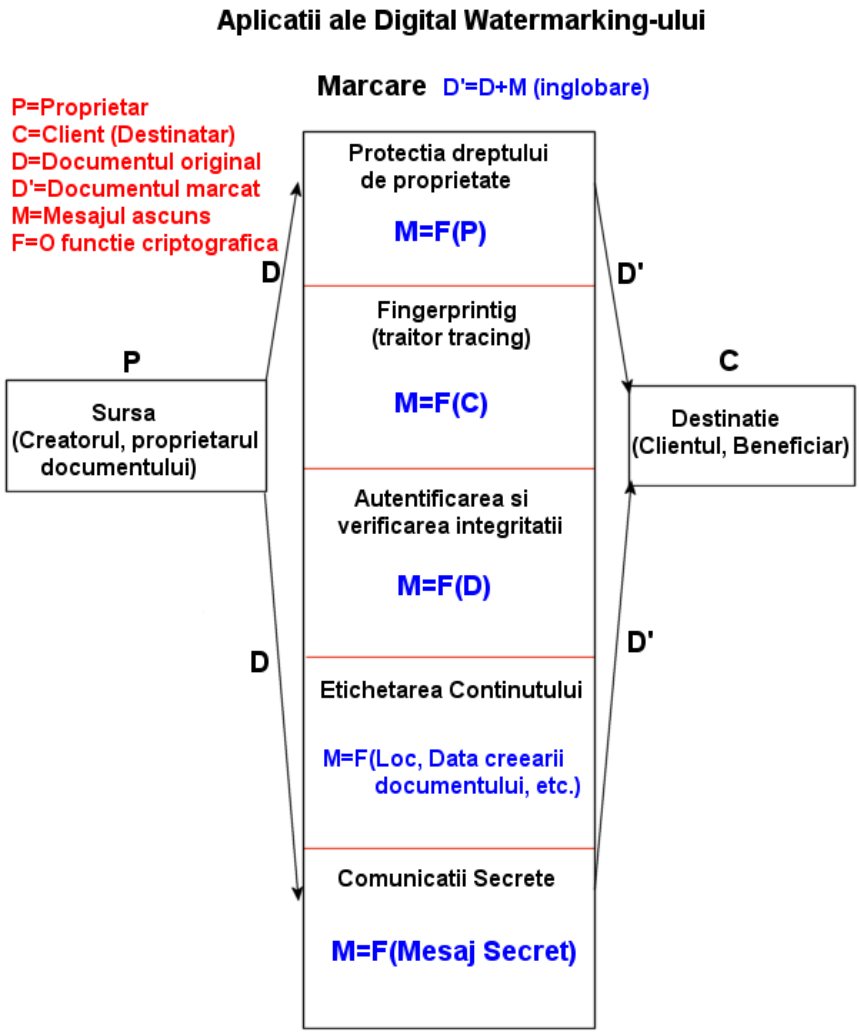


Fig.10.1. Aplicaţii ale Digital Watermarking

2.2. Principalele caracteristici ale unui watermark

Invizibilitate perceptuală: Modificările aduse de inserarea watermark-ului nu ar trebui să afecteze calitatea obiectului. Totuşi, diferenţele

dintre produsul original şi cel care conţine watermark-ul pot fi vizibile dacă se face o comparare directă între cele două produse. Ca urmare a acestui lucru, aceste diferenţe rămân neobservabile pentru că produsul original este accesibil numai proprietarului legal.

Complexitate: Semnalele watermark ar trebui să fie caracterizate printr-o mare complexitate. Acest lucru este necesar pentru a putea fi produsă o gamă largă de
watermark-uri diferite între ele. Un set imens de watermark-uri previne recuperarea unui watermark prin încercări şi proceduri de eroare.

Chei asociate: Watermark-urile trebuiesc asociate cu un număr de identificare numit cheie watermark. Cheia este folosită să genereze , detecteze şi să şteargă watermark-ul. De asemenea cheia trebuie să fie particulară şi să caracterizeze exclusiv doar proprietarul legal.

