Curs 5+6

LFSR = linear feedback stream register

Dacă se cunosc $2L$ valori, se pot afla coeficienții:

$$\begin{pmatrix} s_{L-1} & s_{L-2} & \cdots & s_1 & s_0 \\ s_L & s_{L-1} & \cdots & s_2 & s_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ s_{2L-2} & s_{2L-3} & \cdots & s_L & s_{L-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_L \end{pmatrix} =$$

$$= \begin{pmatrix} s_L \\ s_{L-1} \\ \vdots \\ s_{2L-1} \end{pmatrix} \Rightarrow \text{aflăm } c_1, c_2, \ldots c_L$$

Deci un LSFR este nesigur pt ptoim text attack!

Definiție $s = s_0, s_1, s_2, s_3 \ldots$

Complexitatea liniară $L(s)$:

- $L(s) = 0$ dacă $s = 0, 0, 0, \ldots$
- $L(s) = \infty$ dacă nici un LSFR nu produce $L(s)$
- $L(s) =$ lungimea celui mai scurt LSFR care produce s.

Definiție Pentru un șir finit $s_0, \ldots s_{n-1}$

- $0 \leq L(s) \leq n$
- s periodic de perioadă N: $L(s) \leq N$
- $L(s \oplus t) \leq L(s) + L(t)$

Combinăm mai multe LFSR

Exemple $f(x_1, x_2, x_3, x_4, x_5) = 1 \oplus x_2 \oplus x_3 \oplus x_4 \odot x_5 \oplus x_1 x_2 x_3 x_5$

$f(L_1, L_2, \dots L_n)$ = complexitatea combinației

unde înlocuim $\oplus$ cu $+_2$ și $\odot$ cu $\times_2$.

Exemplu $f(x_1, x_2, x_3) = x_1 x_2 \oplus x_2 x_3 \oplus x_3$ Geffe generator

Complexitate liniară $L_1 L_2 + L_2 L_3 + L_3$

Perioada: $(2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$

x_1	x_2	x_3	z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$P_2(z = x_1) = \frac{3}{4}$ mare !

$P_2(z = x_3) = \frac{3}{4}$ mare !

Atac bazat pe corelație

Sp că cunoaștem L_1, L_2, L_3.

for all primitive connection polynomials of degree L_1
  forall initial states of LFSR1
    Compute $2L_1$ bits of LFSR1
    compute how many are equal with Geffe.
  end
end

Repeat for LFSR 3.
Recover LFSR2 from $x_1x_2 \oplus x_2x_3 \oplus x_3$

– Multe alte forme de stream cyphers, de exemplu A5/1 care a codificat telefoanele GSM ....

RC4 = Ron's Cipher (MIT)

S = array 0, 1, ... , 255 (permutate cumva)

$i = 0, j = 0$

$$\begin{cases} i = (i+1) \bmod 256 \\ j = (j + S_i) \bmod 256 \\ \text{swap}(S_i, S_j); \quad t = (S_i + S_j) \bmod 256; \\ K = S_t \longrightarrow \text{bits} \end{cases}$$

Starea inițială e generată folosind cheia

for $i=0$ to 255 $S_i = i$
$j = 0$

for $i=0$ to 255) do
$$\begin{cases} j = j + S_i + K_i \bmod 256 \\ \text{swap}(S_i, S_j) \end{cases}$$

Nu face valori atât de random cât s-ar dori.

Cifrări bloc

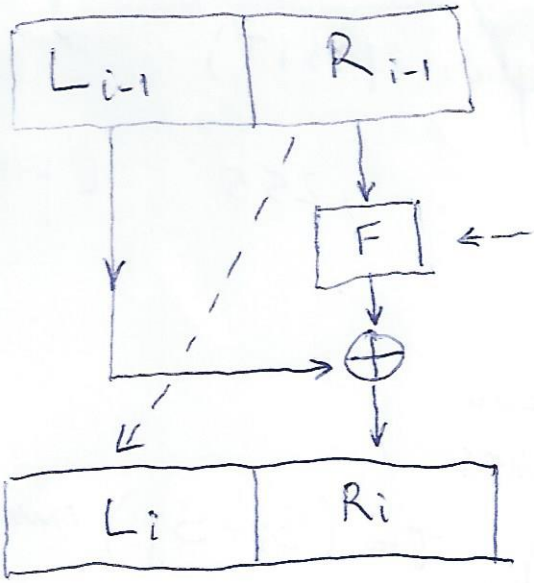
- Cel mai celebru, DES = data encryption standard
  bloc de 64 biți, cheie de 56 biți
  s-au dovedit prea scurte!

