Capitolul 4

Sisteme fluide de criptare

4.1 Sisteme sincrone 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 de criptare dacă cifrează 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 cheie de criptare fluidă (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.*

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] *(deci asigură un secret perfect). 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" (noţiunea va fi prezentată detaliat în Capitolul 6). 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 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

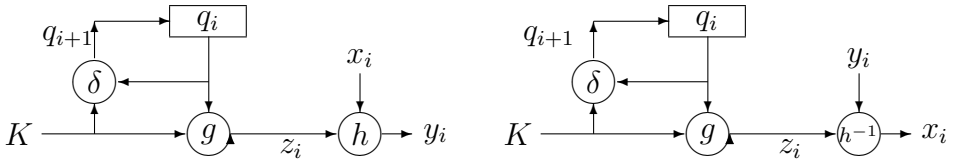
$$z_i = \begin{cases} k_i & dacă \quad 1 \leq i \leq m \\ z_{i-m} & dacă \quad i \geq m+1 \end{cases}$$

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, δ 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.



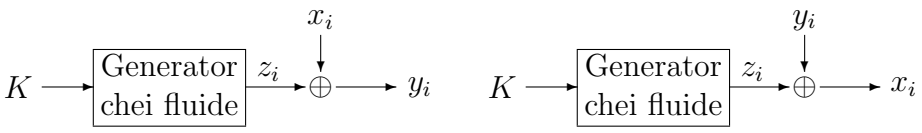
(a) Criptare (b) Decriptare

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 mecanisme 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 funcţia de ieşire* h *este* $\oplus$ *(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 ($LFSR$), 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:*

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

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* $BUCURESTI$*.*
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:
$y_1 = e_{13}(x_1) = 1 + 13 \ (mod\ 26) = 14$; $y_2 = e_{14}(x_2) = 20 + 14 \ (mod\ 26) = 8$;
$y_3 = e_8(x_3) = 2 + 8 \ (mod\ 26) = 10$; $y_4 = e_{10}(x_4) = 20 + 10 \ (mod\ 26) = 4$;
$y_5 = e_4(x_5) = 17 + 4 \ (mod\ 26) = 21$; $y_6 = e_{21}(x_6) = 4 + 21 \ (mod\ 26) = 25$;

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

$y_7 = e_{25}(x_7) = 18 + 25 \ (mod\ 26) = 17$; $y_8 = e_{17}(x_8) = 19 + 17 \ (mod\ 26) = 10$;
$y_9 = e_{10}(x_9) = 8 + 10 \ (mod\ 26) = 18$;
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. *Difuzia 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 (bit cu bit);
2. $A \vee B, A \wedge B, A \oplus B$: operaţiile OR, AND şi XOR (definite 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. $f(B, C, D) = (B \wedge C) \vee (\overline{B} \wedge D)$;
   $g(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
   $h(B, C, D) = B \oplus C \oplus D$.
7. $A||B$: concatenarea (alăturarea) lui A cu B;
8. $(X_1, X_2, \ldots, X_n) \longleftarrow (Y_1, Y_2, \ldots, Y_n)$: atribuire simultană (variabila Y_i ia valoarea X_i simultan pentru toate valorile lui $i = 1, 2, \ldots, n$).

