Capitolul 8

Sistemul de criptare RSA

8.1 Descrierea sistemului RSA

Sistemul de criptare RSA (**R**ivest - **S**hamir - **A**dleman) este în acest moment cel mai cunoscut şi utilizat sistem cu cheie publică[1]. Aceasta se datorează în primul rând modalităţii foarte simple de criptare şi decriptare, care se realizează similar – cu aceleaşi module de calcul (proprietate întâlnită la multe sisteme simetrice: Beaufort, Enigma, AES etc).

Iată în ce constă sistemul de criptare RSA:

Fie p, q numere prime impare distincte şi $n = pq$.
Indicatorul său Euler este $\phi(n) = (p-1)(q-1)$.
Fie $\mathcal{P} = \mathcal{C} = Z_n$. Se defineşte
$\mathcal{K} = \{(n, p, q, a, b) |\ n = pq,\ ab \equiv 1\ (mod\ \phi(n))\}$
Pentru $K = (n, p, q, a, b)$ se definesc $(\forall x, y \in Z_n)$:

$$e_K(x) = x^b\ (mod\ n)$$

şi

$$d_K(y) = y^a\ (mod\ n)$$

Valorile n şi b sunt publice, iar p, q şi a sunt secrete.

Deoarece $ab \equiv 1\ (mod\ \phi(n))$, avem $ab = t\phi(n) + 1$.

Atunci, pentru un $x \in Z_n^* = Z_n \setminus \{0\}$, putem scrie (toate calculele se fac în Z_n):

$$(x^b)^a \equiv x^{t\phi(n)+1} \equiv \left(x^{\phi(n)}\right)^t x \equiv 1^t x \equiv x.$$

Pentru $x = 0$ afirmaţia este banală.

Exemplul 8.1. *Să presupunem că Bob alege* $p = 101,\ q = 113$. *Atunci* $n = 11413$, $\phi(n) = 11200$. *Deoarece* $11200 = 2^6 5^2 7$, *un număr* b *poate fi utilizat ca exponent de*

[1]Sistemul este prezentat în 1977 de Ron Rivest, Adi Shamir şi Len Adleman în cadrul unui proiect de cercetare la MIT. Totuşi, după declasificarea în 1997 a unor documente din Marea Britanie, se pare că matematicianul Clifford Cocks a elaborat în 1973 un sistem echivalent, prezentat într-un document intern $GCHQ$ (Government Communications Headquarters).

criptare dacă şi numai dacă nu este divizibil cu $2, 5$ *sau* 7 *(deci Bob poate să nu factorizeze* $\phi(n)$*; este suficient să verifice dacă* $cmmdc(\phi(n), b) = 1$ *folosind algoritmul lui Euclid).*

Fie de exemplu $b = 3533$*. Avem atunci* $b^{-1} = 6597$ $(mod\ 11200)$*.*

Deci, exponentul (secret) de decriptare este $a = 6597$*.*

Bob face public $n = 11413$ *şi* $b = 3533$*.*

Dacă Alice doreşte să-i transmită lui Bob mesajul $m = 9726$*, ea calculează*

$$9726^{3533}\ (mod\ 11413) = 5761$$

şi trimite prin canal textul criptat $c = 5761$*. Când Bob primeşte acest număr, el determină*

$$5761^{6597}\ (mod\ 11413) = 9726.$$

Securitatea sistemului de criptare RSA se bazează pe ipoteza că funcţia de criptare $e_K(x) = x^b\ mod\ n$ este neinversabilă din punct de vedere al complexităţii, deci este imposibil pentru *Oscar* să o determine. Trapa secretă de care dispune *Bob* pentru decriptare este descompunerea $n = pq$. Deoarece *Bob* ştie această factorizare, el poate calcula $\phi(n) = (p-1)(q-1)$ şi apoi determina exponentul de decriptare a folosind algoritmul lui Euclid extins (a se vedea *Anexa*).

8.2 Implementarea sistemului RSA

Pentru a realiza criptarea, *Bob* trebuie să efectueze următorii paşi (fiecare din ei va fi detaliat mai târziu):

Tabelul 8.1:

1. Generează două numere prime mari $p,\ q$.
2. Calculează $n = pq$ şi $\phi(n) = (p-1)(q-1)$.
3. Alege aleator un număr b $(1 < b < \phi(n))$ astfel ca $cmmdc(b, \phi(n)) = 1$.
4. Calculează $a = b^{-1}\ mod\ \phi(n)$ folosind algoritmul lui Euclid extins.
5. Face public n şi b.

Un atac evident al sistemului constă în încercarea de factorizare a lui n. Dacă se realizează aceasta, este uşor de determinat $\phi(n) = (p-1)(q-1)$ şi de calculat exponentul de decriptare a plecând de la b.

Deci, pentru ca sistemul *RSA* să fie sigur, este necesar ca n să fie suficient de mare pentru ca factorizarea sa să fie imposibilă (din punct de vedere al complexităţii). Algoritmii de factorizare actuali pot descompune rapid numere de până la 200 cifre zecimale. Se recomandă de aceea – pentru siguranţă – să se lucreze cu numere prime p şi q de cel puţin 300 cifre fiecare, deci n va avea peste 500 cifre. Aproape toate implementările actuale ale sistemului folosesc chei de $1024 - 2048$ biţi[2].

Cu intenţia că vom reveni asupra problemelor legate de numere prime mari, să studiem întâi operaţiile necesare pentru criptare şi decriptare. Orice astfel de calcul se bazează pe o exponenţiere modulo n. Cum n este foarte mare, vom utiliza aritmetica numerelor mari pentru lucrul în Z_n, timpul de calcul necesar fiind direct proporţional cu numărul de biţi ai lui n.

Dacă n ocupă k biţi în memorie (deci $k = [log_2 n] + 1$), prin metode de calcul uzuale se ajunge la concluzia că suma a două numere de k biţi se face în $O(k)$, iar înmulţirea în $O(k^2)$. La fel şi reducerea modulo n. Deci, pentru $x, y \in Z_n$, numărul $xy \ mod \ n$ se poate determina prin calcule de complexitate $O(k^2)$. Vom numi această operaţie *multiplicare modulară*.

Să cercetăm acum *exponenţierea modulară* $x^c \ mod \ n$. O modalitate de calcul constă în efectuarea de $c-1$ multiplicări modulare – proces foarte ineficient pentru c mare, deoarece algoritmul devine de complexitate exponenţială.

Există însă un algoritm de exponenţiere rapidă, care realizează $x^c \ mod \ n$ cu complexitate $O(k^3)$ (deci polinomial). Acesta utilizează descompunerea binară a lui c,

$$c = \sum_{i=0}^{s-1} c_i 2^i$$

unde s ($s \leq k$) este numărul de biţi ai lui c, iar $c_i \in \{0, 1\}$.

Exponenţierea se face doar prin ridicări la pătrat şi maxim s înmulţiri modulare, conform algoritmului:

```
z ⟵ 1;
for i ⟵ s − 1 downto 0 do
    z ⟵ z^2 (mod n);
    if c_i = 1 then z ⟵ z · x (mod n)
```

Exemplul 8.2. *Să reluăm datele din Exemplul 8.1. Calculul lui* $9726^{3533} \ mod \ 11413$ *se efectuează cu algoritmul de sus în numai* 12 *paşi; anume:*

[2]Un număr n de maxim 256 biţi poate fi factorizat de un PC obişnuit în câteva ore, folosind un soft free. Dacă n are până la 512 biţi, el poate fi factorizat folosind o reţea de câteva sute de calculatoare, conform unei scheme prezentate în 1999. În 2003 a fost pusă sub semnul întrebării securitatea modulelor de 1024 biţi.

i	c_i	z
11	1	$1^2 \cdot 9726 = 9726$
10	1	$9726^2 \cdot 9726 = 2659$
9	0	$2659^2 = 5634$
8	1	$5634^2 \cdot 9726 = 9167$
7	1	$9167^2 \cdot 9726 = 4958$
6	1	$4958^2 \cdot 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \cdot 9726 = 10185$
2	1	$10185^2 \cdot 9726 = 105$
1	0	$105^2 = 11025$
0	1	$11025^2 \cdot 9726 = 5761$

Deci textul clar 9726 *este criptat de Alice în* 5761*.*

Pentru aplicarea sistemului de criptare RSA, trebuie generate întâi numerele prime p, q – despre care vom discuta în secţiunea următoare. Etapa a doua (din Tabelul 8.1) se efectuează evident în $O((log_2 n)^2)$. Etapele 3 şi 4 folosesc algoritmul lui Euclid extins. Ca rezultat general, calculul celui mai mare divizor comun dintre a şi b ($cmmdc(a, b)$) cu $a > b$ se poate realiza cu complexitatea $O((log_2 a)^2)$.

În general, un algoritm RSA este cam de 1000 ori mai lent decât DES pentru o implementare harwdare şi cam de 100 ori la o implementare software. În [46] se dau câteva tabele cu astfel de valori comparative, la nivelul anului 1995.