- Când a picat DES, s-a făcut un concurs pentru AES
= advanced encryption standard (org. de NIST = national institute of standards and technology)

⇒ Rijndael declarat câștigător la sf. lui 2000

Idee: Stream cyphers better for hardware (less flexible)
Block cyphers better for software

Feistel

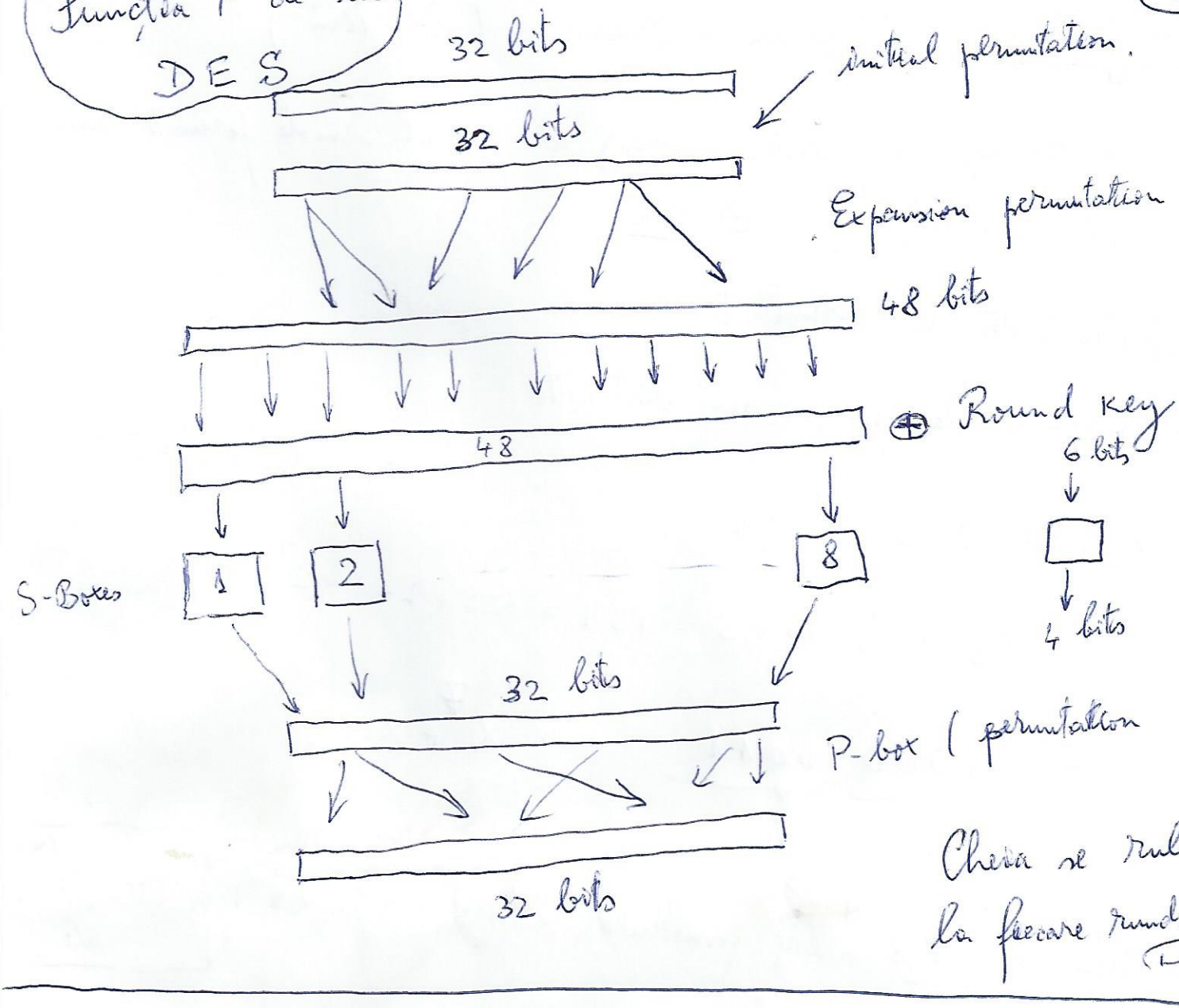


$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus F(R_{i-1}) \end{cases} \Rightarrow \begin{cases} R_{i-1} = L_i \\ L_{i-1} = F(L_i) \oplus R_i \end{cases}$$

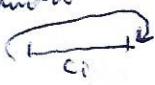
Obs: F nu trebuie să fie inversabilă.

DES 16 runde Feistel

Funcția F de la DES

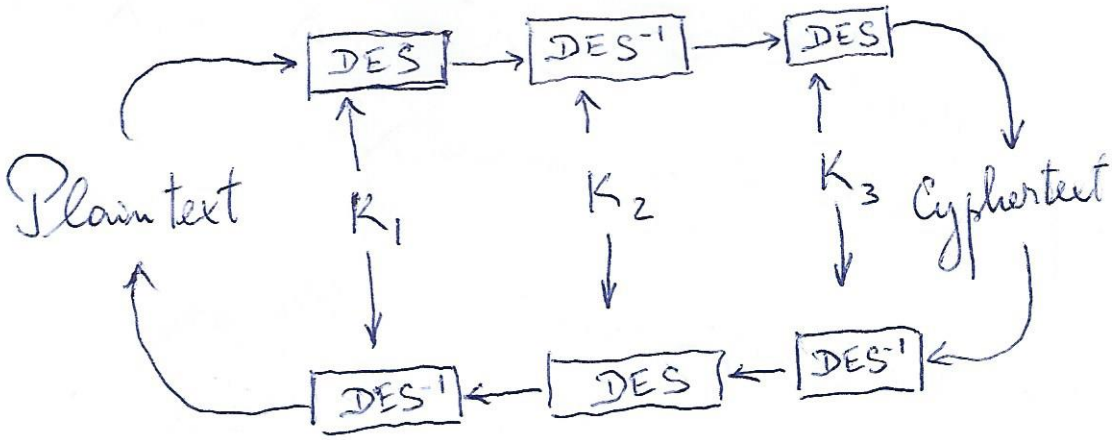


Cheia se rulează la fiecare rundă



~~Rețele~~

Triple DES



Rijndael — Rijmen + Daemen (belgieni)

- nu se bazează pe Feistel, dar are și el runde formate din permutări, substituții și $\oplus$ cheie.
- Aritmetică în corpul $\mathbb{F}_{2^8}$.
- criptarea și decriptarea sunt distincte.

$0, \dots 9, A, B, C, D, E, F$

$1,0,0,0,0,0,1,1$

$0x83 = 8 \cdot 16 + 3 = 131 = 128 + 2 + 1 = 2^7 + 2 + 1$

$\rightsquigarrow \underbrace{1000}_{8}\ \underbrace{0011}_{3} \rightsquigarrow X^7 + X + 1$

Aritmetica în $\mathbb{F}_{2^8}$ este modulo $x^8 + x^4 + x^3 + x + 1$ $\quad \mathbb{F}_{256} = \frac{\mathbb{F}_2[X]}{(x^8 + \dots)\ \text{ireducibil}}$

32 bit word identificat cu $\mathbb{F}_{2^8}[X]$ de grad ≤ 4 în *big endian*

$a_0 a_1 a_2 a_3 \rightsquigarrow a_3 X^3 + a_2 X^2 + a_1 X + a_0$

$a_i \in 2^8$

Aritmetica pe $\mathbb{F}_{2^8}[X]$ se face modulo $X^4 + 1$.

