CRIPTANALIZA. REZULTATE ŞI TEHNICI MATEMATICE

Ediţia I apărută la: Ed. Univ. Buc, 2004, ISBN 973575975-6.

Vasile PREDA, Emil SIMION şi Adrian POPESCU

Ediţia a doua 2011

Cuprins

Capitolul 9

CRIPTANALIZA CIFRURILOR CU CHEI PUBLICE

No matter resistant the cryptogram, all that is really needed is an entry, the identification of one word, of three or four letters.
Hellen Fouché Gaines, 1939

9.1. Principii de bază

9.1.1. Introducere

Algoritmii cu cheie publică îşi bazează securitatea pe dificultatea computaţională a unor probleme din domeniile informaticii teoretice sau a teoriei numerelor. Astfel acest capitol va aborda algoritmii de tip rucsac (domeniul informaticii teoretice) şi algoritmii de tip RSA, Pohlin-Hellman, Rabin, ElGamal şi ai curbelor eliptice (domeniul teoriei numerelor). Se vor studia modul de comportare al algoritmilor amintiţi mai sus punându-se accent şi pe principalele vulnerabilităţi ale acestora. Capitolul se va încheia cu o serie de concluzii referitoare la acest domeniul al criptografiei precum şi cu un paragraf destinat aplicaţiilor specifice (ca de exemplu *PKI* (*Public Key Infrastructure)*). Conceptul de criptografie cu chei publice a fost introdus de *Whitfield Diffie* şi *Martin Hellman* în [16] şi independent de aceştia de Ralph Merkle. În criptografia cu chei publice sunt două tipuri de chei: o cheie publică şi o cheie privată (termenul de cheie secretă va fi folosit pentru criptografia simetrică). Cele două tipuri de chei nu se pot deduce (computaţional acest lucru se realizează într-

un timp foarte mare). Criptografia cu chei publice se poate folosi atât la asigurarea confidenţialităţii cât şi la asigurarea autenticităţii (semnătura digitală). Numai trei algoritmi sunt siguri, din punct de vedere criptografic, pentru a fi folosiţi atât la cifrare cât şi la semnătură digitală: RSA, ElGamal şi Rabin. Acesta este un alt motiv pentru care am optat pentru prezentarea în detaliu a acestora.

De obicei criptografia simetrică şi cea asimetrică (cu chei publice) duc la construcţia sistemelor criptografice hibride: se foloseşte un algoritm simetric (bazat pe o cheie secretă aleatoare) pentru a cifra mesaje şi se utilizează un algoritm asimetric (bazat pe o cheie publică şi o cheie privată) pentru a cifra cheia secretă de sesiune.

Pentru implementarea efectivă a algoritmilor ce se vor prezenta se poate consulta *Welschenbach* [96].

9.1.2. Securitatea algoritmilor cu cheie publică

Securitatea algoritmilor cu cheie publică ce se vor prezenta se bazează pe dificultatea computaţională a următoarelor probleme:

- a problemelor $NP-$complete ca de exemplu *problema rucsacului*: dându-se mulţimea $\{M_1, ..., M_n\} \subset \mathbf{N}^*$ şi numărul $S \in \mathbf{N}^*$ să se calculeze $b_i \in \{0, 1\}$ astfel ca

$$S = \sum_{i=1}^{n} b_i M_i.$$

-a factorizării unui număr compozit (algoritmul *RSA*): dându-se un număr natural compozit (produs de numere prime) $n = p \cdot q$ să se găsească un algoritm eficient de factorizare a acestuia;

-a calculului rădăcinii pătrate modulo un număr compozit (algoritmul *Rabin*): dându-se un număr natural compozit (produs de numere prime) n să se găsească un algoritm eficient de rezolvare a ecuaţiei: $y = x^2 \bmod n$.

-a logaritmului discret (algoritmul *ElGamal*): să se găsească numărul natural $x < p$ care îndeplineşte relaţia $y = g^x \bmod p$ unde p este număr prim şi $g < p$;

9.1.3. Comparaţii ale algoritmilor asimetrici şi a algoritmilor simetrici

Din punct de vedere al cheilor *algoritmii asimetrici* (cu *cheie publică*) se deosebesc de cei *simetrici* (cu *cheie secretă*) prin tipul şi numărul de chei: o *cheie publică* şi o *cheie privată* (pentru *fiecare utilizator*) pentru cei *asimetrici* respectiv o *cheie secretă* (pentru *toţi utilizatorii*) pentru cei *simetrici*. Evaluarea unui sistem asimetric se bazează pe dificultatea computaţională a rezolvării problemelor enunţate mai sus. Evaluarea sistemelor criptografice simetrice se bazează pe demonstrarea unor proprietăţi efectiv demonstrabile ale acestuia.

9.2. Algoritmi de tip rucsac

Algoritmul de cifrare *Merkle-Hellman* constă în codificarea mesajului ca o soluţie a unei probleme de tip rucsac: ponderile $\{M_1, ..., M_n\}$ fiind cheia de cifrare, textul clar fiind $\{b_1, ..., b_n\}$ iar textul cifrat $\sum_{i=1}^{n} b_i M_i$.

9.2.1. Algoritmi rucsac supercrescător

Definiţia 9.2.1. Un şir de ponderi $\{M_1, ..., M_n\}$ se numeşte *supercrescător* dacă:

$$M_k > \sum_{i=1}^{k-1} M_i \text{ pentru orice } k.$$

Problema rucsacului supercrescător este uşor de rezolvat folosind următoarea schemă:

pentru $k = n, ..., 1$ dacă $M_k < S$ atunci $b_k = 1$ şi $S = S - M_k$
altfel $b_k = 0$.

