TEMA 7: CRIPTOGRAFIA ASIMETRICĂ

Obiectivele temei:

- Introducere în criptarea cu cheie publică.
- Analiza algoritmului schimbului de chei Diffie-Hellman.
- Analiza algoritmului de criptare ElGamal.
- Analiza Algoritmului de criptare RSA.
- Familiarizarea cu infrastructura cheilor publice (PKI).

Cuvinte-cheie:

criptare asimetrică,	calculul puterii modulo *n*,
cheie publică,	calculul inversului modulo *n*,
cheie privată,	autoritate de certificare,
distribuirea cheilor,	autoritate de înregistrare,
problema logaritmului discret,	arhitecturi PKI.

Conceptul de criptografie asimetrică sau criptografie cu cheie publică a fost inventat de *Whitfield Diffie* şi *Martin Hellman*. Contribuţia lor constă în propunerea de a folosi un nou concept de criptare în care cheile de criptare şi decriptare sunt diferite, iar cheia de decriptare (care este secretă) nu poate fi dedusă din cheia de criptare (care este publică). În anul 1976 conceptul a fost prezentat în premieră la National Computer Conference în SUA, iar câteva luni mai târziu lucrarea a fost publicată.

7.1. Noțiune de criptare cu cheie publică

Algoritmii de criptare cu cheie publică (sau *algoritmii asimetrici*) au un mare avantaj faţă de algoritmii cu cheie secretă: oricine poate transmite un mesaj secret utilizatorului (cunoscându-i *cheia publică*), iar mesajul rămâne protejat faţă de interceptor. Cu un sistem cu cheie secretă pentru fiecare pereche de utilizatori este necesară câte o cheie separată.

În general, un sistem cu *n* utilizatori ai unui algoritm de criptare cu cheie secretă necesită $\frac{n(n-1)}{2}$ chei, pentru ca oricare pereche de utilizatori să poată comunica între ei şi mesajele lor să rămână secrete faţă de ceilalţi utilizatori. Numărul de chei creşte rapid o dată cu numărul de utilizatori, iar generarea, distribuirea şi menţinerea securităţii cheilor constituie o problemă datorită numărului lor mare.

Într-un algoritm cu cheie publică, un utilizator deţine două chei: o cheie publică şi o cheie privată (secretă). Utilizatorul îşi poate face cunoscută oricui cheia publică. Fie *SK* cheia secretă şi *PK* cheia publică corespunzătoare. Atunci:

$$m=d(SK, e(PK, m)),$$

adică utilizatorul poate decripta cu cheia privată ceea ce oricine altcineva a criptat cu cheia publică corespunzătoare.

Cu al doilea algoritm de criptare cu cheie publică utilizatorul poate cripta un mesaj cu cheia privată, iar mesajul poate fi decriptat doar cu cheia publică corespunzătoare

$$m=d(PK, e(SK, m)).$$

Aceste două proprietăţi presupun că cele două chei, publică şi privată, pot fi aplicate în orice ordine (algoritmul asimetric RSA nu face distincţie între cheia publică şi cheia privată; orice cheie din perechea de chei poate fi folosită fie ca cheie publică, fie ca cheie privată).

O schemă de criptare cu cheie publică conține trei algoritmi (figura 7.1):

1. *generatorul de chei*, care returnează o pereche cheie secretă-cheie publică (*SK*, *PK*);
2. *algoritmul de criptare*, care primeşte la intrare un mesaj *m* din mulţimea mesajelor posibile, o cheie publică *PK* şi returnează textul cifrat *c*.
3. *algoritmul de decriptare*, care ia ca intrare un text cifrat *c* din mulţimea textelor cifrate, o cheie secretă *SK* şi returnează un mesaj *m*.

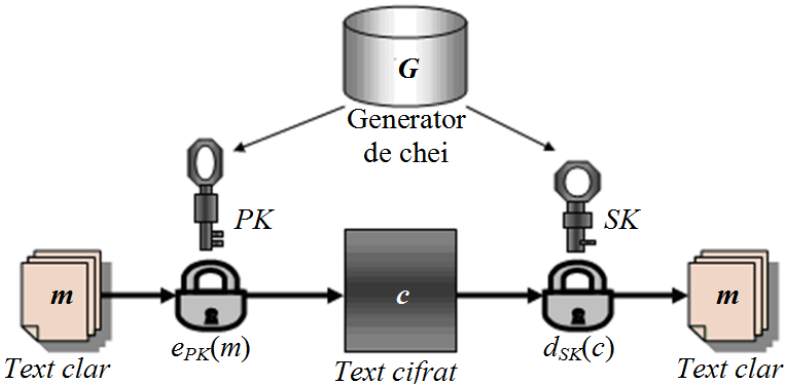

Figura 7.1. *Schema unui algoritm cu cheie publică*

Trebuie de menţionat aici unele condiţii care trebuie respectate:

- receptorul *B* poate uşor să genereze cheia publică PK_B şi cheia privată SK_B;
- emiţătorul *A*, ştiind cheia publică a lui *B* şi mesajul clar *m*, poate să genereze textul cifrat corespunzător:

$$c = e_{PK_B}(m);$$

- receptorul *B* poate uşor să decripteze textul cifrat *c*:

$$m = d_{SK_B}(c) = d_{SK_B}(e_{PK_B}(m));$$

- un atacator care ştie PK_B nu poate să determine cheia privată SK_B;
- un atacator care ştie cheia publică PK_B şi textul cifrat *c* nu poate să determine mesajul original *m*;
- are loc următoarea relaţie (menţionată mai sus):

$$m = d_{SK_B}(e_{PK_B}(m)) = d_{PK_B}(e_{SK_B}(m)).$$

La momentul actual cei mai cunoscuți algoritmi de criptare cu cheie publică sunt:

- algoritmul *RSA*: se bazează pe dificultatea descompunerii în factori primi a numerelor mari (de sute de cifre); este sistemul cel mai larg utilizat în acest moment;
- algoritmul *ElGamal*: se bazează pe dificultatea calculului logaritmului discret într-un corp finit;
- algoritmul *Merkle-Hellman*: primul algoritmul definit cu cheie publică, bazat pe problema {0, 1} a rucsacului, problemă *NP* – completă;
- algoritmul *McEliece*: este bazat pe teoria algebrică a codurilor, decodificarea unui cod liniar fiind de asemenea o problemă *NP* – completă;
- *curbe eliptice*: sunt algoritmi de criptare care îşi desfăşoară calculele pe mulţimea punctelor unei curbe eliptice.

