Capitolul 9

Sistemul de criptare El Gamal

9.1 Descrierea algoritmului de criptare El Gamal

Sistemul de criptare El Gamal[1], prezentat în 1985 (vezi [21]) de Taher ElGamal, se bazează pe problema logaritmului discret (PLD), care este următoarea:

> Fie p număr prim şi $\alpha, \beta \in Z_p,\ \beta \neq 0$.
> Să se determine $a \in Z_{p-1}$ astfel ca
> $$\alpha^a \equiv \beta\ (mod\ p).$$
> Acest întreg a – dacă există – este unic şi se notează $log_\alpha \beta$.

Exemplul 9.1. *Fie $p = 11$ şi $\alpha = 6$. Toate elementele din Z_{11}^* pot fi exprimate ca puteri ale lui α:*

a	0	1	2	3	4	5	6	7	8	9
$6^a\ (mod\ 11)$	1	6	3	7	9	10	5	8	4	2

De aici rezultă imediat tabelul logaritmilor în baza 6:

β	1	2	3	4	5	6	7	8	9	10
$log_6\beta$	0	9	2	8	6	1	3	7	4	5

Pentru $\alpha = 3$ însă nu vom avea totdeauna soluţie. Deoarece

a	0	1	2	3	4	5	6	7	8	9
$3^a\ (mod\ 11)$	1	3	9	5	4	1	3	9	5	4

valorile $\beta \in \{2, 6, 7, 8, 10\}$ nu pot fi exprimate ca logaritmi în baza 3. Altfel spus, ecuaţia $log_3 x = \beta$ nu are soluţie în Z_{11} pentru aceste valori ale lui β.

[1] Implementări ale sistemului sunt conţinute în softuri pentru GNU Privacy Guard şi PGP – pentru a lista doar cele mai cunoscute aplicaţii.

Observaţia 9.1. *Pentru problema logaritmului discret, nu este obligatoriu ca p să fie număr prim. Important este ca α să fie rădăcină primitivă de ordinul $p-1$ a unităţii: $\forall i\ (0 < i < p-1),\ \alpha^i \not\equiv 1\ (mod\, p)$. Teorema lui Fermat asigură $\alpha^{p-1} \equiv 1\ (mod\, p)$.*

La o alegere convenabilă a lui p, PLD este $\mathcal{NP}$ - completă. Pentru siguranţă, p se alege de minim 512 biţi[2] iar $p-1$ să aibă cel puţin un divizor prim "mare". Pentru un astfel de modul p, spunem că *problema logaritmului discret este dificilă în Z_p*. Utilitatea acestei cerinţe rezidă în faptul că, deşi este foarte dificil de calculat un logaritm discret, operaţia inversă – de exponenţiere – este foarte simplă (după cum s-a văzut la sistemul RSA).

Sistemul de criptare El Gamal este următorul:

> Fie p număr prim pentru care PLD este dificilă în Z_p, şi fie $\alpha \in Z_p^*$ primitiv.
> Definim $\mathcal{P}= Z_p^*$, $\mathcal{C}= Z_p^* \times Z_p^*$ şi $\mathcal{K}= \{(p, \alpha, a, \beta) \mid \beta \equiv \alpha^a\ (mod\, p)\}$.
> Valorile p, α, β sunt publice, iar a este secret.
>
> Pentru $K = (p, \alpha, a, \beta)$ şi $k \in Z_{p-1}$ aleator (secret) se defineşte
>
> $$e_K(x, k) = (y_1, y_2)$$
>
> unde $y_1 = \alpha^k\ (mod\, p)$, $\quad y_2 = x \cdot \beta^k\ (mod\, p)$.
> Pentru $y_1, y_2 \in Z_p^*$ se defineşte
>
> $$d_K(y_1, y_2) = y_2 \cdot (y_1^a)^{-1}\ (mod\, p)$$

Verificarea este imediată:

$$y_2 \cdot (y_1^a)^{-1} \equiv x \cdot \beta^k \cdot (\alpha^{ka})^{-1} \equiv x \cdot \beta^k (\beta^k)^{-1} \equiv x\ (mod\, p)$$

Sistemul este evident nedeterminist: criptarea depinde de x şi de o valoare aleatoare aleasă de *Alice*. Există deci mai multe texte criptate corespunzătoare unui anumit text clar.

Exemplul 9.2. *Să alegem $p = 2579$, $\alpha = 2$, $a = 765$. Prin calcul se obţine $\beta = 2^{765}\ (mod\, 2579) = 949$.*

Să presupunem că Alice vrea să trimită mesajul $x = 1299$. Ea alege aleator k (să spunem $k = 853$) şi calculează $y_1 = 2^{853} = 435$, apoi $y_2 = 1299 \cdot 949^{853} = 2396$ (toate calculele se fac modulo 2579).

Când Bob primeşte mesajul criptat $y = (435, 2396)$, el va determina

$$x = 2396 \cdot (435^{765})^{-1} = 1299\ (mod\, 2579).$$

[2] Pentru o securitate pe termen lung se recomandă 1024 biţi ([38]).

Observaţia 9.2.