Algoritmii de tip rucsac care nu sunt supercrescători nu sunt uşor de rezolvat şi nu există nici un algoritm rapid care să rezolve problema. Singura modalitate cunoscută de a determina dacă $b_i = 1$ constă în testarea tuturor soluţiilor. Cei mai rapizi algoritmi de testare au o complexitate exponenţială. Algoritmul Merke-Hellman se bazează pe această proprietate: *cheia privată* este şirul ponderilor pentru un rucasac supercrescător iar *cheia publică* este şirul ponderilor pentru un rucsac care are aceeiaşi soluţie. *Merkle* şi *Hellman* au găsit o metodă prin care se poate transforma o problemă a rucsacului supercrescător într-o problemă normală a rucsacului. Tehnica de conversie face apel la aritmetica modulară.

9.2.2. Crearea cheii publice din cheia privată

Având la dispoziţie o problemă de tip rucsac supercrescător (cheia privată) cu ponderile $\{M_1, ..., M_n\}$ atunci aceasta se transformă într-o problemă de tip rucsac normală (cheia publică) cu şirul ponderilor

$$\{mM_1 \bmod p, ..., mM_n \bmod p\},$$

unde m şi p sunt numere naturale prime între ele (acestea fac parte din cheia privată) şi $p > \sum_{i=1}^{n} M_i$.

9.2.3. Cifrarea

Pentru a cifra un mesaj binar acesta se va împărţi în blocuri de lungimi egale cu cardinalul mulţimii ponderilor. Cifrarea unui bloc $b_1...b_n$ va fi numărul natural

$$\sum_{i=1}^{n} b_i(mM_i \bmod p).$$

9.2.4. Descifrarea

Destinatarul mesajului cunoaşte cheia privată: ponderile originale şi valorile lui m şi p. Pentru a descifra un mesaj acesta va calcula mai întâi pe $m^{-1} \bmod p$. Se va multiplica apoi textul cifrat cu $m^{-1} \bmod p$ iar după aceea se va rezolva problema rucsacului supercrescător pentru a recupera textul original.

9.2.5. Implementarea efectivă

Implementările practice ale algoritmilor de tip rucsac au cel puţin 250 de ponderi. Ponderile sunt cu lungimi între 200 şi 400 biţi, iar modulul are o lungime între 100 şi 200 biţi. În cazul implementărilor reale atât ponderile cât şi numere m şi n sunt generate aleator. Atacul brut nu este fezabil la valorile prezentate anterior. *Shamir* a indicat o serie de condiţii în care se poate sparge un astfel de algoritm.

9.3. Algoritmul RSA

Algoritmul *RSA* a fost inventat[1] de către *Ron Rivest, Adi Shamir* şi *Leonard Adleman* fiind studiat în cadrul unor studii criptanalitice extinse. Securitatea RSA-ului se bazează pe dificultatea factorizării numerelor mari (vezi *Salomaa* [71], *Koblitz* [41] şi [42] pentru o introducere în domeniu). Cheia publică şi cheia privată sunt funcţie de o pereche de numere prime mari (de 200 de cifre sau chiar mai mari). Recuperarea textului clar din cheia publică şi textul cifrat este chivalent cu factorizarea produsului a două numere prime.

9.3.1. Descrirea principiilor de cifrare şi descifrare

Pentru generarea a două chei (publică şi privată) se aleg aleatoriu două numere prime mari p şi q. Din raţionamente de securitate p şi q au acelaşi ordin de mărime. Se va calcula produsul $n = p \cdot q$. Se va alege apoi, aleatoriu, cheia de cifrare e astfel

[1] În anul 1997 a fost făcută public faptul că James H. Ellis, Clifford Cocks şi Malcolm Williamson de la Government Communications Headquarters (GCHQ) au propus, în anul 1973, utilizarea acestui tip de algoritm.

ca e şi $(p-1)(q-1)$ să fie relativ prime. Utilizând algoritmul extins al lui Euclid vom calcula exponentul de descifrare d astfel ca

$$ed \equiv 1 \bmod (p-1)(q-1).$$

Cu alte cuvinte

$$d \equiv e^{-1} \bmod (p-1)(q-1).$$

Remarcăm faptul că d şi n sunt relativ prime. Numerele e şi n constituie cheia publică iar d este cheia privată. Cele două numere p şi q nu sunt necesare dar nu vor fi niciodată făcute publice.

Pentru a cifra un mesaj m îl vom diviza în blocuri de lungime mai mică n (cu date binare vom alege cea mai mare putere a lui 2 mai mică decât n). Dacă p şi q sunt numere prime de 100 cifre atunci n va avea sub 200 de cifre iar fiecare mesaj bloc m_i va avea sub 200 de cifre. Dacă trebuie cifrate blocuri de lungime fixă atunci vom apela la operaţia de padding cu zero. Mesajul cifrat c se va obţine prin concatenarea mesajelor c_i care au aproximativ aceeiaşi lungime. Formula de cifrare va fi:

$$c_i \equiv m_i^e \bmod n.$$

Pentru a descifra un mesaj se calculează:

$$m_i \equiv c_i^d \bmod n,$$

deoarece

$$\begin{aligned} c_i^d &\equiv (m_i^e)^d \equiv m_i^{ed} \equiv m_i^{k(p-1)(q-1)+1} \\ &\equiv m_i m_i^{k(p-1)(q-1)} \equiv m_i \bmod n. \end{aligned}$$

Observaţia 9.3.1. Pentru a evita metodele de factorizare cunoscute numerele p şi q trebuie să fie numere *prime tari*. Un număr prim p se numeşte număr prim tare dacă:

i) $p-1$ are un factor mare r;

ii) $p+1$ are un factor mare s;

iii) $r-1$ are un factor mare t.