7.2. Algoritmul schimbului de chei Diffie-Hellman

Atât conceptul criptării cu chei publice cât şi primul algoritm publicat care folosea chei publice a fost dezvoltat de W. Diffie şi M. Hellman. Algoritmul este numit *Diffie-Hellman key exchange* (1976) este folosit în numeroase produse comerciale. Acest algoritm nu se aplică la criptarea mesajelor sau la crearea de semnături digitale. Scopul său este *distribuirea cheilor*, adică scopul este ca doi utilizatori să poată schimba o cheie secretă în siguranţă, deci algoritmul este limitat la schimbul cheilor secrete.

7.2.1. Problema logaritmului discret

Algoritmul Diffie-Hellman este bazat pe problema complicată a determinării logaritmului discret. Să încercăm o explicaţie succintă acestei noţiuni.

În matematică, un *grup* este o structură algebrică ce constă dintr-o mulțime pe care este definită o operație (în continuare vom considera operația de înmulțire) care combină două elemente ale mulțimii pentru a forma un al treilea element al aceleiași mulțimi. Pentru a fi un grup, mulțimea și operația trebuie să satisfacă o serie de condiții, denumite axiomele grupurilor, și anume asociativitatea, existența elementului neutru și a elementului simetric. Grupul pe care este definită operația de înmulțire se numește *grup multiplicativ*. Ordinul unui grup *G*, notat *ord*(*G*) sau |*G*|, este numărul elementelor grupului.

Un *grup ciclic* este un grup ale cărui elemente sunt puteri ale unui element *a* din grup (când operația de grup este considerată a fi de natură aditivă, se preferă termenul multipli). În notația multiplicativă, elementele grupului sunt: ..., a^{-3}, a^{-2}, a^{-1}, $a^0 = e$, a, a^2, a^3, ..., unde a^2 înseamnă $a \cdot a$, și a^{-3} reprezintă $a^{-1} \cdot a^{-1} \cdot a^{-1} = (a \cdot a \cdot a)^{-1}$ etc. Un astfel de element *a* se numește generator sau element primitiv al grupului, care de obicei se notează cu α sau g.

Pentru comoditate şi exactitate vom considera G – un grup multiplicativ $\boldsymbol{Z}_p^*$ de ordin $p-1$, cu p – număr prim, unde operaţia este înmulţirea *modulo p*.

Exemplu. Fie $p = 11$ și $\alpha=2$. Atunci $\boldsymbol{Z}_{11}^* = \{2^1 mod 11, 2^2 mod 11, \ldots, 2^{10} mod 11\} =$

$=\{2, 4, 8, 5, 10, 9, 7, 3, 6, 1\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Așadar, $\boldsymbol{Z}_{11}^*$ este un grup ciclic multiplicativ de ordinul $n=10$ și elementul generator $\alpha = 2$.

Fie G - un grup ciclic de ordinul n şi α – elementul generator al său. Fie β – un element din G. *Logaritmul discret* β în baza α, care se notează cu $\log_\alpha \beta$, este unicul număr întreg a, $0 \le a \le |G| - 1$, astfel încât $\beta = \alpha^a$.

Exemplu. Fie $p = 97$. Atunci $\boldsymbol{Z}_{97}^*$ este un grup ciclic multiplicativ de ordinul $n = 96$. Elementul generator al grupului este $\alpha = 5$. Deoarece $5^{32} \equiv 35 \pmod{97}$, avem $\log_5 35 = 32$ în $\boldsymbol{Z}_{97}^*$.

Unele proprietăţi ale logaritmilor discreţi:

- $\log_\alpha(\beta\gamma) = (\log_\alpha \beta + \log_\alpha \gamma) \bmod n;$
- $\log_\alpha \beta^s = s \cdot \log_\alpha \beta \bmod n.$

Problema logaritmului discret (DLP – *Discrete Logarithms Problem*) constă în următoarele:

fiind date un grup ciclic finit $G = \boldsymbol{Z}_p^*$, un generator α al lui G şi un element y din G, se cere să se găsească un număr întreg a, $0 \le a \le p-2$, astfel încât $y = \alpha^a \pmod p$.

7.2.2. Descrierea algoritmului Diffie-Hellman

Algoritmul Diffie-Hellman pentru schimbul de chei constă în următoarele (figura 7.2):

- *Alice* şi *Bob* convin asupra unui număr mare prim p şi a unui generator α;
- Alice alege (generează aleator) un număr secret a, şi îl trimite lui Bob numărul A:

$$A = \alpha^a \pmod p;$$

- Bob alege (generează aleator) un număr secret b, şi îl trimite lui Alice numărul B:

$$B = \alpha^b \pmod p;$$

- Alice calculează $B^a \bmod p = (\alpha^b \bmod p)^a \bmod p) = k$;
- Bob calculează $A^b \bmod p = (\alpha^a \bmod p)^b \bmod p) = k$.