1. *Un dezavantaj al sistemului El Gamal constă în dublarea lungimii textului criptat (comparativ cu lungimea textului clar).*

2. *Dacă* (y_1, y_2), (z_1, z_2) *sunt textele criptate ale mesajelor* m_1, m_2 *atunci se poate deduce imediat un text criptat pentru* m_1m_2 : (y_1z_1, y_2z_2). *Similar poate fi dedusă o criptare pentru* $2m_1$ *(sau* $2m_2$*). Acest lucru face sistemul El Gamal permeabil la un atac cu text clar ales.*

3. *Indicaţia ca pentru criptarea a două texte diferite să se folosească valori diferite ale parametrului k este esenţială: astfel, să prsupunem că mesajele* m_1, m_2 *au fost criptate în* (y_1, y_2) *respectiv* (z_1, z_2) *folosind acelaşi* k. *Atunci* $y_2/z_2 = m_1/m_2$ *şi cunoaşterea unuia din mesaje îl determină imediat pe celălalt.*

9.2 Calculul logaritmului discret

În această secţiune vom presupune că p este număr prim, iar α este o rădăcină primitivă de ordinul $p-1$ a unităţii. Aceste două valori fiind fixate, PLD se poate reformula astfel:

Fiind dat un $\beta \in Z_p^*$*, să se determine exponentul* $a \in Z_{p-1}$ *astfel ca* $\alpha^a \equiv \beta\ (mod\ p)$.

Evident această problemă se poate rezolva printr-o căutare directă (se calculează puterile lui α) în timp $O(p)$ şi folosind $\mathcal{O}(1)$ memorie. Pe de-altă parte, dacă se calculează anterior într-o tabelă toate valorile $(a, \alpha^a\ mod\ p)$, aflarea valorii căutate se poate face în $\mathcal{O}(1)$, dar cu un spaţiu de complexitate $\mathcal{O}(p)$.

Toţi algoritmii construiţi pentru calculul logaritmului discret folosesc un compromis spaţiu - timp.

9.2.1 Algoritmul Shanks

Fie $m = \left\lceil \sqrt{p-1} \right\rceil$. Algoritmul Shanks este:

1.	Se construieşte lista $L_1 = \{(j, \alpha^{mj}\ (mod\ p)) \mid 0 \le j \le m-1\}$;
2.	Se construieşte lista $L_2 = \{(i, \beta\alpha^{-i}\ (mod\ p)) \mid 0 \le i \le m-1\}$;
3.	Se determină perechile $(j, y) \in L_1,\ (i, y) \in L_2$ (identice pe a doua poziţie);
4.	Se defineşte $log_\alpha \beta = m \cdot j + i\ (mod\ (p-1))$

De remarcat că prin alegerea perechilor $(j, y) \in L_1,\ (i, y) \in L_2$ vom avea

$$\alpha^{mj} = y = \beta\alpha^{-i}, \quad \text{deci } \alpha^{mj+i} = \beta.$$

Invers, pentru orice β putem scrie $log_\alpha \beta = m \cdot j + i$ cu $0 \le i, j \le m-1$; deci căutarea de la pasul 3 se termină totdeauna cu succes.

Implementarea acestui algoritm se poate face în timp $\mathcal{O}(m)$ şi spaţiu $O(m)$.

Exemplul 9.3. *Fie* $p = 809$ *şi să determinăm* $log_3 525$. *Avem deci* $\alpha = 3,\ \beta = 525,\ m = \lceil\sqrt{808}\rceil = 29,$ *iar* $\alpha^{29}\ mod\ 809\ = 99$.

Lista L_1 *a perechilor* $(j, 99^j\ (mod\ 809)),\ 0 \le j \le 28$ *este:*

$(0,1)$	$(1,99)$	$(2,93)$	$(3,308)$	$(4,559)$
$(5,329)$	$(6,211)$	$(7,664)$	$(8,207)$	$(9,268)$
$(10,644)$	$(11,654)$	$(12,26)$	$(13,147)$	$(14,800)$
$(15,727)$	$(16,781)$	$(17,464)$	$(18,632)$	$(19,275)$
$(20,528)$	$(21,496)$	$(22,564)$	$(23,15)$	$(24,676)$
$(25,586)$	$(26,575)$	$(27,295)$	$(28,81)$	

Lista L_2 *a cuplurilor* $(i, 525\cdot(3^i)^{-1}\ (mod\ 809)),\ 0 \le i \le 28$ *este:*

$(0,525)$	$(1,175)$	$(2,328)$	$(3,379)$	$(4,396)$
$(5,132)$	$(6,44)$	$(7,554)$	$(8,724)$	$(9,511)$
$(10,440)$	$(11,686)$	$(12,768)$	$(13,256)$	$(14,355)$
$(15,388)$	$(16,399)$	$(17,133)$	$(18,314)$	$(19,644)$
$(20,754)$	$(21,521)$	$(22,713)$	$(23,777)$	$(24,259)$
$(25,356)$	$(26,658)$	$(27,489)$	$(28,163)$	

Parcurgând (eventual simultan) cele două liste se găseşte $(10,644) \in L_1,\ (19,644) \in L_2$.

Se poate scrie deci

$$log_3 525 = 29\cdot 10 + 19 = 309.$$

Se verifică uşor că $3^{309} \equiv 525\ (mod\ 809)$.

9.2.2 Algoritmul Pohlig - Hellman

Mai întâi, un rezultat matematic:

Lema 9.1. *Fie* $x \in Z_p$ *un element primitiv. Atunci*

$$x^m \equiv x^n\ (mod\ p) \qquad \Longleftrightarrow \qquad m \equiv n\ (mod\ (p-1))$$

Demonstraţie. Relaţia $x^m \equiv x^n\ (mod\ p)$ se poate rescrie $x^{m-n} \equiv 1\ (mod\ p)$. Dar – conform Teoremei lui Fermat – $x^{p-1} \equiv 1\ (mod\ p)$ şi $x^i \not\equiv 1\ (mod\ p)$ pentru $0 < i < p-1$. Deci $p-1 | m-n$, sau $m - n \equiv 0\ (mod\ (p-1))$, relaţie echivalentă cu $m \equiv n\ (mod\ (p-1))$. □

Revenind la sistemul de criptare El Gamal, să considerăm descompunerea în factori primi

$$p - 1 = \prod_{i=1}^{k} q_i^{c_i}.$$

Dacă s-ar putea calcula $a\ (mod\ q_i^{c_i})$ pentru toţi $i = 1, \ldots, k$, atunci – folosind Teorema chineză a resturilor – s-ar putea determina $a\ mod\ (p-1)$.