9.3.2. Viteza algoritmilor tip RSA

În implementările hard RSA este cam de 1000 de ori mai lent decât algoritmul DES. Cea mai rapidă implementare hardware VLSI pentru RSA (cu 512 biţi modulul) este de 64 kilobiţi pe secundă (estimare realizată în anul 1996).

Alegerea judicioasă a parametrului de cifrare e poate mări viteza de cifrare a algoritmului RSA. Cele mai utilizate valori pentru e sunt 3 (recomandat de standardul PEM (*Privacy Enhanced Mail*) şi standardul PKCS#1 (*Public Key Cryptographic System*)), 17 şi $2^{16}+1$ (recomandat de standardul de *certificate digitale* X.509 şi standardul PKCS#1). Nu apar probleme de securitate prin folosirea oricăror acestor trei valori pentru e (presupunând faptul că se face un padding al mesajelor cu valori aleatoare), chiar dacă un grup de utilizatori au aceiaşi valoare pentru parametru e. Viteaza operaţiile private se pote mări prin utilizarea *lemei chinezeşti a resturilor* (CRT) dacă se stochează valorile lui p, q, $d \bmod (p-1), d \bmod (q-1)$ şi $q^{-1} \bmod p$. Utilizarea lemei chinezeşti a resturilor înlesneşte aplicarea atacurilor de tip timp (*timing attacks*: în funcţie de timpul de execuţie se pot deduce anumiţi biţi din cheie) sau atacuri de tip eroare hardware.

Utilizarea CRT

Operaţia de semnare a unui mesaj M se realizează prin exponenţierea amprentei digitale a documentului $H(M)$ cu ajutorul cheii private: $s = H(M)^d \bmod n$. Verificarea semnăturii se realizează prin comparaţia valorii $H(M)$ cu $s^e \bmod n$.

În cazurile practice valoarea lui e este un număr relativ mic, deci d are o valoare mare. Acest lucru conduce la timpi de rulare diferiţi între operaţiile private (descifrare/semnare) şi cele publice(cifrare/verificare semnătură).

Pentru optimizarea calculelor de verificare a semnăturii se poate utiliza lema chinezească a resturilor (CRT), însă acest lucru induce vulnerabilităţi în mediul de implementare.

Pentru fixarea ideilor vom nota prin $C = M^e \bmod n$ trasformarea cifrată a mesajului $M < n$. Presupunem că am calculat apriori valorile: $d_p := d \bmod (p-1)$, $\quad d_q := d \bmod (q-1)$ şi $r := q^{-1} \bmod p$. Atunci folosind faptul că $\gcd(d, (p-1)(q-1)) = 1$ avem $\gcd(d, (p-1)) = 1$ şi $\gcd(d, (q-1)) = 1$. Utilizarea teoremei lui Fermat ($C^{p-1} \equiv 1 \bmod p$ şi $C^{q-1} \equiv 1 \bmod q$) conduce la:

$$C^d \equiv C^{d_p} \bmod p$$

respectiv

$$C^d \equiv C^{d_q} \bmod q.$$

Utilizând CRT vom avea:

$$M = C^{d_q} \bmod q + q((C^{d_p} \bmod p - C^{d_q} \bmod q) r \bmod p).$$

Astfel, dacă $p > q$, sunt **precalculate** valorile:

$$d_p := (e^{-1} \bmod n) \bmod (p-1),$$
$$d_q := (e^{-1} \bmod n) \bmod (q-1),$$
$$r := q^{-1} \bmod p.$$

În **faza de calcul** se execută:

$$m_1 = c^{d_p} \bmod p,$$
$$m_2 = c^{d_q} \bmod q,$$
$$h = r(m_1 - m_2) \bmod p,$$
$$m = m_2 + hq.$$

Cheia privată ce se stochează fiind (p, q, d_p, d_q, r).

9.3.3. Securitatea RSA-ului

Securitatea algoritmului RSA este direct proporţională cu problema factorizării numerelor mari. Din punct de vedere tehnic acest lucru nu este adevărat. Este *conjecturat* faptul că securitatea algoritmului RSA depinde de problema factorizării numerelor mari. Nu s-a dovedit matematic că este necesar un factor al lui n pentru a calcula pe m din c.

Se poate ataca algoritmul RSA prin ghicirea lui $\varphi(n) = (p-1)(q-1)$. Acest lucru nu este mai simplu decât problema factorizării.

O serie de variante ale RSA s-au dovedit a fi la fel de dificile ca problema factorizării. Deasemenea au fost realizate studii care au pus în evidenţă faptul că recuperarea unor biţi de informaţie dintr-un mesaj cifrat, este la fel de greu de realizat ca decriptarea întregului mesaj.

Factorizarea lui n este cea mai uzuală metodă de atac. Orice adversar are la dispoziţie cheia publică e şi modulul de cifrare n. Pentru a găsi cheia de descifrare d trebuie factorizat n. Acest lucru se poate realiza cu ajutorul tehnologiei de factorizare. La nivelul anului 1996, factorizarea unui număr de 129 cifre era fezabilă. Deci n trebuie să fie mai mare (se poate lua n de 2048 biţi). Metoda de atac brut este mai puţin eficientă decât tehnica factorizării.

Cei mai mulţi algoritmi folosesc tehnici probabiliste pentru a a calcula numere prime p şi q. Se pune, în mod natural, întrebarea dacă algoritmii de testare a primalităţii detectează numerele compozite. O serie de algoritmi de testare a primalităţii detectează cu probabilitate tinzând la 1 numerele compozite (vezi *Schroeder* [72]).