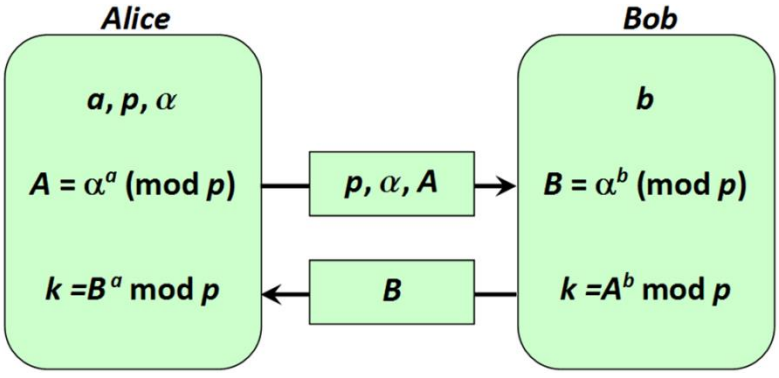

Figura 7.2. *Schema schimbului de chei Diffie-Hellman*

Exemplu de utilizare a schimbului de chei Diffie-Hellman:

- Alice şi Bob convin asupra lui p = **23** şi α = **5**.
- Alice alege aleatoriu a = **6** şi îi trimite lui Bob $A = 5^6 \bmod 23 = \mathbf{8}$.
- Bob alege aleatoriu b = **15** şi trimite lui Alice $5^{15} \bmod 23 = \mathbf{19}$.
- Alice calculează $19^6 \bmod 23 = \mathbf{2}$.
- Bob calculează $8^{15} \bmod 23 = \mathbf{2}$.
- Alice şi Bob au abţinut aleşi rezultat, deci cheia secretă comună este k = **2**.

Cheia secretă stabilită de cei doi utilizatori Alice şi Bob este k. Menţionăm că p şi α nu este necesar de ţinut în secret.

În realizările practice ale Algoritmului Diffie-Hellman *NIST* recomandă pentru perioada 2016-2030 utilizarea cheilor de cel puțin 224 biți, iar a modulului p – de cel puțin 2048 biți, ceea ce corespunde în reprezentarea zecimală la 68 și respectiv 617 cifre (acești parametri sunt recomandați pentru toți algoritmii care funcționează în baza problemei logaritmului discret). Numărul α (generatorul) nu este neapărat mare şi de obicei are valori mai mici ca 10. Ţinând cont de aceste mărimi ale parametrilor a, b şi p putem afirma cu certitudine că securitatea algoritmului este asigurată, deoarece determinarea cheii secrete cunoscând numai α, p, $\alpha^b \bmod p$, $\alpha^b \bmod p$ (fără a cunoaşte a şi b – ele fiind menţinute în secret) este o problemă extrem de complicată care nu poate fi rezolvată într-un timp rezonabil.

Algoritmul Diffie-Hellman poate fi utilizat şi pentru criptarea cu cheie secretă. În acest caz schema este aceeaşi ca şi mai sus însă cu unele particularităţi. Alice nu transmite direct lui Bob valorile p, α şi A însă le publică din timp, ele având calitatea de cheie publică. Bob la rândul său efectuează calculele sale, după care cifrează mesajul cu un algoritm simetric, folosind k în calitate de cheie, apoi transmite spre Alice textul cifrat împreună cu valoarea lui B. În practică însă algoritmul Diffie-Hellman nu este utilizat pentru criptare, deoarece există alți algoritmi cu cheie publică mai eficienți.

7.3. Algoritmul de criptare ElGamal

Unul dintre algoritmii bazați pe problema logaritmului discret, este algoritmul *ElGamal*, care conţine în sine şi algoritmul de semnătură digitală. *Schema ElGamal* se află la baza standardului de semnătură digitală în SUA (standardul DSA), precum şi în Federația Rusă (ГОСТ Р 34.10-94) ambele suferind modificări pentru sporirea securității.

Schema a fost propusă de către Taher Elgamal[1] în anul 1984. Elgamal a elaborat una din versiunile algoritmului Diffie-Hellman. El a perfecţionat algoritmul Diffie-Hellman şi a obţinut doi

[1] *Taher Elgamal*, (născut la 18 august 1955) este un criptograf egiptean. Este autorul algoritmului de criptare și a schemei de semătură digitală ce îi poartă numele. De asemenea, a participat la elaborarea protocolului „SET” de plată cu carduri de credit și la o serie de scheme de plată prin Internet.

algoritmi care sunt utilizaţi pentru criptare şi pentru asigurarea autenticităţii. Algoritmul ElGamal nu a fost brevetat şi deci este o alternativă mai puţin costisitoare decât altele, deoarece nu este necesară achitarea unei taxă pentru licenţă. Se consideră că algoritmul ElGamal este acoperit de brevetul Diffie-Hellman.

Ca și oricare alt algoritm de criptare, ElGmal constă din trei module.

1. *Generarea cheilor*:
   - se generează aleatoriu un număr prim p de n biţi;
   - se alege aleatoriu elementul primitiv α al lui $\boldsymbol{Z}_{\boldsymbol{p}}^{*}$;
   - se alege aleatoriu un număr întreg a încât $1 < a \leq p - 2$;
   - se calculează $y = \alpha^{a} \pmod{p}$;
   - cheia publică este tripletul (p, α, y), cheia privată – numărul a.
2. *Cifrarea* mesajului m, (m trebuie mai întâi codificat cu ajutorul unuia dintre sistemele de codificare a caracterelor, aspre exemplu ASCII) în algoritmul ElGamal este de fapt o modalitate de generare a cheii publice Diffie-Hellman:
   - se alege aleatoriu cheia de sesiune – un număr întreg k, $1 < k \leq p - 2$;
   - se calculează $A = \alpha^{k} \pmod{p}$ şi $B = y^{k} \cdot m \pmod{p}$;
   - textul cifrat este perechea de numere (A, B).

   Este evident că lungimea textului cifrat cu algoritmul ElGamal este de două ori mai mare decât lungimea textului clar m.
3. *Descifrarea*. Cunoscând cheia privată a, textul clar m poate fi calculat din textul cifrat (A, B) în felul următor:
   - $m = B \cdot (A^{a})^{-1} \pmod{p}$;

   aceasta este posibili deoarece
   - $(A^{a})^{-1} = \alpha^{-ka} \bmod p$;
   - $B \cdot (A^{a})^{-1} = (\alpha^{ka} \cdot m) \cdot \alpha^{-ka} = m \bmod p$;

   în scopuri practice pentru descifrare este mai convenabilă formula
   - $m = B \cdot (A^{a})^{-1} = B \cdot A^{(p-1-a)} \pmod{p}$.

