# Capitolul 11

# Alte sisteme de criptare cu cheie publică

## 11.1 Sistemul de criptare Merkle - Hellman

Sistemul de criptare Merkle - Hellman sau "sistemul de criptare rucsac" a fost prezentat în linii mari în Capitolul 7, ca exemplificare pentru ideea de criptare cu cheie publică (apărut în 1978 în [36], a fost oficial primul astfel de sistem de criptare). În această secţiune vom face o analiză a sistemului, însoţită de un atac cu succes asupra variantei iniţiale.

### 11.1.1 Considerente generale

Numim *vector rucsac* un tablou unidimensional $A = (a_1, a_2, \ldots, a_n)$ $(n \geq 3)$ cu elemente numere întregi pozitive distincte.

O *apariţie* a problemei rucsacului este o pereche $(A, x)$ unde $A$ este un vector rucsac, iar $x$ - un număr întreg pozitiv.

O soluţie a apariţiei problemei $(A, x)$ este un vector $B = (a_{i_1}, \ldots, a_{i_k})$ de elemente din $A$ astfel ca $x = \sum_{j=1}^{k} a_{i_j}$.

Cea mai obişnuită abordare a problemei rucsacului este de a decide dacă o anumită apariţie $(A, x)$ a problemei rucsacului are soluţie sau nu. Varianta utilizată în criptografie constă în obţinerea acestei soluţii pentru $(A, x)$, ştiind că ea există. Ambele probleme sunt $\mathcal{NP}$ - complete.

După cum am văzut în Capitolul 7, un vector rucsac este folosit pentru criptarea unui bloc $C$ de $n$ biţi realizând suma acelor elemente din $A$ a căror poziţie corespunde lui 1 în $C$. Matematic, privind $C$ ca un vector coloană, criptarea este realizată de produsul scalar

$$x = A \cdot C$$

Decriptarea revine la a determina $C$ ştiind $x$ sau – pentru varianta cu cheie publică a sistemului de criptare rucsac – de a afla $C$ din $A$ şi $x$.

**Exemplul 11.1.** *Fie $n = 6$ şi vectorul rucsac $A = (3, 41, 5, 1, 21, 10)$. Atunci textul clar $C_1 = (1, 1, 0, 0, 1, 0)$ este criptat în $x_1 = 65$, iar $C_2 = (1, 0, 1, 1, 0, 1)$ în $x_2 = 19$. Pentru vectorul $A$ astfel definit, mulţimea textelor criptate este inclusă în intervalul $[0, 81]$.*

Pentru a putea fi utilizat în criptare, un vector rucsac $A$ trebuie să fie *injectiv*: pentru orice $x \in \mathcal{N}$, apariţia problemei rucsacului $(A, x)$ are cel mult o soluţie.

**Exemplul 11.2.** *Pentru vectorul rucsac $A = (14, 28, 56, 82, 90, 132, 197, 284, 341, 455)$ textul criptat $x = 515$ se poate obţine din trei texte clare distincte:*

$$(1, 1, 0, 0, 0, 1, 0, 0, 1, 0), \quad (0, 1, 1, 0, 1, 0, 0, 0, 1, 0), \quad (1, 0, 0, 1, 1, 1, 1, 0, 0, 0).$$

Sunt anumiţi vectori $A$ pentru care toate aparţiile $(A, x)$ sunt uşor de rezolvat. Am văzut că vectorii cu creştere mare au această proprietate.

**Definiţia 11.1.** *Un vector rucsac $A = (a_1, a_2, \ldots, a_n)$ este crescător (super-crescător) dacă*

$$\forall j \geq 2, \quad a_j > a_{j-1} \quad (\textit{respectiv } a_j > \sum_{i=1}^{j-1} a_i).$$

Evident, orice vector super-crescător este crescător.

Pentru un vector rucsac $A = (a_1, \ldots, a_n)$ definim $max(A) = max\{a_j \mid 1 \leq j \leq n\}$.

Fie $A$ un vector rucsac, $m > max(A)$ şi $t \in (0, m)$ astfel ca $cmmdc(t, m) = 1$. Dacă $B = (b_1, b_2, \ldots, b_n)$ este un vector definit prin

$$b_i \equiv t \cdot a_i \ (mod\ m) \quad (1 \leq i \leq n)$$

spunem că $B$ *rezultă din $A$ prin înmulţire modulară în raport cu perechea* $(m, t)$.

După cum ştim, condiţia $cmmdc(t, m) = 1$ asigură existenţa unui element $u \in (1, m)$ (obţinut prin algoritmul lui Euclid extins), astfel ca $t \cdot u \equiv 1 \ (mod\ m)$.

Deci va avea loc şi proprietatea reciprocă: vectorul rucsac $A$ rezultă din $B$ prin înmulţire modulară în raport cu $(m, u)$[1].

Dacă relaţia $m > max(A)$ este înlocuită cu condiţia $m > \sum_{i=1}^{n} a_i$, spunem că $B$ *rezultă din $A$ prin înmulţire modulară tare în raport cu* $(m, t)$. În acest caz nu va mai rezulta valabilitatea proprietăţii inverse, deoarece inegalitatea $m > \sum_{i=1}^{n} b_i$ nu mai este adevărată totdeauna. Se poate spune însă, evident, că $A$ rezultă din $B$ prin înmulţire modulară în raport cu $(m, u)$.

[1]Evident, $m > max(B)$, pentru că $b_i < m$ pentru orice $i = 1, \ldots, n$.

Construcţia sistemului de criptare rucsac este imediată:

1. *Bob* alege numerele $m, t$ prime între ele, precum şi vectorul super-crescător $A = (a_1, \ldots, a_n)$, astfel ca $m > \sum_{i=1}^{n} a_i$.
2. Determină vectorul rucsac $B$ obţinut din $A$ prin inmulţire modulară tare în raport cu $(m, t)$.
3. Face public $B$ şi păstrează drept cheie secretă elementele $A, m, t$ şi $u \equiv t^{-1} \ (mod\ m)$.

Dacă *Alice* doreşte să cripteze un mesaj, va proceda în felul următor:

1. Sparge mesajul în blocuri $B_x \in Z_2^n$ ($n$ – lungimea vectorului $B$), unde $x \in [0, 2^n - 1)$ are $B_x$ drept reprezentare binară.
2. Calculează $y = B \cdot B_x^T$ şi îl trimite lui *Bob*.

În acest fel, un criptanalist va trebui să rezolve apariţia $(B, y)$ a problemei rucsacului. *Bob* va determina întâi $x \equiv u \cdot y \ (mod\ m)$, după care va rezolva apariţia $(A, x)$.

Această simplificare este asigurată de lema:

**Lema 11.1.** *Presupunem că $A = (a_1, a_2, \ldots, a_n)$ este un vector super-crescător, iar $B$ rezultă din $A$ prin înmulţire modulară tare în raport cu $(m, t)$. Fie $u \equiv t^{-1} \ (mod\ m)$, $y$ un întreg arbitrar şi $x \equiv u \cdot y \ (mod\ m)$. Atunci aserţiunile următoare sunt adevărate:*

1. *Problema rucsacului $(A, x)$ este rezolvabilă în timp liniar. Dacă soluţia există, ea este unică.*
2. *Problema rucsacului $(B, y)$ are cel mult o soluţie.*
3. *Dacă există o soluţie pentru $(B, y)$, atunci ea este egală cu soluţia apariţiei $(A, x)$.*

*Demonstraţie.* Am arătat în Capitolul 7 că orice apariţie a problemei rucsacului având $A$ super-crescător poate fi rezolvată printr-un algoritm liniar (Exemplul 7.4), parcurgând $A$ de la dreapta spre stânga. Metoda arată de asemenea şi faptul că există cel mult o soluţie.