Observație $X^4 + 1 = (X^2 + 1)^2 = (X+1)^4$

iar $\mathbb{F}_{256}[X]/(X+1)^4$ nu e corp! ci doar *inel*

Rijndael pt blocuri 128 cu cheie 128

Internal state $S = \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ \cdots & \cdots & \cdots & \cdots \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix}$ bytes

state matrix

= 4 32 bit words (coloanele)

Each round key $K_i = \begin{pmatrix} K_{00} & \cdots & K_{03} \\ \vdots & & \vdots \\ K_{30} & \cdots & K_{33} \end{pmatrix}$

Operații

Sub Bytes (S-box)

$s = [s_7, \ldots s_0] \in S$ $\quad s \rightsquigarrow y$

$s \in \mathbb{F}_{2^8}$

① $s \rightsquigarrow x = s^{-1}$, $0 \rightsquigarrow 0$

② $x \rightsquigarrow y$

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Shift Rows

Cyclic shift on the state matrix

$$\begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix} \rightsquigarrow \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{11} & s_{12} & s_{13} & s_{10} \\ s_{22} & s_{23} & s_{20} & s_{21} \\ s_{33} & s_{30} & s_{31} & s_{32} \end{pmatrix}$$

Mix Columns

Fiecare coloană din state matrix este considerată polinom de grad ≤ 4 cu coeficienți în $\mathbb{F}_{2^8}$

$$b_0 + b_1 X + b_2 X^2 + b_3 X^3 = \left(a_0 + a_1 X + a_2 X^2 + a_3 X^3 \right) \cdot \left(0x02 + 0x01\, X + 0x01\, X^2 + 0x03\, X^3 \right) \text{ modulo } \left(X^4 + 1 \right)$$

Această operație se prezintă mai ușor așa:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Matricea e inversabilă, deci operația inversă este tot o înmulțire cu o matrice!

Add Round Key

$S \rightsquigarrow S \oplus K_i$ bitwise

Rijndael Encryption

AddRoundKey(S, K_0)

for i = 1 to 9 do
    SubBytes(S)
    ShiftRows(S)
    MixColumns(S)
    AddRoundKey(S, K_i)
end
SubBytes(S)
ShiftRows(S)
AddRoundKey(S, K_0)

Decryptam ↑

Mai rămâne de văzut cum se produce o cheie de rundă

Fie K cheia principală, |K| = 128.

$K = (K_0, K_1, K_2, K_3)$ fiecare K_i are 32 biți

$RC_i = X^i \bmod X^8 + X^4 + X^3 + X + 1$ round constant $\in \mathbb{F}_{2^8}$

Cheia de rundă $K_i = (W_{4i}, W_{4i+1}, W_{4i+2}, W_{4i+3})$

RotBytes = rotirea unui cuvânt cătră stânga, cu 1 byte

SubBytes = se aplică pe fiecare octet din-un cuvânt.