Detecţie/Căutare automată: Watermark-urile trebuie să suporte o procedură de căutare care scanează automat orice domeniu accesibil din reţea.

Detecţie demnă de încredere: Watermark-urile trebuie să constituie o dovadă suficientă a dreptului de proprietate al unui produs digital. Un watermark oarecare este o dovadă clară pentru a demonstra dreptul de proprietate al unei imagini digitale chiar dacă există o probabilitate nesemnificativă de eroare.

***Invizibilitate statistică*:** Watermark-urile nu trebuiesc recuperate folosind metode statistice. De exemplu posesia unui număr mare de produse digitale marcate cu aceiaşi cheie nu trebuie să dezvăluie watermark-ul prin aplicarea de metode statistice. De aceea watermark-urile trebuie să fie independente de produs.

Robusteţea: O imagine digitală trebuie să suporte o gamă largă de modificări diferite care în mod deliberat (atacuri pirat) sau nu (compresie, filtrare pentru eliminarea zgomotului, redimensionare) afectează watermark-ul inserat. În mod evident un watermark care se foloseşte în protecţia dreptului de proprietate trebuie să fie detectabil în condiţiile în care calitatea produsului rămâne în limite acceptabile. indiferent de modificări.

2.3. Modelul matematic

Un model matematic este foarte greu de definit întrucât variate forme ale watermark-urilor digitale pot fi găsite în literatura de specialitate. În general, un watermark poate fi definit ca fiind un semnal digital *W*

$$W = \{w(k) \mid w(k) \in U, k \in W^d\} \tag{10.1}$$

care este introdus în produsele digitale printr-o procedură de inserţie. W^d denotă domeniul dimensiunilor watermark-ului; d=1,2,3 pentru audio, imagini respectiv video. Semnalul watermark poate avea o formă binară (U={0,1} sau U={-1,1}) sau forma unui zgomot gaussian ($U = (-1,1) \subset IR$) Uneori, *W* este numit watermark-ul original pentru a putea fi deosebit de versiunile transformatei F(*W)* care pot apare în timpul inserţiei sau extragerii.

Watermark-ul în mod general (*GWF -general watermarking framework*) se defineşte ca fiind sextuplul ($X,W,K,\gamma,\varepsilon,\delta$), unde:

1. *X* denotă setul de produse digitale care trebuiesc a fi protejate
2. *W* este setul de posibile semnale watermark definite de ecuaţia (10.1).
3. *K* este un set de numere ID (de exemplu un set de parametri întregi) şi sunt numite chei pentru watermark.
4. γ denotă algoritmul care generează watermark-urile folosind cheia şi produsul digital care urmează a fi marcat

$$\gamma : X \times K \to W,\ W = \gamma(X,K) \tag{10.2}$$

5. ε este algoritmul de inserţie care introduce watermark-ul W în produsul digital X_0

$$\varepsilon : X \times K \times IR \to X,\ X_w = \varepsilon(X_0, W) \tag{10.3}$$

X_w denotă versiunea marcată a lui X_0.
6. În final, δ denotă algoritmul de detecţie definit după cum urmează

$$\delta : X \times K \to \{0,1\}$$
$$\delta(X,W) = \begin{cases} 1 \text{ daca W exista in X} \\ 0 \text{ altfel} \end{cases} \tag{10.4}$$

2.4. Protecţia dreptului de proprietate cu GWF

Violarea dreptului de proprietate atacă interesele furnizorilor mai mult decât cele ale clienţilor. GWF răspunde direct la întrebarea pusă de un

furnizor particular : "Sunt eu proprietarul produsului X?". Prin inserarea watermark-ului în produse înainte de distribuţia lor către clienţi sau în reţele public accesibile, întrebarea de mai sus devine "Există watermark-ul meu în X?".

Deci protecţia dreptului de proprietate devine o problemă a detecţiei watermark-urilor. Trebuie remarcat faptul că GWF nu poate răspunde la întrebarea "Este produsul X protejat?" sau "Al cui este produsul X?". Aceasta deoarece GWF este bazat pe detecţia watermark-ului folosind chei private (fig.10.2).