9.3.4. Tipuri de atacuri asupra algoritmilor RSA

Atac cu text cifrat ales

O serie de atacuri se pot aplica asupra implementărilor algoritmului RSA. Aceste atacuri nu sunt asupra algoritmului propriu-zis ci asupra protocoalelor. Trebuie conştientizat faptul că folosirea algoritmului RSA nu este suficientă iar detaliile sunt de maximă importanţă. Vom prezenta trei cazuri de atac cu text cifrat ales.

Cazul 1. Interceptorul pasiv E, va monitoriza comunicaţiile lui A şi va stoca toate mesajele c cifrate cu ajutorul cheii publice a lui A. Interceptorul doreşte ca să citească mesajele clare. Matematic acest lucru revine la a afla pe m astfel ca:

$$m = c^d.$$

Pentru recuperarea lui m se va alege pentru început un număr aleatoriu $r < n$. Interceptorul va intra în posesia cheii publice e alui A şi va calcula:

$$\begin{aligned} x &\equiv r^e \bmod n, \\ y &\equiv xc \bmod n, \\ t &\equiv r^{-1} \bmod n. \end{aligned}$$

Acum interceptorul E va forţa pe A să semneze y folosind cheia sa privată. (Utilizatorul A trebuie să semneze mesajul nu să-i facă hash). Reţinem faptul că A nu a fost anterior în posesia lui y. Utilizatorul A va trimite lui E :

$$u \equiv y^d \bmod n.$$

Interceptorul E va calcula:

$$tu \bmod n \equiv r^{-1}y^d \bmod n \equiv r^{-1}x^d c^d \bmod n \equiv c^d \bmod n \equiv m.$$

Cazul 2. Fie T este un notar public digital. Dacă A doreşte un document notarial atunci acesta va apela la T. Notarul T va semna documentul cu ajutorul semnăturii digitale RSA şi-l va trimite lui A. (Nu se utilizează funcţii hash : notarul T va cifra mesajul cu ajutorul cheii sale private.)

Interceptorul M doreşte ca notarul T să semneze un document $m^{'}$ pe care acesta refuză iniţial să-l semneze (de exemplu un document care nu are o ştampilă de timp corectă). Pentru început M va alege un număr aleatoriu x şi va calcula $y \equiv x^e \bmod n$. Acesta va putea să acceseze pe e deoarece este cheia publică a notarului T. Interceptorul M va calcula $m \equiv ym^{'} \bmod n$ pe care îl trimite lui T să-l semneze. Notarul T va returna lui M mesajul semnat: $m^d \bmod n$. Falsificatorul M va calcula:

$$(m^d \bmod n)x^{-1} \bmod n \equiv (m^{'})^d \bmod n.$$

Slăbiciunea ce a fost exploatată a fost aceea că exponenţierea produsului este produsul exponenţierilor:

$$(xm)^d \bmod n \equiv x^d m^d \bmod n.$$

Cazul 3. Interceptorul E doreşte ca utilizatorul A să semneze mesajul m_3. Acesta va genera două mesaje m_1 şi m_2 astfel ca

$$m_3 \equiv m_1 m_2 \bmod n.$$

Interceptorul E va forţa pe A să semneze mesajele m_1 şi m_2 iar apoi acesta va calcula semnătura lui m_3 :

$$m_3^d \equiv (m_1^d \bmod n)(m_2^d \bmod n).$$

Concluzie: Nu folosiţi niciodată algoritmul RSA pentru semnarea unui document necunoscut. Folosiţi apriori o funcţie hash. Formatul $ISO9796$ previne acest tip de atac.

Atac cu ajutorul modulelor comune

O posibilă implementare a RSA dă aceeaşi valoare n, dar valori diferite pentru exponenţii e şi d. Cea mai evidentă problemă este aceea că dacă acelaşi mesaj este cifrat cu doi exponenţi diferiţi (dar având aceleaşi module), iar acei exponenţi sunt numere relativ prime, atunci textul clar poate fi descoperit fără a cunoaşte exponenţi de descifrare.

Fie m mesajul în clar. Cele două chei publice de cifrare sunt e_1 şi e_2. Modulul comun este n. Cele două mesaje cifrate sunt:

$$c_1 \equiv m^{e_1} \bmod n,$$

$$c_2 \equiv m^{e_2} \bmod n.$$

Criptanalistul cunoaşte n, e_1, e_2, c_1 şi c_2. În continuare se prezintă cum se calculează mesajul clar m.

Întrucât e_1 şi e_2 sunt relativ prime, algoritmul extins al lui Euclid poate găsi r şi s astfel încât:

$$re_1 + se_2 = 1.$$

Presupunând r negativ, algoritmul extins al lui Euclid poate fi folosit din nou pentru a calcula c_1^{-1}. Astfel,

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

Există alte două atacuri împotriva acestui tip de sistem. Unul dintre ele foloseşte o metodă probabilistă pentru a factoriza pe n. Celălalt foloseşte un algoritm deterministic pentru a calcula ceva din cheia secretă fără a factoriza modulul.

Concluzie: Nu distribuiţi niciodată un modul de cifrare n comun unui grup de utilizatori.

Atac asupra exponenţilor de cifrare de dimensiuni mici

Cifrarea şi verificarea semnăturii RSA sunt mai rapide dacă se foloseşte o valoare mică a lui e, dar aceasta poate fi nesigură. Dacă se cifrează $e(e+1)/2$ mesaje lineare dependente cu diferite chei publice având aceeaşi valoare a lui e, atunci există un atac împotriva sistemului. Dacă sunt mai puţin de $e(e+1)/2$ mesaje, sau dacă mesajele nu sunt linear dependente, atunci nu este nici o problemă. Dacă mesajele sunt identice, atunci sunt suficiente e mesaje cifrate (se aplică teorema chinezească a resturilor pentru aflarea lui $c = m^e < \prod_{i=1}^{e} n_i$, unde n_i sunt modulele de cifrare, deci vom obţine prin calcul direct $m = c^{\frac{1}{3}}$). Cea mai simplă soluţie este să se completeze mesajele cu valori aleatoare independente. Aceasta ne asigură de faptul că $m^e \bmod n \neq m^e$. Implementările RSA din produsele PEM şi PGP realizează acest lucru.