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):
$$y_1 = 5A827999, \quad y_2 = 6ED9EBA1, \quad y_3 = 8F1BBCDC, \quad y_4 = CA62C1D6$$
2. a. $X_0 \longleftarrow i$;
   b. **for** $j \longleftarrow 1$ **to** 15 **do** $X_j \longleftarrow 00000000$;
   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. $(M_0, M_1, M_2, M_3, M_4) \longleftarrow (H_0, H_1, H_2, H_3, H_4)$ **where** $a = H_0H_1H_2H_3H_4$;
   e. (Runda 1): **for** $j \longleftarrow 0$ **to** 19 **do**
$$t \longleftarrow ((M_0 <<< 5) + f(M_1, M_2, M_3) + M_4 + X_j + y_1);$$
$$(M_0, M_1, M_2, M_3, M_4) \longleftarrow (t, M_0, M_1 <<< 30, M_2, M_3);$$
   f. (Runda 2): **for** $j \longleftarrow 20$ **to** 39 **do**
$$t \longleftarrow ((M_0 <<< 5) + h(M_1, M_2, M_3) + M_4 + X_j + y_2);$$
$$(M_0, M_1, M_2, M_3, M_4) \longleftarrow (t, M_0, M_1 <<< 30, M_2, M_3);$$
   g. (Runda 3): **for** $j \longleftarrow 40$ **to** 59 **do**
$$t \longleftarrow ((M_0 <<< 5) + g(M_1, M_2, M_3) + M_4 + X_j + y_3);$$
$$(M_0, M_1, M_2, M_3, M_4) \longleftarrow (t, M_0, M_1 <<< 30, M_2, M_3);$$
   h. (Runda 4): **for** $j \longleftarrow 60$ **to** 79 **do**
$$t \longleftarrow ((M_0 <<< 5) + h(M_1, M_2, M_3) + M_4 + X_j + y_4);$$
$$(M_0, M_1, M_2, M_3, M_4) \longleftarrow (t, M_0, M_1 <<< 30, M_2, M_3);$$
   i. $(H_0, H_1, H_2, H_3, H_4) \longleftarrow (H_0 + M_0, H_1 + M_1, H_2 + M_2, H_3 + M_3, H_4 + M_4)$;
$$G_a(i) = H_0||H_1||H_2||H_3||H_4.$$

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

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

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 constanta $Y = 000007FC$ şi 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** $S[j] \longleftarrow F_a(00001000 + j)$;

   c. **for** $k \longleftarrow 0$ **to** $4 \cdot \lceil (L-1)/8192 \rceil - 1$ **do** $R[k] \longleftarrow F_a(00002000 + k)$;

2. Descrierea procedurii $Initialize(n, l, M_1, M_2, M_3, M_4, n_1, n_2, n_3, n_4)$ cu intrările n (cuvânt) şi l (întreg). Ieşirile sunt $M_1, M_2, M_3, M_4, n_1, n_2, n_3, n_4$ (cuvinte).

   a. $M_1 \longleftarrow n \oplus R[4 \cdot l]$, $M_2 \longleftarrow (n >> 8) \oplus R[4 \cdot l + 1]$,
   $M_3 \longleftarrow (n >> 16) \oplus R[4 \cdot l + 2]$, $M_4 \longleftarrow (n >> 24) \oplus R[4 \cdot l + 3]$.

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

   $P \longleftarrow M_1 \wedge Y$, $M_2 \longleftarrow M_2 + T[P/4]$, $M_1 \longleftarrow (M_1 >> 9)$,
   $P \longleftarrow M_2 \wedge Y$, $M_3 \longleftarrow M_3 + T[P/4]$, $M_2 \longleftarrow (M_2 >> 9)$,
   $P \longleftarrow M_3 \wedge Y$, $M_4 \longleftarrow M_4 + T[P/4]$, $M_3 \longleftarrow (M_3 >> 9)$,
   $P \longleftarrow M_4 \wedge Y$, $M_1 \longleftarrow M_1 + T[P/4]$, $M_4 \longleftarrow (M_4 >> 9)$,

   $(n_1, n_2, n_3, n_4) \longleftarrow (M_4, M_1, M_2, M_3)$;

   $P \longleftarrow M_1 \wedge Y$, $M_2 \longleftarrow M_2 + T[P/4]$, $M_1 \longleftarrow (M_1 >> 9)$,
   $P \longleftarrow M_2 \wedge Y$, $M_3 \longleftarrow M_3 + T[P/4]$, $M_2 \longleftarrow (M_2 >> 9)$,
   $P \longleftarrow M_3 \wedge Y$, $M_4 \longleftarrow M_4 + T[P/4]$, $M_3 \longleftarrow (M_3 >> 9)$,
   $P \longleftarrow M_4 \wedge Y$, $M_1 \longleftarrow M_1 + T[P/4]$, $M_4 \longleftarrow (M_4 >> 9)$,