Fie q un număr prim astfel ca $p-1 \equiv 0\ (mod\ q^c)$ şi $p-1 \not\equiv 0\ (mod\ q^{c+1})$. Să arătăm cum se poate calcula atunci $x \equiv a\ (mod\ q^c)$ pentru orice x, $(0 \le x \le q^c - 1)$.

Să descompunem întâi x în baza q folosind egalitatea

$$x = \sum_{i=0}^{c-1} a_i q^i, \quad (0 \le a_i \le q-1).$$

Atunci se poate scrie $a = x + q^c \cdot s$ pentru un anumit număr întreg pozitiv s.

La primul pas trebuie calculat a_0. Se porneşte de la observaţia că

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)a_0/q}\ (mod\ p).$$

Pentru a arăta aceasta, deoarece $\beta^{(p-1)/q} \equiv \alpha^{(p-1)(x+q^c s)/q}\ (mod\ p)$, este suficient să se verifice că $\alpha^{(p-1)(x+q^c s)/q} \equiv \alpha^{(p-1)a_0/q}\ (mod\ p)$.

Această relaţie este adevărată dacă şi numai dacă

$$\frac{(p-1)(x+q^c s)}{q} \equiv \frac{(p-1)a_0}{q} \quad (mod\ (p-1)),$$

ceea ce se poate verifica prin calcul direct:

$$\frac{(p-1)(x+q^c s)}{q} - \frac{(p-1)a_0}{q} = \frac{p-1}{q}(x + q^c s - a_0) = \frac{p-1}{q}\left(\sum_{i=0}^{c-1} a_i q^i + q^c s - a_0\right) =$$

$$= \frac{p-1}{q}\left(\sum_{i=1}^{c-1} a_i q^i + q^c s\right) = (p-1)\left(\sum_{i=1}^{c-1} a_i q^{i-1} + q^{c-1} s\right) \equiv 0 \quad (mod\ (p-1)).$$

Putem acum să începem calculul lui $\beta^{(p-1)/q}\ (mod\ p)$. Dacă $\beta^{(p-1)/q} \equiv 1\ (mod\ p)$, atunci $a_0 = 0$. Altfel se calculează în Z_p $\gamma = \alpha^{(p-1)/q}$, $\gamma^2, \ldots$ până se obţine un număr întreg pozitiv i pentru care $\gamma^i \equiv \beta^{(p-1)/q}$. Atunci $a_0 = i$.

Dacă $c = 1$, algoritmul se termină; altfel, $(c > 1)$, se caută valoarea lui a_1. Pentru aceasta se defineşte

$$\beta_1 = \beta\alpha^{-a_0}$$

şi se notează $x_1 = log_\alpha \beta_1\ (mod\ q^c)$.

Deoarece (evident) $x_1 = \sum\limits_{i=1}^{c-1} a_i q^i$, se va ajunge la relaţia $\beta_1^{(p-1)/q^2} \equiv \alpha^{(p-1)a_1/q}\ (mod\ p)$.

Se calculează atunci $\beta_1^{(p-1)/q^2}\ (mod\ p)$ şi se caută i astfel ca

$$\gamma^i \equiv \beta_1^{(p-1)/q^2}\ (mod\ p).$$

Se ia $a_1 = i$.

Dacă $c = 2$, s-a terminat; în caz contrar, se mai efectuează $c-2$ paşi pentru determinarea coeficienţilor $a_2, \ldots, a_{c-1}$.

Formal, algoritmul Pohlig - Hellman este următorul:

1.	Se calculează $\gamma^i = \alpha^{(p-1)i/q}\ (mod\ p), \quad 0 \le i \le q-1$;
2.	$\beta_0 \longleftarrow \beta$;
3.	**for** $j = 0$ **to** $c-1$ **do**
	3.1 $\delta \longleftarrow \beta_j^{(p-1)/q^{j+1}}\ (mod\ p)$;
	3.2. Se caută i astfel ca $\delta = \gamma^i$;
	3.3. $a_j \longleftarrow i$;
	3.4. $\beta_{j+1} \longleftarrow \beta_j \alpha^{-a_j q^j}\ mod\ p$.

Algoritmul calculează $a_0, a_1, \ldots, a_{c-1}$ unde $log_\alpha \beta\ (mod\ q^c) = \sum_{i=0}^{c-1} a_i q^i$.

Exemplul 9.4. *Fie* $p = 29$. *Avem* $n = p - 1 = 28 = 2^2 7^1$.

Să alegem $\alpha = 2,\ \beta = 18$ *şi ne punem problema determinării lui* $a = \log_2 18$. *Pentru aceasta se va calcula* $a\ (mod\ 4)$ *şi* $a\ (mod\ 7)$.

Să începem cu $q = 2,\ c = 2$. *Avem (toate calculele se efectuează modulo* 29*):*

$\gamma^0 = 1, \quad \gamma^1 = \alpha^{28/2} = 2^{14} = 28$, *deci* $\delta = \beta^{28/2} = 18^{14} = 28$, *de unde rezultă* $a_0 = 1$.

$\beta_1 = \beta_0 \cdot \alpha^{-1} = 9, \quad \beta_1^{28/4} = 9^7 = 28$. *Cum* $\gamma_1 = 28$, *rezultă* $a_1 = 1$.