8.3 Teste de primalitate probabiliste

În realizarea sistemului de criptare RSA trebuie să generăm aleator numere prime cu număr mare de cifre. Practic, se realizează secvenţe de numere, a căror primalitate se testează, până se ajunge la un număr prim. Pentru teste se folosesc algoritmi probabilişti, al căror avantaj este rapiditatea (complexitatea lor este $log\ n$) dar care pot afirma uneori primalitatea unor numere care nu sunt prime. Aceste erori se pot reduce la o marjă acceptabilă prin multiplicarea testelor.

Problema generării aleatoare este posibilă din următorul considerent. Un rezultat din teoria numerelor (numit *Teorema rarefierii numerelor prime*) afirmă că sunt circa $n/log\ n$ numere prime mai mici decât n. Astfel, pentru un modul de 512 biţi, un număr p de 256 biţi are o probabilitate $1/logp \approx 1/177$ de a fi prim. Deci se fac în medie cam 177 generări de numere p pentru a obţine un număr prim (dacă se foloseşte şi faptul că se generează numai numere impare, aceasta reduce la jumătate numărul de încercări). Rezultă că este practic să se construiască numere mari, care sunt *probabil prime*, pe baza cărora să se realizeze criptarea RSA. Vom detalia acest procedeu.

Definiţia 8.1. *O problemă de decizie este o problemă care pune o întrebare al cărui răspuns este dicotomic (Da/Nu).*

Un algoritm probabilist este un algoritm care foloseşte numere aleatoare.

Definiţia 8.2. *Un algoritm Monte - Carlo pozitiv este un algoritm probabilist care rezolvă o problemă de decizie în care orice răspuns pozitiv este corect, dar pentru care un răspuns negativ poate fi incorect.*

În mod similar se defineşte algoritmul Monte - Carlo negativ.

Un algoritm Monte - Carlo pozitiv are o probabilitate de eroare ϵ dacă pentru orice problemă al cărei răspuns ar trebui să fie pozitiv, algoritmul dă un răspuns negativ cu probabilitatea cel mult ϵ.

Problema de decizie folosită aici, numită **Problema de descompunere** este

Fiind dat un număr întreg n, se poate el descompune în produs de alte numere suprau-nitare mai mici ?

Vom prezenta în această secţiune doi algoritmi de tip Monte Carlo pozitiv care rezolvă această problemă de de decizie.

8.3.1 Algoritmul Solovay - Strassen

Să reamintim întâi câteva noţiuni matematice:

Definiţia 8.3. *Fie $p \geq 3$ număr prim şi $a \in Z_p^*$. Spunem că a este rest (reziduu) pătratic modulo p dacă ecuaţia $x^2 \equiv a \ (mod\ p)$ are soluţie în Z_p.*

În caz contrar, un număr $a \neq 0$ nu este rest pătratic.

Exemplul 8.3. *Resturile pătratice modulo 11 sunt $1, 3, 4, 5, 9$ Aceasta deoarece în Z_{11} avem $(\pm 1)^2 = 1,\ (\pm 5)^2 = 3,\ (\pm 2)^2 = 4,\ (\pm 4)^2 = 5,\ (\pm 3)^2 = 9$.*

Problema resturilor pătratice constă în a decide dacă un număr n dat este sau nu un rest pătratic. Un algoritm determinist pentru rezolvarea acestei probleme se bazează pe

Teorema 8.1. *(Criteriul lui Euler). Dacă $p \geq 3$ este prim, un număr a este rest pătratic modulo p dacă şi numai dacă*

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

Demonstraţie. Să presupunem $a \equiv x^2 \ (mod\ p)$. Cum $x^{p-1} \equiv 1 \ (mod\ p)$ (Teorema lui Fermat) pentru $x \not\equiv 0 \ (mod\ p)$,vom avea

$$a^{\frac{p-1}{2}} \equiv (x^2)^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \ (mod\ p).$$

Invers, fie $a^{\frac{p-1}{2}} \equiv 1 \ (mod\ p)$ şi $b \in Z_p$ un element primitiv (de ordin $p-1$). Atunci $a \equiv b^i \ (mod\ p)$ pentru un anumit i. Calculăm

$$1 \equiv a^{\frac{p-1}{2}} \equiv (b^i)^{\frac{p-1}{2}} \equiv b^{\frac{i(p-1)}{2}} \ (mod\ p).$$

Ordinul $p-1$ al lui b va divide $i(p-1)/2$. Deci i este par şi rădăcinile pătrate ale lui a sunt $\pm b^{i/2}$. □

Definiţia 8.4. *Dacă $p \geq 3$ este prim, pentru orice număr $a \geq 0$ se defineşte simbolul Legendre prin*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{dacă } a \equiv 0 \ (mod\ p) \\ 1 & \text{dacă } a \text{ este rest pătratic modulo } p \\ -1 & \text{dacă } a \text{ nu este rest pătratic modulo } p \end{cases}$$

Teorema 8.1 asigură că $a^{(p-1)/2} \equiv 1 \ (mod\ p)$ dacă şi numai dacă a este rest pătratic modulo p. Dacă a este multiplu de p, evident $a^{(p-1)/2} \equiv 0 \ (mod\ p)$. În sfârşit, dacă a nu este rest pătratic modulo p, avem $a^{(p-1)/2} \equiv -1 \ (mod\ p)$ deoarece $a^{p-1} \equiv 1$, $a^{(p-1)/2} \not\equiv 1 \ (mod\ p)$ şi -1 este singura rădăcină pătrată diferită de 1 modulo p. Este deci adevărată teorema următoare:

Teorema 8.2. *Dacă p este număr prim impar, atunci*

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \ (mod\ p)$$

Simbolul lui Legendre se poate generaliza astfel:

Definiţia 8.5. *Fie $n = p_1^{e_1} \dots p_k^{e_k}$ un număr impar descompus în factori primi. Dacă $a \geq 0$ este un număr întreg, se defineşte simbolul Jacobi prin*

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}$$

Exemplul 8.4. *Să calculăm simbolul Jacobi $\left(\frac{6278}{9975}\right)$. Descompunerea în factori primi a lui 9975 este $9975 = 3 \cdot 5^2 \cdot 7 \cdot 19$. Avem atunci*

$$\left(\frac{6278}{9975}\right) = \left(\frac{6278}{3}\right)\left(\frac{6278}{5}\right)^2\left(\frac{6278}{7}\right)\left(\frac{6278}{19}\right) = \left(\frac{2}{3}\right)\left(\frac{3}{5}\right)^2\left(\frac{6}{7}\right)\left(\frac{8}{19}\right) =$$
$$= (-1)(-1)^2(-1)(-1) = -1$$

Fie $n > 1$ un număr impar.

Dacă n este prim, atunci pentru orice a, avem $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \ (mod\ n)$.

Invers, dacă n nu este prim, este posibil ca egalitatea de sus să fie falsă. Dacă congruenţa se verifică, spunem că n este *număr Euler pseudo - prim* pentru baza a.

De exemplu, 91 este pseudo-prim pentru baza 10 deoarece

$$\left(\frac{10}{91}\right) = -1 = 10^{45} \ (mod\ 91).$$

Putem enunţa acum testul de primalitate Solovay - Strassen pentru un număr impar n:

1. Se generează aleator un număr $a \in Z_n^*$;
2. $x \longleftarrow \left(\frac{a}{n}\right)$;
3. **if** $x = 0$ **then** "n nu este prim"
4. $y \longleftarrow a^{\frac{n-1}{2}} \ (mod\ n)$;
5. **if** $x \equiv y \ (mod\ n)$ **then** "n este prim",
   else "n nu este prim".

Pentru evaluarea $a^{\frac{n-1}{2}} \ (mod\ n)$ se poate folosi un algoritm de complexitate $\mathcal{O}((log\ n)^3)$. Problema este cum putem evalua simbolul Jacobi $x \longleftarrow \left(\frac{a}{n}\right)$ fără a factoriza pe n (altfel ne învârtim într-un cerc vicios !!).

Acest lucru se poate realiza folosind câteva proprietăţi. Anume:

Lema 8.1. *Fie n un întreg pozitiv impar. Atunci*

1. *Dacă $x \equiv y \ (mod\ n)$ atunci $\left(\frac{x}{n}\right) = \left(\frac{y}{n}\right)$;*
2. $\left(\frac{2}{n}\right) = \begin{cases} 1 & dacă \quad n \equiv \pm 1 \ (mod\ 8) \\ -1 & dacă \quad n \equiv \pm 3 \ (mod\ 8) \end{cases}$
3. $\left(\frac{x \cdot y}{n}\right) = \left(\frac{x}{n}\right) \cdot \left(\frac{y}{n}\right)$;

Lema 8.2. *Fie m, n două numere întregi pozitive impare. Atunci*

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & dacă \quad m \equiv 3 \ (mod\ 4), \ \ n \equiv 3 \ (mod\ 4) \\ \left(\frac{n}{m}\right) & altfel \end{cases}$$

Lăsăm ca exerciţiu demonstraţiile celor două leme.