Pentru celelalte două serţiuni, să presupunem că vectorul $D \in Z_2^n$ este soluţie a apariţiei problemei rucsacului $(B, y)$; deci $y = B \cdot D^T$. Atunci

$$x \equiv u \cdot y = u \cdot B \cdot D^T = u \cdot (t \cdot A) \cdot D^T \equiv A \cdot D^T \ (mod\ m)$$

Pentru că $m$ depăşeşte suma componentelor lui $A$, trebuie să avem $A \cdot D^T < m$. Cum de asemenea $x < m$ (din definiţia lui $x$), deducem $x = A \cdot D^T$. Deci vectorul $D$ este unica soluţie a apariţiei problemei $(A, x)$ a rucsacului.

Deoarece s-a folosit o soluţie arbitrară a lui $(B, y)$, va rezulta şi aserţiunea (3). □

**Exemplul 11.3.** *Fie $n = 10$ şi vectorul super-crescător*

$$A = (103, 107, 211, 430, 863, 1718, 3449, 6907, 13807, 27610)$$

*Alegem modulul $m = 55207$ (care este mai mare decât suma componentelor lui $A$) şi $t = 25236$. Deoarece $cmmdc(t, m) = 1$, algoritmul lui Euclid extins va calcula $u = t^{-1} = 1061$.*

*Ca rezultat al înmulţirii modulare tari în raport cu $(m, t)$ se obţine vectorul rucsac*

$$B = (4579, 50316, 24924, 30908, 27110, 17953, 32732, 16553, 22075, 53620)$$

*Acest vector $B$ este cheia publică de criptare, în timp ce elementele $A, t, m, u$ formează trapa secretă.*

*Folosind cheia publică $B$, să criptăm textul clar PELIN DE MAI.*

*În prima fază acesta se împarte în perechi de câte două litere, care se codifică în secvenţe binare de 10 biţi (folosind codificarea propusă în Capitolul 7). Obţinem*

| | | |
|---|---|---|
| $PE$ | 10000 00101 | 74.752 |
| $LI$ | 01100 01001 | 161.592 |
| $N\_$ | 01110 00000 | 106.148 |
| $DE$ | 00100 00101 | 95.097 |
| $MA$ | 01101 00001 | 155.970 |
| $I\_$ | 01001 00000 | 77.426 |

*Pe ultima coloană sunt scrise criptările blocurilor de câte două litere (aflate pe prima coloană).*

*Să decriptăm primul bloc:* 74.752 *(cu celelalte se procedează analog). Avem*

$$74752 \cdot 1061 = 79311872 = 1436 \cdot 55207 + 34620$$

*Considerăm apariţia problemei rucsacului $(A, 34.620)$. Soluţia se obţine parcurgând vectorul $A$ de la dreapta spre stânga:*

| *Număr* | *Componenta lui A* | *Bit* |
|---|---|---|
| 35.620 | 27.610 | 1 |
| 7.010 | 13.807 | 0 |
| 7.010 | 6.907 | 1 |
| 103 | 3.449 | 0 |
| 103 | 1.718 | 0 |
| 103 | 863 | 0 |
| 103 | 430 | 0 |
| 103 | 211 | 0 |
| 103 | 107 | 0 |
| 103 | 103 | 1 |

*Citind ultima coloană din tabel, de jos în sus, obţinem* 10000 00101*, care este codificarea binară pentru perechea de litere* $PE$*.*

*Să încercăm să procedăm invers: să criptăm textul clar* $PE$ *folosind vectorul* $A$*. Se obţine, evident,* 34.620*. Aplicăm înmulţirea modulară tare în raport cu* $(55.207, 25.236)$ *şi avem*

$$34620 \cdot 25236 = 873670320 = 15825 \cdot 55207 + 19545$$

*Dar apariţia* $(B, 19.545)$ *nu are soluţie, lucru evident pentru că, în* $B$*, singurele numere mai mici decât* 19.545 *sunt* $4.579, 19.953$ *şi* $16.553$*, iar* 19.545 *nu se poate obţine din nici o combinaţie a lor.*

**Exemplul 11.4.** *Datele din Exemplul 11.3 sunt simple, putând fi prelucrate şi cu un calculator de buzunar. Criptările reale folosesc numere mult mai mari. Tot în exemple se foloseşte frecvent şi varianta* $n = 20$*,* $m = 53939986$*,* $t = 54377$ *(cu inversa* $t^{-1} = u = 17521047$*). Un exemplu de vector super-crescător cu* 20 *componente este*
$A = (101, 102, 206, 412, 823, 1.647, 3.292, 6.584, 13.169, 26.337, 52.676, 105.352, 210.703,$
$421.407, 842.812, 1.685.624, 3.371.249, 6.742.497, 13.484.996, 26.969.992)$

### 11.1.2 Criptanaliza sistemului de criptare rucsac

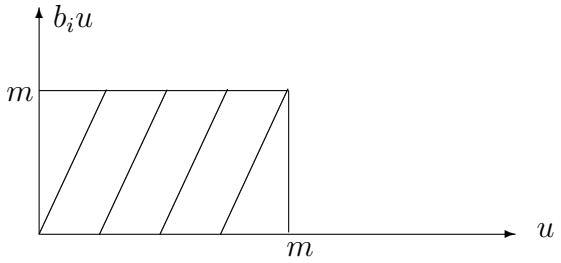
Ne aflăm în faţa următoarei probleme de criptanaliză: se ştie un vector rucsac $B = (b_1, b_2, \ldots, b_n)$ folosit drept cheie publică de criptare în maniera descrisă anterior. Se ştie de asemenea că $B$ este obţinut dintr-un vector super-crescător $A$ printr-o înmulţire modulară tare în raport cu un modul $m$ şi un înmulţitor $t$. Nu cunoaştem $A, m, t$; vrem să le aflăm. Dacă determinăm $m$ şi $u = t^{-1}$, putem găsi imediat vectorul super-crescător $A$ şi decripta mesajele interceptate. Calculul lui $u$ plecând de la $t$ (sau invers) se bazează pe Algoritmul extins al lui Euclid şi nu prezintă nici o dificultate.

În această secţiune vom dezvolta metoda de criptanaliză a lui Adi Shamir ([48]).

Algoritmul operează în timp polinomial, gradul polinomului de complexitate fiind determinat de numărul şi mărimea componentelor vectorului $B$. Shamir face o analiză detaliată asupra stabilirii acestui grad, analiză care poate fi studiată în [48].

Atacul are ca prim scop găsirea valorilor $m$ şi $u$. O primă observaţie: nu este necesară aflarea exact a valorilor pe care le-a definit *Bob* pentru sistemul de criptare. Orice pereche $(m, u)$ care, plecând de la vectorul $B$ duce la un vector super-crescător $A$, poate fi utilizată la decriptare. Astfel de perechi se numesc "*perechi trapă*". Odată găsită o astfel de pereche trapă, Lema 11.1 devine valabilă şi se poate începe decriptarea. Existenţa cel puţin a unei perechi trapă este asigurată de faptul că sistemul de criptare rucsac se bazează pe o astfel de construcţie.

În aflarea unei perechi trapă, vom considera graful funcţiei $f_i(u) = b_i u \ (mod\ m)$ pentru $i = 1, 2, \ldots, n$. Acesta este format din segmente de linii paralele, "rupte" în punctele $u = p \cdot m/b_i, \ \ p = 1, 2, \ldots$




În particular, $a_1 \equiv b_1 u \ (mod\ m)$. Pentru că $a_1$ este prima componentă într-un vector super-crescător, iar $m$ depăşeşte suma tuturor componentelor, elementul $a_1$ trebuie să fie foarte mic în comparaţie cu $m$. Deci valoarea lui $u$ trebuie să fie suficient de aproape de un minim al grafului $y = f_1(u)$, corespunzător lui $b_1$. O exprimare explicită – cât de aproape trebuie să fie de acest minim – trebuie să ţină cont de anumite considerente relative la mărimile $a_1, b_1$ şi $m$. De obicei raportul $b_i/a_i$ este foarte mare pentru valori mici ale lui $i$ (bineînţeles, *Bob* poate ţine seama de acest lucru şi să construiască sistemul în aşa fel încât această remarcă să nu fie valabilă; atunci însă vor apare alte particularităţi de care un bun criptanalist se poate folosi).