Concluzie: Mesajele trebuie completate cu valori aleatoare înaintea cifrării lor; trebuie să fie sigur faptul că m este aproape de aceeşi lungime ca n.

Atac asupra exponenţilor de descifrare de dimensiuni mici

Un alt atac, elaborat de *Michael Wiener*, va descoperi cheia privată d, dacă d este mai mult de un sfert din lungimea lui n şi cheia publică e este mai mică decât modul n. Acest lucru se întâmplă mai rar dacă e şi d sunt alese aleator şi nu se întâmplă deloc dacă e are o valoare mică.

Concluzie: Trebuie aleasă o valoare mare pentru cheia privată d.

Atac în cazul numerelor prime apropiate

În cazul în care numerele prime p şi q sunt suficient de apropiate următoarea observaţie conduce la un algoritm eficient de factorizare:

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2.$$

Pentru factorizarea lui n se testează toate numerele întregi $x > \sqrt{n}$ pentru care $x^2 - n$ este pătrat perfect; fie acesta y. Atunci vom avea:

$$\begin{cases} p = x + y \\ q = x - y. \end{cases}$$

Concluzie: Numerele prime p şi q, care formează factorizarea $n = p \cdot q$, trebuie să fie de ordine de mărime diferite.

Atac de tip eroare hardware

Acest tip de atac permite, în cazul în care se utilizează pentru optimizarea operaţiilor private (de exemplu pentru realizarea semnăturii), factorizarea modulului n. Pentru fixarea ideilor fie M mesajul care urmează a fi semnat cu cheia privată d. Semnătura acestui mesaj $E \equiv M^d \bmod n$ este calculată ca

$$E \equiv aE_1 + bE_2 \bmod n,$$

unde

$$\begin{cases} a \equiv 1 \bmod p \\ a \equiv 0 \bmod q \end{cases}$$

şi

$$\begin{cases} b \equiv 0 \bmod p \\ b \equiv 1 \bmod q. \end{cases}$$

Reţinem faptul că factorizarea $n = p \cdot q$ este trapa secretă a sistemului. Presupunem acum că dispunem de semnătura E a mesajului M şi de o semnătură greşită a aceluiaşi mesaj (a apărut *numai o eroare hardware* în calculul lui E_1 sau E_2). Presupunem, fără restrângerea generalităţii că E_1 a fost calculat eronat ca fiind $\hat{E}_1$ şi că E_2 a fost calculat corect. Putem scrie:

$$E - \hat{E} = a(E_1 - \hat{E}_1).$$

Dacă $E_1 - \hat{E}_1$ nu este divizibil prin p (ceea ce se întâmplă cu o probabilitate foarte mare) atunci:

$$\gcd(E_1 - \hat{E}_1, n) = \gcd(a(E_1 - \hat{E}_1), n) = q$$

deci modulul n poate fi factorizat fără probleme.

Atacuri de tip eroare harware (ca urmare a erorilor latente, tranziente sau induse) pot fi aplicate algoritmului de semnătură a lui Rabin, protocolului de identificare Fiat-Shamir precum şi protocolului de identificare Schnorr.

Concluzie: Trebuie realizat un padding cu valori aleatore al mesajului înaintea realizării semnăturii.

Atacul asupra cifrării şi semnării cu algoritmul RSA

Este natural ca mesajele să fie semnate *înainte* de a fi cifrate, dar nu mereu se respectă această succesiune a operaţiilor. Vom prezenta un atac asupra protocoalelor RSA care realizează semnarea *după* operaţia de cifrare.

Fie utilizatorul A care doreşte să trimită un mesaj lui B. Pentru început A va cifra mesajul cu ajutorul cheii publice a lui B iar apoi în va semna cu ajutorul cheii sale private. Se va trimite mesajul cifrat şi semnat către utilizatorul B :

$$(m^{e_B} \bmod n_B)^{d_A} \bmod n_A.$$

Vom vedea cum poate B pretinde faptul că a primit mesajul $m^{'}$ în loc de m. Deoarece B dispune de factorizarea lui n_B atunci el poate găsi numărul x astfel ca (se rezolvă problema logaritmului discret):

$$m^{'x} \equiv m \bmod n_B.$$

Utilizatorul B va înlocui cheia sa publică e cu xe_B, modulul n_B fiind invariant. Într-o asemenea situaţie B poate pretinde că a primit mesajul $m^{'}$ de la A. Funcţiile hash nu rezolvă problema. Se poate opta pentru utilizarea unui exponent de cifrare fixat.

Condiţii necesare dar nu şi suficiente pentru asigurarea securităţii criptografice a RSA

O serie de restricţii folosite în implementarea RSA se bazează pe succesul atacurilor menţionate:

-cunoaşterea unei perechi de exponenţi cifrare/descifrare pentru un modul dat permite factorizarea modulului;

-cunoaşterea unei perechi de exponenţi cifrare/descifrare pentru un modul dat permite generarea de noi exponenţi de cifrare/descifrare;

-nu trebuie utilizat un modul pentru un grup de utilizatori;

-trebuie realizat un padding al mesajelor pentru a preveni atacul cu ajutorul exponenţilor de cifrare mici, precum şi atacul de tip eroare hardware (în cazul utilizării CRT pentru optimizarea operaţiilor private);

-exponentul de descifrare trebuie să fie mare.