Exemplul 8.5. *Să calculăm simbolul Jacobi $\left(\frac{7411}{9283}\right)$. Vom avea succesiv:*

$$\left(\frac{7411}{9283}\right) = -\left(\frac{9283}{7411}\right) = -\left(\frac{1872}{7411}\right) = -\left(\frac{2}{7411}\right)^4 \left(\frac{117}{7411}\right) = -\left(\frac{117}{7411}\right) =$$
$$= -\left(\frac{7411}{117}\right) = -\left(\frac{40}{117}\right) = -\left(\frac{2}{117}\right)^3 \left(\frac{5}{117}\right) = \left(\frac{5}{117}\right) = \left(\frac{117}{5}\right) = \left(\frac{2}{5}\right) = -1$$

O analiză sumară arată că se poate calcula simbolul Jacobi $\left(\frac{m}{n}\right)$ folosind cel mult $\mathcal{O}(log\ n)$ reduceri modulare, fiecare în timp $\mathcal{O}((log\ n)^2)$. Deci complexitatea poate fi estimată la $\mathcal{O}((log\ n)^3)$[3].

Se poate arăta că numărul de baze a pentru care un număr neprim n este pseudo - prim Euler, este cel mult $n/2$. Aceasta duce la concluzia că testul de primalitate Solovay - Strassen este un algoritm Monte Carlo pozitiv pentru problema de descompunere, cu probabilitate de eroare 1/2.

Un studiu referitor la complexitatea aplicării acestui test de primalitate şi a probabilităţii de eroare se poate găsi în [53].

8.3.2 Algoritmul Miller - Rabin

Acest algoritm este cunoscut şi sub numele de *testul de tare pseudo - primalitate*. Forma sa este:

1. Se descompune $n-1 = 2^k m$ unde m este impar;
2. Se alege aleator întregul $a \in [2, n-2]$;
3. $b \longleftarrow a^m\ (mod\ n)$
4. **if** $b \equiv 1\ (mod\ n)$ **then** "n este prim", Stop;
5. **for** $i \longleftarrow 0$ **to** $k-1$ **do**
   5.1. **if** $b \equiv -1\ (mod\ n)$ **then** "n este prim", Stop,
   else $b \longleftarrow b^2\ (mod\ n)$

6 "n nu este prim", Stop

Evident, algoritmul este polinomial, de complexitate $O((log\ n)^3)$.

Teorema 8.3. *Algoritmul Miller - Rabin este un algoritm Monte Carlo pozitiv pentru problema de descompunere.*

Demonstraţie. Să presupunem prin absurd că algoritmul răspunde că un număr prim n se poate descompune, adică $a^m \not\equiv 1\ (mod\ n)$. Vom urmări şirul de valori pe care le ia b. Cum la fiecare iterare b este ridicat la pătrat, acest şir este $a^m, a^{2m}, \ldots, a^{2^{k-1}m}$.

Vom avea deci

$$a^{2^i m} \not\equiv -1\ (mod\ n) \text{ pentru } 0 \leq i \leq k-1.$$

[3]O analiză mai detaliată poate reduce această complexitate la $\mathcal{O}((log\ n)^2)$.

Deoarece n este prim, teorema lui Fermat dă $a^{2^k m} \equiv 1 \ (mod\ n)$. Deci $a^{2^{k-1}m}$ este o rădăcină pătrată a lui 1 modulo n.

Din faptul că n este prim, singurele rădăcini pătrate ale lui 1 sunt ± 1. Această afirmaţie se poate arăta astfel:

x este rădăcină pătrată a lui 1 modulo n dacă şi numai dacă $n|(x-1)(x+1)$. Cum n este prim, avem $n|(x-1)$ (deci $x \equiv 1 \ (mod\ n)$) sau $n|(x+1)$ (adică $x \equiv -1 \ (mod\ n)$).

Cum prin ipoteză $a^{2^{k-1}m} \not\equiv -1 \ (mod\ n)$, avem $a^{2^{k-1}m} \equiv 1 \ (mod\ n)$.

Atunci $a^{2^{k-2}m}$ trebuie să fie rădăcină pătrată a lui 1, diferită de -1, deci

$$a^{2^{k-2}m} \equiv 1 \ (mod\ n).$$

Procedând iterativ, se ajunge la $a^m \equiv 1 \ (mod\ n)$, ceea ce contrazice faptul că algoritmul nu s-a oprit la Pasul 4. □

Dacă n este un număr impar neprim, atunci maxim 1/4 din numerele $a \in Z_n^*$ conduc la un rezultat fals. În [38] se apreciază că numărul maxim de astfel de valori este $\phi(n)/4$, pentru $n \neq 9$.

De exemplu, pentru $n = 91$ (neprim), mulţimea valorilor a pentru care algoritmul dă răspuns incorect este $\{9, 10, 12, 16, 17, 22, 29, 38, 53, 62, 69, 74, 75, 79, 81, 82\}$. Pentru $n = 105$ orice valoare a lui a conduce la un rezultat corect.

Deci, algoritmul Miller-Rabin este un algoritm Monte-Carlo pozitiv de probabilitate $\epsilon = 1/4$.

În general se consideră că testul Miller - Rabin este mai bun decât Solovay - Strassen. Câteva motive:

1. Solovay - Strassen este mai complex computaţional.

2. Implementarea lui Solovay - Strasen este mai dificilă din cauza calculului simbolului Jacobi.

3. Probabilitatea de eroare pentru Solovay - Strasen este 1/2, pe când la Miller - Rabin ea se reduce la 1/4.

4. Deoarece orice valoare a lui a pentru care testul Miller - Rabin este greşit este un număr Euler pseudo-prim (vezi Exerciţiul 8.6), un test Miler - Rabin nu este niciodată inferior unui test Solovay - Strasen.

Deoarece orice implementare a unui sistem RSA trebuie însoţită de un generator de numere prime mari, sunt necesare construcţii care să genereze rapid astfel de numere. Bruce Schneier propune următoarea variantă ([46]):

1. Se generează un număr aleator p de n biţi.

2. Se verifică dacă primul şi ultimul bit sunt 1.

3. Se verifică dacă p nu este divizibil cu numere prime mici $(3, 5, 7, 11, \ldots)$[4].

4. Se aplică testul Miller - Rabin cu o valoare aleatoare a. Dacă p trece testul, se ia altă valoare pentru a. Cinci teste sunt suficiente. Pentru viteză, se recomandă să se ia valori mici pentru a. Dacă p eşuează la unul din cele cinci teste, se reia algoritmul.

Se apreciază că utilizarea pasului 3, cu o testare a tuturor numerelor prime până la 256 elimină aproape 80% din cazurile nefavorabile.

8.4 Securitatea sistemului RSA

Vom trece în revistă câteva modalităţi de atac ale sistemelor de criptare RSA.

Ca o primă observaţie, RSA nu rezistă la un atac de tipul *man-in-the middle*, strategia fiind cea prezentată în cazul general al sistemelor de criptare cu cheie publică. De aceea, un sistem RSA este însoţit permanent de un certificat generat conform unui protocol PKI (Public Key Infrastructure) şi – bineînţeles – de un generator de numere prime.

8.4.1 Informaţii despre p şi q

Evident, cunoaşterea lui $\phi(n)$ este suficientă pentru spargerea sistemului. În acest caz, totul se reduce la rezolvarea în $N \times N$ a sistemului

$$\begin{cases} pq &= n \\ (p-1)(q-1) &= \phi(n) \end{cases}$$

sau - după substituţie - a ecuaţiei

$$X^2 - (n - \phi(n) + 1)X + n = 0$$

Deci, dacă *Oscar* determină $\phi(n)$, el poate factoriza n şi sparge sistemul.

Cu alte cuvinte, calculul lui $\phi(n)$ nu este mai simplu decât factorizarea lui n.

De asemenea, o slăbiciune constă în alegerea unor numere p, q prime apropiate unul de altul. În acest caz (cu $p > q$), vom avea $(p-q)/2$ un număr foarte mic, iar $(p+q)/2$ un număr foarte apropiat de $\sqrt{n}$. În plus,

$$\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4},$$

deci membrul stâng este pătrat perfect.

Atunci, pentru factorizarea lui n se testează toate numerele întregi $x > \sqrt{n}$ până se găseşte unul astfel încât $x^2 - n$ este pătrat perfect; fie acesta y^2. Atunci vom avea imediat $p = x + y,\ q = x - y$.

[4]Multe implementări testează divizibilitatea cu numerele prime mai mici decât 256. Eficienţa este crescută dacă se merge până la 2000

Exemplul 8.6. *Pentru* $n = 97343$ *se găseşte* $\sqrt{n} = 311,998$. *Apoi* $312^2 - n = 1$, *ceea ce conduce la factorizarea* $p = 313,\ q = 311$.

Deci, în general este recomandabil ca cele două numere prime p şi q să difere – în reprezentarea binară – prin numărul de biţi.

8.4.2 Exponentul de decriptare

Factorizarea modulului ştiind exponentul de decriptare

Dacă există un algoritm care calculează exponentul de decriptare a fără a-l cunoaşte pe $\phi(n)$, acesta poate fi utilizat ca *oracol*[5] într-un algoritm probabilist care descompune n. Deci, se poate spune că dacă a este descoperit, secretul factorizării lui n este compromis; atunci *Bob* va trebui să schimbe nu numai exponentul de decriptare, ci şi modulul n.

Algoritmul de descompunere care va fi descris este de tip Las Vegas.

Definiţia 8.6. *Fie* ϵ $(0 \leq \epsilon < 1)$. *Un algoritm tip Las Vegas este un algoritm probabilist care, pentru orice apariţie a unei probleme, poate oferi un răspuns – totdeauna corect – sau poate eşua şi să nu dea nici un răspuns, cu probabilitate* ϵ.