Rijndael Key Schedule

$W_0 = K_0$, $W_1 = K_1$, $W_2 = K_2$, $W_3 = K_3$

for $i = 1$ to 10 do

$T = \text{RotBytes}(W_{4i-1})$

$T = \text{SubBytes}(T)$

$T = T \oplus RC_i$

$W_{4i} = W_{4i-4} \oplus T$

$W_{4i+4} = W_{4i-3} \oplus W_{4i}$

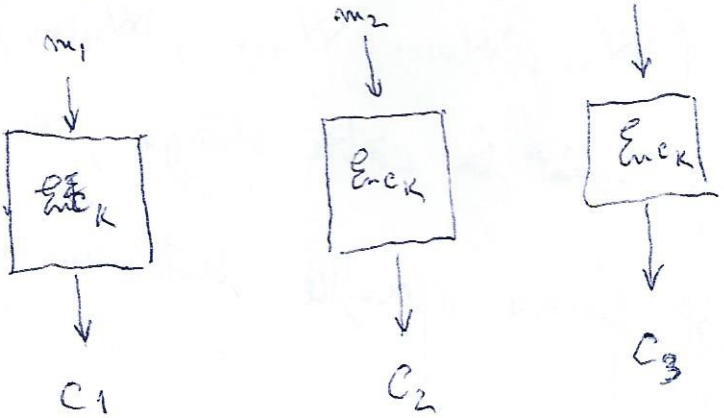
$W_{4i+2} = W_{4i-2} \oplus W_{4i+1}$

$W_{4i+3} = W_{4i-1} \oplus W_{4i+2}$

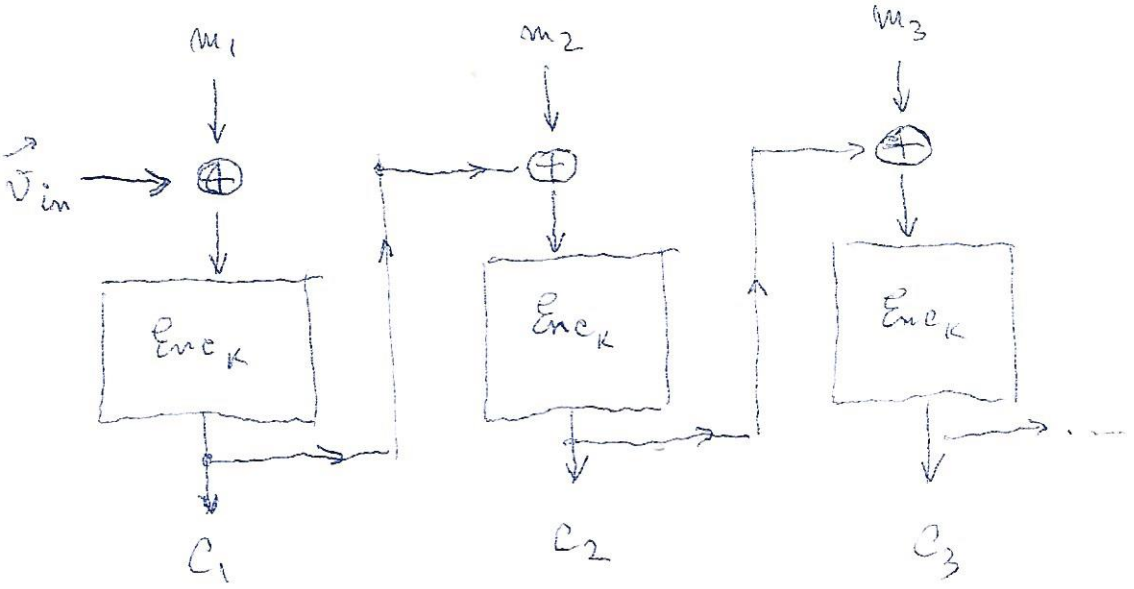
end

Moduri de operare pentru block-codes (DES, Rijndael, etc...)

ECB = electronic code-book mode



CBC mode = cypher block chaining (entrelacé)



