# Prelegerea 4

# Sisteme de criptare fluide

## 4.1 Sisteme sincronizabile şi auto-sincronizabile

În sistemele de criptare prezentate până acum, elementele succesive ale textului clar erau criptate folosind aceeaşi cheie $K$. Deci, dacă

$$\mathbf{x} = x_1 x_2 x_3 \dots$$

este textul clar, textul criptat este

$$\mathbf{y} = y_1 y_2 y_3 \dots = e_K(x_1) e_K(x_2) e_K(x_3) \dots$$

Sistemele de criptare de acest tip se numesc *sisteme de criptare bloc (block cyphers).*

Altă manieră utilizată este aceea a sistemelor de criptare *fluide (stream cyphers).*

**Definiţia 4.1** *Fie* $\mathcal{M}= (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *un sistem de criptare. O secvenţă de simboluri* $k_1 k_2 k_3 \dots \in \mathcal{K}^+$ *se numeşte cheie fluidă.*

**Definiţia 4.2** $\mathcal{M}= (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *este un sistem fluid dacă criptează un text clar*

$$\mathbf{x} = x_1 x_2 x_3 \dots$$

*în*

$$\mathbf{y} = y_1 y_2 y_3 \dots = e_{k_1}(x_1) e_{k_2}(x_2) e_{k_3}(x_3) \dots$$

*unde*

$$\mathbf{k} = k_1 k_2 k_3 \dots$$

*este o cheie fluidă din* $\mathcal{K}^+$.

Problema principală este aceea de a genera o astfel de cheie de criptare (teoretic infinit). Aceasta se poate realiza fie aleator, fie pe baza unui algoritm care pleacă de la o secvenţă mică de chei de criptare. Un astfel de algoritm se numeşte *generator de chei fluide.* Mai multe detalii vor fi prezentate în cadrul prelegerii dedicate generatorilor de numere pseudo-aleatoare.

**Exemplul 4.1** *(sistemul de criptare Vernam):*
*În acest sistem* $x_i, k_i, y_i \in \{0,1\}$. *Criptarea se realizează conform formulei*

$$y_i = x_i \oplus k_i, \quad (i \geq 1)$$

*Dacă cheia fluidă este aleasă aleator şi nu mai este folosită ulterior, sistemul de criptare Vernam se numeşte "one - time pad".*

*Un sistem de criptare one-time pad este teoretic imposibil de spart.* [1] *Într-adevăr, fiind dat un text criptat cu un astfel de sistem, Oscar nu are nici o informaţie privind cheia fluidă sau textul clar. Dificultatea constă însă atât în lungimea cheii (egală cu cea a textului clar), cât şi în modalitatea de transmitere a ei între Alice şi Bob.*

*Pentru sistemul Vernam există o modalitate de atac cunoscută sub numele de "criptanaliză diferenţială" (prezentată mai târziu, în cadrul sistemului de criptare DES). Anume:*

*Dacă* $y_1y_2 \ldots y_n$ *şi* $y'_1y'_2 \ldots y'_n$ *sunt două texte criptate cu aceiaşi chee fluidă* $k_1k_2 \ldots k_n$, *atunci*

$$y_i = x_i \oplus k_i, \qquad y'_i = x'_i \oplus k_i \quad (1 \leq i \leq n)$$

*deci* $y_i \oplus y'_i == x_i \oplus x'_i$, *şi cheia nu mai este necesară, ci doar informaţia privind lungimea sa; redundanţa ultimei relaţii va permite criptanaliza.*

Sistemele de criptare fluide sunt clasificate în *sisteme sincrone* şi *auto-sincronizabile.*

**Definiţia 4.3** *Un sistem de criptare fluid sincron este o structură* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{E}, \mathcal{D})$, *unde:*

