Prelegerea 10

Sistemul de criptare RSA

10.1 Descrierea sistemului RSA

Sistemul de criptare RSA (**R**ivest - **S**hamir - **A**dleman) este în acest moment cel mai cunoscut şi uzitat 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 în special la multe sisteme simetrice).

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 10.1 *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 criptare dacă*

[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).

şi numai dacă nu este divizibil cu $2, 5$ *sau* 7 *(practic, Bob nu trebuie să factorizeze* $\phi(n)$*; este suficient să verifice dacă* $(\phi(n), b) = 1$ *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 9726*, ea calculează*

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

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

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

Securitatea sistemului de criptare RSA se bazează pe ipoteza că funcţia $e_K(x) = x^b \ mod \ n$ 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*).

10.2 Implementarea sistemului RSA

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

Tabelul 10.1:

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 $(b, \phi(n)) = 1$;
4. Calculează $a = b^{-1} \ mod \ \phi(n)$ folosind algoritmul lui Euclid;
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 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 $O(k^2)$. Conform prelegerii anterioare, vom numi aceasta *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 10.2 *Să reluăm datele din Exemplul 10.1. 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, trebuiesc generate întâi numerele prime p, q - despre care ne ocupăm în secţiunea următoare. Etapa a doua (din Tabelul 10.1) 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 (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, cam de 100 ori la o implementare software. În [3] se dau câteva tabele cu astfel de valori comparative, la nivelul anului 1995.

10.3 Teste de primalitate probabiliste

În realizarea sistemului de criptare RSA trebuiesc generate aleator numere prime cu număr mare de cifre. Practic, se realizează aleator 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 10.1 *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 10.2 *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 mai mici ?

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

10.3.1 Algoritmul Solovay - Strassen

Să reamintim întâi câteva noţiuni matematice:

Definiţia 10.3 *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 10.3 *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 10.1 *(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 10.4 *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 10.1 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 10.2 *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 10.5 *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 10.4 *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 10.1 *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 10.2 *Fie m, n două numere întregi pozitive impare. Atunci*

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

Lăsăm ca exerciţiu demonstraţiile celor două leme.

Exemplul 10.5 *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 [1].

10.3.2 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**
6. **if** $b \equiv -1\ (mod\ n)$ **then** "n este prim", Stop,
   else $b \longleftarrow b^2\ (mod\ n)$
7. "n nu este prim", Stop

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

Teorema 10.3 *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 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 [4] 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, algorituml 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 computaţional mai complex.

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 10.8), 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. Schneier propune următoarea variantă ([3]):

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.

10.4 Anexă

10.4.1 Algoritmul lui Euclid extins

După cum se ştie, algoritmul lui Euclid constituie o modalitate eficace de determinare a celui mai mare divizor comun a două numere întregi pozitive. El poate fi extins pentru a determina şi inversele elementelor dintr-un corp finit Z_n.

Să reamintim întâi algoritmul lui Euclid (forma clasică):

Fie $r_0, r_1 \in N^*$. Se efectuează secvenţa de împărţiri succesive:

$$\begin{aligned} r_0 &= q_1 r_1 + r_2 && 0 < r_2 < r_1 \\ r_1 &= q_2 r_2 + r_3 && 0 < r_3 < r_2 \\ &\vdots \\ r_{m-2} &= q_{m-1} r_{m-1} + r_m && 0 < r_m < r_{m-1} \\ r_{m-1} &= q_m r_m. \end{aligned} \tag{1}$$

Deoarece $(r_0, r_1) = (r_1, r_2) = \ldots = (r_{m-1}, r_m) = r_m$, rezultă că cel mai mare divizor comun dintre r_0 şi r_1 este r_m.

Să definim acum şirul $t_0, t_1, \ldots, t_m$ astfel:

$$\begin{aligned} &t_0 = 0, \quad t_1 = 1 \\ &t_j = t_{j-2} - q_{j-1} t_{j-1} \ (mod\ r_0),\ j \geq 2 \end{aligned} \tag{2}$$

Teorema 10.4 *Pentru $0 \leq j \leq m$ avem $r_j \equiv t_j r_1 \ (mod\ r_0)$ unde r_j şi t_j sunt definite de (1) respectiv (2).*

Demonstraţie: Se foloseşte o inducţie după j. Pentru $j = 0$ şi $j = 1$ afirmaţia este banală. O presupunem adevărată pentru $j = i-1$ şi $j = i-2$ $(i \geq 2)$ şi să o arătăm pentru $j = i$. Toate calculele se fac modulo r_0.