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

4. **repeat**

   a. $Initialize(n, l, M_1, M_2, M_3, M_4, n_1, n_2, n_3, n_4)$;

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

   $P \longleftarrow M_1 \wedge Y$, $M_2 \longleftarrow M_2 + T[P/4]$, $M_1 \longleftarrow (M_1 >> 9)$, $M_2 \longleftarrow M_2 \oplus M_1$;
   $Q \longleftarrow M_2 \wedge Y$, $M_3 \longleftarrow M_3 + T[Q/4]$, $M_2 \longleftarrow (M_2 >> 9)$, $M_3 \longleftarrow M_3 \oplus M_2$;

$P \longleftarrow (P + M_3) \wedge Y,\ M_4 \longleftarrow M_4 + T[P/4],\ M_3 \longleftarrow (M_3 >> 9),\ M_4 \longleftarrow M_4 \oplus M_3;$
$Q \longleftarrow (Q + M_4) \wedge Y,\ M_1 \longleftarrow M_1 + T[Q/4],\ M_4 \longleftarrow (M_4 >> 9),\ M_1 \longleftarrow M_1 \oplus M_4;$
$P \longleftarrow (P + M_1) \wedge Y,\ M_2 \longleftarrow M_2 + T[P/4],\ M_1 \longleftarrow (M_1 >> 9);$
$Q \longleftarrow (Q + M_2) \wedge Y,\ M_3 \longleftarrow M_3 + T[Q/4],\ M_2 \longleftarrow (M_2 >> 9);$
$P \longleftarrow (P + M_3) \wedge Y,\ M_4 \longleftarrow M_4 + T[P/4],\ M_3 \longleftarrow (M_3 >> 9);$
$Q \longleftarrow (Q + M_4) \wedge Y,\ M_1 \longleftarrow M_1 + T[Q/4],\ M_4 \longleftarrow (M_4 >> 9);$
$y \longleftarrow y||(M_2 + S[4 \cdot i - 4])||(M_3 \oplus S[4 \cdot i - 3])||(M_4 + S[4 \cdot i - 2])||(M_1 \oplus S[\cdot i - 1]).$

if $|y| \geq L$ **then return**(y) **STOP**
else if $i\ (mod\ 2) = 1$ **then** $(M_1, M_3) \longleftarrow (M_1 + n_1, M_3 + n_2)$
else $(M_1, M_3) \longleftarrow (M_1 + n_3, M_3 + n_4)$
c. $l \longleftarrow l + 1$.

Observaţia 4.2 *([29]) 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 $S - box$.

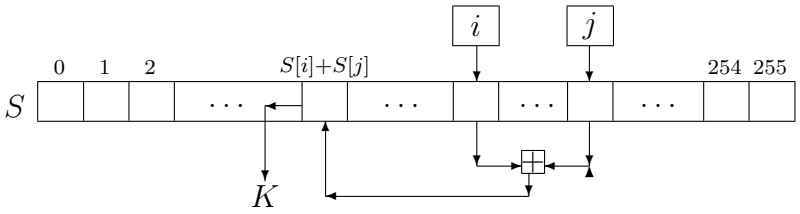
[4]Când un sistem de criptare este marcat *opţional*, însemnă că el este una din variantele oferite pentru implementare.

- 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).