Trebuie remarcat faptul că aceste elemente *nu sunt suficiente* pentru a proiecta un algoritm criptografic sigur. Întregul sistem trebuie să fie sigur din punct de vedere criptografic ca şi protocoalele criptografice. O vulnerabilitate în oricare din aceste trei domenii conduce la un sistem nesigur din punct de vedere criptografic.

9.3.5. Trape în generarea cheilor RSA

Dezvoltatorul unui sistem de tip PKI, poate induce trape, la nivelul generării cheilor, pentru utilizarea acestora în activitatea de criptanaliză. Evidenţierea existenţei acestor trape conduce la compromite imaginea producătorului produsului criptografic. Vom exemplifica o serie de astfel de trape şi anume:

-ascunderea exponentului privat mic d, prin utilizarea unei funcţii de permutare $\pi_\beta(\cdot)$ a numerelor impare mai mici ca n (modulul de cifrare);

-ascunderea exponetului public mic e, precum şi a unei informaţii referitoare la exponentul privat d. Acest lucru se face prin utilizarea unei funcţii de permutare a numerelor impare mai mici ca n (modulul de cifrare);

-ascunderea factorului prim p în produsul $n = p \cdot q$.

Pentru fiecare modalitate de inducere a trapelor menţionate mai sus trebuie dezvoltate teste de detecţie a acestora.

Vom prezenta un algoritm de inducere a trapei prin ascunderea exponentului privat mic d în cadrul operaţiei de generare a perechilor de chei publice/private de tip RSA.

Ascunderea exponentului privat mic d

PASUL 0. Selectăm două numere prime mari, de acelaşi ordin de mărime, p şi q. Modulul de cifrare va fi în acest caz $n = p \cdot q$. Fie k numărul de biţi ai lui n.

PASUL 1. Generăm aleatoriu δ număr impar astfel ca $\gcd(\delta, \varphi(n)) = 1$ şi $|\delta| \leq \frac{k}{4}$.

PASUL 2. Calculăm $\varepsilon = \delta^{-1} \bmod \varphi(n),\ e = \pi_\beta(\varepsilon)$ unde $\pi_\beta(\cdot)$ este o permutare a numerelor impare;

PASUL 3. Dacă $\gcd(e, \varphi(n)) \neq 1$ atunci trecem la **PASUL 1.**

PASUL 4. Calculăm $d = e^{-1} \bmod \varphi(n)$.

PASUL 5. Cheia publică (cu trapă) este (n, e) iar cheia privată (cu trapă) este (n, p, q, d).

Evident cunoaşterea permutării $\pi_\beta(\cdot)$ permite, conform celor prezentate în paragraful 9.3.4 (algoritmul lui Wiener), aflarea exponentului privat d.

9.4. Algoritmul Pohlig-Hellman

Algoritmul Pohlin-Hellman este similar algoritmului RSA. Acesta nu este un algoritm simetric deoarece sunt folosite chei diferite pentru cifrare respectiv descifrare. Nu este un algoritm cu cheie publică deoarece cheile se pot deduce uşor una din cealaltă, atât cheia de cifrare cât şi cheia de descifrare trebuie să fie secrete. La fel

ca în algoritmul RSA:

$$\begin{aligned} C &\equiv P^e \bmod n \\ P &\equiv C^d \bmod n \end{aligned}$$

unde

$$ed \equiv 1(\text{mod un număr complicat}).$$

Spre deosebire de RSA, numărul n nu este definit de produsul a două numere prime ci trebuie să fie parte a unei chei secrete. Dacă dispunem de e şi n atunci putem calcula pe d. Fără cunoaşterea lui e sau d interceptorul trebuie să rezolve problema (dificilă din punct de vedere computaţional):

$$e \equiv \log_P C \bmod n.$$

9.5. Algoritmul Rabin

Securitatea schemei lui Rabin se bazează pe dificultatea găsirii rădăcinii pătrate modulo un număr compozit. Această problemă este echivalentă cu problema factorizării. Vom prezenta o implementare a acestei scheme.

Pentru început vom alege două numere prime p şi q ambele congruente cu $3 \bmod 4$. Aceste numere constituie cheia privată iar produsul acestora $n = p \cdot q$ constituie cheia publică.

Pentru a cifra un mesaj M (M trebuie să fie mai mic decât n) vom calcula

$$C \equiv M^2 \bmod n.$$

Deoarece receptorul cunoaşte cele două numere p şi q acesta va rezolva două dintre congruenţe cu ajutorul lemei chinezeşti a resturilor (vezi Anexa I). Calculăm:

$$\begin{aligned} m_1 &\equiv C^{\frac{p+1}{4}} \bmod p \\ m_2 &\equiv (p - C^{\frac{p+1}{4}}) \bmod p \\ m_3 &\equiv C^{\frac{q+1}{4}} \bmod q \\ m_3 &\equiv (q - C^{\frac{q+1}{4}}) \bmod q \end{aligned}$$

Se aleg două numere întregi $a := q(q^{-1} \bmod p)$ şi $b := p(p^{-1} \bmod q)$. Cele patru soluţii posibile sunt:

$$\begin{aligned} M_1 &\equiv (am_1 + bm_3) \bmod n \\ M_2 &\equiv (am_1 + bm_4) \bmod n \\ M_3 &\equiv (am_2 + bm_3) \bmod n \\ M_4 &\equiv (am_2 + bm_4) \bmod n \end{aligned}$$

Unul dintre aceste numere este egal cu M. Dacă textul este un mesaj dintr-o limbă bine precizată atunci se poate spune care este alegerea corectă M_i. Dacă textul este o secvenţă aleatoare (o cheie sau o semnătură digitală) atunci nu putem să indicăm nici o variantă. Soluţia pentru rezolvarea acestei probleme constă în adăugarea unui header mesajului înainte de realizarea cifrării.