Avem deci $a \equiv 3\ (mod\ 4)$.

Să considerăm acum $q = 7,\ c = 1$. *Vom avea (modulo* 29*):*

$\beta^{28/7} = 18^4 = 25, \quad \gamma^1 = \alpha^{28/7} = 2^4 = 16$, *apoi* $\gamma^2 = 24,\ \gamma^3 = 7,\ \gamma^4 = 25$, *deci* $a_0 = 4$ *şi* $a \equiv 4\ (mod\ 7)$.

Se obţine sistemul $a \equiv 3\ (mod\ 4), \quad a \equiv 4\ (mod\ 7)$, *de unde – folosind teorema chineză a resturilor –* $a \equiv 11\ (mod\ 28)$. *Deci,* $log_2 18 = 11$ *în* Z_{29}.

9.2.3 Algoritmul Pollard Rho

Fie p un număr prim şi $\alpha \in Z_p$ un element de ordin n $(n < p)$. Vom considera $G_\alpha \subseteq Z_p$ subgrupul ciclic generat de α. Ne punem problema calculării lui $log_\alpha \beta$, unde $\beta \in G_\alpha$ este arbitrar.

Fie $Z_p = S_1 \cup S_2 \cup S_3$ o partiţie a lui Z_p în trei mulţimi de dimensiuni aproximativ egale; considerăm funcţia

$$f : G_\alpha \times Z_n \times Z_n \longrightarrow G_\alpha \times Z_n \times Z_n$$

definită prin

$$f(x, a, b) = \begin{cases} (\beta x, a, b+1) & \text{dacă} \quad x \in S_1 \\ (x^2, 2a, 2b) & \text{dacă} \quad x \in S_2 \\ (\alpha x, a+1, b) & \text{dacă} \quad x \in S_3 \end{cases}$$

Pe baza acestei funcţii vom genera recursiv triplete (x, a, b) cu proprietatea $x = \alpha^a \beta^b$.

Fie $(1,0,0)$ tripletul iniţial (el are această proprietate). În continuare

$$(x_i, a_i, b_i) = \begin{cases} (1,0,0) & \text{dacă} \quad i = 0 \\ f(x_{i-1}, a_{i-1}, b_{i-1}) & \text{dacă} \quad i \geq 1 \end{cases}$$

În etapa a doua, se compară tripletele (x_{2i}, a_{2i}, b_{2i}) şi (x_i, a_i, b_i) până se găseşte o valoare a lui i pentru care $x_{2i} = x_i$. În acel moment,

$$\alpha^{a_{2i}}\beta^{b_{2i}} = \alpha^{a_i}\beta^{b_i}.$$

Notând $c = log_\alpha \beta$, relaţia poate fi rescrisă

$$\alpha^{a_{2i}+cb_{2i}} = \alpha^{a_i+cb_i}.$$

Cum α are ordinul n, rezultă

$$a_{2i} + cb_{2i} \equiv a_i + cb_i \ (mod\ n)$$

sau

$$c(b_{2i} - b_i) \equiv a_i - a_{2i} \ (mod\ n).$$

Dacă $cmmdc(b_{2i} - b_i, n) = 1$, atunci se poate obţine c:

$$c = (a_i - a_{2i}) \cdot (b_{2i} - b_i)^{-1} \quad (mod\ n)$$

Exemplul 9.5. *Să considerăm $p = 809$ şi $\alpha = 89$; ordinul lui α în Z_{809}^* este $n = 101$. Se verifică uşor că $\beta = 618 \in G_{89}$. Vom calcula $log_{89}618$.*

Să presupunem că alegem partiţia

$$S_1 = \{x \mid x \in Z_{809},\ x \equiv 1\ (mod\ 3)\}$$
$$S_2 = \{x \mid x \in Z_{809},\ x \equiv 0\ (mod\ 3)\}$$
$$S_3 = \{x \mid x \in Z_{809},\ x \equiv 2\ (mod\ 3)\}$$

Pentru $i = 1, 2, 3, \ldots$ se obţin următoarele triplete:

i	(x_i, a_i, b_i)	(x_{2i}, a_{2i}, b_{2i})
1	$(618, 0, 1)$	$(76, 0, 2)$
2	$(76, 0, 2)$	$(113, 0, 4)$
3	$(46, 0, 3)$	$(488, 1, 5)$
4	$(113, 0, 4)$	$(605, 4, 10)$
5	$(349, 1, 4)$	$(422, 5, 11)$
6	$(488, 1, 5)$	$(683, 7, 11)$
7	$(555, 2, 5)$	$(451, 8, 12)$
8	$(605, 4, 10)$	$(344, 9, 13)$
9	$(451, 5, 10)$	$(112, 11, 13)$
10	$(422, 5, 11)$	$(422, 11, 15)$

Deci $x_{10} = x_{20} = 422$. *Se poate calcula atunci*
$$log_{89}618 = (11-5)\cdot(11-15)^{-1}\ (mod\ 101) = 6\cdot 25\ (mod\ 101) = 49$$
(în grupul multiplicativ Z_{809}^**).*

O formalizare a algoritmului Pollard Rho pentru calculul logaritmului discret[3] este:

Algoritm Pollard Rho(Z_p, n, α, β)
1 Se defineşte partiţia $Z_p = S_1 \cup S_2 \cup S_3$;
2. $(x,a,b) \longleftarrow f(1,0,0)$, $(x_1,a_1,b_1) \longleftarrow f(x,a,b)$
3. **while** $x \neq x_1$ **do**
   3.1. $(x,a,b) \longleftarrow f(x,a,b)$;
   3.2. $(x_1,a_1,b_1) \longleftarrow f(x_1,a_1,b_1)$, $(x_1,a_1,b_1) \longleftarrow f(x_1,a_1,b_1)$;
4. **if** $cmmdc(b_1-b,n) > 1$ **then return**(Eşec)
   else return$((a-a_1)\cdot(b_1-b)^{-1}\ (mod\ n))$

procedure $f(x,a,b)$
1. **if** $x \in S_1$ **then** $f \longleftarrow (\beta\cdot x,\ a,\ (b+1)\ (mod\ n))$;
2. **if** $x \in S_2$ **then** $f \longleftarrow (x\cdot x,\ 2\cdot a\ (mod\ n),\ 2\cdot b\ (mod\ n))$;
3. **if** $x \in S_3$ **then** $f \longleftarrow (\alpha\cdot x,\ (a+1)\ (mod\ n),\ b)$;
4. **return**(f).
end procedure

În cazul $cmmdc(b_1-b,n) = d > 1$, congruenţa $c\cdot(b_1-b) \equiv a-a_1\ (mod\ n)$ are d soluţii posibile. Dacă d este destul de mic, aceste soluţii se pot afla, iar o simplă căutare exhaustivă printre ele va determina soluţia corectă.

9.2.4 Metoda de calcul a indicelui

Această metodă seamănă cu unul din cei mai buni algoritmi de descompunere în factori. Vom da doar o descriere informală a acestui algoritm.

Se foloseşte o *bază de divizori* $\mathcal{B}$ compusă din B numere prime "mici". Prima etapă constă în aflarea logaritmilor elementelor din baza $\mathcal{B}$.

În a doua etapă, folosind aceşti logaritmi, se va determina logaritmul discret al lui β.

I: Se construiesc $C = B + 10$ congruenţe modulo p de forma

$$\alpha^{x_j} \equiv p_1^{a_{ij}} p_2^{a_{2j}} \ldots p_B^{a_{Bj}}\ (mod\ p), \qquad (1 \le j \le C).$$

Cu aceste C ecuaţii de necunoscute $log_\alpha p_i$ $(1 \le i \le B)$ se încearcă aflarea unei soluţii unice modulo $(p-1)$. În caz de reuşită, primul pas este încheiat.

[3]Un algoritm similar Pollard Rho poate fi construit pentru factorizarea unui număr. Detalii se găsesc de exemplu în [53].

Problema ar fi cum să se găsească aceste C congruenţe. O metodă elementară constă din trei paşi: alegerea aleatoare a unui x, calculul lui $\alpha^x \ (mod\ p)$ şi verificarea dacă acest număr are toţi divizorii în $\mathcal{B}$.

II: Acum se poate determina $log_\alpha\beta$ cu un algoritm de tip Las Vegas. Se alege aleator un număr întreg s $(1 \le s \le p-2)$ şi se determină $\gamma = \beta\alpha^s \ (mod\ p)$.

Se încearcă apoi descompunerea lui γ în baza $\mathcal{B}$. Dacă acest lucru este posibil, se obţine o relaţie de forma

$$\beta\alpha^s \equiv p_1^{c_1}p_2^{c_2}\dots p_B^{c_B} \ (mod\ p)$$

care poate fi transformată în

$$log_\alpha\beta + s \equiv c_1 log_\alpha p_1 + \dots + c_B log_\alpha p_B \quad (mod\ (p-1)).$$

De aici - prin evaluarea membrului drept, se poate determina $log_\alpha\beta$.

Exemplul 9.6. *Fie* $p = 10007$ *şi* $\alpha = 5$ *(element primitiv). Să considerăm* $\mathcal{B} = \{2, 3, 5, 7\}$ *ca bază de divizori. Cum – evident –* $log_5 5 = 1$*, trebuie determinaţi doar trei logaritmi de bază.*

Trei numere aleatoare "norocoase" pot fi $4063, 5136, 9865$*.*

Pentru $x = 4063$ *calculăm* $5^{4063} \ (mod\ 10007) = 42 = 2 \cdot 3 \cdot 7$*, care conduce la congruenţa*

$$log_5 2 + log_5 3 + log_5 7 \equiv 4063 \ (mod\ 10006).$$

În mod similar se obţin $5^{5136} \ (mod\ 10007) = 54 = 2 \cdot 3^3$*,* $5^{9865} \ (mod\ 10007) = 189 = 3^3 \cdot 7$*.*

Pe baza lor se obţin alte două relaţii:

$$log_5 2 + 3 log_5 3 \equiv 5136 \ (mod\ 10006),$$
$$3 log_5 3 + log_5 7 \equiv 9865 \ (mod\ 10006).$$

Rezolvarea acestui sistem de trei ecuaţii în Z_{10006} *conduce la soluţia unică*

$$log_5 2 = 6578, \quad log_5 3 = 6190, \quad log_5 7 = 1301.$$

Să presupunem acum că se caută $log_5 9451$*. Dacă se generează aleator numărul* $s = 7736$*, avem* $9451 \cdot 5^{7736} \ (mod\ 10007) = 8400 = 2^4 3^1 5^2 7^1$*.*

Cum acesta se poate factoriza în $\mathcal{B}$*, avem*

$log_5 9451 = 4 log_5 2 + log_5 3 + 2 log_5 5 + log_5 7 - s = 4 \cdot 6578 + 6190 + 2 \cdot 1 + 1301 - 7736 = 6057$*, calculele fiind realizate modulo* 10006*.*

Se verifică uşor că $5^{6057} \equiv 9451 \ (mod\ 10007)$*.*

9.3 Securitatea PLD faţă de informaţii parţiale

În această secţiune vom considera un tip de atac care încearcă să determine valoarea unuia sau mai multor biţi din reprezentarea binară a logaritmilor discreţi.