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 la 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])
```

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

Implementarea sistemului $RC4$ este foarte simplă: se lucrează cu octeţi şi sunt necesari 256 octeţi 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).

Securitatea $RC4$

Conform cu Bruce Schneier ([37]), 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 necesară 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 soluţie 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 slăbiciune este exploatată de Fluhrer, Mantin şi Shamir pentru a sparge ulterior sistemul de criptare WEP (wired equivalent privacy) 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.

4.2.3 Sistemul $A5/1$

$A5/1$ este un sistem de criptare fluid utilizat pe scară largă în reţelele de telefonie mobilă GSM[7]. Construit în 1987 special pentru Europa şi SUA (o versiune intenţionat mai slabă – $A5/2$ – a fost definită în 1989 pentru export în alte zone geografice), el a fost făcut public după ce în 2000 s-a demonstrat vulnerabilitatea sa ([4]).

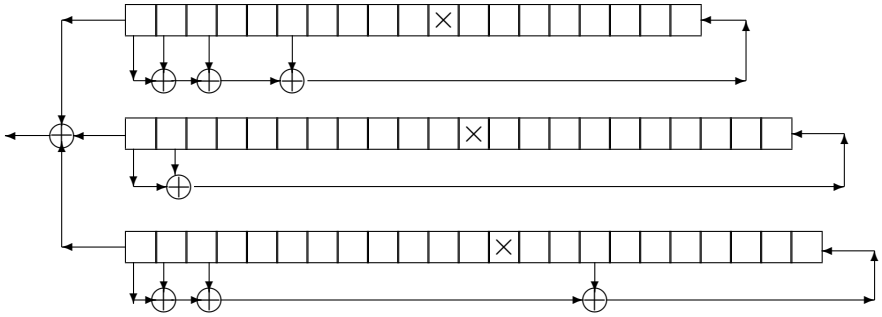
$A5/1$ este un automat finit bazat pe trei regiştri seriali cu feedback ($LFSR$) sincroni. Cei trei regiştri R_1, R_2, R_3 conţin 19, 22 şi respectiv 23 flip - flopuri de date (unităţi de memorie de un bit). La un tact, fiecare registru poate rămâne pe loc sau îşi poate deplasa conţinutul spre stânga, aducând pe ultima poziţie o combinaţie liniară de alţi biţi. Cele

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

[7] Înafara algoritmului de criptare $A5/1$, reţelele GSM folosesc şi un algoritm de autentificare $A3/8$.

trei combinaţii liniare sunt definite astfel:

$$\begin{aligned} R_1[0] &\longleftarrow R_1[13] \oplus R_1[16] \oplus R_1[17] \oplus R_1[18] \\ R_2[0] &\longleftarrow R_2[20] \oplus R_2[21] \\ R_3[0] &\longleftarrow R_3[17] \oplus R_3[20] \oplus R_3[21] \oplus R_3[22] \end{aligned}$$



Automatul $A5/1$

Pentru a decide ce regiştri se deplasează la un tact, se folosesc trei biţi *de tact* (cei notati $\times$). La fiecare tact se determina valoarea majoritară din acesti biţi; un registru se deplasează (şi scoate o valoare) dacă bitul său de tact are aceeaşi valoare cu cea majoritară. Deci, la fiecare pas, cel puţin doi regiştri vor suferi o deplasare.

La fiecare tact, valoarea emisă de automat este

$$k = R_1[18] \oplus R_2[21] \oplus R_3[22]$$

Apoi y se combină prin XOR cu bitul curent din mesajul de informaţie (similar sistemului *one-time-pad*).

Exemplul 4.4 *Să presupunem că cei trei regiştri sunt (biţii de tact au fost marcaţi):*

R_1 = 0110 1011 10**1**0 1001 011,
R_2 = 1010 1100 001**0** 1101 1001 11,
R_3 = 1100 0101 1011 **0**010 1010 010

Să urmărim comportarea automatului câţiva tacţi:

1. **Tact 1***: Ieşire:* $y \longleftarrow 0 \oplus 1 \oplus 1 = 0$.

   Bit majoritar: 0. *Se deplasează* R_2 *şi* R_3. *Noua stare este:*