Observaţia 8.1. *Un algoritm Las Vegas poate să nu dea răspuns; dar dacă dă – acest răspuns este sigur corect. Algoritmii Monte Carlo în schimb dau totdeauna răspuns, deşi acesta uneori este incorect.*

Deci, dacă avem un algoritm Las Vegas pentru rezolvarea unei probleme, putem să îl apelăm de mai multe ori, până se obţine un răspuns. Probabilitatea ca el să nu răspundă la m tentative consecutive este ϵ^m.

Să considerăm un algoritm ipotetic **A** care calculează exponentul de decriptare a plecând de la exponentul de criptare b. Se poate descrie atunci un algoritm Las Vegas care utilizează **A** ca oracol. El este bazat pe studiul rădăcinilor pătrate ale unităţii modulo n, când $n = pq$, p şi q fiind numere prime impare.

În acest caz $x^2 \equiv 1\ (mod\ p)$ are ca singure soluţii $x \equiv \pm 1\ (mod\ p)$. La fel, $x^2 \equiv 1\ (mod\ q)$ are soluţiile $x \equiv \pm 1\ (mod\ q)$.

Din Teorema chineză a resturilor (a se vedea Anexa 2) rezultă că congruenţa $x^2 \equiv 1\ (mod\ n)$ este echivalentă cu $x^2 \equiv 1\ (mod\ p)$ şi $x^2 \equiv 1\ (mod\ q)$.

Vom avea deci patru rădăcini pătrate ale unităţii modulo n, care pot fi calculate cu Teorema chineză a resturilor. Două sunt soluţiile *triviale* $\pm 1\ (mod\ n)$, iar celelalte – numite *netriviale* – sunt opuse modulo n.

Exemplul 8.7. *Fie* $n = 403 = 13 \cdot 31$. *Cele patru rădăcini pătrate ale lui* 1 *modulo* 403 *sunt* $1, 92, 311$ *şi* 402.

[5] program care răspunde numai cu *Da/Nu* la o întrebare - tip a utilizatorului

Să presupunem acum că x este o rădăcină pătrată netrivială a lui 1 modulo n, deci o soluţie a ecuaţiei $x^2 \equiv 1 \ (mod\ n)$. Avem

$$n|(x-1)(x+1)$$

Dar n nu poate divide nici unul din factorii din membrul drept. Deci va trebui ca $cmmdc(x+1, n) = p$, $cmmdc(x-1, n) = q$ – sau invers – $cmmdc(x+1, n) = q$, $cmmdc(x-1, n) = p$. Acest cel mai mare divizor comun se poate calcula fără a şti descompunerea lui n, aplicând algoritmul de mai jos, care foloseşte **A** ca oracol:

1. Se generează aleator $w \in Z_n^*$, $w \neq 1$;
2. $x \longleftarrow cmmdc(w, n)$;
3. **if** $x > 1$ **then** Stop (cu $p = x$ sau $q = x$);
4. $a \longleftarrow \mathbf{A}(b)$;
5. Se descompune $ab - 1 = 2^s r$, r impar;
6. $v \longleftarrow w^r \ (mod\ n)$;
7. **if** $v \equiv 1 \ (mod\ n)$ **then** Stop (eşec);
8. **while** $v \not\equiv 1 \ (mod\ n)$ **do**
   - 8.1. $v_0 \longleftarrow v$;
   - 8.2. $v \longleftarrow v^2 \ (mod\ n)$;
9. **if** $v_0 \equiv -1 \ (mod\ n)$ **then** Stop (eşec);
   else $x \longleftarrow cmmdc(v_0 + 1, n)$, Stop ($p = x$ sau $q = x$).

Deci, cunoaşterea unei rădăcini pătrate netriviale a lui 1 modulo n determină descompunerea lui n printr-un calcul de complexitate polinomială.

Exemplul 8.8. *Fie* $n = 89855713$, $b = 34986517$, $a = 82330933$ *şi să considerăm că s-a generat aleator* $w = 5$. *Vom avea:*

$$ab - 1 = 2^3 \cdot 360059073378795.$$

La pasul 6 *se obţine* $v = 85877701$, *iar la pasul* 8.2, $v = 1$. *La pasul* 9 *se va obţine atunci* $cmmdc(85877702, n) = 9103$.

Acesta este un factor al lui n; *celălalt este* $n/9103 = 9871$.

Trebuie demonstrată următoarea afirmaţie:

Afirmaţia 8.1. *Procedeul descris este un algoritm.*

Demonstraţie. Ca o primă observaţie, dacă există suficientă şansă şi w este multiplu de p sau q, atunci el se factorizează imediat (pasul 2).

Dacă w este prim cu n, atunci se calculează succesiv $w^r, w^{2r}, \ldots$ prin ridicări succesive la pătrat, până se ajunge la un t cu $w^{2^t r} \equiv 1 \ (mod\ n)$. Deoarece $ab - 1 = 2^s r \equiv 0 \ (mod\ \phi(n))$, se ştie că $w^{2^s r} \equiv 1 \ (mod\ n)$. Deci bucla **while** va efectua maxim s iteraţii.

La sfârşitul buclei se va găsi o valoare $v_0 \not\equiv 1 \ (mod\ n)$ cu $v_0^2 \equiv 1 \ (mod\ n)$. Dacă $v_0 \equiv -1 \ (mod\ n)$, algoritmul eşuează; altfel, v_0 este o rădăcină pătrată netrivială a lui 1 modulo n care – la pasul 12 – permite descompunerea lui n. □

Se poate arăta ([53]) că acest algoritm se termină cu succes cu probabilitate 1/2.

Atacul lui Wiener

În [54] este dezvoltat un atac asupra sistemului de criptare RSA în care exponentul de decriptare a este mic; mai exact, trebuie verificate condiţiile

$$3a < n^{1/4}, \qquad q < p < 2q.$$

Deci, dacă n are j biţi, atunci atacul va fi eficient pentru orice sistem de criptare RSA în care a are mai puţin de $j/4 - 1$ biţi, iar p şi q au valori suficient de apropiate[6].

Din condiţia $a \cdot b \equiv 1 \ (mod\ \phi(n))$ rezultă că există un număr întreg t astfel ca

$$a \cdot b - t \cdot \phi(n) = 1.$$

Relaţia se poate rescrie

$$\left| \frac{b}{\phi(n)} - \frac{t}{a} \right| = \frac{1}{a \cdot \phi(n)}$$

Pe de-altă parte, din $n = p \cdot q > q^2$ rezultă $q < \sqrt{n}$; deci

$$0 < n - \phi(n) = p \cdot q - (p-1) \cdot (q-1) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}$$

Pe baza acestor relaţii, putem evalua

$$\left| \frac{b}{n} - \frac{t}{a} \right| = \left| \frac{a \cdot b - t \cdot n}{a \cdot n} \right| = \left| \frac{1 + t \cdot (\phi(n) - n)}{a \cdot n} \right| < \frac{3t\sqrt{n}}{a \cdot n} = \frac{3 \cdot t}{a\sqrt{n}}$$

Deoarece $t < a$ (evident), vom avea $3 \cdot t < 3 \cdot a < n^{1/4}$ şi deci

$$\left| \frac{b}{n} - \frac{t}{a} \right| < \frac{1}{a \cdot n^{1/4}} < \frac{1}{3 \cdot a^2}.$$

Rezultă că valoarea fracţiei t/a este foarte apropiată de valoarea lui b/n. Din teoria fracţiilor continue se ştie că orice aproximare suficient de bună a lui b/n este una din convergenţele dezvoltării în fracţie continuă a lui b/n. Să descriem acest procedeu.

Definiţia 8.7. *O fracţie continuă (finită) este un m-tuplu $[q_1, q_2, \ldots, q_m]$ de numere naturale care reprezintă notarea expresiei*

$$q_1 + \frac{1}{q_2 + \frac{1}{q_3 + \ldots + \frac{1}{q_m}}}$$

[6]Bob poate fi tentat să aleagă astfel de parametri, pentru creşterea vitezei de decriptare; reamintim, RSA este un sistem relativ lent.

Fie a, b două numere întregi pozitive prime între ele şi $(q_1, q_2, \ldots, q_m)$ secvenţa câturilor obţinute prin aplicarea algoritmului lui Euclid. Se verifică uşor că $\frac{a}{b} = [q_1, q_2, \ldots, q_m]$. Vom spune că $[q_1, q_2, \ldots, q_m]$ *este dezvoltarea în fracţie continuă a lui* a/b.