Deoarece în schema ElGamal se introduce o mărime aleatoare k acest cifru poate fi considerat cifru polialfabetic. Alegerea aleatorie a lui k a generat noţiunea de schemă probabilistică de cifrare. Caracterul probabilistic cal cifrării reprezintă o prioritate a cifrului ElGaml, deoarece schemele probabilistice au o rezistenţă mai mare decât alte scheme cu un proces determinist de cifrare. Dezavantajul schemei ElGamal îl reprezintă dublarea lungimii textului cifrat în raport cu textul clar. Pentru schemele probabilistice de cifrare mesajul m şi cheia nu determină univoc textul cifrat. În schema ElGamal este necesar de a alege valori diferite ale mărimii aleatoare k pentru cifrarea a două mesaje diferite m şi m'. Dacă am folosi aceleaşi valori ale lui k pentru textele cifrate

(A, B) şi (A', B') atunci din relaţia $B \cdot (B')^{-1} = m \cdot (m')^{-1}$, se poate uşor de calculat m' dacă cunoaştem m.

Exemplu de utilizare a algoritmului ElGamal; textul clar este m = ”*A*” = 065 = 65, pe care Alice trebuie să-l trimită cifrat lui Bob. Algoritmul ElGamal va consta în următoarele:

1. *Generarea cheilor*
   - se alege aleatoriu numărul prim p = 281;
   - se alege aleatoriu elementul primitiv α = *3* al câmpului $\boldsymbol{Z}_{281}^{*}$;
   - se alege aleatoriu un număr întreg a = 57 încât $1 < a \leq 279$;
   - se calculează $y = \alpha^a \pmod p = 3^{57} mod\ 281 = 258$;
   - cheia publică este tripletul (p, α, y) = (**281**, **3**, **258**);
   - cheia privată este numărul a = **57**.
2. *Cifrarea*
   - Alice alege aleatoriu cheia de sesiune k, $1 < k \leq 279$, fie k = 49;
   - ea calculează;

     $A = \alpha^k \pmod p = 3^{49} \ mod\ 281 = 146$ şi

     $B = y^k \cdot m \pmod p = (3^{49} \cdot 65)\ mod\ 281 = (152 \cdot 65)\ mod\ 281 = 45$;
   - textul cifrat este perechea de numere (A, B) = (146, 45);
   - Alice transmite lui Bob textul cifrat (146, 45).
3. *Descifrarea*. Bob, fiind titularul cheii secrete a = 57, primeşte textul cifrat (146, 45) de la Alice. Pentru descifrarea acestui mesaj el procedează în felul următor:
   - $m = B \cdot (A^a)^{-1} (mod\ p) = 45 \cdot (146^{57})^{-1} (mod\ 281)$.
   - pentru aceasta Bob poate calcula mai întâi $146^{57} \bmod 281 = 152$;
   - apoi calculează $152^{-1}\ mod\ 281 = 220$ (pentru aceasta poate aplica algoritmul Euclid Extins).
   - calculează textul clar

     $m = (45 \cdot 220)\ mod\ 281 = 9900\ mod\ 281 = 65 = 065 =$ “*A*”.
   - Textul clar putea fi calculat şi fără a aplica calculul inversului – aplicând formula $m = B \cdot A^{(p-1-a)} (mod\ p)$:

$m = 45 \cdot 146^{(281-1-57)} (mod\ 281) = 45 \cdot 146^{223} (mod\ 281) = 45 \cdot 220 (mod\ 281) = 9900 (mod\ 281) = 65.$

7.4. Algoritmul de criptare RSA

Un alt algoritm de criptare bazat pe o problemă dificilă este *Algoritmul RSA*, numit astfel după inventatorii săi, Rivest, Shamir[2] şi Adelman[3]. A fost publicat în 1978 şi rămâne un algoritm foarte folosit şi astăzi, în ciuda eforturilor criptanaliştilor de a-l sparge.

7.4.1. Descrierea algoritmului RSA

Algoritmul de criptare RSA incorporează rezultate din teoria numerelor, combinate cu dificultatea determinării factorilor primi pentru un număr ţintă. RSA la fel operează cu aritmetica modulo *n*. Un bloc în text clar este tratat ca un întreg, iar pentru criptare şi decriptare se folosesc două chei, *e* şi *d*, care sunt interschimbabile.

Problema pe care se bazează algoritmul de criptare este cea a factorizării numerelor mari. Problema factorizării nu se cunoaşte a fi *NP*-completă; cel mai rapid algoritm cunoscut este exponenţial în timp.

La momentul actual în RSA se utilizează numere prime găsite prin intermediul algoritmilor probabilistici, cel mai performant fiind testul Miller-Rabin. El este considerat suficient de bun pentru generarea numerelor prime aplicate în criptografie. Dacă însă acest test ar genera la un moment dat un număr compus, urmările vor fi imprevizibile pentru utilizatorii algoritmului RSA cu cheile respective.

Teoretic sunt trei posibilităţi de abordare a unui atac în cazul algoritmului RSA: atacul în forţă, atacul bazat pe metode matematice (încercarea factorizării produsului a două numere prime mari) şi atacul temporal. Analiza acestor atacuri duce la concluzia că nici unul nu are sorţi de izbândă. În pofida unor intense cercetări, au fost identificate doar probleme minore în comparaţie cu cele din cazul algoritmului rucsacului a lui Merkle şi Hellman.

RSA constă din următorii 3 algoritmi: generatorul de chei, algoritmul de criptare şi algoritmul de decriptare.

1. *Generatorul de chei*:
   - generează două numere prime mari *p* şi *q*;
   - calculează $n = pq$ şi indicatorul Euler $\varphi(n) = (p - 1)(q - 1)$;
   - alege aleator un număr *e*, astfel încât $1 < e < \varphi(n)$ și cmmdc(e, $\varphi(n)$) = 1;
     calculează $d = e^{-1} \bmod \varphi(n)$, adică $d \cdot e = 1 \bmod \varphi(n)$) folosind spre exemplu algoritmul extins al lui Euclid;
   - face public *n* şi *e*; astfel cheia publică este (*e, n*) iar cea privată este (*d, n*).