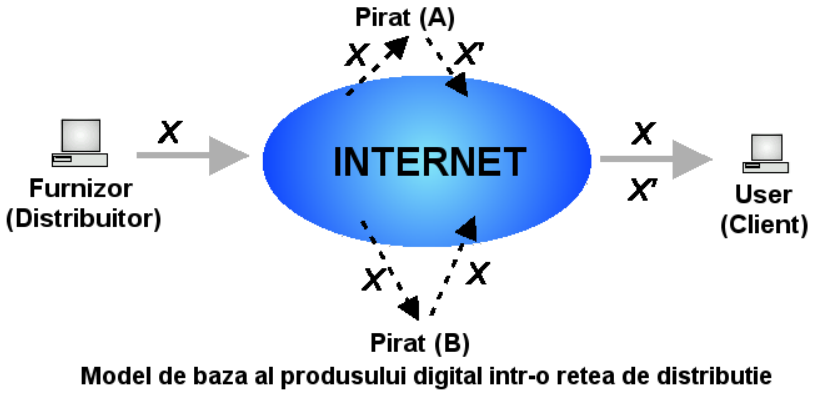


Fig.10.2. Detecţia Watermark cu chei private într-o reţea de distribuţie

3. MOD DE LUCRU

3.1. Descrierea sitului de marcare şi verificare watermark

Implementarea software a sistemului de marcare şi verificare watermark constă dintr-un sit ce are ca scop marcarea şi verificarea online a mai multor tipuri de documente în scopul realizării unei protecţii a dreptului de proprietate în Internet.

Sunt recunoscute 7 tipuri de documente din 4 categorii: *executabile, imagini, audio* şi *documente de tip text*. Aceste formate recunoscute sunt:

1. executabile: *.exe
2. imagini: *.gif, *.bmp
3. audio: *.wav,*.voc
4. texte: *.txt, *doc.

Situl a fost realizat în PHP sub Windows 2000, dar este funcţionabil şi în Windows 9x şi Me. Ca bază de date foloseşte MySQL pentru a ţine o evidenţă a tuturor utilizatorilor înregistraţi. Procesul de marcare şi verificare al watermark-ului nu este accesibil decât utilizatorilor înregistraţi (aflaţi în baza de date).

Situl poate fi împărţit în 3 mari module:

(1) Modulul de înregistrare/conectare a utilizatorului;

(2) Modulul de inserare a watermark-ului în documente;

(3) Modulul de extragere a watermark-ului din documente.

În plus, pe lângă protecţia dreptului de proprietate, situl are şi următoarele domenii de aplicaţie:

- comunicaţii secrete între 2 utilizatori înregistraţi
Documentul marcat de un utilizator înregistrat este citit la destinaţie de către celălalt utilizator înregistrat. Această operaţiune presupune stabilirea în prealabil a unei parole secrete între cei doi utilizatori.

- detecţia utilizatorului neautorizat
Înainte de distribuţia unui document (o aplicaţie software, o imagine etc.) către un client, se poate marca acest document cu un cod unic, care să identifice clientul căruia îi este destinat documentul. În cazul descoperirii unei copii neautorizate, se poate extrage codul din această copie şi astfel se poate afla şi clientul care a distribuit ilegal documentul.

- detecţia modificării/validităţii şi integrităţii unui document
Se marchează documentul înainte de trimitere cu un watermark care depinde de informaţia trimisă şi la destinaţie se extrage acest cod, testând validitatea documentului.

- refacerea unui document modificat
Prin introducerea unui watermark care reprezintă un cod detector şi corector de erori se pot reface documentele transmise prin medii de distribuţie nesigure.

- etichetarea conţinutului
Pentru imagini sau sunete este necesară etichetarea conţinutului acestora prin adăugarea de informaţii privitoare la locul şi data creării documentului .

- Se recomandă explorarea tuturor posibilităţilor legate de protecţia dreptului de proprietate oferite de modulele sitului, ca şi a manierei în care pot fi realizate aplicaţii din categoriile mai sus menţionate.