Conform ipotezei de inducţie, $r_{i-2} = t_{i-2} r_1,\ r_{i-1} = t_{i-1} t_1$. Acum:
$r_i = r_{i-2} - q_{i-1} r_{i-1} = t_{i-2} r_1 - q_{i-1} t_{i-1} r_1 = (t_{i-2} - q_{i-1} r_{i-1}) r_1 = t_i r_1$. □

Corolarul 10.1 *Dacă $(r_0, r_1) = 1$ atunci $t_m = r_1^{-1} \ (mod\ r_0)$.*

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

Se poate da acum algoritmul extins al lui Euclid care pentru $n > 1$ şi $b \in Z_n^*$ va determina $b^{-1} \ mod \ n$ (dacă există).

1. $n_0 \longleftarrow n,\ b_0 \longleftarrow b,\ t_0 \longleftarrow 0,\ t \longleftarrow 1$
2. $q \longleftarrow \left[\frac{n_0}{b_0}\right], \quad r \longleftarrow n_0 - q \cdot b_0$
3. **while** $r > 0$ **do**
   - 3.1. $temp \longleftarrow t_0 - q \cdot t$
   - 3.2. **if** $temp \geq 0$ **then** $temp \longleftarrow temp \quad (mod\ n)$
     else $temp \longleftarrow n - ((-temp) \quad (mod\ n))$
   - 3.3. $n_0 \longleftarrow b_0,\ b_0 \longleftarrow r,\ t_0 \longleftarrow t,\ t \longleftarrow temp$
   - 3.4. $q \longleftarrow \left[\frac{n_0}{b_0}\right],\ r \longleftarrow n_0 - q \cdot b_0$
4. **if** $b_0 \neq 1$ **then** b nu are inversă $mod\ n$
   else $b^{-1}\ (mod\ n) = t$.

Exemplul 10.6 *Să calculăm* 28^{-1} *mod* 75*, folosind algoritmului lui Euclid extins. Vom avea pe rând:*

n_0	b_0	q	r	t_0	t	$temp$
75	28	2	19	0	1	73
28	19	1	9	1	73	3
19	9	2	1	73	3	67
9	$\underline{1}$	9	0	3	$\underline{67}$	

Deci $28^{-1}\ mod\ 75\ = 67$.

10.4.2 Teorema chineză a resturilor

Teorema 10.5 *Se dau numerele* $p_1, p_2, \ldots, p_r$ *prime între ele şi fie* $n = p_1 p_2 \ldots p_r$*. Atunci sistemul de ecuaţii*

$$x \equiv a_i \quad (mod\ p_i), \qquad 1 \leq i \leq r$$

are soluţie comună în intervalul $[0, n-1]$.

Demonstraţie: Pentru fiecare i, $(p_i, n/p_i) = 1$; deci există numerele y_i astfel încât

$$\frac{n}{p_i} \cdot y_i \equiv 1\ (mod\ p_i).$$

De asemenea, pentru $j \neq i$, deoarece $p_j | (n/p_i)$, avem $\frac{n}{p_i} \cdot y_i \equiv 0\ (mod\ p_j)$.

Alegem

$$x = \sum_{i=1}^{r} \frac{n}{p_i} \cdot y_i \cdot a_i\ (mod\ n).$$

Pentru orice i, x este o soluţie a ecuaţiei $x \equiv a_i \ (mod\ p_i)$ deoarece în Z_{p_i} avem $x = \dfrac{n}{p_i} \cdot y_i \cdot a_i = a_i$. □

Exemplul 10.7 *Fie* $r = 3,\ p_1 = 7,\ p_2 = 11,\ p_3 = 13$, *deci* $n = 1001$. *Notând* $m_i = \dfrac{n}{p_i}$, *avem* $m_1 = 143,\ m_2 = 91$ *şi* $m_3 = 77$. *Folosind algoritmul lui Euclid, se obţine* $y_1 = 5,\ y_2 = 4,\ y_3 = 12$. *Soluţia generală este atunci*

$$x = 715a_1 + 364a_2 + 924a_3 \ (mod\ 1001).$$

De exemplu, pentru sistemul

$$x \equiv 5 \ (mod\ 7), \quad x \equiv 3 \ (mod\ 11), \quad x \equiv 10 \ (mod\ 13)$$

formula de sus dă

$x = 715 \cdot 5 + 364 \cdot 3 + 924 \cdot 10 \ (mod\ 1001) = 13907 \ (mod\ 1001) = 894.$

Verificarea se realizează reducând x *modulo* 7, 11 13.