   R_1 = 0110 1011 10**1**0 1001 011,
   R_2 = 0101 1000 010**1** 1011 0011 11,
   R_3 = 1000 1011 0110 **0**101 0100 100

2. **Tact 2***: Ieşire:* $y \longleftarrow 0 \oplus 0 \oplus 1 = 1$.

   Bit majoritar: 1. *Se deplasează* R_1 *şi* R_2. *Noua stare este:*

   R_1 = 1101 0111 01**0**1 0010 110,
   R_2 = 1011 0000 101**1** 0110 0111 11,
   R_3 = 1000 1011 0110 **0**101 0100 100

3. **Tact 3***: Ieşire:* $y \longleftarrow 1 \oplus 1 \oplus 1 = 1$.

Bit majoritar: 0. *Se deplasează* R_1 *şi* R_3. *Noua stare este:*

$R_1 = 1010\ 1110\ 10\mathbf{1}0\ 0101\ 101,$
$R_2 = 1011\ 0000\ 101\mathbf{1}\ 0110\ 0111\ 11,$
$R_3 = 0001\ 0110\ 1100\ \mathbf{1}010\ 1001\ 000$

4. **Tact 4***: Ieşire:* $y \longleftarrow 1 \oplus 1 \oplus 0 = 0$.

Bit majoritar: 1. *Se deplasează toţi regiştrii. Noua stare este:*

$R_1 = 0101\ 1101\ 01\mathbf{0}0\ 1011\ 011,$
$R_2 = 0110\ 0001\ 011\mathbf{0}\ 1100\ 1111\ 11,$
$R_3 = 0010\ 1101\ 1001\ \mathbf{0}101\ 0010\ 000$

Deci cheia fluidă de criptare este $z = 01100\ldots$

$A5/1$ necesită deci şi o iniţializare a stării interne, bazată pe o cheie de criptare K de 64 biţi şi anumiţi parametrii GSM stocaţi într-o constantă $Count$ de 22 biţi. Deoarece sistemul GSM transmite mesajele în pachete de câte 114 biţi, $A5/1$ va genera chei fluide de 114 biţi; deci este necesară o iniţializare a automatului la fiecare pachet nou primit. Algoritmul de iniţializare este:

1. Se resetează toţi regiştrii.
2. **for** $i \longleftarrow 0$ **to** 63 **do**
   - 2.1. $R_1[0] \longleftarrow R_1[0] \oplus K[i]$;
   - 2.2. $R_2[0] \longleftarrow R_2[0] \oplus K[i]$;
   - 2.3. $R_3[0] \longleftarrow R_3[0] \oplus K[i]$;
   - 2.4. Deplasare stânga toţi regiştrii;
3. **for** $i \longleftarrow 0$ **to** 21 **do**
   - 3.1. $R_1[0] \longleftarrow R_1[0] \oplus Count[i]$;
   - 3.2. $R_2[0] \longleftarrow R_2[0] \oplus Count[i]$;
   - 3.3. $R_3[0] \longleftarrow R_3[0] \oplus Count[i]$;
   - 3.4. Deplasare stânga toţi regiştrii;
4. **for** $i \longleftarrow 0$ **to** 99 **do**

   Funcţionează un tact automatul $A5/1$, cu ignorarea ieşirii.

4.3 Exerciţii

4.1 *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* Σ *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 $|\Sigma|$.

4.2 *Criptaţi mesajul* 10110101 *folosind automatul* $A5/1$ *din Exemplul 4.4 (considerând starea iniţială cea dinaintea Tactului* 1*).*

4.3 *Construim următorul sistem de criptare cu cheie fluidă:*

Fie $\pi \in S_{26}$ *o permutare fixată; cheia este* $K \in Z_{26}$.

Secvenţa de chei fluide este definită $z_i = (K + i - 1) \bmod 26 \quad (i \geq 1)$

Funcţia de criptare este $e_z(x) = \pi((x + z) \bmod 26)$

iar cea de decriptare $d_z(y) = (\pi^{-1}(y) - z) \bmod 26$.

Să presupunem că permutarea fixată este

$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\ 23 & 13 & 24 & 0 & 7 & 15 & 14 & 6 & 25 & 16 & 22 & 1 & 19 & 18 & 5 & 11 & 17 & 2 & 21 & 12 & 20 & 4 & 10 & 9 & 3 & 8 \end{pmatrix}$$

Folosind un atac prin forţă brută, să se decripteze mesajul $BSTKXOUXMWJK$

