Sisteme de programe pentru timp real

Universitatea “Politehnica” din Bucuresti
2003-2004
Adina Magda Florea
http://turing.cs.pub.ro/sptr_2004

Curs Nr. 2

- Algoritmul deplaseaza-aduna – continuare

Criptologie si criptosisteme

- **Numere aleatoare**
- **Operatii aritmetice cu numere mari**
- **Criptologie – generalitati**
- **Criptosisteme conventionale**
- **Criptosisteme publice**
- **Standarde actuale**

2

Algoritmul deplaseaza-aduna

Generalizare la:

1. “Don’t care symbols”, deci identificarea cu orice caracter pe o pozitie
2. Identificarea cu un caracter dintr-o clasa de caractere (un interval)
3. Identificarea cu un caracter din complementul unei clase.
4. Identificarea cu un numar dat K de nepotriviri

3

Identificarea claselor de sabloane

a - sirul (de lungime **N**) in care se executa cautare
p - sablonul (de lungime **M**)
i - index in sablon
j - index in text

- x – un caracter din alfabetul Σ
- \- un caracter ce poate identifica cu orice caracter din Σ
- $[x_1 \ldots x_n]$ – identifica orice caracter dintre cele specificate
- $\{x_1 \ldots x_n\}$ - identifica orice caracter din Σ diferit de cele specificate
- 2 dimensiuni: M – sablon, M1 – descriere sablon

4

Identificarea claselor de sabloane

Modificarea tabelei T

$$T[x] = \sum_{i=0}^{M-1} \delta(x \neq clasa_{i+1}) \cdot 2^{b \cdot i}$$

Σ = {a, b, c, d}

Sablonul este **a{b}[ab]b{bc}**

M = 5 si M1 = 15

Sir_1: **a b a b c** Sir_2: **a c a b a**

Care sunt valorile T[a], T[b], T[c] si T[d] ?

5

Algoritmul pt calculul lui T

```
valinit ← 111...111 (w biti)
for i = 1 to M do
  if p1 = '*' or pi este un complement
  then valinit ← valinit & 111…0i…111
endfor
for i = 1 to |Σ| do
  T[xi] ← valinit
endfor
for i = 1 to M do
  foreach x ∈ p_i do
    if p_i este un complement
      then T[x] ← T[x] | 000…1_i…000
      else T[x] ← T[x] & 111…0_i…111
  endforeach
endfor
```

6

Implementarea calculului lui T

```
void tinit(char *p)
{ unsigned int T[MAXSYM], initval, mask=1;
  int i=0, M=0, M1=strlen(p);
/* calculeaza initval<-initval & 111..0i..111 */
  initval=~0;
  while(i<M1)
      { M++;
        if(p[i]=='*')initval&=~mask;
        else if(p[i]=='{'){initval&=~mask;
                          while(p[i]!=EOS &&
                              p[i]!='}')  i++;}
             else if(p[i]=='[')
                       while(p[i]!=EOS && p[i]!=']')i++;
        mask<<=B;i++;
      }
```

7

Implementarea calculului lui T - continuare

```
/* calculeaza T[xi]<-initval pentru fiecare
  xi din alfabet*/
  for(i=0;i<MAXSYM;i++)T[i]=initval;
/* Actualizeaza T[x], cu x din sablon */
  i=0; mask=1;
  while(i<M1)
  { if(p[i]=='{') while(p[++i]!='}')
                                   T[p[i]]|=mask;
   else if(p[i]=='[') while(p[++i]!=']')

     T[p[i]]&=~mask;
        else T[p[i]]&=~mask;
   i++; mask<<=B;
}
```

8

Identificarea cu max k nepotriviri

Fiecare stare individuala poate fi reprezentata printr-un numar de maxim O(logM) biti.

- **$b = sup|log_2(M+1)|+1$**
- daca $s^j_M \le k$, am gasit o identificare cu maximum **k** nepotriviri
- operatorul **op** este adunarea
- Deoarece ne intereseaza numai k nepotriviri, este suficient sa alegem

$b = sup|log_2(k+1)| + 1$

Problema transportului – un bit de transport (overflow) pt fiecare stare

- **Obtinerea unei noi stari:** **$stare^{j+1}=(stare^j << b) + T[x_{j+1}]$**
- **Conditia de identificare cu max k nepotriviri**

$s^j_M + overf_j < (k+1) * 2^{b(M-1)}$

9

Exemplu

- **Sablonul ababc**
- **Alfabetul {a, b, c, d}**
- **T construit anterior**
- **k = 2, b = 3**
- **starea initiala 00000**
- **overflow initial 44444**

sir	a	b	d	a	b
T[x]	11010	10101	11111	11010	10101
stare	11010	20201	13121	02220	32301
overf	44440	44400	44000	40000	00000

sir	a	b	a	b	c
T[x]	11010	10101	11010	10101	01111
stare	30020	10301	10020	10301	00121
overf	04000	4000	04000	40000	40000