- $\mathcal{P}, \mathcal{C}, \mathcal{K}$ *sunt mulţimi finite nevide ale căror elemente se numesc "texte clare", "texte criptate" şi respectiv "chei";*
- $\mathcal{L}$ *este o mulţime finită nevidă numită "alfabet şir de chei";*
- $g : \mathcal{K} \longrightarrow \mathcal{L}^+$ *este un generator de chei fluide: pentru*
$$\forall k \in \mathcal{K}, \quad g(k) = k_1k_2k_3 \ldots \in \mathcal{L}^+$$
*este o cheie fluidă (teoretic infinită);*
- $\forall z \in \mathcal{L}$ *există o regulă de criptare* $e_z \in \mathcal{E}$ *şi o regulă de decriptare* $d_z \in \mathcal{D}$ *astfel ca*
$$\forall x \in \mathcal{P}, \quad d_k(e_k(x)) = x$$

**Exemplul 4.2** *Sistemul de criptare Vigenere poate fi definit ca un sistem de criptare fluid sincron. Fie m lungimea cuvântului cheie din sistemul Vigenere. Definim*

[1] În anii '90 comunicarea între Moscova şi Washington era securizată în acest mod, transportul cheii de criptare fiind asigurat de curieri.

$$\mathcal{P} = \mathcal{C} = \mathcal{L} = Z_{26}, \quad \mathcal{K} = (Z_{26})^m,$$
$$e_z(x) = x + z \ (mod\ 26), \quad d_z(y) = y - z \ (mod\ 26)$$

*Cheia* $z_1 z_2 z_3 \ldots$ *este definită*

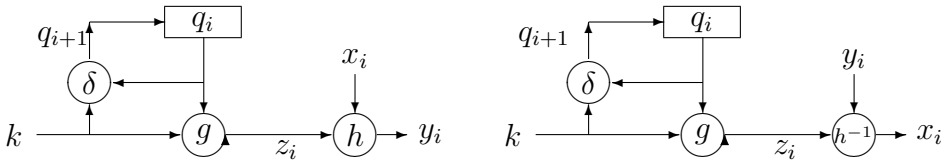
$$z_i = \begin{cases} k_i & dacă \quad 1 \leq i \leq m \\ z_{i-m} & dacă \quad i \geq m+1 \end{cases}$$

*Ea va genera din cheia fixă* $K = (k_1, k_2, \ldots, k_m)$*, cheia fluidă* $k_1 k_2 \ldots k_m k_1 k_2 \ldots k_m$ $k_1 k_2 \ldots$

Procesul de criptare cu un sistem fluid sincron poate fi reprezentat ca un automat descris de relaţiile

$$q_{i+1} = \delta(q_i, k), \quad z_i = g(q_i, k), \quad y_i = h(z_i, x_i)$$

unde $q_0$ este *starea iniţială* – care poate fi determinată din cheia $k$, $\delta$ este *funcţia de tranziţie a stărilor*, $g$ este *funcţia care produce cheia fluidă* $z_i$, iar $h$ este *funcţia de ieşire* care produce textul criptat $y_i$ pe baza textului clar $x_i$ şi a cheii fluide $z_i$.




(a) Criptare (b) Decriptare

**Observaţia 4.1**

- *Un sistem de criptare bloc poate fi privit ca un caz particular de sistem de criptare fluid, în care* $\forall i \geq 1, \quad z_i = K$.

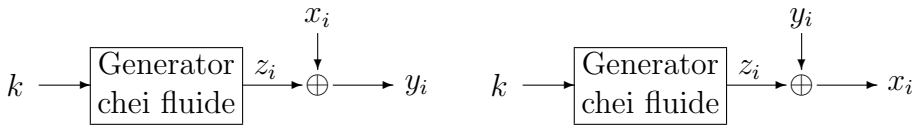
- *(sincronizare): În sistemele de criptare fluide sincrone, Alice şi Bob trebuie să-şi sincronizeze cheia fluidă pentru a putea obţine o criptare/decriptare corectă. Dacă în timpul transmisiei sunt inseraţi sau eliminaţi biţi în textul criptat, atunci decriptarea eşuează şi ea poate fi reluată numai pe baza unor tehnici de resincronizare (cum ar fi de exemplu reiniţializarea sau plasarea unor marcatori speciali la intervale regulate în textul transmis).*

- *Modificarea unui bit din textul criptat (fără a se elimina sau adăuga nimic) nu va afecta decriptarea altor caractere (nepropagarea erorii).*