Bibliografie

[1] Anderson R. ş.a. - *Serpent: A proposal for the Advanced Encryption Standard*, http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/serpent.pdf

[2] Atanasiu A. - *Teoria codurilor corectoare de erori*, Editura Univ. Bucureşti, 2001;

[3] D. Bayer, S. Haber, W. Stornetta; Improving the efficiency and reliability of digital time-stamping. Sequences II, Methods in Communication, Security and Computer Science, Springer Verlag (1993), 329-334.

[4] A. Biryukov, A. Shamir, D. Wagner, *Real Time Cryptanalysis of* $A5/1$ *on a PC*, Fast Software Encryption - FSE 2000, pp 118.

[5] A. Bruen, M. Forcinito, *Cryptography, Information Theory, and Error - Correction*, Wiley Interscience 2005.

[6] Bos J.N., Chaum D. - Provably unforgable signatures; Lecture Notes in Computer Science, $740(1993), 1 - 14$

[7] D. Chaum, H. van Antwerpen - Undeniable signatures; Lecture Notes in Computer Science, $435(1990), 212 - 216$

[8] D. Chaum, E. van Heijst, B. Pfitzmann; Cryptographically strong undeniable signatures, unconditionally secure for the signer. Lecture Notes in Computer Science, 576 (1992), 470-484.

[9] Brigitte Collard - *Secret Language in Graeco-Roman antiquity* (teză de doctorat) $http://bcs.fltr.ucl.ac.be/FE/07/CRYPT/Intro.html$

[10] Cook S., $http://www.claymath.org/millennium/P\_vs\_NP/Official\_Problem\_Description.pdf$

[11] Coppersmith D. ş.a. - *MARS - a candidate cypher for AES*, http://www.research.ibm.com/security/mars.pdf

[12] Daemen J., Rijmen V. - *The Rijndael Block Cipher Proposal*, http://csrc.nist.gov/CryptoToolkit/aes/

[13] I.B. Damgard; A design principle for hash functions. Lecture Notes in Computer Science, 435 (1990), 516-427.

[14] Diffie D.W., Hellman M.E. - *New Directions in Cryptography*, IEEE Transactions on Information Theory, IT-22, 6 (1976), pp. 644-654

[15] W. Diffie, M.E. Hellman - Multiuser cryptographic techniques; AFIPS Conference Proceedings, $45(1976), 109 - 112$

[16] L´ Ecuyer P. - *Random Numbers for Simulation*, Comm ACM 33, 10(1990), 742-749, 774.

[17] Enge A. - *Elliptic Curves and their applications to Cryptography*, Kluwer Academic Publ, 1999

[18] El Gamal T., *A public key cryptosystem and a signature scheme based on discrete algorithms*, IEEE Transactions on Information Theory, 31 (1985), 469-472

[19] Fog A. - *http://www.agner.org/random/theory*;

[20] Gibson J., *Discrete logarithm hash function that is collision free and one way.* IEEE Proceedings-E, 138 (1991), 407-410.

[21] S. Haber, W. Stornetta; How to timestamp a digital document. Journal of Cryptology, 3(1991), 99-111.

[22] van Heyst E., Petersen T.P. - How to make efficient fail-stop signatures; Lecture Notes in Computer Science, $658(1993), 366 - 377$

[23] Kahn D. - *The Codebreakers*, MacMillan Publishing Co, New York, 1967

[24] Kelly T. - *The myth of the skytale*, Cryptologia, Iulie 1998, pp. 244 - 260.