Acum, pentru fiecare j $(1 \leq j \leq m)$ definim $C_j = [q_1, q_2, \ldots, q_j]$ ca fiind a j-a convergenţă a lui $[q_1, q_2, \ldots, q_m]$. Fiecare C_j se poate scrie ca un număr raţional c_j/d_j, unde valorile c_j şi d_j se pot defini recursiv astfel:

$$c_j = \begin{cases} 1 & \text{dacă } j = 0 \\ q_1 & \text{dacă } j = 1 \\ q_j \cdot c_{j-1} + c_{j-2} & \text{dacă } j \geq 2 \end{cases} \qquad d_j = \begin{cases} 0 & \text{dacă } j = 0 \\ 1 & \text{dacă } j = 1 \\ q_j \cdot d_{j-1} + d_{j-2} & \text{dacă } j \geq 2 \end{cases}$$

Exemplul 8.9. *Să dezvoltăm numărul raţional* $34/99$ *în fracţie continuă. Folosind algoritmul lui Euclid se obţine* $[0, 2, 1, 10, 3]$*, care este notarea fracţiei*

$$\frac{34}{99} = 0 + \frac{1}{2 + \frac{1}{1 + \frac{1}{10 + \frac{1}{3}}}}$$

Convergenţele acestei fracţii sunt:

$$\begin{aligned} [0] &= 0 \\ [0, 2] &= 1/2 \\ [0, 2, 1] &= 1/3 \\ [0, 2, 1, 10] &= 11/32 \\ [0, 2, 1, 10, 3] &= 34/99 \end{aligned}$$

Este adevărată următoarea teoremă ([54],[53]):

Teorema 8.4. *Dacă* $cmmdc(a, b) = cmmdc(c, d) = 1$ *şi*

$$\left| \frac{a}{b} - \frac{c}{d} \right| < \frac{1}{2 \cdot d^2}$$

atunci c/d *este una din convergenţele dezvoltării în fracţie continuă a lui* a/b.

Să revenim acum la sistemul de criptare RSA. În condiţiile $3a < n^{1/4}$ şi $q < p < 2q$, putem da următorul algoritm de factorizare a lui n:

1. Plecând de la n şi b (publice), se află dezvoltarea în fracţie continuă a lui b/n (folosind algoritmul lui Euclid).
2. Se parcurg pe rând convergenţele acestei dezvoltări. Dacă există convergenţa t/a care verifică $t|(a \cdot b - 1)$, se calculează $\phi(n) = \dfrac{a \cdot b - 1}{t}$.
3. Cu n şi $\phi(n)$ se află p şi q, conform metodei din secţiunea 8.4.1.

Dacă sunt îndeplinite ipotezele de la începutul acestui paragraf, Teorema 8.4 asigură existenţa unei convergenţe care satisface pasul 2 al algoritmului.

Ţinând cont de observaţiile anterioare, algoritmul lui Wiener poate fi detaliat:

Intrare: $[q_1, q_2, \ldots, q_m]$ – dezvoltarea în fracţie continuă a lui b/n.
Algoritm:
1 $c_0 \longleftarrow 1, \quad c_1 \longleftarrow q_1, \quad d_0 \longleftarrow 0, \quad d_1 \longleftarrow 1;$
2. **for** $j \longleftarrow 1$ **to** m **do**
  2.1. **if** $c_j | (d_j \cdot b - 1)$ **then**
    2.1.1. $m \longleftarrow (d_j \cdot b - 1)/c_j;$
    2.1.2. Fie p, q rădăcinile ecuaţiei $x^2 - (n - m + 1)x + n = 0$
    2.1.3. **if** $p, q \in Z_n$ **then return**(p, q);
  2.2. $j \longleftarrow j + 1;$
  2.3. $c_j \longleftarrow q_j \cdot c_{j-1} + c_{k-2}, \qquad d_j \longleftarrow q_j \cdot d_{j-1} + d_{j-2};$
3. **return("eşec")**;

Exemplul 8.10. *Să presupunem că* $n = 160523347,\ b = 60728973$. *Dezvoltarea în fracţie continuă a lui* b/n *este*

$$[0, 2, 1, 1, 1, 4, 12, 102, 1, 1, 2, 3, 2, 2, 36]$$

Primele convergenţe sunt: $0,\ \frac{1}{2},\ \frac{1}{3},\ \frac{2}{5},\ \frac{3}{8},\ \frac{14}{37} \ldots$

Primele cinci convergenţe nu verifică condiţia de divizibilitate. Pentru $14/37$ *avem însă:*

$$m = \frac{37 \cdot 60728973 - 1}{14} = 160498000.$$

Rezolvând ecuaţia $x^2 - 25348x + 160523347 = 0$ *obţinem rădăcinile* 12347 *şi* 13001. *Deci avem factorizarea*

$$160523347 = 12347 \cdot 13001$$

8.4.3 Informaţie parţială despre textul clar

Să studiem puţin informaţia din textul clar care ar putea *trăda* sistemul de criptare RSA. Ştiind că $y = e_K(x)$, vom considera două exemple de informaţie parţială dată de y despre x:

1. $par(y)$ - dă valoarea ultimului bit din scrierea binară a lui x;

2. $jum(y)$ - va da 0 dacă $0 \leq x < n/2$, 1 dacă $n/2 \leq x \leq n - 1$.

Vom arăta ([45]) că orice algoritm care poate calcula $par(y)$ sau $jum(y)$ poate fi utilizat ca oracol pentru regăsirea textului clar x. Altfel spus, a calcula una din aceste funcţii este la fel de dificil cu a decripta tot textul y.

Faptul că cele două funcţii sunt polinomial echivalente rezultă din

$$jum(y) = par(y \cdot e_K(2) \ mod \ n) \qquad par(y) = jum(y \cdot e_K(2^{-1}) \ mod \ n)$$

şi din relaţia $e_K(x_1x_2) = e_K(x_1)e_K(x_2)$.

Să arătăm acum cum se poate calcula $x = d_K(y)$ cu ajutorul unui oracol care dă valoarea $jum(y)$:

1. $k \longleftarrow [log_2 n]$;
2. **for** $i = 0$ **to** k **do**
   - 2.1. $y_i \longleftarrow jum(y)$
   - 2.2. $y \longleftarrow (y \cdot e_K(2)) \ mod \ n$
3. $jos \longleftarrow 0$;
4. $sus \longleftarrow n$;
5. **for** $i = 0$ **to** k **do**
   - 5.1. $mijloc \longleftarrow (jos + sus)/2$;
   - 5.2. **if** $y_i = 1$ **then** $jos \longleftarrow mijloc$ **else** $sus \longleftarrow mijloc$
6. $x \longleftarrow [sus]$

La pasul 2 se calculează $y_i = jum(y \cdot (e_K(2))^i) = jum(e_K(x \cdot 2^i))$ pentru $0 \leq i \leq [log_2 \ n]$. Se observă că

$jum(e_K(x)) = 0 \Longleftrightarrow x \in \left[0, \frac{n}{2}\right)$

$jum(e_K(2x)) = 0 \Longleftrightarrow x \in \left[0, \frac{n}{4}\right) \cup \left[\frac{n}{2}, \frac{3n}{4}\right)$

$jum(e_K(4x)) = 0 \Longleftrightarrow x \in \left[0, \frac{n}{8}\right) \cup \left[\frac{n}{4}, \frac{3n}{8}\right) \cup \left[\frac{n}{2}, \frac{5n}{8}\right) \cup \left[\frac{3n}{4}, \frac{7n}{8}\right)$, etc.

În acest mod, x se poate localiza printr-o căutare binară, realizată la paşii $7 - 11$.

Exemplul 8.11. *Fie $n = 1457$, $b = 779$, iar textul criptat este $y = 722$. Calculăm $e_K(2) = 946$. Să presupunem că oracolul jum din pasul 2.1. dă următoarele răspunsuri:*

i	0	1	2	3	4	5	6	7	8	9	10
y_i	1	0	1	0	1	1	1	1	1	0	0

Căutarea binară este realizată în tabelul:

i	jos	$mijloc$	sus
0	0,00	728,50	1457,00
1	728,50	1092,75	1457,00
2	728,50	910,62	1092,75
3	910,62	1001,69	1092,75
4	910,62	956,16	1001,69
5	956,16	978,92	1001,69
6	978,92	990,30	1001,69
7	990,30	996,00	1001,69
8	996,00	998,84	1001,69
9	998,84	1000,26	1001,69
10	998,84	999,55	1000,26
	998,84	999,55	999,55

Textul clar este deci $x = [999, 55] = 999$.

8.4.4 Algoritmi de descompunere în factori primi

Sunt extrem de numeroase lucrările care tratează descompunerea numerelor în factori primi. De aceea aici vom face doar o trecere în revistă a celor mai cunoscuţi algoritmi de factorizare.

Astfel, cel mai simplu pare a fi *ciurul lui Eratostene* care constă în încercarea de împărţi numărul n impar prin toate numerele întregi impare din intervalul $[3, \sqrt{n}]$. Pentru $n < 10^{12}$ tehnica este destul de eficientă.

Metoda $p - 1$

Un algoritm simplu care se poate aplica uneori şi la numere mari este metoda $p - 1$ enunţată de Pollard în 1974. În esenţă, el foloseşte trei variabile de intrare: numărul n (impar) care trebuie descompus, o margine B şi un număr oarecare $g \in [2, n - 1]$. Descrierea algoritmului este:

Intrare: n, B, g.
1. $a \longleftarrow g$
2. **for** $j = 2$ **to** B **do** $a \longleftarrow a^j \ mod \ n$
3. $d \longleftarrow cmmdc(a - 1, n)$
4. **if** $d > 1$ **then** "d este factor al lui n", **Stop**
   else "nu s-a găsit divizor al lui n"

Să vedem cum funcţionează acest algoritm:

Presupunem că p este un divizor prim al lui n şi că toţi divizorii primi ai lui $p-1$ – la puterile la care apar în descompunerea lui $p-1$ – sunt mai mici decât B. Atunci $p-1|B!$.