- *Un adversar activ care elimină, inserează sau retrimite componente ale mesajului criptat, va provoca desincronizări şi va fi deci detectat la recepţie. De asemenea, el poate face modificări în textul criptat şi să observe cum vor afecta ele textul clar. Deci un text criptat cu un sistem fluid sincron necesită meca-nisme separate de autentificare şi de garantare a integrităţii datelor.*

**Definiţia 4.4** *Un sistem aditiv fluid binar de criptare este un sistem fluid sincron în care* $\mathcal{P} = \mathcal{C} = \mathcal{L} = Z_2$ *iar* $h$ *este funcţia* $\oplus$ *(XOR).*

Un astfel de sistem este reprezentat mai jos:




(a) Criptare (b) Decriptare

**Definiţia 4.5** *Un sistem de criptare fluid este auto-sincronizabil (sau "asincron") dacă funcţia de generare a cheii fluide depinde de un număr fixat de caractere criptate anterior.*

Deci comportamentul unui astfel de sistem poate fi descris de ecuaţiile

$$q_i = (y_{i-t}, y_{i-t+1}, \ldots, y_{i-1}), \quad z_i = g(q_i, k), \quad y_i = h(z_i, x_i)$$

unde $q_0 = (y_{-t}, y_{-t+1}, \ldots, y_{-1})$ este *starea iniţială* (cunoscută), $k$ este cheia, $g$ este funcţia de generare a cheii fluide, iar $h$ este *functia de ieşire* care criptează textul clar $x_i$. Cele mai cunoscute sisteme auto-sincronizabile sunt regiştrii liniari cu feedback, utilizaţi la generarea secvenţelor de numere pseudo-aleatoare (în criptografie) şi la generarea codurilor ciclice (în teoria codurilor).

**Exemplul 4.3** *(Criptare cu auto-cheie)*[2]*:*
*Fie* $\mathcal{P} = \mathcal{C} = \mathcal{L} = Z_{26}$*. Se defineşte cheia fluidă astfel:*

$$z_1 = K, \quad z_i = y_{i-1}, \ (i \geq 2)$$

*Pentru un element oarecare al cheii* $z \in Z_{26}$*, se defineşte*

$$e_z(x) = x + z \ (mod\ 26)$$
$$d_z(y) = y - z \ (mod\ 26)$$

*Să considerăm* $K = 13$ *şi să criptăm textul clar* $BUCURESTI$*.*

[2]Se pare că sistemul este atribuit lui Vigenere.

*Transformat în secvenţă numerică vom avea*

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = (1, 20, 2, 20, 17, 4, 18, 19, 8)$$

*În faza de criptare, vom calcula pe rând:*
$y_1 = e_{13}(x_1) = 1 + 13 \ (mod\ 26) = 14$; $y_2 = e_{14}(x_2) = 20 + 14 \ (mod\ 26) = 8$;
$y_3 = e_8(x_3) = 2 + 8 \ (mod\ 26) = 10$; $y_4 = e_{10}(x_4) = 20 + 10 \ (mod\ 26) = 4$;
$y_5 = e_4(x_5) = 17 + 4 \ (mod\ 26) = 21$; $y_6 = e_{21}(x_6) = 4 + 21 \ (mod\ 26) = 25$;
$y_7 = e_{25}(x_7) = 18 + 25 \ (mod\ 26) = 17$; $y_8 = e_{17}(x_8) = 19 + 17 \ (mod\ 26) = 10$;
$y_9 = e_{10}(x_9) = 8 + 10 \ (mod\ 26) = 18$;
*Deci textul criptat este* $(14, 8, 10, 4, 21, 25, 17, 10, 18)$ *sau – în litere:* $OIKEVZRKS$.

*Pentru decriptare, vom avea:*
$x_1 = d_{13}(y_1) = 14 - 13 \ (mod\ 26) = 1$; $x_2 = d_{14}(y_2) = 8 - 14 \ (mod\ 26) = 20$; *ş.a.m.d.*

*Se observă că textul decriptat poate fi obţinut de oricine, aproape în totalitate, fără a cunoaşte nici o cheie (aceasta fiind necesară doar pentru decriptarea primului caracter). Deci gradul său de securitate este nul.*

*Ceva mai dificil este sistemul în care generarea cheii fluide se realizează cu ecuaţia* $z_i = x_{i-1}$ $(i \geq 2)$.