Similar, observăm că valoarea lui $u$ din perechea trapă $(m, u)$ trebuie să fie destul de apropiată de un minim al grafului lui $f_2(u)$. Aceasta duce, pe baza inegalităţii triunghiului, la concluzia că două minime ale lui $f_1(u)$ şi $f_2(u)$ trebuie să fie apropiate. Putem proceda în acest fel şi pentru alte valori. Faptul că valoarea lui $u$ este apropiată de un minim al fiecărei curbe $f_i(u)$ implică faptul că aceste minime sunt apropiate unul de altul.

Deci, în loc să aflăm pe $u$, vom căuta puncte de acumulare ale minimelor curbelor $f_i(u)$. Ele vor duce la determinarea unor intervale de valori reale, în care se află astfel de minime. În final se va alege un $u$ dintr-un astfel de interval.

Prin argumente şi calcule euristice, Shamir arată că în general sunt suficiente patru astfel de curbe pentru aflarea unei mulţimi rezonabil de mică de astfel de intervale (sau – echivalent – puncte de acumulare).

Să "traducem" aceste idei sub o formă matematică.

Primul obstacol: nu ştim nici o valoare a lui $m$ (care apare în perechea trapă). Vom considera temporar valoarea $m = 1$. Această particularizare este de fapt o normalizare a graficelor şi nu afectează localizarea punctelor de acumulare care ne interesează.

Algoritmul constă din două părţi: în prima etapă vom afla o mulţime de numere întregi $p$ cu proprietatea că al $p$-lea punct de minim al curbei $f_1(u)$ este punct de acumulare. Pentru a evita generarea unui număr prea mare de valori $p$, se fixează o limită: un parametru $r$ care să indice numărul maxim de valori posibile permise. Dacă prima parte a algoritmului generează mai mult de $r$ valori, el se va termina cu eşec.

În etapa următoare vom explora pe rând punctele de acumulare găsite anterior. Unul din teste va reuşi, deoarece valoarea lui $u$ folosită de *Bob* în sistemul de criptare desemnează un astfel de punct de acumulare.

I. Coordonata $u$ a celui de-al $p$-lea punct de minim al curbei $f_1(u)$ este $p/b_1$ (reamintim, momentan $m = 1$). Deci condiţia ca minimele curbelor $f_1(u)$ şi $f_2(u)$ să fie apropiate

este

$$-e < \frac{p}{b_1} - \frac{q}{b_2} < e, \qquad 1 \le p \le b_1 - 1, \quad 1 \le q \le b_2 - 1$$

unde $e$ este o valoare rezonabil de mică. Înmulţind această relaţie cu $b_1 b_2$ se obţine

$$-\delta < b_2 p - b_1 q < \delta, \qquad 1 \le p \le b_1 - 1, \quad 1 \le q \le b_2 - 1.$$

Considerăm $s$ astfel de puncte de minim şi scriem $s-1$ inegalităţi de acest tip, pentru $b_1, b_2, \ldots, b_s$. Relativ la o valoare estimativă a lui $\delta$, Shamir arată că dacă se alege $\delta < \sqrt{b_1/2}$, atunci probabilitatea ca algoritmul să eşueze este mai mică de $(2/r)^{s-1}$.

Prima parte a algoritmului rezolvă acest sistem de inecuaţii, aflând toate valorile lui $p$ pentru care există $q, \ldots$ astfel încât să fie satisfăcute cele $s-1$ inecuaţii.

II. Fie $p$ un punct de minim fixat arbitrar. Toate punctele de discontinuitate ale curbelor $f_i(u)$ $(1 \le i \le n)$ aflate în intervalul $\left[\frac{p}{b_1}, \frac{p+1}{b_1}\right]$ sunt ordonate crescător. Considerăm $x_j, x_{j+1}$ două puncte consecutive de discontinuitate. Atunci în intervalul $[x_j, x_{j+1}]$ fiecare curbă $f_i(u)$ este un segment reprezentat prin $f_i(u) = b_i u - c_i^j$ , unde $c_i^j$ este o constantă care depinde de valorile lui $i, j$ şi $p$.

Construim sistemul de inecuaţii liniare

$$\begin{cases} x_j \le u \le x_{j+1} \\ \sum_{i=1}^{n} (b_i u - c_i^j) < 1 \\ (b_1 u - c_1^j) + \ldots + (b_{i-1} u - c_{i-1}^j) < b_i u - c_i^j, \qquad (2 \le i \le n) \end{cases}$$

Soluţia acestui sistem este un subinterval (posibil vid) al lui $[x_j, x_{j+1}]$.

O condiţie necesară şi suficientă ca două numere $u, m$ să formeze o pereche trapă este să existe un $p$ şi un $j$ astfel ca $u/m$ să aparţină unui astfel de subinterval. Într-adevăr, în sistemul de sus, a doua inecuaţie asigură un $m$ mai mare decât suma componentelor, iar ultimele $n-1$ inecuaţii reprezintă condiţia de super-creştere.

Deci etapa a doua face o căutare exhaustivă printre perechile $(p, j)$, unde $p$ este dat de prima etapă a algoritmului, iar $j$ este un index al listei ordonate de puncte de discontinuitate din intervalul dat de $p$. Căutarea se face până la aflarea unui interval nevid (soluţie care există totdeauna). La un astfel de interval corespunde cel puţin o pereche trapă.

**Exemplul 11.5.** *Să presupunem că vectorul public este* $B = (7, 3, 2)$ *(exemplul este extrem de simplu, dar va permite detalierea algoritmului).*

*Prima etapă a algoritmului solicită rezolvarea unui sistem de două inegalităţi duble:*

$$-\delta < 3p - 7q < \delta, \qquad -\delta < 2p - 7r < \delta, \qquad (1 \le p \le 6, \;\; 1 \le q \le 2, \;\; r = 1).$$

*Valoarea recomandată pentru* $\delta$ *este* $\sqrt{b_1/2} = \sqrt{7/2} = 1,87$. *Această alegere nu dă însă nici o soluţie pentru* $p$ *(de fapt, pe exemple mici, orice rezultat asimptotic poate fi eronat). Vom trece la etapa a doua, considerând drept candidaţi pentru testare toate valorile lui* $p$. *Deci împărţim intervalul* $(0,1)$ *în subintervalele*
$\left(0,\frac{1}{7}\right), \left(\frac{1}{7},\frac{2}{7}\right), \left(\frac{2}{7},\frac{1}{3}\right), \left(\frac{1}{3},\frac{3}{7}\right), \left(\frac{3}{7},\frac{1}{2}\right), \left(\frac{1}{2},\frac{4}{7}\right), \left(\frac{4}{7},\frac{2}{3}\right), \left(\frac{2}{3},\frac{5}{7}\right), \left(\frac{5}{7},\frac{6}{7}\right), \left(\frac{6}{7},1\right)$.