La terminarea ciclului de la pasul 2, avem

$$a \equiv g^{B!} \ (mod \ n) \quad \text{deci} \quad a \equiv g^{B!} \ (mod \ p)$$

deoarece $p|n$. Cum $g^{p-1} \equiv 1 \ (mod\ p)$, conform teoremei lui Fermat (în afară de cazul când $p|g$) şi cum $(p-1)|B!$, se obţine $a \equiv 1 \ (mod\ p)$.

Deci, la pasul 3 se ajunge la $p|(a-1)$ şi $p|n$, de unde rezultă $p|d = cmmdc(a-1, n)$.

Numărul d este un divizor netrivial al lui n (în afară de cazul $a = 1$ la pasul 3).

Având un divizor netrivial d, procesul se poate itera.

Exemplul 8.12. *Să considerăm* $n = 15770708441$. *Aplicând metoda* $p-1$ *cu* $B = 180$, *se găseşte* $a = 11620221425$, *iar* $d = 135979$. *Se ajunge la descompunerea finală*

$$15770708441 = 135979 \cdot 115979.$$

Descompunerea a reuşit deoarece 135978 *are numai factori primi "mici":* $135978 = 2 \cdot 3 \cdot 131 \cdot 173$. *Luând deci* $B \geq 173$ *se obţine* $135978|B!$.

Observaţia 8.2. *Condiţia ca metoda să funcţioneze este ca divizorii primi* **la puterile la care apar în descompunerea** *lui* $p-1$ *să fie mai mici decât* b. *Dacă s-ar solicita ca doar divizorii primi să verifice această condiţie, rezultatul ar fi fals. Astfel, să considerăm* $p = 17$ *şi* $B = 3$. *Atunci* $p - 1 = 2^4$. *Vom avea* $2 < 3$ *dar* 16 *nu este un divizor al lui* 3 *!!*

Pentru valori relativ mici ale lui B algoritmul este de complexitate polinomial scăzută ($O(BlogB(log\ n)^3)$). Dacă B creşte până la $\sqrt{n}$, el va reuşi totdeauna, dar nu va fi mai rapid decât ciurul lui Eratostene.

Deci slăbiciunea metodei rezidă în faptul că n trebuie să admită un divizor p cu proprietatea ca $p-1$ să aibă numai factori primi mici.

Pentru a rezista la acest atac, se recomandă folosirea numerelor *prime tari*.

Definiţia 8.8. *Se numeşte număr prim tare un număr prim* p *care verifică condiţiile:*

1. $p-1$ *are un divizor prim mare* r;
2. $p+1$ *are un divizor prim mare;*
3. $r-1$ *are un divizor prim mare.*

Există diverşi algoritmi pentru generarea numerelor prime tari. Pentru exemplificare am ales algoritmul lui Gordon:

1. Se generează aleator două numere prime mari distincte s, t.
2. Se alege un număr aleator i_0. Se află primul număr prim de forma $2 \cdot i \cdot t + 1$, unde $i \longleftarrow i_0, i_0 + 1, \ldots$. Fie $r = 2 \cdot i \cdot t + 1$ acest număr prim.
3. $p_0 \longleftarrow 2 \cdot (s^{r-2} \ (mod\ r)) \cdot s - 1$;
4. Se alege un număr aleator j_0. Se află primul număr prim de forma $p_0 + 2 \cdot j \cdot r \cdot s$, unde $j \longleftarrow j_0, j_0 + 1, \ldots$. Fie $p = p_0 + 2 \cdot j \cdot r \cdot s$ acest număr prim.
5. **return**(p)

Teorema 8.5. *Numărul* p *generat de algoritmul Gordon este un număr prim tare.*

Demonstraţie. Cum $r \neq s$, vom avea $s^{r-1} \equiv 1 \ (mod\ r)$ (Fermat). Deci $p_0 \equiv 1 \ (mod\ r)$ şi $p_0 \equiv -1 \ (mod\ s)$. Acum:

(1) $p - 1 = p_0 + 2 \cdot j \cdot r \cdot s - 1 \equiv 0 \ (mod\ r)$, deci $p - 1$ are pe r drept divizor prim.

(2) $p + 1 = p_0 + 2 \cdot j \cdot r \cdot s + 1 \equiv 0 \ (mod\ s)$, deci s este un divizor prim al lui $p + 1$.

(3) $r - 1 = 2 \cdot i \cdot t \equiv 0 \ (mod\ t)$, deci numărul prim t divide pe $r - 1$. □

Practic, generarea unui număr prim tare se realizează în trei paşi:

(i) Cu un generator de numere aleatoare, se generează numerele s, t, i_0, j_0;

(ii) Se testează dacă s şi t sunt numere prime, folosind algoritmul Miller - Rabin;

(iii) În caz afirmativ, se aplică algoritmul lui Gordon, bazat de asemenea pe algoritmul Miller - Rabin.

De multe ori, pentru criptarea RSA este suficient să se folosească numere prime mari p cu proprietatea că $\frac{p-1}{2}$ este de asemenea număr prim.

Exemplul 8.13. *În practică este folosit frecvent exponentul de criptare $b = 3$. În acest caz însă, este necesar ca $p - 1$ şi $q - 1$ să nu fie divizibile cu 3. Rezultatul este o criptare extrem de rapidă, deoarece se foloseşte o singură înmulţire modulară şi o singură ridicare la pătrat modulară.*

De asemenea este utilizat frecvent şi exponentul $b = 2^{16} + 1 = 65537$. Acest număr are numai doi de 1 în reprezentarea binară, aşa că o criptare foloseşte 16 ridicări la pătrat modulare şi o singură înmulţire modulară.

Algoritmul lui Dixon şi sita pătratică

Algoritmul lui Dixon se bazează pe o idee extrem de simplă: dacă se pot afla două numere x, y cu $x \not\equiv y \ (mod\ n)$ dar $x^2 \equiv y^2 \ (mod\ n)$, atunci $cmmdc(x - y, n)$ este un divizor netrivial al lui n.

Metoda utilizează o bază $\mathcal{B}$ de factori primi "mici". Se caută întâi mai multe numere x pentru care divizorii primi ai lui $x^2 \ mod\ n$ sunt în $\mathcal{B}$. Se formează apoi produse cu aceste numere, în aşa fel încât fiecare factor prim al pătratului produsului să apară de un număr par de ori. Aceasta conduce la o relaţie $x^2 \equiv y^2 \ (mod\ n)$ care va da – eventual – o descompunere a lui n.

Exemplul 8.14. *Fie $n = 15770708441$ şi alegem mulţimea $\mathcal{B}$= $\{2, 3, 5, 7, 11, 13\}$. Selectăm*

$$8340934156^2 \equiv 3 \cdot 7 \ (mod\ n)$$
$$12044942944^2 \equiv 2 \cdot 7 \cdot 13 \ (mod\ n)$$
$$2773700011^2 \equiv 2 \cdot 3 \cdot 13 \ (mod\ n)$$

Dacă se ia produsul acestor trei congruenţe, se ajunge la

$$(8340934156 \cdot 12044942944 \cdot 2773700011)^2 \equiv (2 \cdot 3 \cdot 7 \cdot 13)^2 \ (mod\ n)$$

Reducând conţinutul parantezelor modulo n, se obţine

$$9503435785^2 \equiv 546^2 \ (mod\ n).$$

Vom calcula acum $cmmdc(9503435785 - 546, 15770708441) = 115759$*,*
care va da un divizor 115759 *al lui* n*.*

Fie $\mathcal{B}$= $\{p_1, p_2, \ldots, p_B\}$; considerăm un număr C "puţin" mai mare decât B (o posibilitate este $C = B + 10$) şi presupunem că am găsit C relaţii de forma

$$x_j^2 \equiv p_1^{\alpha_{1j}} \cdot p_2^{\alpha_{2j}} \cdot \ldots \cdot p_B^{\alpha_{Bj}}, \quad (1 \leq j \leq C).$$

Pentru fiecare j se consideră vectorul binar (elementele sale se iau modulo 2)

$$\alpha_j = (\overline{\alpha}_{1j}, \ldots, \overline{\alpha}_{Bj}) \in Z_2^B.$$

Dacă se poate determina o submulţime, formată din astfel de vectori, a căror sumă modulo 2 să fie $(0, 0, \ldots, 0)$, atunci pătratul produsului elementelor x_j corespunzătoare va avea în $\mathcal{B}$ toţi divizorii reprezentaţi de un număr par de ori.

Exemplul 8.15. *Revenind la Exemplul 8.14, cei trei vectori care se construiesc sunt*
$\alpha_1 = (0, 1, 0, 1, 0, 0), \quad \alpha_2 = (1, 0, 0, 1, 0, 1), \quad \alpha_3 = (1, 1, 0, 0, 0, 1).$
Se verifică imediat că $\alpha_1 + \alpha_2 + \alpha_3 \equiv (0, 0, 0, 0, 0, 0)\ (mod\ 2)$*.*

Evident, a căuta o submulţime de C vectori de sumă nulă modulo 2 revine la a căuta o relaţie de dependenţă liniară (în Z_2) între aceşti vectori. Dacă $C > B$, o asemenea relaţie există şi poate fi găsită uşor prin eliminare gaussiană.