$$c_1 = Enc_K(m_1 \oplus \vec{v}_i)$$

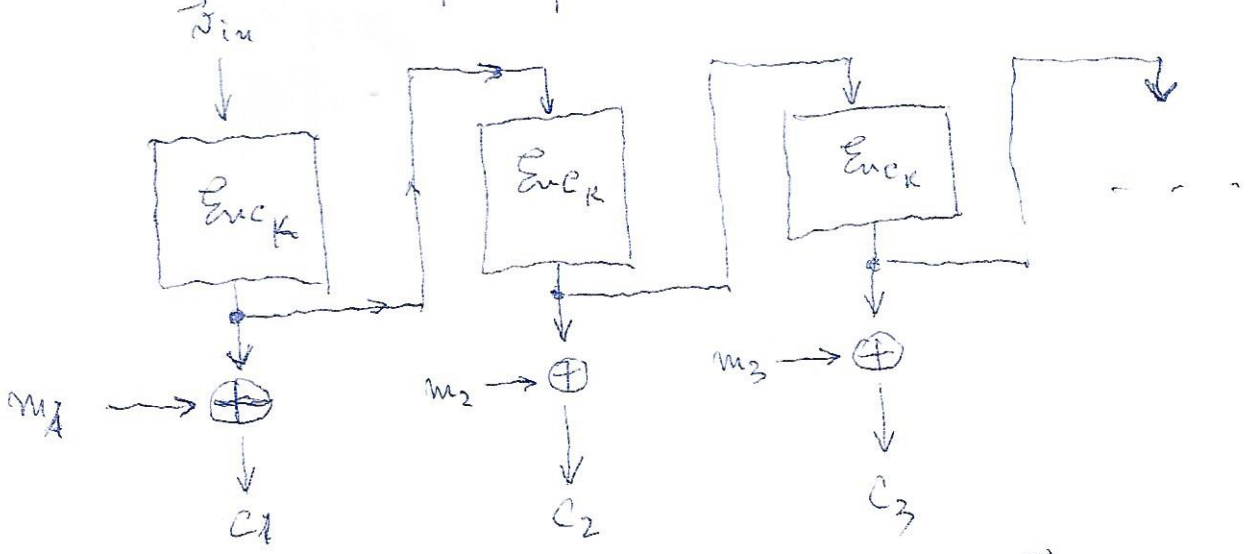
$$c_i = Enc_K(m_i \oplus c_{i-1})$$

$$\rightsquigarrow$$

$$m_1 = Dec_K(c_1) \oplus \vec{v}_{in}$$

$$m_i = Dec_K(c_i) \oplus c_{i-1}$$

OFB Mode = output feedback mode



$$c_1 = m_1 \oplus Enc_K(\vec{v}_{in})$$

$$c_2 = m_2 \oplus Enc_K^2(\vec{v}_{in})$$

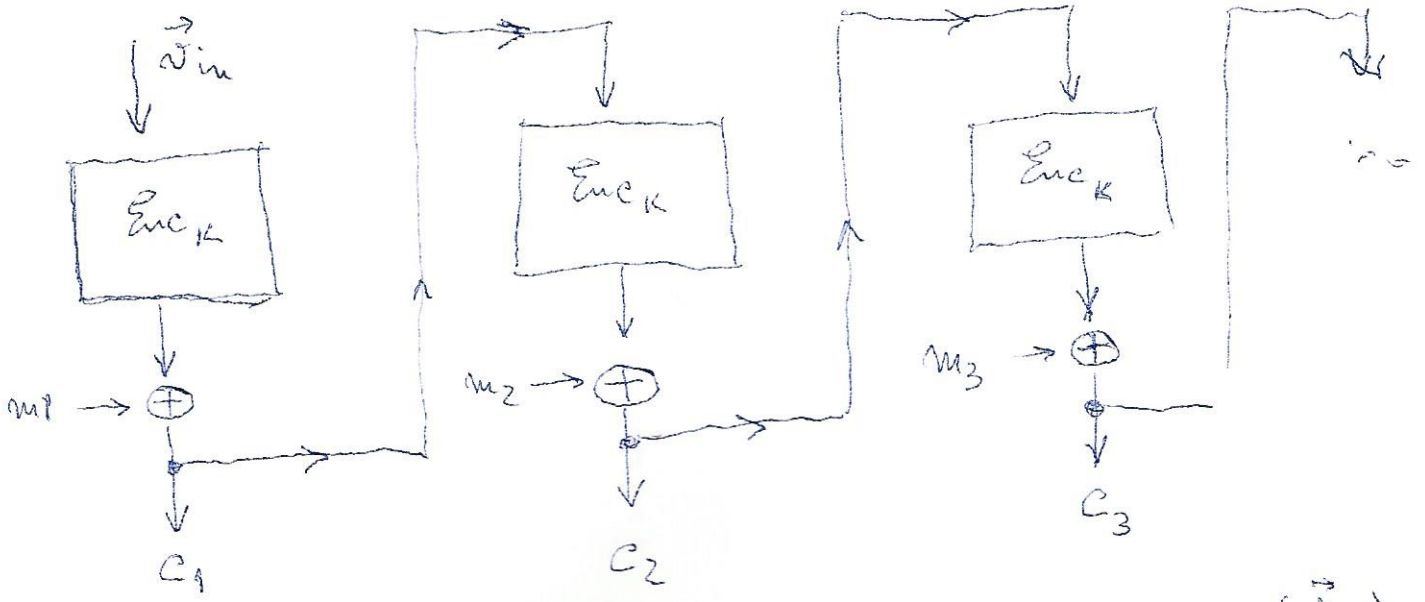
$$\vdots$$

$$m_1 = c_1 \oplus Enc_K(\vec{v}_{in})$$

$$m_2 = c_2 \oplus Enc_K^2(\vec{v}_{in})$$

$$\vdots$$

CFB Mode = Cypher Feed-Back mode



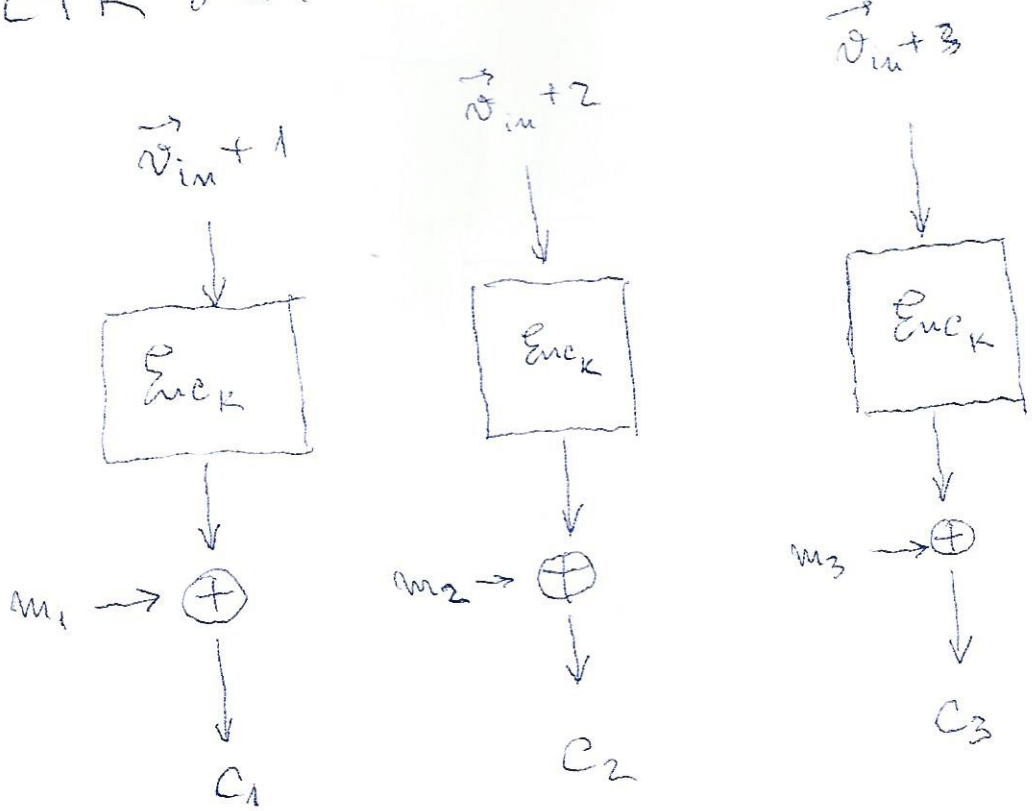
$$c_1 = m_1 \oplus Enc_K(\vec{v}_{in})$$

$$c_i = m_i \oplus Enc_K(c_{i-1})$$

$$m_1 = c_1 \oplus Enc_K(\vec{v}_{in})$$

$$m_i = c_i \oplus Enc_K(c_{i-1})$$

CTR Mode = counter mode



$$c_i = m_i \oplus Enc_K(\vec{v}_{in}+i)$$

$$m_i = c_i \oplus Enc_K(\vec{v}_{in}+i)$$

Hash Functions and Message Authentification

Codes

A alfabet, $h: A^* \to A^n$, $h(x) = y$ hash-value

- rezistență la preimagine:

  $|y| = n \Rightarrow \approx O(2^n)$ operații pt a găsi un x cu $h(x) = y$

- rezistență la coliziune
  - greu de găsit $x \neq x'$ cu $h(x) = h(x')$
  - dacă se calculează $h(x_1), h(x_2), h(x_3), \ldots$ se găsește $O(2^{n/2})$ pași până la prima coliziune.