Mai exact se încearcă calculul lui $L_i(\beta)$: al i-lea bit (numărând de la cel mai puţin bit semnificativ) din scrierea în binar a lui $log_\alpha\beta$ peste Z_p^*; deci $1 \le i \le \lceil log_2(p-1) \rceil$.

Afirmaţia 9.1. *$L_1(\beta)$ poate fi calculat printr-un algoritm de complexitate polinomială.*

Demonstraţie. Să considerăm funcţia $f : Z_p^* \longleftarrow Z_p^*$ definită

$$f(x) = x^2 \ (mod\, p).$$

Notăm $RP(p)$ mulţimea resturilor pătratice modulo p:

$$RP(p) = \{x \mid \exists y \in Z_p^*,\ x \equiv y^2 \ (mod\, p)\}.$$

Pe baza observaţiilor

1. $f(x) = f(p - x)$,
2. $x^2 \equiv y^2 \ (mod\, p) \quad \Longleftrightarrow \quad x = \pm y \ (mod\, p)$

rezultă $card(RP(p)) = (p-1)/2$ (deci exact jumătate din elementele lui Z_p^* sunt resturi pătratice).

Să presupunem acum că $\alpha \in Z_p$ este primitiv. Deci $\alpha^i \in RP(p)$ pentru i par. Cum $(p-1)/2$ astfel de puteri sunt distincte, rezultă

$$RP(p) = \left\{ \alpha^{2i} \,\middle|\, 0 \le i \le \frac{p-3}{2} \right\}.$$

Deci β este rest pătratic dacă şi numai dacă $log_\alpha \beta$ este par, adică $L_1(\beta) = 0$.

Conform Teoremei 8.1 (Capitolul 8), β este rest pătratic dacă şi numai dacă

$$\beta^{\frac{p-1}{2}} \equiv 1 \ (mod\, p)$$

fapt care poate fi testat cu un algoritm de complexitate polinomială. Deci putem da o formulă pentru calculul lui $L_1(\beta)$:

$$L_1(\beta) = \begin{cases} 0 & \text{dacă} \quad \beta^{(p-1)/2} \equiv 1 \ (mod\, p) \\ 1 & \text{altfel} \end{cases}$$

□

Afirmaţia 9.2. *Dacă $p - 1 = 2^s(2t + 1)$, atunci*

1. *Calculul lui $L_i(\beta)$ pentru $1 \le i \le s$ este uşor.*
2. *Orice algoritm (sau oracol) care poate calcula $L_{s+1}(\beta)$ permite rezolvarea problemei logaritmului discret în Z_p.*

Prima parte a afirmaţiei este simplă.

Vom demonstra a doua parte pentru cazul $s = 1$. Mai exact,vom arăta că dacă p este prim şi $p \equiv 3$ (mod 4), atunci orice oracol care dă $L_2(\beta)$ poate fi folosit la rezolvarea problemei logaritmului discret în Z_p.

Se ştie (algoritmul de criptare al lui Rabin, Capitolul 8) că dacă β este rest pătratic în Z_p şi $p \equiv 3$ (mod 4), atunci rădăcinile pătrate ale lui β modulo p sunt $\pm\beta^{(p+1)/4}$ ($mod\, p$).

Lema 9.2. *Dacă* $p \equiv 3 \ (mod\ 4)$ *şi* $\beta \neq 0$, *atunci* $L_1(p-\beta) = 1 - L_1(\beta)$.

Demonstraţie. Fie $\alpha^a \equiv \beta \ (mod\ p)$. Atunci $\alpha^{a+(p-1)/2} \equiv -\beta \ (mod\ p)$. Deoarece $p \equiv 3 \ (mod\ 4)$, numărul $(p-1)/2$ este impar. Deci $L_1(\beta) \neq L_1(p-\beta)$. □

Fie acum $\beta = \alpha^a$ pentru un exponent par a, necunoscut. Atunci

$$\pm\beta^{(p+1)/4} \equiv \alpha^{a/2} \ (mod\ p).$$

Cum $L_2(\beta) = L_1(\alpha^{a/2})$, valoarea $L_2(\beta)$ poate determina care din cele două variante (cu $+$ sau $-$) este corectă. Acest lucru este folosit de următorul algoritm care dă valoarea logaritmului discret $log_\alpha\beta$ (s-a presupus că valoarea $L_2(\beta)$ se poate afla, folosind de exemplu un oracol):

Algoritm aflare bit(p, α, β)
1. $x_0 \longleftarrow L_1(\beta)$;
2. $\beta \longleftarrow \beta/\alpha^{x_0} \ (mod\ p)$
3. $i \longleftarrow 1$;
4. **while** $\beta \neq 1$ **do**
    - 4.1. $x_i \longleftarrow L_2(\beta)$;
    - 4.2. $\gamma \longleftarrow \beta^{(p+1)/4} \ (mod\ p)$;
    - 4.3. **if** $L_1(\gamma) = x_i$ **then** $\beta \longleftarrow \gamma$
      else $\beta \longleftarrow p - \gamma$;
    - 4.4. $\beta \longleftarrow \beta/\alpha^{x_i} \ (mod\ p)$;
    - 4.5. $i \longleftarrow i + 1$;
5. **return**$(x_{i-1}, x_{i-2}, \ldots, x_0)$.