Ar mai fi de văzut cum se pot obţine acei x_j pentru care x_j^2 admit descompuneri în factori primi din baza $\mathcal{B}$. Sunt mai multe metode posibile pentru aceasta; de exemplu, ciurul pătratic – construit de Pomerance – foloseşte numere întregi de forma

$$x_j = j + [\sqrt{n}],\ j = 1, 2, \ldots.$$

De remarcat că dacă B este mare, este foarte posibil ca un întreg x_j să se descompună în $\mathcal{B}$, dar numărul acestor x_j trebuie să crească pentru a căuta relaţiile de dependenţă. Se arată că alegerea optimă pentru B este în jur de

$$\sqrt{e^{\sqrt{logn\ log\ logn}}}.$$

8.4.5 Alte tipuri de atac

Atac bazat pe proprietăţi multiplicative ale criptării *RSA*

Fie m_1, m_2 două texte clare şi c_1 respectiv c_2 textele criptate corespunzătoare. Vom avea

$$(m_1 m_2)^b \equiv m_1^b m_2^b \equiv c_1 c_2 \ (mod\ n)$$

Această proprietate de homomorfism a criptării poate oferi unui adversar activ posibilitatea unui atac cu text clar ales.

Să presupunem că *Oscar* vrea să decripteze mesajul $c = m^b \ (mod\ n)$ trimis de *Bob* lui *Alice*, iar *Alice* are amabilitatea să îi decripteze lui *Oscar* orice text criptat primit (înafară de c, bineinţeles).

Atunci *Oscar* va alege un număr aleator $x \in Z_n^*$, va calcula $c_1 = c \cdot x^b \ (mod\ n)$ şi va solicita decriptarea lui. *Alice* va decripta pentru el $m_1 = c_1^a \ (mod\ n)$. Deoarece

$$m_1 \equiv c_1^a \equiv c^a (x^b)^a \equiv mx \ (mod\ n)$$

Oscar va afla imediat $m = m_1 \cdot x^{-1} \ (mod\ n)$.

Acest tip de atac este prevenit de obicei impunând anumite structuri speciale asupra textelor clare.

Atac bazat pe un exponent mic de criptare

Asa cum s-a arătat în Exemplul 8.13, pentru mărirea vitezei de criptare se preferă exponenţi mici de criptare (cum este $b = 3$). Această alegere are unele slăbiciuni, care permit atacuri în anumite condiţii.

Astfel, să presupunem că *Alice* doreşte să trimită acelaşi text clar m la trei destinatari diferiţi, care au modulele n_i, dar acelaşi exponent de criptare $b = 3$. Deci textele criptate vor fi $c_i = m^3 \ (mod\ n_i) \quad i = 1, 2, 3$. *Oscar* le interceptează şi rezolvă sistemul

$$x \equiv c_1 \ (mod\ n_1), \qquad x \equiv c_2 \ (mod\ n_2), \qquad x \equiv c_3 \ (mod\ n_3)$$

folosind teorema chineză a resturilor, care asigură existenţa unui $x \in [0, n_1 \cdot n_2 \cdot n_3)$. Deoarece $m^3 < n_1 \cdot n_2 \cdot n_3$, va rezulta că $x = m^3$.

Deci *Oscar* va afla textul clar m extrăgând rădăcina de ordinul 3 din soluţia x.

În general, exponenţi mici de criptare asigură o securitate redusă pentru mesajele mici m: dacă $m < n^{1/b}$ atunci textul clar poate fi dedus din textul criptat $c = m^b$ calculând rădăcina de ordin b a lui c.

Folosirea unui modul comun de criptare de către mai mulţi utilizatori permite de asemenea un atac uşor. Să presupunem că *Alice* trimite acelaşi mesaj m către doi utilizatori care au cheile publice b_1, b_2 dar acelaşi modul n. Deci

$$c_1 \equiv m^{b_1} \ (mod\ n), \qquad c_2 \equiv m^{b_2} \ (mod\ n).$$

Fără a micşora generalitatea, putem presupune că $cmmdc(b_1, b_2) = 1$. *Oscar* va folosi algoritmul lui Euclid pentru a determina numerele întregi r, s astfel ca

$$r \cdot b_1 + s \cdot b_2 = 1$$

Unul din numerele r sau s este negativ; să presupunem că este r (pentru s negativ se procedează analog). Atunci *Oscar* obţine textul clar m folosind egalitatea

$$(c_1^{-1})^{-r} \cdot c_2^s \equiv m \ (mod\ n)$$

Atacuri ciclice

Fie $c = m^b \ (mod\ n)$ un text criptat şi k un întreg pozitiv astfel ca

$$c^{b^k} \equiv c \ (mod\ n)$$

(un astfel de k există totdeauna, deoarece criptarea RSA este de fapt o permutare în spaţiul Z_n al textelor clare). Rezultă că

$$c^{b^{k-1}} \equiv m \ (mod\ n)$$

Această observaţie conduce la un *atac ciclic*: *Oscar* calculează $c^{b^i} \ (mod\ n)$ pentru $i = 1, 2, \ldots$ până găseşte un k astfel ca $c^{b^k} \equiv c \ (mod\ n)$. Atunci va decripta c în $c^{b^{k-1}} \equiv m \ (mod\ n)$.

Un *atac ciclic generalizat* constă în aflarea celui mai mic întreg pozitiv k astfel ca $cmmdc(c^{b^k} - c, n) > 1$. Vom avea implicaţia

$$c^{b^k} \equiv c \ (mod\ p), \qquad c^{b^k} \not\equiv c \ (mod\ q) \quad \Longrightarrow \quad k = p$$

şi similar

$$c^{b^k} \not\equiv c \ (mod\ p), \qquad c^{b^k} \equiv c \ (mod\ q) \quad \Longrightarrow \quad k = q$$

În ambele cazuri s-a obţinut factorizarea lui n.

Pe de-altă parte, dacă

$$c^{b^k} \equiv c \ (mod\ p), \qquad c^{b^k} \equiv c \ (mod\ q) \quad \Longrightarrow \quad k = n \ \text{ şi } \ c^{b^k} \equiv c \ (mod\ n)$$

În acest caz s-a reuşit decriptarea $m = c^{b^{k-1}} \ (mod\ n)$.

Folosirea unor numere prime tari asigură o protecţie suficientă pentru acest gen de atac.

8.5 Sisteme de criptare înrudite cu RSA

8.5.1 Sistemul de criptare Rabin

Sistemul de criptare Rabin (propus în 1979) este o variantă a sistemului RSA, care oferă o securitate de calcul echivalentă. Descrierea sa este:

Fie $n = pq$ unde p, q sunt numere prime distincte, $p, q \equiv 3 \ (mod\ 4)$. Se ia $\mathcal{P} = \mathcal{C} = Z_n$ şi $\mathcal{K} = \{(n, p, q, B) \mid 0 \leq B \leq n - 1\}$.
Pentru cheia $K = (n, p, q, B)$ se definesc:

$$e_K(x) = x(x + B) \ (mod\ n) \qquad d_K(y) = \sqrt{\frac{B^2}{4} + y} - \frac{B}{2}$$

Observaţia 8.3. *Numerele prime n cu $n \equiv 3 \ (mod\ 4)$ se numesc "numere Blum".*

Există patru texte clare distincte care se pot cripta în acelaşi text. Să detaliem această afirmaţie:

Fie α una din cele patru rădăcini pătrate modulo n ale unităţii, şi $x \in Z_n$. Efectuăm calculele

$$e_K\left(\alpha\left(x+\frac{B}{2}\right)-\frac{B}{2}\right)=\alpha^2\left(x+\frac{B}{2}\right)^2-\left(\frac{B}{2}\right)^2=x^2+Bx=e_K(x)$$

(calculele s-au realizat în Z_n, iar împărţirea la 2 şi 4 s-a făcut prin înmulţirea în Z_n cu 2^{-1} respectiv 4^{-1}).

Cele patru texte clare care se cifrează în $e_K(x)$ sunt

$$x, \quad -x-B, \quad \alpha(x+B/2)-B/2 \ \text{ şi } -\alpha(x+B/2)-B/2,$$

unde α este o rădăcină pătrată netrivială modulo n a unităţii.

Verificarea este imediată.

În general, dacă nu dispune de informaţii suplimentare, *Bob* nu are nici un mijloc de a distinge care din cele patru mesaje este cel corect.

Să vedem cum se realizează decriptarea. *Bob* primeşte mesajul criptat y şi încearcă să să determine x astfel ca $x^2+Bx \equiv y \ (mod\ n)$.

Aceasta este o ecuaţie de gradul doi în x. Termenul de gradul 1 se poate elimina folosind substituţia $x_1 = x + B/2$ (sau – echivalent – $x = x_1 - B/2$).

Se ajunge la ecuaţia

$$x_1^2 \equiv \frac{B^2}{4}+y \quad (mod\ n).$$

Notând membrul drept cu C, această ecuaţie se scrie $x_1^2 \equiv C \ (mod\ n)$. Deci decriptarea se reduce la extragerea rădăcinilor pătrate modulo n; operaţie echivalentă cu rezolvarea sistemului

$$x_1^2 \equiv C \ (mod\ p) \qquad x_1^2 \equiv C \ (mod\ q)$$

care, prin combinarea soluţiilor fiecărei ecuaţii va da patru rădăcini pătrate modulo n.