9.6. Algoritmul ElGamal

Schema ElGamal este folosită atât pentru cifrare cât şi pentru semnătură electronică. Securitatea sa se bazează pe dificultatea rezolvării problemei logaritmului discret într-un corp finit. Pentru a genera o pereche de chei vom alege pentru început un număr prim p şi două numere g şi x ambele mai mici decât p. Vom calcula:

$$y \equiv g^x \bmod p.$$

Cheia publică este y, g, p iar cheia privată este x. Ambele numere g şi p pot fi comune unui grup de utilizatori. Algoritmul de semnătură digitală ElGamal este prezentat în capitolul 10.

Pentru a cifra un mesaj M vom alege mai întâi un număr aleatoriu k relativ prim cu $p-1$. Vom calcula apoi:

$$\begin{aligned} a &\equiv g^k \bmod p \\ b &\equiv y^k M \bmod p \end{aligned}$$

Perechea (a, b) constituie textul cifrat (acesta este de două ori mai mare decât textul clar).

Pentru a descifra (a, b) calculăm:

$$M \equiv ba^{-x} \bmod p,$$

deoarece $a^x \equiv g^{xk} \bmod p$ şi $ba^{-x} \bmod p \equiv y^k M a^{-x} \bmod p \equiv g^{kx} M g^{-kx} \bmod p \equiv M \bmod p$.

9.7. Curbe eliptice

Folosirea curbelor eliptice în criptografie a fost propusă pentru prima oară în 1985 de *Victor Miller* (IBM) şi, independent, de *Neal Koblitz* [39]. Ideea de bază era folosirea grupului de puncte de pe o curbă eliptică în locul grupului $\mathbf{Z}_p^*$. Recent curbele eplitice au fost folosite pentru conceperea unor algoritmi eficienţi de factorizare a întregilor şi pentru demonstrarea primalităţii. Ele au fost utilizate în construirea criptosistemelor cu chei publice, în construcţia generatoarelor pseudoaleatoare

de biţi. Curbele eliptice şi-au găsit aplicabilitate şi în teoria codurilor, unde au fost întrebuinţate pentru obţinerea unor coduri corectoare de erori, foarte bune. Curbele eliptice au jucat un rol foarte important şi în demonstrarea recentă a *Ultimei Teoreme a lui Fermat.*

Definiţia 9.7.1. O *curbă eliptică*, E, constă din elemente (numite puncte) de tipul (x, y) ce satisfac ecuaţia:

$$y^2 \equiv x^3 + ax + b \bmod p$$

(unde a şi b sunt constante astfel încât $4a^3 + 27b^2 \neq 0 \bmod p$, iar p este un număr prim) împreună cu un elemente singular, notat O şi numit *punctul de la infinit.* Acest punct poate fi privit ca fiind punctul din vârful şi de la baza oricărei linii verticale.

O curbă eliptică E are o structură de grup abelian împreună cu operaţia adunare. Adunarea a două puncte de pe o curbă eliptică este definită în concordanţă cu o mulţime simplă de reguli (vezi figura 9.1).

Fiind date două puncte pe E, $P_1(x_1, y_1)$ şi $P_2(x_2, y_2)$, avem următoarele cazuri:
-dacă $x_2 = x_1$ şi $y_2 = -y_1$ atunci $P_1 + P_2 = 0$.
-altfel $P_1 + P_2 = (x_3, y_3)$, unde:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

cu

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, \text{ dacă } P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1}, \text{dacă } P_1 = P_2 \end{cases}$$

Operaţia de adunare pe o curbă eliptică este corespondenta operaţiei de înmulţire în sistemele cu chei publice obijnuite, iar adunarea multiplă este corespondenţa exponenţierii modulare din acestea.

Deşi regulile de calcul în grupul punctelor unei curbe eliptice par destul de complicate, aritmetica acestora poate fi implementată extrem de eficient, calculele în acest grup fiind realizate mult mai rapid decât cele din grupul $\mathbf{Z}_p$.

Corespondenţa problemei logaritmilor discreţi la curbele eliptice este problema logaritmilor pe curbe eliptice, definită după cum urmează: fiind dat un punct G pe o curbă eliptică, de ordin r (numărul punctelor de pe curba eliptică) şi un alt punct Y, să se găsească un unic x cu $0 \leq x \leq r - 1$ astfel încât $Y = xG$.

Cele mai bune atacuri asupra problemelor logaritmilor pe curbe eliptice sunt metodele general aplicabile oricărui grup, ceea ce făcea ca acestea să fie deosebit de

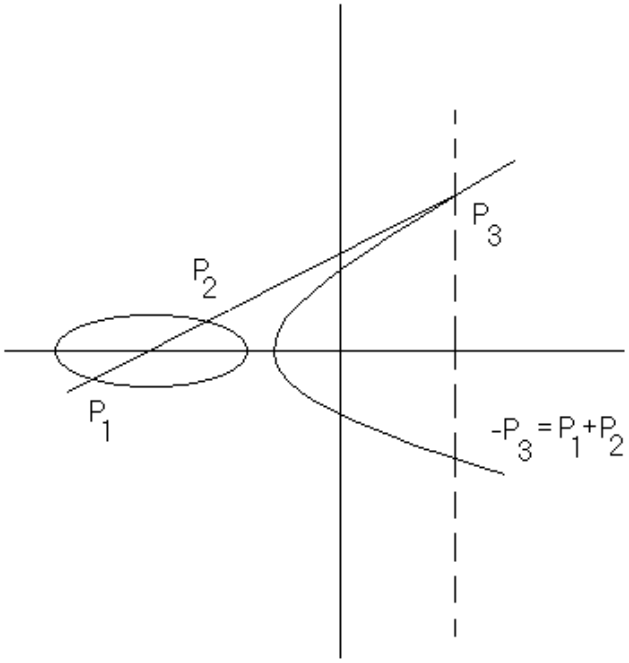

Figura 9.1: Operaţia de adunare pe o curbă eliptică.