În final, se obţine

$$log_\alpha\beta = \sum_{j\geq 0} x_j \cdot 2^j.$$

Exemplul 9.7. *Fie* $p = 19$, $\alpha = 2$, $\beta = 6$. *Deoarece numerele sunt foarte mici, se pot determina uşor valorile pentru* L_1 *şi* L_2*; ele sunt cele din tabelul*

x	$L_1(x)$	$L_2(x)$	x	$L_1(x)$	$L_2(x)$	x	$L_1(x)$	$L_2(x)$
1	0	0	7	0	1	13	1	0
2	1	0	8	1	1	14	1	1
3	1	0	9	0	0	15	1	1
4	0	1	10	1	0	16	0	0
5	0	0	11	0	0	17	0	1
6	0	1	12	1	1	18	1	0

Pe baza acestor informaţii, aplicăm algoritmul. Se obţine:

$x_0 \longleftarrow 0, \quad \beta \longleftarrow 6, \quad i \longleftarrow 1;$

$x_1 \longleftarrow L_2(6) = 1, \quad \gamma \longleftarrow 5, \quad L_1(5) = 0 \neq x_1, \quad \beta \longleftarrow 14, \quad \beta \longleftarrow 7, \quad i \longleftarrow 2;$

$x_2 \longleftarrow L_2(7) = 1, \quad \gamma \longleftarrow 11, \quad L_1(11) = 0 \neq x_2, \quad \beta \longleftarrow 8, \quad \beta \longleftarrow 4, \quad i \longleftarrow 3;$

$x_3 \longleftarrow L_2(4) = 1, \quad \gamma \longleftarrow 17, \quad L_1(17) = 0 \neq x_3, \quad \beta \longleftarrow 2, \quad \beta \longleftarrow 1, \quad i \longleftarrow 4.$

return$(1, 1, 1, 0)$.

Deci $log_2 6 = 1110_2 = 14$.

9.4 Generalizarea sistemului de criptare El Gamal

Sistemul de criptare *El Gamal* se poate construi pentru orice grup (în locul grupului multiplicativ Z_n^*) în care problema logaritmului (definită corespunzător) este dificilă.

Fie $(G, \circ)$ un grup finit. Problema logaritmului discret (PLD) se defineşte în G astfel:

Fie $\alpha \in G$ şi $H = \{\alpha^i \mid i \geq 0\}$ subgrupul generat de α. Dacă $\beta \in H$, să se determine un a (unic) ($0 \leq a \leq card(H) - 1$) cu $\alpha^a = \beta$, unde $\alpha^a = \underbrace{\alpha \circ \alpha \circ \ldots \circ \alpha}_{a \text{ ori}}$

Definirea sistemului de criptare *El Gamal* în subgrupul H în loc de Z_n^* este uşor de realizat; anume:

Fie $(G, \circ)$ un grup şi $\alpha \in G$ pentru care PLD în $H = \{\alpha^i \mid i \geq 0\}$ este dificilă.
Fie $\mathcal{P} = G$, $\mathcal{C} = G \times G$ şi $\mathcal{K} = \{(G, \alpha, a, \beta) \mid \beta = \alpha^a\}$.
Valorile α, β sunt publice iar a este secret.
Pentru $K = (G, \alpha, a, \beta)$ şi un $k \in Z_{card(H)}$ aleator (secret), se defineşte

$$e_K(x, k) = (y_1, y_2) \text{ unde } y_1 = \alpha^k, \; y_2 = x \circ \beta^k.$$

Pentru $y = (y_1, y_2)$, decriptarea este

$$d_K(y) = y_2 \circ (y_1^a)^{-1}.$$

De remarcat că pentru criptare/decriptare nu este necesară cunoaşterea ordinului $card(H)$ de mărime al subgrupului; *Alice* poate alege aleator un k, $(0 \leq k \leq card(G) - 1)$ cu care cele două procese funcţionează fără probleme.

Se poate observa de asemenea că G nu este neapărat abelian (H în schimb este, fiind subgrup ciclic).

Să studiem acum problema logaritmului discret "generalizat". Deoarece H este subgrup ciclic, orice versiune a problemei este echivalentă cu PLD într-un grup ciclic.

În schimb, se pare că dificultatea problemei depinde mult de reprezentarea grupului utilizat.

Astfel în grupul aditiv Z_n, problema este simplă; aici exponenţierea α^a este de fapt înmulţirea cu a modulo n. Deci, PLD constă în aflarea unui număr întreg a astfel ca

$$a\alpha \equiv \beta \ (mod\ n).$$

Dacă se alege α astfel ca $cmmdc(\alpha, n) = 1$ (α este generator al grupului), α are un invers multiplicativ modulo n, care se determină uşor cu algoritmul lui Euclid extins. Atunci,

$$a = log_\alpha\beta = \beta\alpha^{-1} \ (mod\ n).$$

Să vedem cum se reprezintă PLD în grupul multiplicativ Z_p^* cu p prim. Acest grup este ciclic de ordin $p-1$, deci izomorf cu grupul aditiv Z_{p-1}. Deoarece PLD se poate rezolva uşor într-un grup aditiv, apare întrebarea dacă putem rezolva această problemă în Z_p^* reducând-o la Z_{p-1}.

Ştim că există un izomorfism $\phi : Z_p^* \longrightarrow Z_{p-1}$, deci pentru care

$$\phi(xy\ mod\ p) = (\phi(x) + \phi(y)) \quad (mod\ (p-1).)$$

În particular, $\phi(\alpha^a\ mod\ p) = a\phi(\alpha) \ \ (mod\ (p-1))$, adică

$$\beta \equiv \alpha^a \ (mod\ p) \quad \Longleftrightarrow \quad a\phi(a) \equiv \phi(\beta) \quad (mod\ (p-1)).$$

Acum, determinarea lui a se realizează cu $log_\alpha\beta = \phi(\beta)(\phi(\alpha))^{-1} \ (mod\ (p-1))$.