10

Algoritmul pentru max k nepotriviri

```
mask ← 1_Mb0…0…1_2b0…01_b0…0
lim ← (k+1) << (M-1)*b
stare ← 0
for i = 1 to N do
  stare ← (stare << b) + T[a_i]
  overf ← (overf << b) | (stare & mask)
  stare ← stare & ~mask
  if (stare+overf) < lim
  then identificare la pozitia i-M+1
endfor
```

11

Criptologie
si criptosisteme

12

1. Numere aleatoare

- *Numar intreg/real aleator* intr-un domeniu dat si cu o precizie fixata / *Numar pseudoaleator*
- *Generator de numere aleatoare* - o multime de stari S, o functie $f:S \rightarrow S$ si o stare initiala s_0 - samanta.
- Starile generatorului evolueaza dupa relatia:

  $s_i = f(s_{i-1})$, cu $i = 1,2,\ldots$ $\quad g:S \rightarrow (0,1)$
- *Perioada* unui generator de numere aleatoare este cel mai mic intreg pozitiv **p** a.i.

  $s_{i+p} = s_i$, $i > p \geq 0$

13

Sunt numere aleatoare?

- Testul $\aleph^2$
- N numere intregi in intervalul [0, r), frecventa fiecarui numar din interval fiind f_i (i = 0, r-1)

$$\aleph^2 = \frac{\sum_{i=0}^{r-1} (f_i - N/r)^2}{N/r} \qquad (\in [\sqrt[2]{r}, r])$$

- Distributii uniforme (aceeasi probabilitate)

14

1.1 Metoda congruent multiplicativa

- **$f(s_{i+1}) = (b*s_i + c)$ *mod* m**

  $g(s_i) = s_i/m$

  s_0 - samanta , b, c < m, pozitivi

  $f(s_i) \in [0, m-1]$
- **Cum se aleg m, s_0 si b ?**
- Fiecare valoare este mai mica decat cel mai mare intreg, dar prima operatie a*b+1 duce la overflow
- Cum se elimina overflow-ul?

15

Cum se elimina overflow-ul?

- Reprezentare pe 32 de biti - intereseaza pentru rezultat numai ultimii 8 digiti.
- **a** si **b** se reprezinta ca doua polinoame in fct. de x.
- a = 1234567 $\qquad$ b = 31415821

$p(x) = 3x^7 + \ldots + 2x + 1$, cu $b = p(10)$

$q(x) = x^6 + \ldots + 7$, cu $a = q(10)$

$grad(p) \leq N-1$ $\quad grad(q) \leq N-1$ $\qquad grad(p*q) \leq 2N-2$

$\mathbf{p = 10^4 * p_1 + p_0}$

$\mathbf{q = 10^4 * q_1 + q_0}$

$\mathbf{p * q = (10^4 * p_1 + p_0)(10^4 q_1 + q_0) =}$

$\mathbf{10^8 * p_1 * q_1 + 10^4 * (p_1 * q_0 + p_0 * q_1) + p_0 * q_0.}$

16

Generare numere aleatoare

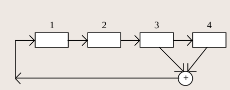
```
#define    m    100000000
#define    m1       10000
#define    b     31415821

long int a =1234567;
long int   mult( long int p , long int q )
{  long int   p0, p1, q0, q1;
   p1 = p / m;    p0 = p % m1;
  q1 = q / m1;  q0 = q % m1;
   return ((p0 * q1 + p1 * q0) % m1)*m1 + p0* q0)% m ;
}
long int random( )
{
   a = ( mult( a, b ) + 1 ) % m ;
   return   a;
}
// nr. aleatoare ∈[0,m-1]
// echiv rand() RAND_MAX=m-1
```

17

1.2 Metoda congruent-aditiva

- **Registru de deplasare cu feed-back**
- Adunare



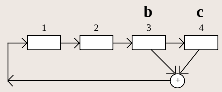
- 1111
- 0111, 0011, 0001, 1000, ...
- Pt. **n** biti se pot obtine secvente de max. 2^n-1 numere distincte
- n = 31 sunt bune pozitiile 0 si una din pozitiile 4, 7, 8, 14, 19, 25, 26 sau 29.