*În acest caz,* $BUCURESTI$ *se criptează în* $OVWWLVWLB$ *(pentru* $K = 13$*).*

*Nici acest sistem de criptare cu auto-cheie nu este sigur, deoarece – evident – sunt posibile doar* 26 *chei.*

**Proprietăţi**:

1. *Auto-sincronizare*: Deoarece funcţia de decriptare $h^{-1}$ depinde numai de un număr fixat de caractere criptate anterior, desincronizarea – rezultată eventual prin inserarea sau ştergerea de caractere criptate – se poate evita. Astfel de sisteme de criptare pot restabili proprietatea de sincronizare afectând doar un număr finit de caractere din textul clar.

2. *Propagarea limitată a erorii*: Dacă starea unui sistem fluid auto-sincronizabil depinde de $t$ caractere anterioare, atunci modificarea (eventual ştergerea sau inserarea) unui caracter din textul criptat poate duce la decriptarea incorectă a maxim $t$ caractere; după aceea decriptarea redevine corectă.

3. *Răspândirea textelor clare*: Deoarece fiecare caracter din textul clar influenţează întregul text criptat care urmează, eventualele proprietăţi statistice ale textului clar sunt dispersate prin textul criptat. Deci, din acest punct de vedere, sistemele de criptare auto-sincronizabile sunt mai rezistente decât cele sincronizabile la atacurile bazate pe redondanţa textului clar.

4. *Rezistenţa la criptanaliză activă*: Din proprietăţile de mai sus rezultă că este destul de dificil de depistat un atac venit din partea unui adversar activ (care poate modifica, insera sau şterge caractere) auto-sincronizarea readucând decriptarea în faza

normală. De aceea sunt necesare mecanisme suplimentare pentru a asigura autentificarea şi integritatea datelor.

## 4.2 Exemple de sisteme fluide de criptare

### 4.2.1 SEAL

*SEAL* (**S**oftware - optimized **E**ncryption **AL**gorithm) este un sistem de criptare aditiv binar (Definiţia 4.4), definit în 1993 de Coppersmith şi Rogaway. Este unul din cele mai eficiente sisteme implementabile pe procesoare de 32 biţi.

La un astfel de sistem este importantă descrierea generatorului de chei fluide (care se adună modulo 2 cu textul clar). *SEAL* este o funcţie pseudo-aleatoare care scoate o cheie fluidă de $L$ biţi folosind un număr $n$ de 32 biţi şi o cheie secretă $a$ de 160 biţi.

Fie $A, B, C, D, X_i, Y_j$ cuvinte de 32 biţi. Vom folosi următoarele notaţii:

1. $\overline{A}$: complementul lui $A$ (pe biţi);
2. $A \vee B, A \wedge B, A \oplus B$: operaţiile $OR, AND$ şi $XOR$ (pe biţi);
3. $A << s$: rotirea ciclică a lui $A$ spre stânga cu $s$ poziţii;
4. $A >> s$: rotirea ciclică a lui $A$ spre dreapta cu $s$ poziţii;
5. $A + B \ (mod\ 2^{32})$: suma lui $A$ şi $B$ (considerate ca numere întregi fără semn);
6. $f(B, C, D) = (B \wedge C) \vee (\overline{B} \wedge D)$;
   $g(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$; $\quad h(B, C, D) = B \oplus C \oplus D$.
7. $A||B$: concatenarea lui $A$ cu $B$;
8. $(X_1, X_2, \ldots, X_n) \longleftarrow (Y_1, , , , , Y_2, \ldots, Y_n)$: atribuire simultană.