3.2. Marcarea fişierelor executabile

Marcarea funcţionează cu succes asupra tuturor categoriilor de fişiere executabile în format PE cu câteva excepţii (programele de tip install cum ar fi HW32V31.EXE - programul de instalare al utilitarului HexWorkShop- şi programele care citesc date din propriul lor fişier executabil). Dimensiunea mesajului secret este limitată la 4096 caractere, dar prin marcarea online nu se permite introducerea de mai mult de 100 de caractere.

În urma vizualizării secţiunilor celor 2 fişiere -cel marcat şi cel original- se observă diferenţele dintre cele două fişiere, şi anume fişierul marcat are secţiunea *wmark* în plus, secţiune care are rolul de încărcător pentru fişierul marcat (fig.10.3., fig.10.4.**).**

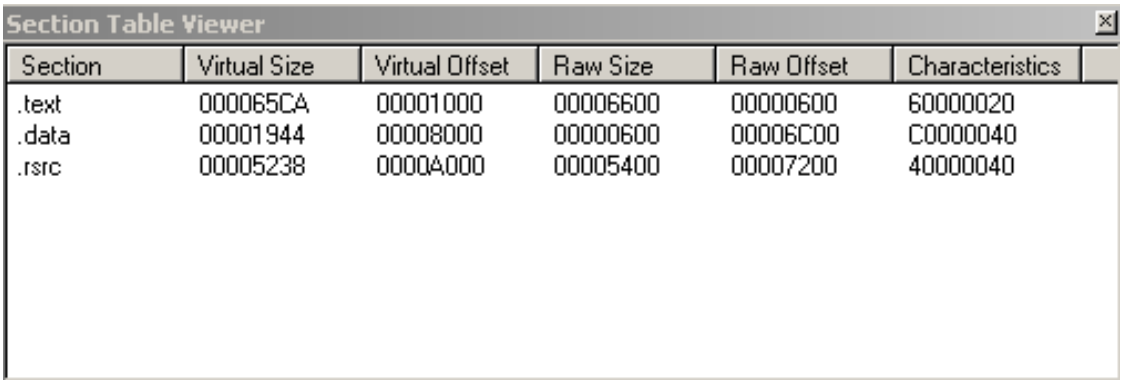


Fig.10.3. Fişierul Notepad.exe original

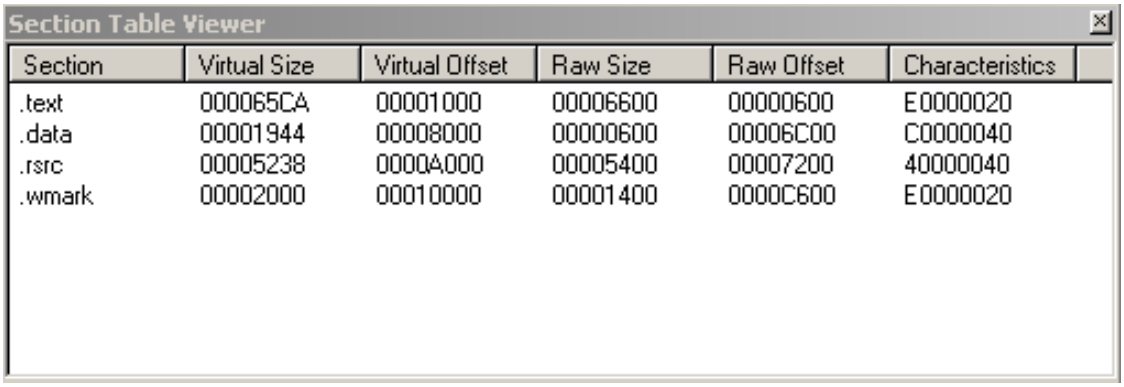


Fig. 10.4. Fişierul Notepad.exe marcat

Din punct de vedere al funcţionalităţii, cele 2 programe se comportă identic, procesul de marcare nu aduce nici o disfuncţionalitate în execuţia programului marcat, ci doar o eventuală creştere a timpului de execuţie, fapt inobservabil la fişierul *Notepad.exe* şi în general la fişierele executabile mici.