*În fiecare din aceste subintervale, cele trei curbe sunt de forma* $f_i(u) = b_i u - c_i^j$ *(*$i = 1,2,3$*, iar indicele* $j$ *indică intervalul). Toate intervalele sunt deschise, pentru că nici un punct de discontinuitate nu va corespunde unei perechi trapă.*

*Pentru fiecare subinterval considerăm inecuaţiile*

$$\begin{cases} (7u - i_1) + (3u - i_2) + (2u - i_3) < 1 \\ 7u - i_1 < 3u - i_2 \\ (7u - i_1) + (3u - i_2) < 2u - i_3 \end{cases}$$

*cu* $0 \le i_1 \le 6,\ \ 0 \le i_2 \le 2,\ \ 0 \le i_3 \le 1$. *Acest sistem se poate rescrie*

$$12u < i, \qquad 4u < j, \qquad 8u < k$$

*unde s-a notat* $i = 1 + i_1 + i_2 + i_3,\ \ j = i_1 - i_2,\ \ k = i_1 + i_2 - i_3$.

*Listăm într-un tabel toate variantele posibile:*

| | $\left(0,\frac{1}{7}\right)$ | $\left(\frac{1}{7},\frac{2}{7}\right)$ | $\left(\frac{2}{7},\frac{1}{3}\right)$ | $\left(\frac{1}{3},\frac{3}{7}\right)$ | $\left(\frac{3}{7},\frac{1}{2}\right)$ | $\left(\frac{1}{2},\frac{4}{7}\right)$ | $\left(\frac{4}{7},\frac{2}{3}\right)$ | $\left(\frac{2}{3},\frac{5}{7}\right)$ | $\left(\frac{5}{7},\frac{6}{7}\right)$ | $\left(\frac{6}{7},1\right)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $i_1$ | 0 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 |
| $i_2$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| $i_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $j$ | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 2 | 3 | 4 |
| $k$ | 0 | 1 | 2 | 3 | 4 | 3 | 4 | 5 | 6 | 7 |
| $12u < i$ | *PT* | *PT* | *NU* | *NU* | *NU* | *NU* | *PT* | *NU* | *PT* | *NU* |
| $4u < j$ | *NU* | *PT* | *DA* | *NU* | *DA* | *NU* | *DA* | *NU* | *PT* | *DA* |
| $8u < k$ | *NU* | *NU* | *NU* | *PT* | *DA* | *NU* | *NU* | *NU* | *PT* | *PT* |

*Pentru fiecare subinterval am notat: DA – dacă inegalitatea din prima coloană este adevărată, NU – dacă nu este adevărată, şi PT – dacă este parţial adevărată (verificată doar la unul din capete). Un interval generează perechi trapă numai dacă pe coloana sa apar numai DA sau PT.*

*În cazul nostru, singurul interval valid este* $(5/7, 6/7)$. *Alegem numere raţionale* $u/m$ *din acest interval, care să verifice cele trei inecuaţii. Căutând cei mai mici numitori posibili, primul număr raţional care verifică este* $8/11$. *Deci luăm* $u = 8$ *şi* $m = 11$. *Atunci relaţia* $a_i \equiv b_i u \pmod{11}$ *transformă vectorul* $B = (7,3,2)$ *în vectorul cu creştere mare* $A = (1,2,5)$.

*Evident, aceasta nu este singura soluţie posibilă. Alte soluţii pot fi de exemplu:*

- $(u, m) = (41, 56)$ *care transformă* $B$ *în vectorul super-crescător* $A = (7, 11, 26)$,
- $(u, m) = (61, 84)$ *care conduce la vectorul super-crescător* $A = (7, 15, 38)$,
- $(u, m) = (223, 308)$ *care conduce la vectorul super-crescător* $A = (25, 35, 138)$.

**Exemplul 11.6.** *Să considerăm vectorul rucsac public*

$$B = (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523)$$

*Este mult prea dificil să scriem lista completă a punctelor de discontinuitate din toate intervalele* $\left(\frac{p}{43}, \frac{p+1}{43}\right)$*; de exemplu, numai funcţia* $f_{10}(u)$ *(pentru* $p = 1253$*) are* 35 *puncte de discontinuitate. Totuşi,* $B$ *conţine destul de multe slăbiciuni criptografice pentru a permite scurtarea algoritmului de atac.*

*Inegalităţile din prima etapă pot fi scrise sub forma*

$$|129p - 43q| \leq \delta, \qquad |215p - 43r| \leq \delta, \qquad |473p - 43s| \leq \delta$$

*Pentru că* $129, 215$ *şi* 473 *sunt multipli de* 43*, rezultă că* $p = 1$ *este un candidat posibil. Nu mai căutăm alte valori pentru* $p$ *şi restrângem investigaţia la intervalul* $\left(\frac{1}{43}, \frac{2}{43}\right)$*. Considerând şi puncte de discontinuitate ale altor curbe în acest interval, vom putea restrânge subintervalul soluţiilor la* $\left(\frac{1}{43}, \frac{36}{1523}\right)$*. Cele* 10 *curbe sunt definite aici sub forma segmentelor de dreaptă de ecuaţii*

$$\begin{array}{llll} f_1(u) = 43u - 1, & f_2(u) = 129u - 3, & f_3(u) = 215u - 5, & f_4(u) = 473u - 11, \\ f_5(u) = 903u - 21 & f_6(u) = 302u - 7, & f_7(u) = 561u - 13, & f_8(u) = 1165u - 27, \\ f_9(u) = 697u - 16, & f_{10}(u) = 1523u - 35. & & \end{array}$$

*Inecuaţia a doua (care se referă la mărimea modulului) este* $\sum_{i=1}^{10} f_i(u) < 1$ *sau* $6011u - 139 < 1$*, care duce la soluţia* $u < 140/6011$*. Pentru că* $140/6011 < 36/1523$*, găsim un nou interval – mai restrâns – al soluţiilor:* $\left(\frac{1}{43}, \frac{140}{6011}\right)$.

*Listăm inecuaţiile care exprimă condiţia de super-creştere, împreună cu soluţiile lor:*

$$\begin{array}{ll} 129u - 3 > 43u - 1 & u > 1/43 \\ 215u - 5 > 172u - 4 & u > 2/43 \\ 473u - 11 > 387u - 9 & u > 1/43 \\ 903u - 21 > 860u - 20 & u > 1/43 \\ 302u - 7 > 1763u - 41 & u < 34/1461 \\ 561u - 13 > 2065u - 48 & u < 35/1504 \\ 1165u - 27 > 2626u - 61 & u < 34/1461 \\ 697u - 16 > 3791u - 88 & u < 72/3094 \\ 1523u - 35 > 4488u - 104 & u < 69/2965 \end{array}$$

*Cea mai mică margine superioară dintre toate aceste soluţii este* $72/3094 = 36/1547$*. În final se obţine intervalul* $\left(\frac{1}{43}, \frac{36}{1547}\right)$*. Alegând numărul* $37/1590$ *din acest interval se obtine vectorul super-crescător* $A$ *din Exemplul* 7.4.*, Capitolul* 7*. Pentru numărul* $72/3095$ *se obţine vectorul super-crescător* $A = (1, 3, 5, 11, 21, 79, 157, 315, 664, 1331)$.

### 11.1.3 Vectori rucsac cu densitate mare

Spargerea sistemului de criptare rucsac definit de Merkle şi Hellman nu înseamnă rezolvarea problemei rucsacului, ci numai exploatarea slăbiciunii provenite din modul de alegere a cheii secrete. Ulterior au fost generate şi alte construcţii de cripto-sisteme rucsac. În această setiune prezentăm un astfel de sistem, prezentat de Arto Salomaa ([45]).

În varianta Merkle - Hellman, sistemul de criptare rucsac se baza pe vectori cu densitate mică, în sensul că elementele vectorului erau foarte rare în raport cu numărul lor. Noul sistem propune o construcţie bazată pe vectori rucsac cu densitate mare.