10.5 Exerciţii

10.1 *Demonstraţi lemele 10.1 şi 10.2.*

10.2 *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.*

10.3 *Să se arate că pentru orice număr* n *egal cu produsul primelor* k $(k \geq 2)$ *numere prime impare testul Miller - Rabin dă rezultat corect pentru orice întreg* $a \in [2, n-2]$.

10.4 *O modalitate curentă de a mări viteza de decriptare foloseşte teorema chineză a restului. 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* $y^d \pmod n$.

(*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.*

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

10.6 *Scrieţi un program pentru calculul simbolului Jacobi, folosind lemele 10.1 şi 10.2. Singura operaţie de factorizare permisă este împărţirea la* 2. *valori de test:*

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

10.7 *Fie*

$$G(n) = \left\{a | a \in Z_n^*,\ \left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \pmod n\right\}.$$

(*a*) *Demonstraţi că* $G(n)$ *este subgrup al lui* Z_n. *Deci, conform Teoremei lui Langrance, dacă subgrupul este propriu, atunci*

$$|G(n)| \leq \frac{|Z_n^*|}{2} \leq \frac{n-1}{2}.$$

(*b*) *Să presupunem că* $n = p^k \cdot q$, *unde* p *şi* q *sunt numere impare prime între ele,* p *este prim şi* $k \geq 2$. *Fie* $a = 1 + p^{k-1} \cdot q$. *Demonstraţi că*

$$\left(\frac{a}{n}\right) \not\equiv a^{\frac{n-1}{2}} \pmod n.$$

(*c*) *Fie* $n = p_1 p_2 \dots p_s$ *unde* p_i *sunt numere prime impare distincte. Să presupunem că* $a \equiv 1 \pmod{p_1}$ *şi* $a \equiv 1 \pmod{p_2 p_3 \dots p_s}$, *unde* u *este un non-reziduu pătratic modulo* p_1 *(un astfel de* a *există, conform teoremei chineze a restului). Demonstraţi că*

$$\left(\frac{a}{n}\right) \equiv -1 \pmod n, \qquad a^{\frac{n-1}{2}} \not\equiv -1 \pmod{p_2 p_3 \dots p_s}$$

(deci $a^{\frac{n-1}{2}} \not\equiv -1 \pmod n$*).*

(*d*) *Dacă* n *este un număr neprim impar, arătaţi că* $|G(n)| \leq \frac{n-1}{2}$.

(*e*) *Pe baza celor de mai sus, arătaţi că probabiltatea de eroare a testului de primalitate Solovay - Strassen este cel mult* $1/2$.

10.8 *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* $n \equiv 3 \pmod 4$.

10.9 *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 eifcientă 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) $\forall i \geq 1$, arătaţi că numărul soluţiilor congruenţei $x^2 \equiv a \ (mod\ p^i)$ este 0 sau 2.

10.10 *Folosind algoritmul lui Euclid extins, calculaţi inversele:*

$$17^{-1} \ (mod\ 101), \qquad 357^{-1} \ (mod\ 1234), \qquad 3125^{-1} \ (mod\ 9987)$$

10.11 *Fie aplicaţia $u : Z_{105} \longrightarrow Z_3 \times Z_5 \times Z_7$ definită prin*

$$u(x) = (x \ (mod\ 3), \quad x \ (mod\ 5), \quad x \ (mod\ 7))$$

Să se găsească o formulă pentru u^{-1} şi să se determine cu ajutorul ei $u^{-1}(2, 2, 3)$.

10.12 *Să se rezolve sistemul de congruenţe*

$$x \equiv 12 \ (mod\ 25), \qquad x \equiv 9 \ (mod\ 26), \qquad x \equiv 23 \ (mod\ 27)$$

10.13 *Să se rezolve sistemul de congruenţe*

$$13 \cdot x \equiv 4 \ (mod\ 99), \qquad 15 \cdot x \equiv 56 \ (mod\ 101)$$

Bibliografie

[1] D. Stinton, *Cryptography, Theory et Pratice*, Second Edition, Chapman & Hall/CRC 2002

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

[3] B. Schneier, *Applied Cryptography*, Second Edition, John Wiley & Sons, 1996

[4] A. Menezes, P. Oorschot, S. Vanstome, *Handbook of Applied Cryptography*,

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