Deci, dacă se găseşte o metodă eficace pentru calculul izomorfismului ϕ, se obţine un algoritm eficace pentru calculul logaritmului discret în Z_p^*.

Problema este că nu se cunoaşte nici o metodă generală de construcţie a lui ϕ pentru un număr prim p oarecare. Deşi se ştie că cele două grupuri sunt izomorfe, nu există încă un algoritm eficient pentru construcţia explicită a unui astfel de izomorfism.

Această metodă se poate aplica problemei logaritmului discret într-un grup finit arbitrar. Implementările au fost realizate în general pentru $Z_p, GF(2^p)$ (unde PLD este dificilă) sau curbe eliptice.

9.5 Exerciţii

9.1. *Implementaţi algoritmul Shanks pentru aflarea logaritmului discret. Aplicaţii pentru aflarea* $log_{106}12375$ *în* Z_{24691}^* *şi* $log_6 248388$ *în* Z_{458009}^*.

9.2. *Numărul* $p = 458009$ *este prim şi* $\alpha = 2$ *are ordinul* 57251 *în* Z_p^*. *Folosind algoritmul Pollard Rho, calculaţi* $log_2 56851$ *în* Z_p^*.

Luaţi valoarea iniţială $x_0 = 1$ *şi partiţia din Exemplul 9.5.*

9.3. *Fie p un număr prim impar şi k un număr pozitiv. Grupul multiplicativ $Z^*_{p^k}$ are ordinul $p^{k-1} \cdot (p-1)$ şi este ciclic. Un generator al acestui grup este numit "element primitiv modulo p^k".*

(*a*) *Dacă α este un element primitiv modulo p, arătaţi că cel puţin unul din numerele $\alpha, \alpha + p$ este element primitiv modulo p^2.*

(*b*) *Descrieţi cum se poate poate verifica eficient că 3 este o rădăcină primitivă modulo 29 şi modulo 29^2. Arătaţi întâi că dacă α este o rădăcină primitivă modulo p şi modulo p^2, atunci ea este rădăcină primitivă modulo p^j pentru orice j întreg.*

(*c*) *Găsiţi un întreg α care este rădăcină primitivă modulo 29 dar nu este rădăcină primitivă modulo 29^2.*

(*d*) *Folosiţi algoritmul Pohlig - Hellman pentru a calcula $log_3 3344$ în Z^*_{24389}.*

9.4. *Să implementăm sistemul de criptare El Gamal în $GF(3^3)$. Polinomul $x^3 + 2x^2 + 1$ este ireductibil peste $Z_3[x]$ şi deci $GF(3^3) = Z_[x]/(x^3 + 2x^2 + 1)$. Asociem cele 26 litere ale alfabetului cu cele 26 elemente nenule ale corpului (ordonate lexicografic):*

A	$\leftrightarrow$	1	B	$\leftrightarrow$	2	C	$\leftrightarrow$	x
D	$\leftrightarrow$	$x+1$	E	$\leftrightarrow$	$x+2$	F	$\leftrightarrow$	$2x$
G	$\leftrightarrow$	$2x+1$	H	$\leftrightarrow$	$2x+2$	I	$\leftrightarrow$	x^2
J	$\leftrightarrow$	x^2+1	K	$\leftrightarrow$	x^2+2	L	$\leftrightarrow$	x^2+x
M	$\leftrightarrow$	x^2+x+1	N	$\leftrightarrow$	x^2+x+2	O	$\leftrightarrow$	x^2+2x
P	$\leftrightarrow$	x^2+2x+1	Q	$\leftrightarrow$	x^2+2x+2	R	$\leftrightarrow$	$2x^2$
S	$\leftrightarrow$	$2x^2+1$	T	$\leftrightarrow$	$2x^2+2$	U	$\leftrightarrow$	$2x^2+x$
V	$\leftrightarrow$	$2x^2+x+1$	W	$\leftrightarrow$	$2x^2+x+2$	X	$\leftrightarrow$	$2x^2+2x$
Y	$\leftrightarrow$	$2x^2+2x+1$	Z	$\leftrightarrow$	$2x^2+2x+2$			

Să presupunem că Bob foloseşte $\alpha = x$ şi $p = 11$ într-un sistem de criptare El Gamal. Apoi alege $\beta = x + 2$. Decriptaţi mesajul

$$(K,H)\ (P,X)\ (N,K)\ (H,R)\ (T,F)\ (V,Y)\ (E,H)\ ((F,A)\ (T,W)\ (J,D)\ (U,J)$$

9.5. *Fie $p = 227$. Elementul $\alpha = 2$ este primitiv în Z^*_p.*

(*a*) *Calculaţi $\alpha^{32}, \alpha^{40}, \alpha^{59}$ şi α^{156} modulo p şi apoi factorizaţi-le pentru baza de factori $\{2, 3, 5, 7, 11\}$.*

(*b*) *Folosind faptul că $log_2 2 = 1$, calculaţi $log_2 3$, $log_2 5$, $log_2 7$, $log_2 11$ folosind factorizarea anterioară.*

(*c*) *Să presupunem că vrem să calculăm $log_2 173$. Înmulţim 173 cu valoarea "aleatoare" 2^{177} (mod p). Factorizaţi rezultatul peste baza de factori dată mai sus şi determinatî $log_2 173$.*

9.6. *Implementaţi algoritmul Pohlig - Hellman de calcul al logaritmului discret în Z_p, unde p este număr prim şi α primitiv. Folosiţi programul pentru a afla $log_5 8563$ în Z_{28703} şi $log_{10} 12611$ în Z_{31153}.*

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_Pro blem_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