Fie $p$ un număr prim şi $h \geq 1$. Un element $\alpha$ este algebric de gradul $h$ peste $Z_p$ dacă satisface o ecuaţie polinomială $P(x) = 0$ de gradul $h$ şi nici o ecuaţie de grad mai mic (adică $P(x)$ este un polinom ireductibil peste $Z_p$). Considerând atunci extensia Galois $GF(p^h)$, elementele sale pot fi reprezentate sub forma

$$x = \sum_{j=0}^{h-1} c_j \alpha^j \qquad (0 \leq c_j \leq p-1)$$

**Exemplul 11.7.** *Fie $p = 3,\ h = 2$ şi $\alpha$ o rădăcină a ecuaţiei $X^2 - X - 1 = 0$. Elementele corpului $GF(3^2)$ pot fi exprimate în funcţie de $\alpha$ astfel:*

$$GF(3^2) = \{0,\ 1,\ 2,\ \alpha,\ \alpha+1,\ \alpha+2,\ 2\alpha,\ 2\alpha+1,\ 2\alpha+2\}$$

Similar aritmeticii modulare definite pe $Z_p$, putem folosi noţiunea de logaritm discret şi în extensiile Galois $GF(p^h)$. Un element $\beta$ este un generator al lui $GF(p^h) \setminus \{0\}$ dacă pentru orice $x \in GF(p^h) \setminus \{0\}$ există un întreg $i \in [0, p^h - 1]$ astfel ca $x = \beta^i$. Deci $i = log_\beta x$.

**Exemplul 11.8.** *Pentru extensia $GF(3^2)$ construită în Exemplul 11.7, $\alpha$ este un generator. Logaritmii elementelor nenule din această extensie sunt:*

| $x$ | 1 | 2 | $\alpha$ | $\alpha+1$ | $\alpha+2$ | $2\alpha$ | $2\alpha+1$ | $2\alpha+2$ |
|---|---|---|---|---|---|---|---|---|
| $log_\alpha x$ | 8 | 4 | 1 | 2 | 7 | 5 | 3 | 6 |

*Tot un generator al lui $GF(3^2) \setminus \{0\}$ este şi $2\alpha + 1$. Tabela de logaritmi în această bază este*

| $x$ | 1 | 2 | $\alpha$ | $\alpha+1$ | $\alpha+2$ | $2\alpha$ | $2\alpha+1$ | $2\alpha+2$ |
|---|---|---|---|---|---|---|---|---|
| $log_{2\alpha+1} x$ | 8 | 4 | 3 | 6 | 5 | 7 | 1 | 2 |

În 1936 a fost enunţată o problemă interesantă, cu aplicaţii în domeniu. Anume, fiind date numerele întregi pozitive $n$ şi $h$, există un vector $A = (a_1, a_2, \ldots, a_n)$ cu elemente nenegative distincte, astfel încât toate sumele de exact $h$ componente de elemente (nu neapărat distincte) din $A$, sunt diferite. Un astfel de vector $A$ este uşor de construit, luând $a_i = h^{i-1}$ $(1 \leq i \leq n)$. Construcţia corespunde vectorilor rucsac cu densitate mică

(categorie în care intră şi vectorii super-crescători). În cazul vectorilor rucsac cu densitate mare (unde valorile componentelor lui $A$ cresc doar polinomial în $n$), Bose şi Chowla au dat o rezolvare, prezentată în Lema următoare:

**Lema 11.2.** *Fie $p$ un număr prim şi $h$ un număr întreg, $(h \geq 2)$. Atunci există un vector rucsac $A = (a_1, a_2, \ldots, a_p)$ care satisface condiţiile*

1. $1 \leq a_i \leq p^h - 1, \quad (1 \leq i \leq p)$,
2. *Dacă $x_, y_i$ $(1 \leq i \leq p)$ sunt numere întregi nenegative cu*

$$(x_1, x_2, \ldots, x_p) \neq (y_1, y_2, \ldots, y_p) \quad iar \quad \sum_{i=1}^{p} x_i = \sum_{i=1}^{p} y_i = h,$$

*atunci* $\sum_{i=1}^{p} a_i x_i \neq \sum_{i=1}^{p} a_i y_i$.

*Demonstraţie.* Să considerăm extensia Galois $GF(p^h)$; fie $\alpha$ un element algebric de grad $h$ peste $Z_p$, iar $g$ – un generator al lui $GF(p^h) \setminus \{0\}$. Definim

$$a_i = log_g(\alpha + i - 1), \qquad (1 \leq i \leq p)$$

Condiţia (1) este evident satisfăcută. Pentru a arăta şi (2), să presupunem prin absurd că există numerele $x_i,\ y_i$ $(1 \leq i \leq p)$ care verifică ipoteza, dar $\sum_{i=1}^{p} a_i x_i = \sum_{i=1}^{p} a_i y_i$. Deci

$$g^{\sum_{i=1}^{p} a_i x_i} = g^{\sum_{i=1}^{p} a_i y_i}$$

sau

$$(\alpha + 0)^{x_1} \ldots (\alpha + p - 1)^{x_p} = (\alpha + 0)^{y_1} \ldots (\alpha + p - 1)^{y_p}.$$

Ţinând cont şi de ipoteză, această expresie este un polinom în $\alpha$ de grad cel mult $h - 1$, ceea ce contrazice faptul că $\alpha$ este algebric de gradul $h$. □

Demonstraţia rămâne valabilă şi pentru $p = q^n$, $q$ număr prim. De asemenea, se poate înlocui concluzia Lemei 11.2 cu un rezultat mai tare:

$$\sum_{i=1}^{p} a_i x_i \not\equiv \sum_{i=1}^{p} a_i y_i \qquad (mod\ p^h - 1)$$

Pe baza acestor rezultate, să construim un sistem de criptare rucsac. Textul clar va conţine cuvinte (blocuri) de $p$ biţi, astfel încât în fiecare bloc sunt exact $h$ biţi egali cu 1. În general, un text clar arbitrar nu poate fi segmentat în astfel de blocuri; totuşi, se pot defini codificări convenabile premergătoare ale textului clar. O astfel de codificare este asigurată de rezultatul următor:

**Lema 11.3.** *Fie $n \geq 3$ şi $h < n$. Atunci există o aplicaţie injectivă a mulţimii secvenţelor binare de lungime $\lfloor log_2 C_n^h \rfloor$ în mulţimea secvenţelor binare de $n$ biţi în care apar $h$ de 1.*

*Demonstraţie.* Să considerăm secvenţele binare de lungime $\lfloor log_2 C_n^h \rfloor$ ca reprezentări binare de numere $a$.

- Ordonăm crescător secvenţele binare de lungime $n$ având $h$ de 1.

- Stabilim o corespundenţă între secvenţa binară asociată numărului $a$ şi al $(a+1)$ - lea bloc binar astfel ordonat.

Aplicaţia definită de această corespondenţă este injectivă; în plus, deoarece $C_n^h < 2^n$, nu sunt epuizate toate secvenţele binare. □

**Exemplul 11.9.** *Fie $n = 5$ şi $h = 2$. Atunci $\lfloor log_2 C_5^2 \rfloor = 3$; deci putem codifica blocuri de 3 biţi. Corespondenţa se realizează după tabelul:*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | $\longrightarrow$ | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | $\longrightarrow$ | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | $\longrightarrow$ | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | $\longrightarrow$ | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | $\longrightarrow$ | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | $\longrightarrow$ | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | $\longrightarrow$ | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | $\longrightarrow$ | 1 | 0 | 0 | 1 | 0 |

*Secvenţele* 10100 *şi* 11000 *rămân neutilizate.*

Să descriem acum sistemul de criptare:

- Se alege un număr prim $p$ şi fie $h < p$. Se alege $\alpha$ algebric de gradul $h$ peste $Z_p$ şi un generator $g$ al lui $GF(p^h) \setminus \{0\}$.
- Se calculează $A = (a_1, a_2 \ldots, a_p)$ definit prin
$$a_i = log_g(\alpha + i - 1), \quad (1 \leq i \leq p) \tag{1}$$
- Se defineşte vectorul $B = (b_1, b_2, \ldots, b_p)$ prin $b_i = a_{\pi(i)} + d$,
unde $\pi \in S_p$ este o permutare, iar $d$ $(0 \leq d \leq p^h - 2)$ este o constantă arbitrară.