- rezistență la a doua preimagine

  dat x, greu de găsit $x' \neq x$ cu $h(x) = h(x')$

  mai exact $O(2^n)$ operații

rezistență la preimagine $\rightsquigarrow$ rezistență la a doua preimagine $\rightsquigarrow$ rezistență la coliziune.

Merkle - Damgård - construction

- presupune că avem deja funcție $f: \{0,1\}^1 \to \{0,1\}^n$ care este rezistentă la coliziuni
- construiește o funcție h generală: $\{0,1\}^* \to \{0,1\}^n$.

- $l := s - n$
- pad m with 0's s.t. $|m| = t \cdot l$.
- divide m into t blocks of length l

$$m_1 \dots m_t$$

- H = fixed bit string, $|H| = n$
- for $i = 1$ to n do

$$H = f(H \| m_i)$$

end

- return H

Observație Fie $f : \{0,1\}^8 \longrightarrow \{0,1\}^4$ compresion

- Dacă aplicăm metoda pt

$m_1 = 010$ și $m_2 = 0100$ obținem

$$h(m_1) = f(01000000) = h(m_2)$$

- Se încearcă să mărim spațiul mesajului cu m $\$$, apoi să adăugăm 0-uri

$$h(m_1) = f(0101\underline{=}0000)$$
$$h(m_2) = f(0100\underline{=}1000)$$

în general diferite!

Acum ne ocupăm cu funcții de compresie.

Exemplu MD4.

MD4

$$f(u,v,w) = (u \wedge v) \vee (\neg u \wedge v)$$

$$g(u,v,w) = (u \wedge v) \vee (u \wedge w) \vee (v \wedge w)$$

$$h(u,v,w) = u \oplus v \oplus w$$

$u, v, w \in \{0,1\}^{32}$

Există o stare curentă (H_1, H_2, H_3, H_4)

$H_i \in \{0,1\}^{32}$

$H_1 = 0x67452301$

$H_2 = 0xEFCDAB89$

$H_3 = 0x98BADCFE = H_2$ ←

$H_4 = 0x10325476 = H_1$ ←

valori inițiale !

+ constante (y_i, z_i, s_i) care depind de fiecare rundă

$M \rightsquigarrow$ câte 16 cuvinte $X_0, \ldots X_{15}$ $\quad 16 \cdot 32 = 512$

$(A,B,C,D) = (H_1, H_2, H_3, H_4)$

for $j = 0$ to 15 do

$t = A + f(B,C,D) + X_{z_j} + y_j$ ⊕ R1

$(A,B,C,D) = (D, t \lll s_j, B, C)$ (rotație)

end

for $j = 16$ to 31 do

$t = A + g(B,C,D) + X_{z_j} + y_j$ R2

$(A,B,C,D) = (D, t \lll s_j, B, C)$

(R3) e la fel, doar cu funcția h.

$(H_1, H_2, H_3, H_4) = (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$

Hash funcțion se poate obține și dintr-un bloc cifru

Matyas - Meyer - Oseas

$H_i = f(x_i, H_{i-1}) = Enc_{H_{i-1}}(x_i) \oplus x_i$

Davies - Meyer

$H_i = f(x_i, H_{i-1}) = Enc_{x_i}(H_{i-1}) \oplus H_{i-1}$

Miyaguchi - Preneel hash

$H_i = f(x_i, H_{i-1}) = Enc_{H_{i-1}}(x_i) \oplus x_i \oplus H_{i-1}$

Message Authentification Codes

trimit $Enc_{K_1}(m) \| Mac_{K_2}(Enc_{K_1}(m))$

pentru Mac folosesc sau hash-function

block-codes : CBC-Mac