[2] *Adi Shamir* (n. 1952, Tel Aviv, Israel) este un informatician și criptograf israelian, cu numeroase contribuții în domeniul informaticii și criptanalizei, cunoscut mai ales pentru rolul său în dezvoltarea algoritmului de criptare cu chei publice RSA și a schemei de indentificare Feige-Fiat-Shamir.

[3] *Leonard Max Adleman* (n. 31 decembrie 1945, San Francisco, SUA) este un informatician și biolog american, cunoscut mai ales pentru rolul său în dezvoltarea algoritmului de criptare cu chei publice RSA.

2. *Algoritmul de criptare*:

Fie *m* – mesajul în clar dat sub forma unui număr natural mai mic decât *n* (*m* poate fi reprezentat zecimal în conformitate cu tabelul *ASCII* de codificare a caracterelor). Atunci textul cifrat, notat cu *c*, este

$$c = m^e \bmod n.$$

3. *Algoritmul de decriptare*:

Fie *c* – textul cifrat primit. Atunci textul clar *m* în reprezentarea numerică este

$$m = c^d \bmod n,$$

după care, folosind aceeași metodă de codificare a caracterelor, se obține reprezentarea originală a lui *m*.

Din motive de securitate, numerele întregi *p* și *q* ar trebui să fie alese la întâmplare și ar trebui să fie similare ca mărime, dar să difere în lungime cu câteva cifre pentru a face factorizarea lui *n* mai dificilă. În același timp este recomandată utilizarea unei chei *n* de cel puțin 2048 biți, iar pe dispozitivele care permit – de 4096 biți, deoarece cheile de lungime mai mică sau deja pot fi sparte sau ar putea fi în viitorul apropiat. În același timp publicația specială NIST privind securitatea informatică (SP 800-78 Rev 1 din august 2007) nu permite fixarea exponentului cheii publice *e* mai mici de 65537, dar nu menționează motivul acestei restricții.

Remarcă: pentru cifrurile care utilizează calculele modulo *n* trebuie ținut cont de faptul că în cazul în care valoarea zecimală a lui *m* depăşeşte mărimea modulului *n*, se va diviza *m* în câteva blocuri, care nu vor începe cu 0 și vor fi mai mici decât valoarea *n* a modului, apoi vor fi criptate separat.

7.4.2. Algoritm de calculare rapidă a puterii modulo n

Pentru calcularea rapidă a lui $a^b \bmod n$ sunt elaborați mai mulți algoritmi, iar unul dintre aceştia constă în următoarele:

- se determină reprezentarea binară a lui *a*;
- fiecărei unităţi din această reprezentare binară îi corespund operaţiile de ridicare la pătrat urmată de înmulţirea cu baza $((x^2 \bmod n) \cdot a) \bmod n$, iar pentru fiecare 0 – numai înmulţirea cu baza $(x \cdot a) \bmod n$, unde *x* este rezultatul obținut la pasul anterior;
- prima unitate din această reprezentare nu o luăm în considerare (pe primul loc numaidecât se află 1și $((1^2 \bmod n) \cdot a) \bmod n = a$), deci calculele le începem cu *x*=*a* și în continuare operăm cu rezultatele curente obținute.

7.4.3. Algoritmul Euclid extins

Pentru calcularea inversului *d* al uni număr întreg *e* în clasele de resturi *modulo n* putem aplica algoritmul Euclid extins, care poate fi reprezentat cu ajutorul tabelului 7.1. În acest tabel avem:

$$x_1 = 1,\ y_1 = 0,\ r_1 = n,\ x_2 = 0,\ y_2 = 1,\ r_2 = e.$$

$$q_i = \left[\frac{r_{i-1}}{r_i} \right],\ i = 2,3,\ldots,$$

unde [*x*] reprezintă partea întreagă a numărului *x*. Pentru $i = 3, 4, 5 \ldots$ avem:

$$r_i = r_{i-2} \bmod r_{i-1},\ ,$$

$$x_i = x_{i-2} - q_{i-1} \cdot x_{i-1},$$

$$y_i = y_{i-2} - q_{i-1} \cdot y_{i-1}.$$

Procesul continuă până când obţinem $r = 1$. În acest caz (la iteraţia *k*) $e^{-1} \bmod n = y_k$. În cazul în care $y_k < 0$ se adună modulul *n*. De menționat că coloana care îi corespunde lui *x* nu este necesară la calcularea inversului unui număr, ea fiind totuși parte a algoritmului Euclid extins și este necesară pentru calcularea cmmdc a două numere în clasa modulo.

i	*x*	*y*	*r*	*q*
1	1	0	*n*	
2	0	1	*a*	$q_2 = \left[\frac{r_1}{r_2} \right]$
3	$x_i = x_{i-2} - q_{i-1} \cdot x_{i-1}$	$y_i = y_{i-2} - q_{i-1} \cdot y_{i-1}$	$r_i = r_{i-2} \bmod r_{i-1}$	$q_3 = \left[\frac{r_2}{r_3} \right]$
…	…	…	…	…
k	$x_k = x_{k-2} - q_{k-1} \cdot x_{k-1}$	$y_k = y_{k-2} - q_{k-1} \cdot y_{k-1}$	1	$q_k = \left[\frac{r_{k-1}}{r_k} \right]$

Tabelul 7.1. *Schema algoritmului Euclid extins*

Notă: În cazul în care dispunem de un algoritm rapid de calcul al puterii *modulo* putem calcula inversul modular folosindu-ne de formula

$$x^{-1} \bmod n = x^{n-2} m \bmod n.$$

7.4.4. Exemplu de utilizare RSA

Fie că *Bob* intenționează să facă schimb de mesaje criptate cu *Alice*. Pentru aceasta *Bob* trebuie mai întâi să genereze o pereche de chei pentru algoritmul *RSA* şi să-i transmită lui *Alice* cheia sa publică. *Alice* va cripta cu această cheie mesajul ‚*A*' şi va trimite lui *Bob* textul cifrat, care, folosind cheia sa privată, trebuie să-l decripteze.