**Algoritmul de generare a tabelei $G$ pentru SEAL 2.0**: [3]

**Intrare**: Un şir $a$ de 160 biţi şi un întreg $i$ $(0 \leq i < 2^{32})$.
**Ieşire**: $G_a(i)$ – şir de 160 biţi.
**Algoritm 1**:

1. Se definesc constantele (de 32 biţi):

   $y_1 = 0x5a827999, \quad y_2 = 0x6ed9eba1,$
   $y_3 = 0x8f1bbcdc, \quad y_4 = 0xca62c1d6$

2. *a.* $X_0 \longleftarrow i$;

   *b.* **for** $j \longleftarrow 1$ **to** $15$ **do** $X_j \longleftarrow 0x00000000$;

   *c.* **for** $j \longleftarrow 16$ **to** $79$ **do** $X_j \longleftarrow ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) <<< 1)$;

   *d.* $(A, B, C, D, E) \longleftarrow (H_0, H_1, H_2, H_3, H_4)$ **where** $a = H_0H_1H_2H_3H_4$;

   *e.* (Runda 1): **for** $j \longleftarrow 0$ **to** $19$ **do**

   $$t \longleftarrow ((A <<< 5) + f(B, C, D) + E + X_j + y_1);$$
   $$(A, B, C, D, E) \longleftarrow (t, A, B <<< 30, C, D);$$

   *f.* (Runda 2): **for** $j \longleftarrow 20$ **to** $39$ **do**

   $$t \longleftarrow ((A <<< 5) + h(B, C, D) + E + X_j + y_2);$$
   $$(A, B, C, D, E) \longleftarrow (t, A, B <<< 30, C, D);$$

   *g.* (Runda 3): **for** $j \longleftarrow 40$ **to** $59$ **do**

   $$t \longleftarrow ((A <<< 5) + g(B, C, D) + E + X_j + y_3);$$
   $$(A, B, C, D, E) \longleftarrow (t, A, B <<< 30, C, D);$$

   *h.* (Runda 4): **for** $j \longleftarrow 60$ **to** $79$ **do**

   $$t \longleftarrow ((A <<< 5) + h(B, C, D) + E + X_j + y_4);$$
   $$(A, B, C, D, E) \longleftarrow (t, A, B <<< 30, C, D);$$

   *i.* $(H_0, H_1, H_2, H_3, H_4) \longleftarrow (H_0 + A, H_1 + B, H_2 + C, H_3 + D, H_4 + E);$

$$G_a(i) = H_0||H_1||H_2||H_3||H_4.$$

$SEAL(a, n)$ (**Generatorul de chei fluide pentru SEAL 2.0**):

[3] Algoritmul SEAL 1.0 este bazat pe funcţia de dispersie $SHA$, iar SEAL 2.0 – pe funcţia $SHA-1$.

**Intrare**: $a$ – cheia secretă (160 biţi), $n \in [0, 2^{32})$ întreg, $L$ - lungimea solicitată pentru cheia fluidă.
**Ieşire**: cheia fluidă $y$, $|y| = L'$, unde $L'$ este primul multiplu de 128 din $[L, \infty)$.

1. Se generează tabelele $T, S, R$ având ca elemente cuvinte de 32 biţi.

   Fie funcţia $F_a(i) = H^i_{i \,(mod\, 5)}$ unde $H^i_0 H^i_1 H^i_2 H^i_3 H^i_4 = G_a(\lfloor i/5 \rfloor)$.

   *a.* **for** $i \longleftarrow 0$ **to** $511$ **do** $T[i] \longleftarrow F_a(i)$;

   *b.* **for** $j \longleftarrow 0$ **to** $255$ **do** $S[j] \longleftarrow F_a(0x00001000 + j)$;

   *c.* **for** $k \longleftarrow 0$ **to** $4 \cdot \lceil (L-1)/8192 \rceil - 1$ **do** $R[k] \longleftarrow F_a(0x00002000 + k)$;

2. Descrierea procedurii $Initialize(n, l, A, B, C, D, n_1, n_2, n_3, n_4)$ cu intrările $n$ (cuvânt) şi $l$ (întreg). Ieşirile sunt $A, B, C, D, n_1, n_2, n_3, n_4$ (cuvinte).