18

Metoda congruent-aditiva

- **Adunare**

$\mathrm{Re}\, g_k = \mathrm{Re}\, g_{k-b} \oplus \mathrm{Re}\, g_{k-c}$



- Considerand un sir de numere aleatoare $a_0 \ldots a_k$, se obtin numere aleatoare in continuare

  in [0, m-1], astfel
- **a [k] = (a [k-b] + a [k-c]) % m** (b < c)
- min 2^c-1 numere distincte
- Valori b = 31, c = 55
- a – coada circulara

19

2 Operatii aritmetice cu numere mari

- Reprezentare
- Intregul **0120200103110001200004012314** cu N = 28 digiti se reprezinta prin **p(10)** unde p este polinomul

$$p(x) = x^{26} + 2 \cdot x^{25} + \ \ldots\ + x + 4$$

- Calcul eficient al inmultirii a doua polinoame p(x) si q(x) de grad *N-1*
- Produs de grad *2N-2* cu *2N-1* termeni
- Produs calculat direct – N^2 inmultiri
- Divide and conquer
- N – par => 2 polinoame de grad N/2

20

Utilizare "Divide and conquer"

$$p(x) = p_0 + p_1 x + \ldots + p_{N-1} x^{N-1}$$
$$p_i(x) = p_0 + p_1 + \ldots + p_{N/2-1} x^{N/2-1}$$
$$p_s(x) = p_{N/2} + \ldots + p_{N-1} x^{N/2-1}$$
$$p(x) = p_i(x) + x^{N/2} \cdot p_s(x)$$
$$q(x) = q_i(x) + x^{N/2} \cdot q_s(x)$$
$$p(x) \cdot q(x) = p_i(x) \cdot q_i(x) + (\, p_i(x) \cdot q_s(x) + q_i(x) \cdot p_s(x)\,) \cdot x^{N/2} + p_s(x) \cdot q_s(x) \cdot x^N$$

- Pentru a calcula p (x) * q (x) sunt necesare numai 3 inmultiri

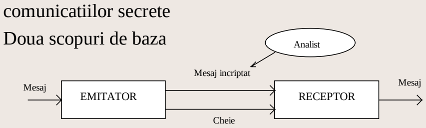
$$r_i(x) = p_i(x) \cdot q_i(x)$$
$$r_s(x) = p_s(x) \cdot q_s(x)$$
$$r_m(x) = (p_i(x) + p_s(x)) \cdot (q_i(x) + q_s(x))$$
$$p(x) \cdot q(x) = r_i(x) + (r_m(x) - r_i(x) - r_s(x)) \cdot x^{N/2} + r_s(x) \cdot x^N$$

- Doua polinoame de grad N pot fi inmultite folosind $\approx N^{1.58}$ inmultiri

21

3 Criptologie - generalitati

- **Criptografia** - Proiectarea sistemelor de comunicatie secreta
- **Criptoanaliza** - Studiul metodelor de intelegere a comunicatiilor secrete
- Doua scopuri de baza



- Doua tipuri de criptosisteme:

  - conventionale (criptosisteme simetrice)

  - publice (criptosisteme asimetrice)

22

4 Criptosisteme conventionale

Metode simple

- ***Cifrul lui Cezar*** - N-a litera din alfabet, se inlocuieste cu litera (N+k) din alfabet, unde k este constant (Cezar lua k = 3)
- ***Substitutie simpla*** - Matrice cu 26 linii si 2 coloane care defineste substitutia literelor
- ***Cifrul Vigenere***: se utilizeaza o cheie pentru a determina valorile lui **k** care trebuie adaugate fiecarei litere.

Fie cheia $c_1 c_2 \ldots c_m$.

$j \leftarrow 0$

pentru fiecare litera l_i din mesaj **executa**

  l_i din mesaj are indexul p in alfabet

  $j \leftarrow (\, j+1\,)$ *mod* m

  alege c_j din cheie

  fie **k** indexul lui c_j in alfabet

  inlocuieste l_i cu litera din alfabet de index (k + p)

sfarsit

23

Criptosisteme conventionale

Cifrul Vigenere se poate combina cu substitutia simpla

- Daca **cheie ≥ mesaj** ➔ ***Cifrul Vernam*** (one time pad)
- ***Masini de criptare/decriptare*** - primeste un numar de chei adevarate, numite ***criptovariabile***, care sunt utilizate pentru a genera chei lungi
- Generarea pseudocheii din criptovariabile este asemanatoare cu metoda congruent aditiva (cu registru) de la numere aleatoare
- Pericol
- Dificultati ale sistemelor conventionale

24

5 Criptosisteme publice

Idee: fiecare utilizator are o **cheie publica P** care poate fi cunoscuta de oricine si o **cheie secreta S** cunoscuta numai de el.

- Mesaj M
- E - cheia publica P a receptorului - C = P (M)
- R - cheia secreta S - M = S (C)

Conditii

- S (P (M)) = M pentru fiecare mesaj M
- Toate perechile (S, P) sa fie distincte
- Deducerea lui S din P sa fie la fel de dificila ca si decriptarea lui C
- Atat S cat si P sunt usor de calculat

5.1 Criptosistemul public RSA

- Cheia de incriptare **P** este o pereche de intregi **(N, p)** cu **p** public
- Cheia de decriptare **S** este o pereche de intregi **(N, s)** unde **s** este secret.