Pentru generarea cheilor *Bob* alege 2 numere prime:

p = 31 şi q = 23.

Calculează produsul

$$n = 31 \cdot 23 = 713.$$

Apoi calculează indicatorul Euler

$$\varphi(n) = (p - 1)\cdot(q - 1) = 30 \cdot 22 = 660.$$

În continuare *Bob* alege aleator un număr e astfel ca $1 < e < \varphi(n)$ și cmmdc(e, $\varphi(n)$) = 1, adică $1<e<660$ şi cmmdc(e, 660) = 1. Fie că el alege

$$e = 223.$$

Calculează $d = 223^{-1} \bmod 660$ aplicând, spre exemplu, algoritmul Euclid extins (tabelul 7.2).

i	y	r	q
1	0	660	
2	1	223	2
3	-2	214	1
…	3	9	23
k	–71	7	1
	74	2	3
	–293	**1**	2
		Stop!	

Tabelul 7.2. *Schema algoritmului Euclid extins pentru* $d = 223^{-1}$ *mod* 660

Aşadar,

$$d = 223^{-1} \textit{ mod } 660 = -293 \textit{ mod } 660 = -293 + 660 = 367$$

Deci cheia privată (secretă) a lui *Bob* este *SK*=(367, 713).

Cheia sa publică *PK*= (223, 713) – Bob o transmite lui *Alice*.

Alice trebuie să trimită mesajul „*A*” lui *Bob* cifrând-ul cu algoritmul *RSA*. Conform tabelului *ASCII* valoarea zecimală a lui „*A*” este 065=65. Deci m = 65. Pentru aceasta *Alice* calculează

$$c = 65^{223} \bmod 713,$$

aplicând algoritmul de calcul rapid al puterii modulare explicat mai sus. Reprezentarea binară a exponentului 223 este:

$$223_{10} = 11011111_2.$$

Aşadar,

$$\begin{aligned} 65^{223} \bmod 713 = (((((((((((65^2 \bmod 713) \cdot 65 \bmod n)^2 \bmod n)^2 \bmod n) \cdot \\ \cdot 65 \bmod n)^2 \bmod n) \cdot 65 \bmod n)^2 \bmod n) \cdot \\ \cdot 65 \bmod n)^2 \bmod n) \cdot 65 \bmod n)^2 \bmod n) \cdot 65 \bmod n). \end{aligned}$$

Efectuând consecutive calculele se obţine

$$65^{223} \bmod 713 = 396.$$

Alice trimite lui *Bob* textul cifrat

$$c = 396.$$

Bob aplică aceeaşi metodă şi calculează

$$m = c^d \bmod n = 396^{367} = 65 = 065,$$

adică conform tabelului *ASCII* – litera „*A*”.

Dacă numărul *n* nu este suficient de mare, atunci algoritmul poate fi spart. În cazul nostru *n*=713. Acest număr poate fi foarte simplu factorizat: $n = 713 = 31 \cdot 23$. Atacatorul cunoaşte cheia publică, deci ştie că $e = 223$ şi la fel ca *Bob* calculează *d* şi obţine cheia lui privată, spărgând astfel algoritmul.

7.5. Infrastructura cheilor publice

Una dintre principalele probleme asociate cu criptografia este punerea cheilor private la dispoziţia utilizatorilor autorizaţi, fără a le dezvălui altcuiva. Alte probleme principale includ fixarea cheilor publice de entităţi, permiţând altor entităţi verificarea acestora, precum şi furnizarea serviciilor necesare pentru gestionarea continuă a cheilor într-un sistem distribuit. Astfel a apărut necesitatea de o infrastructură accesibilă care să organizeze circulația cheilor publice și anume - *Infrastructura Cheii Publice* (sau *PKI - Public key infrastructure*). PKI poate accelera şi simplifica livrarea produselor şi serviciilor prin oferirea de abordări electronice.

7.5.1. Noțiune de Infrastructura Cheii Publice

O infrastructură de chei publice este un set de roluri, politici și proceduri necesare pentru crearea, gestionarea, distribuirea, utilizarea, stocarea și revocarea certificatelor digitale și gestionarea criptării cu chei publice. Scopul unei PKI este de a facilita transferul electronic sigur de informații pentru o serie de activități de rețea, cum ar fi comerțul electronic, Internet banking, e-mailurile confidențiale, etc. PKI Este necesară pentru activitățile în care parolele simple sunt o metodă inadecvată de autentificare și sunt necesare probe mai riguroase pentru a confirma identitatea părților implicate în comunicare și pentru a valida informațiile transferate.

Termenul „infrastructura cheilor publice” este derivat din criptografia cu chei publice, tehnologie pe care se bazează PKI. Ea are caracteristici unice care o fac nepreţuită ca bază pentru funcţiile de securitate în sisteme distribuite. PKI este o combinaţie de software, tehnologii de criptare şi servicii care permite organizaţiilor să protejeze securitatea comunicaţiilor şi a tranzacţiilor lor de afaceri în reţele. PKI integrează certificatele digitale, criptografia cu cheie publică şi autorităţile de certificare într-un set complet al unei arhitecturi de securitate cu un nivel înalt de organizare.

Un exemplu tipic de PKI cuprinde: emiterea de certificate digitale pentru utilizatorii individuali şi servere; software-ul necesar; instrumente de integrare a certificatelor; instrumente de gestionare, de reînnoire, precum şi de retragere a certificatelor; servicii conexe şi de asistenţă.