Cheia publică de criptare este $B, p, h$. Trapa secretă constă din $\alpha, g, d, \pi$.

Fie $C$ o secvenţă binară de lungime $p$ în care suma elementelor este $h$. Considerat sub forma unui vector, $C$ este criptat prin produsul scalar

$$e(C) = B \cdot C^T \quad (mod\ p^h - 1)$$

Protocolul de decriptare urmat de *Bob* este:

- Dacă $x$ este mesajul criptat, în prima fază se determină $y = x - h \cdot d \ (mod\ p^h - 1)$.
- Se calculează $g^y$ în $GF(p^h)$. Acesta este un polinom în $\alpha$, de grad cel mult $h-1$. Pe de altă parte, $\alpha$ satisface o ecuaţie de forma $\alpha^h = r(\alpha)$, unde $r(X) \in Z_q[X]$ este un polinom de grad cel mult $h-1$.
- Polinomul $s(\alpha) = \alpha^h + g^y - r(\alpha)$ se descompune în factori liniari peste $Z_p$ (lucru posibil deoarece $s(\alpha)$ este un produs de puteri ale lui $g$, fiecare exponent fiind de forma (1)). Fie $s(\alpha) = (\alpha + i_1 - 1)(\alpha + i_2 - 1) \ldots (\alpha + i_h - 1)$ această descompunere.
- Poziţia elementelor 1 din textul clar este dată de valorile $(\pi^{-1}(i_1), \pi^{-1}(i_2), \ldots, \pi^{-1}(i_h))$.

Unicitatea decriptării este asigurată de Lema 11.2.

**Exemplul 11.10.** *Pentru facilitarea înţelegerii, vom face abstracţie de permutarea $\pi$ şi deplasarea $d$. Fie extensia $GF(3^2)$ definită în Exemplele 11.7 şi 11.8, în care $\alpha$ verifică ecuaţia $X^2 = X + 1$, iar $2\alpha + 1$ este generatorul extensiei. Deoarece logaritmii elementelor $\alpha, \alpha + 1$ şi $\alpha + 2$ sunt $3, 6$ şi respectiv $5$, se obţine cheia publică de criptare $A = (3, 6, 5)$ (în acest exemplu $B = A$). În plus, $p = 3$ şi $h = 2$.*

*Un text clar este compus din vectori binari cu 3 componente, în care suma componentelor este 2.*

*Să considerăm vectorii $(2, 0, 0)$ şi $(0, 1, 1)$. Ei sunt criptaţi în numerele 6 respectiv 3 (calculele se efectuează modulo $p^h - 1 = 8$).*

*La decriptare, Bob:*

*- calculează întâi puterile $(2\alpha + 1)^6 = \alpha + 1$ şi $(2\alpha + 1)^3 = \alpha$.*

*- La ambele expresii, adună $\alpha^2 - \alpha - 1$, rezultând polinoamele $\alpha^2$ respectiv $\alpha^2 - a = (\alpha + 1)(\alpha + 2)$.*

*- Se deduc textele clare $(2, 0, 0)$ respectiv $(0, 1, 1)$.*

**Exemplul 11.11.** *Să relăm elementele principale din Exemplul 11.10, şi să construim vectorul public, aplicând lui $A$ permutarea $\pi = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ urmată de deplasarea $d = 7$. Vectorul rezultat este $B = (5, 4, 2)$; acesta, împreună cu $p = 3$ şi $h = 2$ constituie cheia publică. Trapa secretă este formată din $\pi$, $d$, polinomul $X^2 - X - 1$ şi generatorul $2\alpha + 1$.*

*Atunci, textul clar $(0, 1, 1)$ este criptat în 6.*

*La recepţie, Bob calculează întâi $6 - 2 \cdot 7 \ (mod\ 8) \ = 0$.*
*Pe urmă determină $\alpha^2 + (2\alpha + 1)^0 - \alpha - 1 = \alpha^2 - \alpha = \alpha(\alpha + 2)$, care conduce la vectorul $(1, 0, 1)$. Acestuia i se aplică permutarea inversă $\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$ pentru a obţine textul clar $(0, 1, 1)$.*

## 11.2 Sistemul de criptare McEliece

Sistemul de criptare *McEliece* – propus în 1978 – este destul de apropiat de problema rucsacului. El utilizează drept cadru teoria codurilor liniare (pentru detalii vezi [2]); aici, în general decodificarea unui cod liniar binar corector de erori este o problemă $\mathcal{NP}$ - completă. Pentru unele clase de coduri sunt construiţi algoritmi de decodificare polinomiali; o astfel de clasă o formează codurile Goppa, care constituie baza sistemului de criptare *McEliece*.

**Definiţia 11.2.** *Fie $k, n \in \mathcal{N}(k \leq n)$. Un $(n, k)$ - cod liniar binar este un subspaţiu liniar $\mathbf{C} \subseteq Z_2^n$ de dimensiune $k$.*

*O matrice generatoare a lui $\mathbf{C}$ este o matrice binară $k \times n$ ale cărei linii formează o bază a lui $\mathbf{C}$.*

*Pentru $\mathbf{a} \in Z_2^n$ se defineşte ponderea $w(\mathbf{a}) =$ numărul de elemente nenule din $\mathbf{a}$.*

*Pentru $\mathbf{a}, \mathbf{b} \in Z_2^n \quad \mathbf{a} = (a_1, \ldots, a_n),\ \mathbf{b} = (b_1, \ldots, b_n)$, se defineşte distanţa Hamming prin $d(\mathbf{a}, \mathbf{b}) = w(\mathbf{a} - \mathbf{b})$.*

*Pentru un $(n, k)$ - cod liniar binar $\mathbf{C}$, distanţa minimă este*

$$d_{\mathbf{C}} = min\{d(\mathbf{a}, \mathbf{b}) \mid \mathbf{a}, \mathbf{b} \in \mathbf{C},\ \mathbf{a} \neq \mathbf{b}\}$$

*Un $(n, k, d)$ - cod este un $(n, k)$ - cod de distanţă minimă $d$.*

Rolul unui cod corector de erori este de a corija modificări aleatoare care apar în transmiterea unui set de date (binare) printr-un canal. În linii mari, acesta funcţionează astfel: dacă $\mathbf{a}$ este un mesaj de informaţie de $k$ biţi, *Alice* îl codifică într-un cuvânt de $n$ biţi $\mathbf{b} = \mathbf{a}G$, unde $G$ este matricea generatoare a codului.

*Bob* primeşte un mesaj $\mathbf{r} \in Z_2^n$ (eventual $\mathbf{r} = \mathbf{b}$) şi caută un cuvânt $\mathbf{b_1} \in \mathbf{C}$ cu $d(\mathbf{r}, \mathbf{b_1})$ minim posibil. Va decodifica $\mathbf{r}$ în $\mathbf{b_1}$ după care va calcula un mesaj de informaţie $\mathbf{a_1}$ astfel ca $\mathbf{b_1} = \mathbf{a_1}G$. Cazul ideal este acela când $\mathbf{b_1} = \mathbf{b},\ \mathbf{a_1} = \mathbf{a}$ (adică erorile au fost acoperite corect). Se cunoaşte că, dacă numărul de erori care apar nu depăşeşte $(d-1)/2$, acest procedeu corectează efectiv erorile.