   *a.* $A \longleftarrow n \oplus R[4 \cdot l]$, $\quad B \longleftarrow (n >> 8) \oplus R[4 \cdot l + 1]$,
   $C \longleftarrow (n >> 16) \oplus R[4 \cdot l + 2]$, $\quad D \longleftarrow (n >> 24) \oplus R[4 \cdot l + 3]$.

   *b.* **for** $j \longleftarrow 1$ **to** $2$ **do**

   $$P \longleftarrow A \wedge 0x000007fc, \quad B \longleftarrow B + T[P/4], \quad A \longleftarrow (A >> 9),$$
   $$P \longleftarrow B \wedge 0x000007fc, \quad C \longleftarrow C + T[P/4], \quad B \longleftarrow (B >> 9),$$
   $$P \longleftarrow C \wedge 0x000007fc, \quad D \longleftarrow D + T[P/4], \quad C \longleftarrow (C >> 9),$$
   $$P \longleftarrow D \wedge 0x000007fc, \quad A \longleftarrow A + T[P/4], \quad D \longleftarrow (D >> 9),$$
   $$(n_1, n_2, n_3, n_4) \longleftarrow (D, A, B, C);$$
   $$P \longleftarrow A \wedge 0x000007fc, \quad B \longleftarrow B + T[P/4], \quad A \longleftarrow (A >> 9),$$
   $$P \longleftarrow B \wedge 0x000007fc, \quad C \longleftarrow C + T[P/4], \quad B \longleftarrow (B >> 9),$$
   $$P \longleftarrow C \wedge 0x000007fc, \quad D \longleftarrow D + T[P/4], \quad C \longleftarrow (C >> 9),$$
   $$P \longleftarrow D \wedge 0x000007fc, \quad A \longleftarrow A + T[P/4], \quad D \longleftarrow (D >> 9),$$

3. $l \longleftarrow 0$, $y \longleftarrow \epsilon$ (şirul vid);

4. **repeat**

   *a.* $Initialize(n, l, A, B, C, D, n_1, n_2, n_3, n_4)$;

   *b.* **for** $i \longleftarrow 1$ **to** $64$ **do**

   $P \longleftarrow A \wedge 0x000007fc,\ B \longleftarrow B + T[P/4],\ A \longleftarrow (A >> 9),\ B \longleftarrow B \oplus A;$
   $Q \longleftarrow B \wedge 0x000007fc,\ C \longleftarrow C + T[Q/4],\ B \longleftarrow (B >> 9),\ C \longleftarrow C \oplus B;$
   $P \longleftarrow (P + C) \wedge 0x000007fc,\ D \longleftarrow D + T[P/4],\ C \longleftarrow (C >> 9),\ D \longleftarrow D \oplus C;$
   $Q \longleftarrow (Q + D) \wedge 0x000007fc,\ A \longleftarrow A + T[Q/4],\ D \longleftarrow (D >> 9),\ A \longleftarrow A \oplus D;$
   $P \longleftarrow (P + A) \wedge 0x000007fc,\ B \longleftarrow B + T[P/4],\ A \longleftarrow (A >> 9);$
   $Q \longleftarrow (Q + B) \wedge 0x000007fc,\ C \longleftarrow C + T[Q/4],\ B \longleftarrow (B >> 9);$

$P \longleftarrow (P + C) \wedge 0x000007fc,\ D \longleftarrow D + T[P/4],\ C \longleftarrow (C >> 9);$
$Q \longleftarrow (Q + D) \wedge 0x000007fc,\ A \longleftarrow A + T[Q/4],\ D \longleftarrow (D >> 9);$
$y \longleftarrow y||(B + S[4 \cdot i - 4])||(C \oplus S[4 \cdot i - 3])||(D + S[4 \cdot i - 2])||(A \oplus S[\cdot i - 1]).$

**if** $|y| \geq L$ **then return**$(y)$ **STOP**
**else if** $i\ (mod\ 2) = 1$ **then** $(A, C) \longleftarrow (A + n_1, C + n_2)$
**else** $(A, C) \longleftarrow (A + n_3, C + n_4)$
$c.\ l \longleftarrow l + 1.$