Elementele funcţionale ale unei infrastructuri de chei publice includ *autorităţile de certificare* (CA - certificate authority), *autorităţile de înregistrare* (Registration authority), repozitorii şi arhive:

- o autoritate de certificare (CA), similar unui notar, eliberează sau revocă certificate;
- o autoritate de înregistrare (RA), o entitate care este încredinţată de CA să înregistreze sau să garanteze asocierea dintre cheile publice şi identitatea titularului certificatului cu o CA;
- un repozitoriu este o bază de date de certificate digitale active şi liste de revocare a certificatelor (CRL - Certificate Revocation List) pentru un mediu specific al PKI;
- o arhivă este o bază de date de informaţii pentru a fi utilizate în soluţionarea eventualelor viitoare litigii;
- utilizatorii PKI sunt organizaţii sau persoane care utilizează PKI, dar care nu eliberează certificate.

O *autoritate de certificare* este componenta de bază a PKI, împreună cu o colecţie de hardware, software şi oameni care le operează. CA este cunoscută după numele acesteia şi cheia sa publică și îndeplineşte patru funcţii de bază ale PKI:

1. eliberează certificate;
2. menţine informaţii cu privire la starea certificatelor şi eliberează fişiere CRL;
3. publică certificate curente şi fişiere CRL;
4. menţine arhivele de informaţii despre starea certificatelor expirate care au fost eliberate.

O autoritate de certificare emite un certificat digital pentru fiecare identitate, confirmând că identitatea are prerogative corespunzătoare. Un certificat digital include de obicei cheia publică, informaţii despre identitatea părţii care deţine cheia privată corespunzătoare, perioada de funcţionare pentru certificat şi semnătură digitală proprie CA. Autoritatea trebuie, de asemenea, să elibereze şi să prelucreze listele de revocare a certificatelor (fişiere CRL), care sunt nişte liste de certificate ce au fost revocate deja.

O *autoritate de înre*gistrare este proiectată pentru a verifica conţinutul certificatului pentru CA. Ca şi autoritatea de certificare, o RA este o colecţie de hardware, software, şi persoana sau persoanele care le operează. Spre deosebire de CA, o RA va fi deseori exploatată de către o singură persoană. Fiecare CA va păstra o listă de autorități de înregistrare acreditate; o RA este cunoscută pentru CA după numele şi cheia publică ale acesteia.

Un *repozitoriu* este necesar pentru distribuirea certificatelor şi a informaţiilor cu privire la starea certificatului. Acest repozitoriu reprezintă un mijloc de stocare şi distribuire a certificatelor, precum şi de gestionare a actualizării certificatelor.

O *arhivă* acceptă responsabilitatea pentru stocarea pe termen lung a informaţiilor în favoarea CA. O arhivă afirmă că informaţia a fost bună în momentul în care a fost primită, şi nu a fost modificată în timp ce s-a aflat în arhivă. Arhiva protejează informaţiile prin intermediul unor mecanisme tehnice şi proceduri adecvate în timp ce le are în grija sa. În cazul în care apare un diferend cu privire la o semnătură la o dată ulterioară, aceste informaţii pot fi utilizate pentru a verifica dacă cheia privată asociată cu certificatul a fost utilizată pentru a semna un document.

Utilizatorii PKI se bazează pe alte componente ale infrastructurii de chei publice ca să obţină certificate şi să verifice certificatele altor entităţi cu care fac afaceri. Aceste entităţi includ:

- partea care se bazează pe certificat pentru ca să ştie, cu certitudine, cheia publică a unei alte entităţi;
- titularul certificatului, adică partea care are eliberat un certificat şi poate semna documente digitale.

7.5.2. Modele actuale de interoperabilitate PKI

Titularii de certificate îşi vor obţine certificatele de la Autorități de Certificare diferite, în funcţie de organizaţia sau comunitatea în care ei sunt membri. Un PKI este de obicei compus din mai multe CA legate prin căi de încredere. O cale de încredere leagă o parte de bază cu una sau mai multe părţi terţe de încredere, astfel încât partea de bază poate avea încredere în valabilitatea certificatului în uz. Există două arhitecturi tradiţionale PKI pentru a sprijini acest obiectiv, *arhitectura ierarhică* şi *arhitectura plasă*. O a treia abordare, *arhitectura CA pod* a fost dezvoltată pentru a se adresa situaţiei în care organizaţiile se confruntă în încercarea de a lega propriile PKI-uri cu cele ale partenerilor lor de afaceri.

În *PKI Ierarhice* (*Hierarchical Architectures*) Autorităţile sunt aranjate ierarhic în cadrul unei CA „rădăcină”, (root) care eliberează certificate la CA subordonate. Aceste CA pot elibera certificate pentru CA sau utilizatori aflaţi mai jos în ierarhie. Într-un PKI ierarhic, fiecare parte de bază cunoaşte cheia publică a CA rădăcină. Orice certificat poate fi verificat prin verificarea căii de certificare a certificatelor de la CA rădăcină.

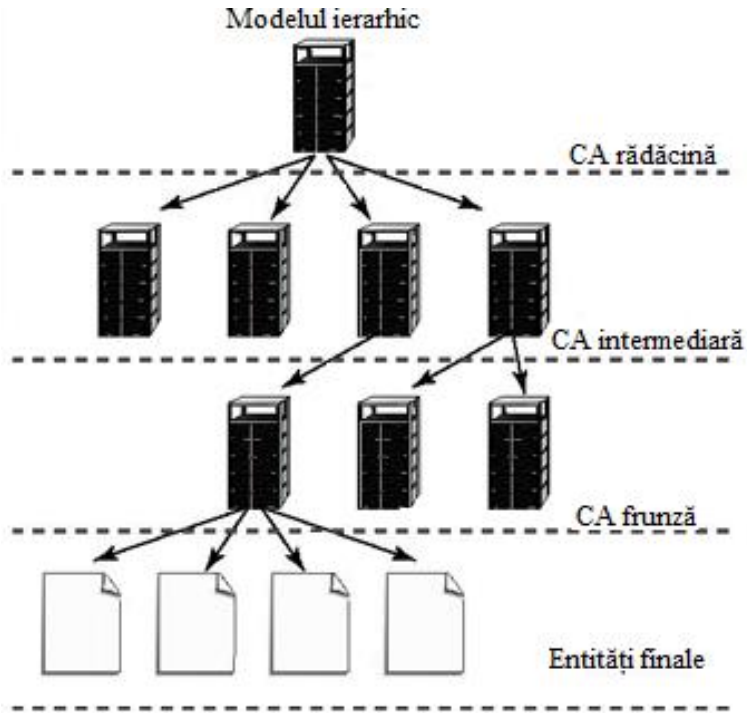

Figura 7.3. *Modelul PKI ierarhic*

Sistemele CA rădăcină pot avea încredere între ele, precum și între CA intermediare și cele de tip frunză. O CA frunză este doar un CA care se află la sfârșitul unei rețele sau lanțuri CA. Această structură permite a fi creativ și eficient la crearea sistemelor hibride.