Dacă *Bob* caută cuvântul - cod cel mai apropiat comparând $\mathbf{r}$ pe rând cu fiecare element din $\mathbf{C}$, cum sunt $2^k$ astfel de cuvinte, algoritmul va fi exponenţial, deci nefuncţional.

Majoritatea algoritmilor de decodificare se bazează pe noţiunea de *sindrom*, definit astfel:

*Matricea de control* a unui $(n, k, d)$ - cod liniar binar de matrice generatoare $G$ este o matrice $H$ de dimensiune $(n-k) \times n$ ale cărei linii formează o bază a unui spaţiu liniar ortogonal. Evident, $G \cdot H^T = \mathbf{0}$.

Pentru un cuvânt $\mathbf{r} \in Z_2^n$, se numeşte *sindrom* secvenţa de $n-k$ biţi definită $H \cdot \mathbf{r}^T$.

**Teorema 11.1.** *$\mathbf{a}$ este un cuvânt - cod dacă şi numai dacă $H \cdot \mathbf{a}^T = \mathbf{0}$.*

*În plus, dacă $\mathbf{a} \in \mathbf{C},\ \mathbf{e} \in Z_2^n$ şi $\mathbf{r} = \mathbf{a} + \mathbf{e}$, atunci $H \cdot \mathbf{r}^T = H \cdot \mathbf{e}^T$.*

Pentru demonstraţie se poate consulta de asemenea [2].

**e** poate fi considerat drept vectorul de erori care au apărut în transmiterea mesajului **a**. Teorema anterioară afirmă că sindromul depinde doar de erori, nu şi de cuvântul - cod transmis.

Această observaţie sugerează o metodă de decodificare bazată pe sindrom. Se calculează întâi $\mathbf{s} = H \cdot \mathbf{r}^T$. Dacă $\mathbf{s} = \mathbf{0}$, decodificarea lui **r** este tot **r**. Altfel, se încearcă toate cuvintele de pondere 1. Pentru fiecare astfel de cuvânt **e** se calculează $H \cdot \mathbf{e}^T$. Dacă s-a găsit un **e** cu $H \cdot \mathbf{e}^T = \mathbf{s}$, **r** se decodifică în $\mathbf{r} - \mathbf{e}$. În caz contrar se încearcă vectorii de pondere $2, 3, \ldots, [(d-1)/2]$. Dacă nu s-a găsit nici un cuvânt **e** cu $H \cdot \mathbf{e}^T = \mathbf{s}$, se deduce că au apărut mai mult de $[(d-1)/2]$ erori în cursul transmisiei.

Metoda prezentată funcţionează pentru toate codurile liniare. Pentru anumite clase speciale de coduri există algoritmi polinomiali de decodificare şi corectare a erorilor; în cazul general însă problema este $\mathcal{NP}$ - completă.

Algoritmul de criptare *McElliece* se bazează pe această idee. Trapa sa secretă o constituie o clasă de coduri pentru care există algoritmi eficace de decodificare – codurile Goppa. În plus, există un număr mare de coduri Goppa neechivalente, având aceiaşi parametri.

Algoritmul de criptare *McEliece* este următorul:

Fie $G$ matricea generatoare a unui $(n, k, d)$ - cod Goppa cu $n = 2^m$, $d = 2t+1$, $k = n - m \cdot t$.
Se definesc:
- $S$ – matrice inversabilă $k \times k$ peste $Z_2$,
- $P$ – matrice de permutare $n \times n$ (matrice în care pe fiecare linie şi coloană există o valoare 1, iar restul elementelor sunt 0).

Fie $\mathcal{P} = Z_2^k$, $\mathcal{C} = Z_2^n$, $\mathcal{K} = \{(G, S, P, G') \mid G' = S \cdot G \cdot P\}$.
$G'$ este publică iar $G, S, P$ sunt secrete.
Pentru $K = (G, S, P, G')$ se defineşte

$$e_K(\mathbf{a}, \mathbf{e}) = \mathbf{a} \cdot G' + \mathbf{e}$$

unde $\mathbf{e} \in Z_2^n$ este un cuvânt aleator de pondere $t$.
*Bob* decriptează un mesaj $\mathbf{b} \in Z_2^n$ astfel:

1. Calculează $\mathbf{b_1} = \mathbf{b} \cdot P^{-1}$;
2. Decodifică $\mathbf{b_1}$ obţinând $\mathbf{b_1} = \mathbf{a_1} + \mathbf{e_1}$ unde $\mathbf{a_1} \in \mathbf{C}$;
3. Calculează $\mathbf{a_0} \in Z_2^k$ astfel ca $\mathbf{a_0} \cdot G = \mathbf{a_1}$;
4. Calculează $\mathbf{a} = \mathbf{a_0} \cdot S^{-1}$.

De remarcat că sistemul *McElliece* este primul sistem de criptare cu cheie publică care a folosit alegerea unei valori aleatoare în procesul de criptare, idee reluată de ulterior de sistemul *El Gamal*.

**Corectitudinea algoritmului de decriptare:**

Deoarece $\mathbf{b_1} = \mathbf{b} \cdot P^{-1} = (\mathbf{a} \cdot G' + \mathbf{e}) \cdot P^{-1} = (\mathbf{a} \cdot S \cdot G \cdot P + \mathbf{e}) \cdot P^{-1} = (\mathbf{a} \cdot S) \cdot G + \mathbf{e} \cdot P^{-1}$ iar $\mathbf{e} \cdot P^{-1}$ este un vector de pondere cel mult $t$, algoritmul de decodificare al codului de matrice generatoare $G$ poate decodifica corect pe $\mathbf{b_1}$ şi obţine un mesaj sursă $\mathbf{a_0} = \mathbf{a} \cdot S$.

La ultimul pas se află mesajul iniţial $\mathbf{a} = \mathbf{a_0} \cdot S^{-1}$.

Nu vom intra în detalii privind definiţia codurilor Goppa ([2]). Acestea pot fi privite însă drept coduri liniare cu parametrii $n = 2^m,\ d = 2t + 1,\ k = n - m \cdot t$. Pentru o implementare practică referitor la criptare, McEliece sugerează $m = 10,\ t = 50$, ceea ce corespunde unui $(1024, 524, 101)$ - cod Goppa[2]. Un text clar este o secvenţă de 524 biţi, iar un text criptat este o secvenţă de 1024 biţi. Cheia publică este o matrice binară de dimensiuni $524 \times 1024$.

**Exemplul 11.12.** *Vom exemplifica algoritmul pe un* $(8, 2, 5)$ *- cod Goppa (deci* $n = 2^3,\ k = 2,\ d = 5$*). Acest cod - extrem de mic (are doar* 4 *cuvinte) este generat de matricea*

$$G = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

*Să presupunem că Bob alege matricile*

$$S = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad cu \qquad S^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

*şi*

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad cu \quad P^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

*Matricea publică generată este deci*

$$G' = S \cdot G \cdot P = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

*Să presupunem că Alice vrea să cripteze textul clar* $\mathbf{a} = (0, 1)$ *folosind vectorul - eroare* $\mathbf{e} = (0, 0, 1, 0, 0, 1, 0, 0)$ *(ales aleator) de pondere* 2*. Textul criptat este*

$$\mathbf{b} = \mathbf{a} \cdot G' + \mathbf{e} = (1, 1, 1, 1, 0, 0, 1, 0).$$

[2]O analiză a securităţii recomandă parametrii $n = 1024,\ t = 38,\ k \geq 644$.