Aceste numere trebuie sa fie foarte mari, in mod tipic N - 200 digiti iar p si s aproximativ 100 digiti

Metoda de criptare/decriptare

1. Se imparte mesajul in k grupri de biti $M_1 \ldots M_k$
2. Se incripteaza mesajul astfel: $C = P(M) = C_1 \ldots C_k$

   unde $C_i = (M^p{}_i)$ *mod* N ➔ R
3. Receptorul decripteaza mesajul

   $M = S(C) = M_1 \ldots M_k$ unde $M_i = (C^s{}_i)$ *mod* N

Modul de alegere a *p* si *s*

1. Se genereaza trei numere aleatoare prime mari (100 digiti) x, y, z.
2. Cel mai mare dintre acestea este ales ca valoare a lui s.
3. Fie celelalte doua numere x si y.
4. N = x * y
5. p se alege astfel incat p * s *mod* (x-1) * (y-1) = 1.

Se poate demonstra ca, pt. aceste alegeri,

$$M^{ps} \bmod N = M$$

Este sigur?

Cum se genereaza un nr. prim f. mare?

Se genereaza un nr. aleator f. mare + se testeaza daca este prim

```
Fie w numarul pentru care se testeaza daca este prim.
1.   i ← 1, n ← 50
2.   Determina a si m a.i. w=1+2^a m , unde m este impar si 2
     este cea mai mare putere a lui 2 care divide w-1.
3.   Genereaza un numar aleator b ∈( 1, w )
4.   j ← 0, z ← b^m mod w
5.   daca  (( j=0 ) si ( z=1 ))  sau ( z=w-1 )
     atunci executa pasul 9
6.   daca ( j>0 ) si ( z=1 ) atunci executa pasul 8
7.   j ← j + 1
     daca  j<a atunci
     executa pasul 6.
8.   w nu este prim
     stop
9.   daca  i<n
     atunci  i i + 1;  executa pasul 3
     altfel w este probabil prim
sfarsit
```


5.2 Discutie criptosisteme publice

- Toate abordarile se bazeaza pe calcule NP
- ***Problema:*** daca se inlocuieste **p** (cu s asociat) cu **p'** (si s' asociat)
- ***Cheile publice*** – **certificate de cheie** prefixate cu un *certificat de semnatura*; contin:
  - User key ID
  - Data la care a fost creata cheia
  - Cheia
- ***Cheia secreta*** – cheie incriptata cu o parola

6. Standarde actuale

Publice

- DSA – Digital Signature Algorithm
- DSS – Digital Signature Standard
- AES – Advanced Encryption Standard
- NIST (National Institute of Standards and Technology, USA) lucreaza la *Federal Public Key Infrastructure* – va sustine semnaturile digitale

Conventionale

- DES – Digital Encryption Standard

6.1 DES – Digital Encryption Standard

Modele de operare DES

- ECB – Electronic Codebook – DES direct
- CBC – Cipher Block Chaining – DES extins care inlantuie blocuri de text incifrat
- CFB – Cipher Feedback – utilizeaza text incriptat anterior ca intrare pt. DES si genereaza iesiri pseudoaleatoare care sunt combinate cu textul neincriptat pentru a produce text incriptat
- TDEA – Triple Data Encryption Standard

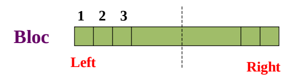
DES – Mod de functionare

- Algoritm pt. criptarea si decriptarea **blocurilor de date de 64 de biti** pe baza unei **chei de 64 de biti**.
- Criptarea si decriptarea utilizeaza **aceeasi cheie k** – decriptarea este reversul criptarii
- **Blocul de criptat:**

1) Permutare initiala **IP**

2) Calcul complex care depinde de cheie = o functie **f** - *functia de criptare*, si o functie **KS** - *planificarea cheii*

3) Permutare inversa a cele initiale $\mathbf{IP^{-1}}$



DES – Mod de functionare

1) IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

3) IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES – Mod de functionare

2) Calcul

- **16 iteratii**; functia **f** opereaza asupra a 2 blocuri: unul de 32 biti si unul de 48 de biti ➔ un bloc de 32 de biti
- Blocul de intrare = 64 biti = **L** (32) **R** (32)
- ***K*** – un bloc de 48 biti ales din cheia **KEY** de 64 biti
- La fiecare iteratie blocul **K** este diferit

$K_n = KS(n,KEY)$

$n \in [1,16]$, K_n – functie care permuta o selectie de biti din KEY

- Pt un bloc $\mathbf{L_{n-1}R_{n-1}}$, iesirea $\mathbf{L_nR_n}$ a unei iteratii este:

$$\mathbf{L_n = R_{n-1} \qquad R_n = L_{n-1} \oplus f(R_{n-1},K_n)}$$