În *PKI Plasă* (*Mesh Architecture*) CA independente se certifică reciproc (adică își eliberează reciproc certificate), rezultând într-o plasă generală a relației de încredere între CA egale. O parte de bază știe cheia publică a unei „CA apropiate", în general, cea care a emis certificatul său. Partea de bază verifică certificatul prin verificarea unei căi de certificare a certificatelor care duce de la CA de încredere.

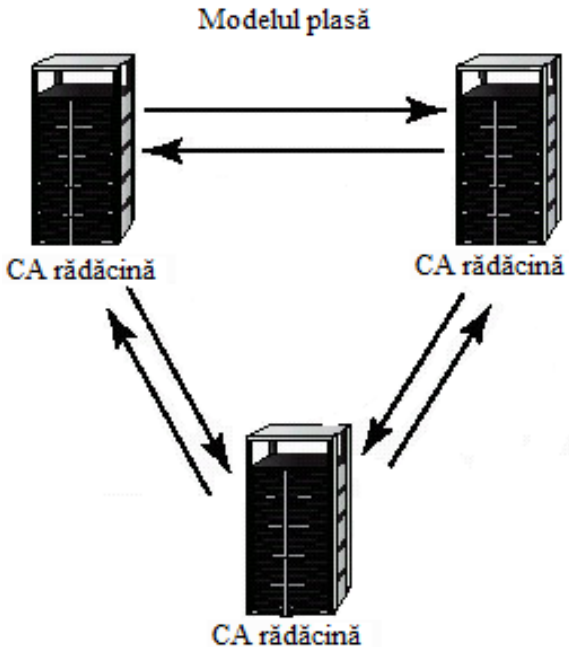

Figura 7.4. *Modelul PKI plasă*

PKI Pod (*Single Architecture*) a fost concepută pentru a conecta PKI-uri organizaţionale, indiferent de arhitectura acestora. Acest lucru este realizat prin introducerea unei noi CA, numită CA Pod, a cărui unic scop este de a stabili relaţii cu PKI-uri organizaţionale.

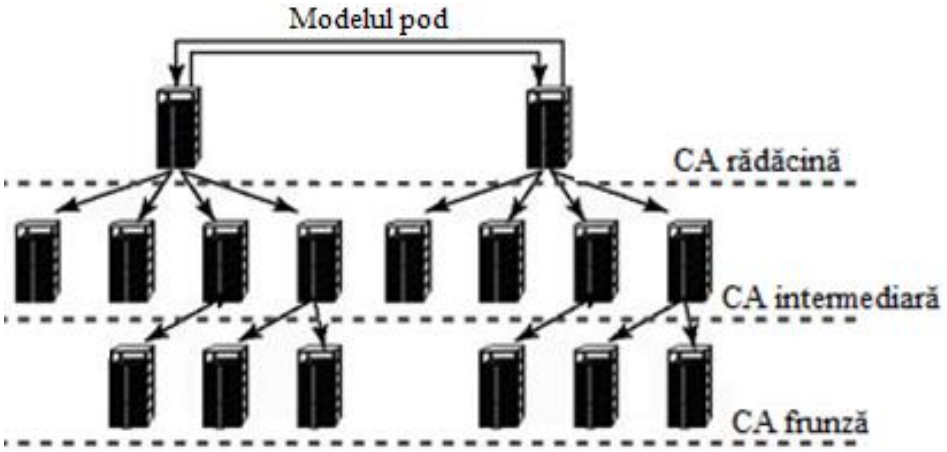

Figura 7.4. *Modelul PKI pod*

Spre deosebire de o CA Plasă, CA Pod nu eliberează certificate direct către utilizatorii finali.

Spre deosebire de o CA rădăcină într-o ierarhie, CA Pod nu este destinată utilizării ca un punct de încredere. Toţi utilizatorii PKI considera CA Pod în calitate de intermediar. CA Pod stabileşte relaţii punct cu punct cu diferite PKI organizaţionale. Aceste relaţii pot fi combinate pentru a forma un pod de încredere de utilizatori conectaţi la PKI diferite.

Dacă domeniul de încredere este implementat ca un PKI ierarhic, CA Pod va stabili o relaţie cu CA rădăcină. Dacă domeniul este implementat ca o PKI Plasă, podul va stabili o relaţie cu numai una dintre CA. În ambele cazuri, CA care intră într-o relaţie de încredere cu podul este definită ca fiind principala CA.

7.6. Întrebări și subiecte pentru aprofundarea cunoștințelor și lucrul individual

1. Care este particularitatea care deosebește criptarea cu chei publice de criptarea cu cheie secretă?
2. Fie $p = 97$ și $\alpha = 5$. Modelați schimbul de chei utilizând algoritmul Diffie-Hellman.
3. Fie $p = 97$ și $\alpha = 5$. Modelați generarea cheilor, criptarea și decriptarea mesajului M=’A’ aplicând algoritmul ElGamal.
4. Fie $p = 97$. Generați o pereche de chei pentru algoritmul RSA. Aplicând algoritmul RSA realizați criptarea și decriptarea mesajului M=’A’. Comparați rezultatele cu exempul precedent.
5. Descrieți infrastructura cheilor publice în RM. Justificați necesitatea existenței PKI.