Într-o criptare corectă, C este totdeauna un rest pătratic modulo p şi q. Dacă $p \equiv 3 \ (mod\ 4)$, există o formulă simplă pentru extragerea rădăcinilor pătrate dintr-un rest pătratic C modulo p. Avem (calculele se fac modulo p):

$$\left(\pm C^{(p+1)/4}\right)^2 \equiv C^{(p+1)/2} \equiv C^{(p-1)/2}C \equiv C$$

(s-a folosit Teorema 8.1). Cele patru rădăcini sunt deci

$$\pm C^{(p+1)/4} \ (mod\ p), \quad \pm C^{(q+1)/4} \ (mod\ q)$$

Acestea, prin combinare folosind teorema chineză a resturilor, dau cele patru rădăcini pătrate ale lui C.

Observaţia 8.4. *Nu se cunoaşte un algoritm polinomial determinist pentru extragerea rădăcinilor pătratice modulo p pentru $p \equiv 1 \pmod 4$; în această situaţie există doar algoritmi Las Vegas.*

După determinarea acestor rădăcini x_1, se află $x = x_1 - B/2$, rezultat care dă formula de decriptare din enunţul metodei Rabin.

Exemplul 8.16. *Fie $n = 77 = 7 \cdot 11$ şi $B = 9$. Funcţia de criptare este*

$$e_K(x) = x^2 + 9x \pmod{77}$$

iar cea de decriptare

$$d_K(y) = \sqrt{1+y} - 43 \pmod{77}.$$

Să presupunem că Bob vrea să decripteze textul $y = 22$. El va trebui să determine rădăcinile pătrate ale lui 23 modulo 7 şi 11. Cum aceste două module sunt congruente cu 3 modulo 4, se poate aplica formula arătată anterior:

$$23^{(7+1)/4} \equiv 2^2 \equiv 4 \pmod 7 \qquad 23^{(11+1)/4} \equiv 1^3 \equiv 1 \pmod{11}.$$

Utilizând teorema chineză a resturilor, se obţin rădăcinile pătrate ale lui 23 modulo 77:

$$11 \cdot 2 \cdot a + 7 \cdot 8 \cdot b \pmod{77}$$

unde $a = \pm 4$, $b = \pm 1$. Calculând, se obţin valorile ± 10, ± 32. Cele patru texte clare posibile (calculate modulo 77) vor fi deci:

$$10 - 43 = 44, \quad 67 - 43 = 24, \quad 32 - 43 = 66, \quad 45 - 43 = 2.$$

Se verifică imediat că toate aceste patru texte clare se criptează în 22.

Să studiem acum securitatea sistemului de criptare Rabin. Să presupunem că există un algoritm de decriptare **A**; acesta poate fi atunci utilizat într-un algoritm Las Vegas care descompune modulul n cu probabilitate $1/2$; algoritmul este următorul:

1. Se alege aleator $r \in Z_n$;
2. $y \longleftarrow r^2 - B^2/4 \ mod \ n$;
3. $x \longleftarrow \mathbf{A}(y)$;
4. $x_1 \longleftarrow x + B/2$;
5. **if** $x_1 \equiv \pm r \pmod n$ **then** Stop (eşec)
   else $cmmdc(x_1 + r, n) = p$ sau q, Stop

Să observăm întâi că $y = e_K\left(r - \dfrac{B}{2}\right)$, deci la pasul 3 se decriptează x sub forma $r - B/2$. Cum la pasul 5. avem $x_1^2 \equiv r^2 \pmod n$, rezultă $x_1 \equiv \pm r \pmod n$ sau $x_1 \equiv \pm \alpha r \pmod n$, unde α este o rădăcină netrivială modulo n a unităţii.

În al doilea caz, $n|(x_1 - r)(x_1 + r)$ şi n nu divide nici unul din cei doi factori. Deci, calculul lui $cmmdc(x_1 + r, n)$ sau $cmmdc(x_1 - r, n)$ va da p sau q, adică o descompunere a lui n.

Să calculăm probabilitatea de succes a algoritmului ([45]), din $n-1$ extrageri posibile ale lui r. Pentru două resturi nenule r_1, r_2, se defineşte

$$r_1 \sim r_2 \quad \Longleftrightarrow \quad r_1^2 \equiv r_2^2 \ (mod\ n)$$

Aceasta este evident o relaţie de echivalenţă. Toate clasele de echivalenţă din Z_n^* au câte patru elemente, fiecare clasă fiind de forma $[r] = \{\pm r,\ \pm\alpha r\}$. În algoritmul anterior, două valori dintr-o clasă de echivalenţă conduc la acelaşi y.

Să considerăm un x_1 calculat plecând de la valoarea x returnată de oracolul **A** pentru un y dat. x_1 este un element din $[r]$. Dacă $x_1 = \pm r$, algoritmul eşuează; dacă $x_1 = \pm\alpha r$, el reuşeşte să descompună n. Cum r este aleator, cele patru posibilităţi sunt echi-probabile, deci algoritmul dă reuşită în 50% din cazuri.

8.6 Exerciţii

8.1. *Demonstraţi lemele 8.1 şi 8.2.*

8.2. *Fie p, q numere prime impare distincte şi $n = pq$. Definim*

$$\lambda(n) = \frac{(p-1)(q-1)}{cmmdc(p-1, q-1)}$$

Folosim un sistem de criptare RSA în care s-a făcut modificarea $a \cdot b \equiv 1\ (mod\ \lambda(n))$.

(a) Demonstraţi că operaţiile de criptare şi decriptare sunt operaţii inverse şi în acest sistem.

(b) Dacă $p = 37$, $q = 79$ şi $b = 7$, calculaţi valoarea exponentului a atât în acest sistem cât şi în sistemul RSA normal.

8.3. *O modalitate curentă de a mări viteza de decriptare foloseşte teorema chineză a resturilor. Să presupunem că $n = pq$ şi $d_K(y) = y^a\ (mod\ n)$. Definim $d_p = d\ (mod\ (p-1))$, $d_q = d\ (mod\ (q-1))$ şi $M_p = q^{-1}\ (mod\ p)$, $M_q = p^{-1}\ (mod\ q)$.*

Vom considera algoritmul

1.	$x_p \longleftarrow y^{d_p}\ (mod\ p)$;
2.	$x_q \longleftarrow y^{d_q}\ (mod\ q)$;
3.	$x \longleftarrow M_p \cdot q \cdot x_p + M_q \cdot p \cdot x_q\ (mod\ n)$;
4.	**return**(x).

(a) Demonstraţi că valoarea x returnată este de fapt $y^d\ (mod\ n)$.

(*b*) *Pentru* $p = 1511,\ q = 2003$ *calculaţi* d_p, d_q, M_p, M_q *şi apoi decriptaţi textul* $y = 152702$ *folosind algoritmul din acest exerciţiu.*

8.4. *Pentru* $n = 837,\ 851\ 1189$ *aflaţi numărul de baze* b *pentru care numărul* n *este Euler pseudo-prim.*

8.5. *Scrieţi un program pentru calculul simbolului Jacobi, folosind Lemele 8.1 şi 8.2. Singura operaţie de factorizare permisă este împărţirea la* 2*. Valori de test:*

$$\left(\frac{610}{987}\right), \qquad \left(\frac{20964}{1987}\right), \qquad \left(\frac{1234567}{11111111}\right)$$

8.6. *Să se arate că orice număr* a *pentru care testul Miller - Rabin dă rezultat fals este un număr Euler pseudo-prim. Reciproca este adevărată dacă şi numai dacă* $n \equiv 3\ (mod\ 4)$.

8.7. *Fie* p *un număr prim impar şi* $cmmdc(a, p) = 1$.
a) Să presupunem că $i \geq 2$ *şi* $b^2 \equiv a\ (mod\ p^{i-1})$. *Demonstraţi că există un* $x \in Z_p$ *unic astfel ca* $x^2 \equiv a\ (mod\ p^i)$ *şi* $x \equiv b\ (mod\ p^{i-1})$.

Găsiţi o modalitate eficientă de calcul a lui x.
(*b*) *Aplicaţi punctul anterior în următoarea situaţie: plecând de la congruenţa* $6^2 \equiv 17\ (mod\ 19)$, *aflaţi rădăcinile pătrate ale lui* 17 *modulo* 19^2 *şi modulo* 19^3.
(*c*) *Arătaţi că numărul soluţiilor congruenţei* $x^2 \equiv a\ (mod\ p^i)$ *este* 0 *sau* 2, $(\forall i \geq 1)$.

8.8. *Folosind metoda* $p - 1$ *şi diverse margini* B, *factorizaţi* 262063 *şi* 9420457. *Cât este* B *pentru fiecare caz ?*

8.9. *Fie* $n = 317940011$ *şi* $b = 7753781$. *Descompuneţi* n *în factori, folosind algoritmul lui Wiener.*

8.10. *Consideräm algoritmul lui Rabin cu* $p = 199,\ q = 211,\ n = pq$ *şi* $B = 1357$.
(*a*) *Criptaţi mesajul* 32767*;*
(*b*) *Determinaţi cele* 4 *mesaje clare care duc la textul criptat* $y = e_K(32767)$.

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