[25] A. Konheim - *Computer Security and Cryptography*, Wiley Interscience, 2007.

[26] Knuth D. - *The art of computer Programming*, vol 2 (Seminumerical Algorithms)

[27] R.C. Merkle; A fast software one-way functions and DES. Lecture Notes in Computer Science, 435 (1990), 428-446

[28] Mitchell C.J., Piper F., Wild, P. - Digital signatures; Contemporary Cryptology, The Science of Information Integrity, IEEE Press, $(1992), 325 - 378$

[29] Menezes A., Oorschot P., Vanstome S., *Handbook pf Applied Cryptography*

[30] B. Preneel, R. Govaerts, J. Vandewalle; Hash functions based on block ciphers: a syntetic approach. Lecture Notes in Computer Science, 773 (1994), 368-378

[31] Rivest R. ş.a - *The RC6^{TM} Block Cipher*,
ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf

[32] R.L. Rivest; The **MD4** message digest algorithm. Lecture Notes in Computer Science, 537, (1991), 303-311

[33] Rivest R., Shamir A., Adleman A., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21 (2), 1978, pages 120–126.

[34] Rosing, M, *Implementing Elliptic Curve Cryptography*, Manning, 1998

[35] D. Salmon, *Data Privacy and Security*, Springer Professional Computing, 2003

[36] Salomaa A., *Criptografie cu chei publice*, Ed. Militară, Bucureşti 1994

[37] Schneier B., *Applied Cryptography*, John Wiley and Sons, 1995

[38] Schneier B ş.a. - *Twofish*, http://www.counterpane.com/twofish.html

[39] Selmer E.S. - *Linear Recurrence over Finite Field*, Univ. of Bergen, Norway, 1966;

[40] Sibley E.H. - *Random Number Generators: Good Ones are Hard to Find*, Comm ACM 31, 10(1988), 1192-1201.

[41] Smid M.E., Branstad, D.K. - Response to comments on the $NIST$ proposed digital signature standard; Lecture Notes in Computer Science, $740(1993), 76 - 88$

[42] Stinton D., *Cryptography, Theory and Practice*, Chapman& Hall/CRC, 2002

[43] Wiener M.J. - *Cryptanalysis of short RSA secret exponents*, IEEE Trans on Information Theory, 36 (1990), 553-558

[44] Williams H.C. - *Some public-key criptofunctions as intractable as factorisation*, Cryptologia, 9 (1985), 224-237.

[45] Zeng K.G., Yang C.H., Wei D.Y., Rao T.R.N.- *Pseudorandom Bit Generators in Stream Cipher Cryptography*, IEEE Computer, 24 (1991), 8.17.

[46] Secure hash Standard. National Bureau of Standards, FIPS Publications 180, 1993

[47] Digital signature standard; National Bureau of Standards, FIPS Publications 186, 1994

[48] $http://en.wikipedia.org/wiki/Enigma\_machine$

[49] $http://en.wikipedia.org/wiki/M-209$

[50] *http://en.wikipedia.org/wiki/Caesar_ cipher# History_ and_ usage*

[51] *http://psychcentral.com/psypsych/Polybius_ square*

[52] *http://www.answers.com/topic/vigen-re-cipher*

[53] *http://en.wikipedia.org/wiki/Rosetta_ stone*

[54] *Serpent homepage*, http://www.cl.cam.ac.uk/ rja14/serpent.html

[55] *P versus NP homepage*, http://www.win.tue.nl/ gwoegi/P-versus-NP.htm

[56] http://www.win.tue.nl/ gwoegi/P-versus-NP.htm

[57] http://en.wikipedia.org/wiki/Complexity_ classes_ P_ and_ NP