**Observaţia 4.2** *([1]) În majoritatea aplicaţiilor pentru SEAL 2.0 se foloseşte* $L \leq 2^{19}$*. Algoritmul funcţionează şi pentru valori mai mari, dar devine ineficient deoarece creşte mult dimensiunea tabelei R. O variantă este concatenarea cheilor fluide* $SEAL(a, 0)||$ $SEAL(a, 1)||SEAL(a, 2)||\ldots$ *Deoarece* $n < 2^{32}$*, se pot obţine astfel chei fluide de lungimi până la* $2^{51}$*, păstrând* $L = 2^{19}$*.*

### 4.2.2 RC4

Sistemul *RC*4 (Rivest Code #4) a fost creat în 1987 de Ron Rivest pentru societatea RSA Data Security Inc (astăzi *RSA Security*). Destinat scopurilor comerciale, el a fost secret, fiind accesibil numai după semnarea unui protocol de confidenţialitate. În septembrie 1994 însă, un anonim publică codul său sursă pe o listă de discuţii, după care se răspândeşte rapid stârnind o serie de polemici referitor la drepturile intelectuale. Se consideră că astăzi există mai multe implementări ilegale.

*RC*4 este un sistem aditiv fluid de criptare.

Printre sistemele care folosesc *RC*4 se pot aminti SQL (Oracle), Lotus Notes, AOCE (Apple Computer), WEP WPA CipherSaber Secure Sockets Layer (opţional) sau Secure shell (opţional)[4] *RC*4 este utilizat pentru criptarea fişierelor în protocoale cum ar fi *RSA SecurPC* sau în standarde de comunicaţii (WEP, WPA pentru carduri, criptarea traficului de date sau a site-urilor de web bazate pe protocolul SSL). De asemenea, el face parte din Cellular Digital Packet Data specification.

La acest sistem, pentru generarea cheii fluide se foloseşte o stare internă (secretă) formată din două componente:

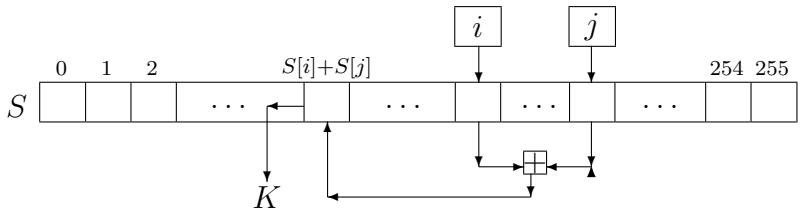
- Un tablou de 256 octeţi: $S[0], S[1], \ldots S[255]$, numit $S - box$.
- Doi pointeri de câte 8-biţi (notaţi "$i$" respectiv "$j$").

$S - boxul$ este iniţializat cu o cheie de lungime variabilă – de obicei între 40 şi 256 biţi, folosind un algoritm numit $KSA$ (key-sheduling algorithm).

În faza a doua, cheia fluidă este generată folosind algoritmul $PRGA$ (pseudo-random generation algorithm).

[4]Când un sistem de criptare este marcat *opţional*, însemnă că el este una din variantele oferite pentru implementare.

**Algoritmul PRGA de generare a cheii fluide**




Conţinuturile $S[i]$ şi $S[j]$ (unde $i, j$ sunt date de cei doi pointeri) se adună modulo 256, iar octetul $K$ de la adresa $S[i] + S[j]$ este introdus în cheia fluidă. În plus cei doi octeţi sunt interschimbaţi.

Procedeul este reluat atât timp cât este nevoie. La fiecare reluare starea celor doi pointeri se modifică ($i$ este incrementat cu 1 iar $j$ este incrementat cu $S[i]$). În acest fel, orice locaţie din $S - box$ este modificată cel puţin odată ;a 256 iteraţii.