*După recepţionarea mesajului, Bob calculează întâi*

$$\mathbf{b_1} = \mathbf{b} \cdot P^{-1} = (1,1,1,1,1,0,0,0),$$

*pe care îl scrie sub forma* $\mathbf{a_1} + \mathbf{e_1}$ *unde* $\mathbf{a_1} = (1,1,1,1,0,1,0,0)$ *este un cuvânt - cod, iar* $\mathbf{e_1} = (0,0,0,0,1,1,0,0) \neq \mathbf{e}$ *(din cauza înmulţirii cu* $P^{-1}$*).*

*Bob calculează apoi mesajul* $\mathbf{a_0} = (1,1)$*, singurul cu proprietatea* $\mathbf{a_0} \cdot G = \mathbf{a_1}$*.*

*Ultimul pas este determinarea lui* $\mathbf{a} = S^{-1} \cdot \mathbf{a_0} = (0,1)$*, care este textul clar expediat de Alice.*

Algoritmul McElliece s-a dovedit sigur. Acest lucru rezultă din analiza celor două tipuri de atac posibile:

1. Din informaţia publică, *Oscar* încearcă să afle matricea $G$ sau o matrice $G_1$ a unui cod Goppa echivalent (având aceeaşi parametri). Nu se cunoaşte nici un algoritm eficient pentru un astfel de demers.

2. *Oscar* încearcă să afle mesajul clar $\mathbf{a}$ direct din textul criptat $\mathbf{b}$. El ia aleator $k$ coloane din matricea publică $G'$. Notând $G'_k, \mathbf{b_k}, \mathbf{e_k}$ restricţiile lui $G', \mathbf{b}$ respectiv $\mathbf{e}$ la aceste $k$ coloane, vom avea $\mathbf{a} \cdot G'_k = \mathbf{b_k} + \mathbf{e_k}$. Dacă $\mathbf{b_k} = \mathbf{0}$ şi $G'_k$ este nesingulară, atunci $\mathbf{a}$ poate fi aflat rezolvând sistemul liniar $\mathbf{a} \cdot G'_k = \mathbf{b_k}$. Probabilitatea ca cei $k$ biţi selectaţi să nu facă parte din eroare (deci $\mathbf{e_k} = \mathbf{0}$) este $C_{n-t}^k / C_n^k$, neglijabilă pentru valorile alese ale parametrior $n, k, t$.

Interesant, dar această securitate este mult diminuată dacă se foloseşte altă clasă de coduri liniare în locul codurilor Goppa.

Totuşi, în ciuda securităţii sale şi a vitezei relativ mari de criptare/decriptare sistemul McElliece nu este folosit practic. Cauza principală o constituie cheia sa excesiv de mare. De exemplu, pentru $n = 1024,\ t = 38,\ k \geq 644$, cheia are aproximativ $2^{19}$ biţi.

## 11.3 Exerciţii

**11.1.** *Folosind vectorul rucsac public* $B = (228, 325, 346, 485, 556, 525)$ *criptaţi textul clar* $CASA\ DE\ PIATRA$*.*

**11.2.** *Se ştiu parametrii* $m = 523,\ u = 28$ *şi vectorul public*

$$B = (355, 131, 318, 113, 21, 135, 215).$$

1. *Aflaţi vectorul rucsac super-crescător* $A = (a_1, a_2, a_3, a_4, a_5, a_7)$*;*

2. *Decriptaţi mesajul* $(113, 689, 379, 350, 346, 697, 355, 355)$*.*

**11.3.** *Folosind vectorul rucsac public* $B = (102, 238, 3400, 284, 1044, 2122, 425)$ *criptaţi textul clar* $FRUNZA\ DE\ STEJAR$*.*

**11.4.** *Se ştiu parametrii* $m = 3989,\ u = 352$ *şi vectorul public*

$$B = (102, 238, 3400, 284, 1044, 2122, 425).$$

1. *Aflaţi vectorul rucsac super-crescător* $A = (a_1, a_2, a_3, a_4, a_5, a_7)$*;*
2. *Decriptaţi mesajul* $(1753, 2122, 5624, 1566, 1809, 3450, 2360, 1049, 0)$.

**11.5.** *Plecând de la criptanaliza vectorului rucsac din din Exemplul 11.6, să se construiască vectorul super-crescător corespunzător valorii* $720/30949$.

**11.6.** *Fie* $A = (a_1, a_2, \ldots, a_n)$ *un vector rucsac. Un întreg pozitiv* $x$ *este reprezentat de* $A$ *dacă şi numai dacă* $x$ *poate fi exprimat ca o sumă de* $a_i$ *în care fiecare element din* $A$ *apare cel mult odată. Dacă* $A$ *este un vector injectiv, atunci este clar că* $2^n - 1$ *valori întregi sunt reprezentate de* $A$*. Acesta este cel mai mare număr posibil de valori. Care este cel mai mic număr posibil de valori (în funcţie de* $n$*) ?*

**11.7.** *Folosind parametrii din Exemplul 11.12 criptaţi mesajele* $\mathbf{x} = (1, 1)$ *şi* $\mathbf{y} = (1, 0)$.

# 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] Atanasiu, A. - *Arhitectura calculatorului*, Editura Infodata, Cluj, 2006;

[4] Blum L., Blum M., Shub M. - *Comparision of two pseudo-random number generators*, Advanced in Cryptology, CRYPTO 82

[5] 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.

[6] Biham E., Shamir A. - *Differential Cryptanalysis of DES - like Cryptosystems*, Journal of Cryptology, vol. 4, 1 (1991), pp. 3-72.

[7] Biham E., Shamir A. - *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[8] Biham E., Shamir A. - *Differential Cryptanalysis of the Full 16-Round DES*, Proceedings of Crypto92, LNCS 740, Springer-Verlag.

[9] Biham E. - *On Matsuis Linear Cryptanalysis*, Advances in Cryptology - EUROCRYPT 94 (LNCS 950), Springer-Verlag, pp. 341-355, 1995.

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

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

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

[13] Cook S., *http : //www.claymath.org/millennium/P_vs_NP/Official_Problem_Description.pdf*

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

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

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

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

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

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

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

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

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

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

[24] Heyes H. M. - *A Tutorial on Linear and Differential Cryptanalysis.*

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

[26] Junod P. - *On the complexity of Matsuiş attack*, in SAC 01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography, pp 199211, London, UK, 2001. Springer-Verlag.

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

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

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

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

[31] Lenstra, H.W. - *Factoring Integers with Eiipltic Curves*, Annals of Mathematics, vol. 126, pp. 649-673, 1987.

[32] Matsui M, Yamagishi A. - *A new method for known plaintext attack of FEAL cipher.* Advances in Cryptology - EUROCRYPT 1992.

[33] Matsui M. - *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology - EUROCRYPT 93, LNCS 765, Springer-Verlag, pp. 386-397, 1994.

[34] Matsui M. - *The first experimental cryptanalysis of the Data Encryption Standard*, in Y.G. Desmedt, editor, Advances in Cryptology - Crypto 4, LNCS 839, SpringerVerlag (1994), 1- 11.

[35] Matsui M. - *New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptalaysis*, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, pp. 205-218.

[36] Merkle R. C., Hellman M. - *Hiding Information and Signatures in Trapdoor Knapsacks*, IEEE Trans. IT 24(5), Sept 1978, pp. 525530.

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

[38] Menezes A., Oorschot P., Vanstome S. - *Handbook of Applied Cryptography*, CRC Press 1996.

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

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

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

[42] 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.

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

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

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

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

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

[48] Shamir, A. - *A polynomial time Algorithm for breaking the basic Merkle - Hellman cryptosystem*,

http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C82/279.PDF

[49] Shoup, V. - *Lower bounds for discrete logarithm and related problems*, Advanced in Cryptology, EUROCRYPT 97, Springer - Verlag LNCS 1233, pp. 313-328, 1997.

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

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

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

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

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

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

[56] 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.

[57] *Secure hash Standard*; National Bureau of Standards, FIPS Publications 180, 1993

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

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

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

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

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

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

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

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

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

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