La nivelul secţiunilor *.data* al fişierelor, în ciuda funcţionalităţii identice, se remarcă o diferenţă totală datorită criptării secţiunilor (fig.10.5).

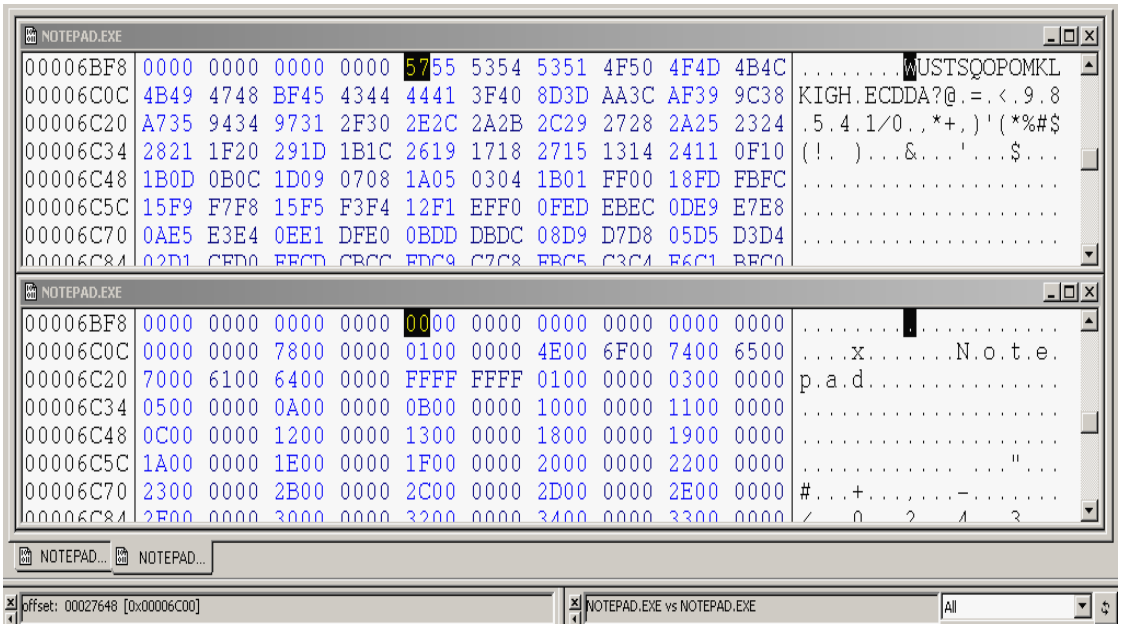


Fig. 10.5. Diferenţe la nivel binar între fişierul marcat şi cel original

3.3.Marcarea fişierelor bmp, wav şi voc

În urma marcării fişierului *pasare.bmp* (fig.10.6), s-a obţinut o imagine (fig.10.7) identică vizual cu cea originală.

Fig. 10.6. Fişierul original *pasare.bmp*

Fig. 10.7. Fişierul marcat *pasare.bmp*

Privind cu atenţie cele 2 fişiere, la nivel binar pot fi observate mici diferenţe de 1 bit ale anumitor pixeli aleşi într-un mod care depinde de parolă, diferenţe care evident fac ca imaginea marcată să arate identic cu cea originală.

De exemplu, la deplasamentul 0x000094D3, octetul 0x51 este înlocuit cu 0x50, iar următoarea diferenţă dintre imagini se află la deplasamentul 0x000098AF, unde octetul 0x9A este înlocuit cu octetul 0x9B.

- Se recomandă exerciţii de marcare din gama *fişiere executabile*, *fişiere bmp, wav şi voc* şi formularea observaţiilor consecutive referitoare la calitatea fişierelor obţinute.

4. CHESTIUNI DE STUDIAT

- Imaginaţi metode de atac asupra documentelor marcate, pentru fiecare dintre tipurile de fişiere menţionate;
- Apreciaţi, în funcţie de posibilităţile de contraatac, eficienţa şi siguranţa metodelor de marcare;
- Propuneţi eventuale modificări ale algoritmilor de marcare în ideea îmbunătăţirii robusteţei lor.