Formal:

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output S[(S[i] + S[j]) mod 256]
```

**Algoritmul $KSA$ de iniţializare**

$KSA$ este utilizat pentru iniţializarea $S - boxului$. Fie $n$ ($1 \leq n \leq 255$) numărul de octeţi din cheie[5]. Iniţial în $S$ se introduce permutarea identică. Ulterior se realizează 256 transpoziţii. Formal

```
for i from 0 to 255 do S[i] := i
j := 0
for i from 0 to 255 do
    j := (j + S[i] + key[i mod n]) mod 256
    swap(S[i],S[j])
```

Implementarea sistemului $RC4$ este foarte simpla: ea lucrează cu octeţi şi necesită 256 pentru $S - box$, n octeţi de memorie pentru cheie, plus trei variabile întregi $i, j, k$. Operaţiile folosite sunt $XOR$ şi $AND$ (sau – pe unele platforme – simpla adunare pe biţi, fără transport).

[5]În majoritatea implementărilor $n$ este în intervalul $[5, 16]$.

**Securitatea** *RC4*

Conform cu Bruce Schneier ([2]), sistemul are circa $256! \times 256^2 = 2^{1700}$ stări posibile; el este rezistent la criptanaliza diferenţială şi liniară, iar ciclurile sunt mai mari de 10.100.

Totuşi, securitatea *RC4* este slabă din mai multe puncte de vedere şi criptografii nu recomandă sistemul pentru aplicaţiile actuale.

Cheia fluidă generată are o uşoară preferinţă pentru anumite secvenţe de octeţi. Aceasta a permis construirea unui atac (Fluhrer şi McGrew), care separă cheia fluidă dintr-o secvenţă aleatoare de maxim 1 GB.

În 2001, Fluhrer, Mantin şi Shamir descoperă că din toate cheile RC4 posibile, primii octeţi de ieşire nu sunt aleatori, oferind informaţii importante despre cheie.

La majoritatea sistemelor de criptare, o măsură de securitate ncesară este alegerea unui număr aleator nou[6] care să fie folosit pentru criptarea fiecărui mesaj. În acest fel, criptarea de două ori a aceluiaşi mesaj va genera texte diferite. O solutîe sigură (care funcţionează pentru orice sistem de criptare) este de a folosi o cheie pe termen lung din care, prin amestec cu un nonce (după un algoritm prestabilit) se obţine cheia necesară unei criptări. Multe aplicaţii însă – care folosesc *RC4* – fac o simplă concatenare a cheii cu nonce. Această slabiciune este exploatată de Fluhrer, Mantin şi Shamir pentru a sparge ulterior sistemul de criptare *WEP* (wired equivalent pruvacy) folosit pe reţelele fără fir 802.11.

Ulterior implementările sistemului *RC4* s-au apărat eliminând primii octeţi (uzual 1024) din cheia fluidă, înainte de a o folosi.

## 4.3 Exerciţii

1. Construiţi coduri de autentificare a mesajelor (MAC) capabile să autentifice mesajele primite printr-un sistem fluid de criptare. (Indicaţie: a se vedea [1], paragraf 9.5.4)

2. Presupunem că definim o cheie fluidă într-un sistem sincronizabil în felul următor: Fie $K \in \mathcal{K}$, $\mathcal{L}$ un alfabet al cheilor şi $\Sigma$ o mulţime finită de stări. Se defineşte o stare iniţială $q_0 \in \sigma$. Apoi, pentru $i \geq 1$, se defineşte recursiv
$$q_i = f(q_{i-1}, K)$$
unde $f : \Sigma \times \mathcal{K} \longrightarrow \Sigma$. De asemenea, $\forall i \geq 1$, elementul $z_i$ din cheia fluidă este definit prin
$$z_i = g(q_i, K)$$
unde $g : \Sigma \times \mathcal{K} \longrightarrow \mathcal{L}$. Arătaţi că orice cheie fluidă rezultată în acest mod are o perioadă de lungime maxim $|\Sigma|$.

[6] Un astfel de număr poartă numele de *nonce* (new number)

# Bibliografie

[1] Menezes, Oorschot, Vanstome - *Handbook of applied cryptography*, 1997

[2] Schneier, B. - *Applied Cryptography*, John Wiley & Sons, second edition, 1997

[3] Stinton, D. – *Cryptography, Theory and Practice*, Chaptan & Hall/CRC, second edition 2002