ineficiente pe un anumit caz particular. Datorită lipsei metodelor de atac specializate, criptosistemele bazate pe curbe eliptice, folosind chei de dimensiuni reduse, oferă aceeaşi securitate ca şi criptosistemele bazate pe problema logaritmilor discreţi ce folosesc chei mult mai mari. Vom menţiona, ca scheme criptografice cu ajutorul curbelor eliptice scheme similare algoritmul de schimbare de chei Diffie-Helman, algorimtmilor ce cifrare Massey-Omura şi ElGamal. Descrierea acestora depăşeşte cadrul acestui curs (a se vedea *Koblitz* [39] şi [40]). Pentru un exemplu privind schema de cifrare *ElGamal pe curbe eliptice* a se vedea exerciţiul 9.11.10.

9.8. Aplicaţii practice

Aplicaţiile practice, de verificare a corectitudinii utilizării şi implementării algoritmului RSA (relativ la problemele semnalate în paragrafele anterioare), pot fi de exemplu din categoriile următoare:

-aplicaţii software cum ar fi programul de protecţie a poştei electronice PGP (Pretty Good Privacy), în special la partea de generare a cheilor publice/private,

anveloparea cheii de sesiune (cu ajutorul cheii publice a destinatarului) şi a semnării mesajului (cu ajutorul cheii private a emitentului);

-smart-carduri folosite în sistemul de plăţi electronice;

-tokenuri utilizate pentru autentificarea utilizatorului şi/sau a mesajelor;

-aplicaţia de generare a cheilor publice/private dintr-o structură de tip PKI (*Public Key Infrastructure*) cum ar fi, de exemplu, implementarea funcţiei de generare a cheilor din OpenSSL (*Secure Socket Layer*).

Direcţii ulterioare de investigare ar consta în identificarea vulnerabilităţilor ce pot apare în cazul utilizării problemelor computaţionale de *calculul a rădăcinii pătrate modulo un număr compozit* (algoritmul *Rabin*) şi a *logaritmului discret* (algoritmul *ElGamal*). Totodată vom avea în vedere investigarea problemelor similare ce pot apare în cazul utilizării curbelor eliptice.

9.9. Teste de primalitate şi metode de factorizare

9.9.1. Teste de primalitate

Prezentăm o serie de teoreme cunoscute şi sub denumirea de *teste de primalitate.*

Teorema lui Euler. *Dacă* $(b, m) = 1$ *atuci* $b^{\phi(m)} \equiv 1 \bmod m$.

Teorema lui Fermat. *Dacă* p *este număr prim care nu divive* b *atunci* $b^{p-1} \equiv 1 \bmod p$.

Teorema lui Wilson. *Numărul* p *este prim dacă şi numai dacă avem congruenţa* $(p-1)! \equiv 1 \bmod p$.

Ciurul lui Erastotene. *Numărul* p *este prim dacă şi numai dacă nu este divizibil cu nici un număr natural mai mic ca* $[\sqrt{p}]$.

Pentru valori mari ale lui p testul lui Wilson şi testul lui Erastotene nu sunt eficiente din punct de vedere al timpului de calcul. Vom investiga acum reciproca teoremei lui Fermat. Vom spune despre un număr natural p ca este *pseudoprim* cu baza b dacă $b^{p-1} \equiv 1 \bmod p$. Numerele pseudoprime care nu sunt prime sunt rare: spre exemplu există numai 22 de numere mai mici ca 10000 pseudoprime în baza 2 care nu sunt prime: $341, 561, 645, 1105, ...$ Un număr p se numeşte *număr Carmichael* dacă este pseudoprim cu orice bază $b \leq p-1$. Numerele Carmichael sunt extrem de rare: de exemplu, există numai 255 de numere mai mici ca 1000000. În concluzie reciproca teoremei lui Fermat poate fi folosită ca test de primalitate cu rata de eroare tinzând la zero când reprezentarea sa binară tinde la infinit.

Testul Miler-Rabin depăşeşte problemele testului simplu de pseudoprimalitate prin două modificări:

-încearcă să aleagă aleator diferite valori ale bazei b în loc de o singură valoare;

-în timp ce calculează fiecare exponenţiere modulară (ridicare la pătrat urmată de stabilirea restului modulo p), stabileşte dacă nu cumva a fost descoperită o rădăcină

netrivială a lui 1 modulo p. În caz afirmativ testul va decide faptul că numărul p este compozit. *Rata de eroare* a testului Miller-Rabin pentru orice număr întreg impar este de $\frac{1}{2^s}$, unde s sunt numărul de valori aleatoare alese pentru baza b.

Testul Fermat

Ideea *testului Fermat* este aceea de a utiliza *reciproca teoremei lui Fermat*. Vom prezenta, sub formă de pseudocod, algoritmul de primalitate al lui Fermat.

FERMAT(n,t)
Intrare: un număr impar $n \geq 3$ şi un parametru de securitate $t \geq 1$.
Ieşire: decizia asupra primalităţii.
PASUL 1. Pentru $i = 1, ..., t$ se execută:
1.1 se alege aleatoriu a cu $2 \leq a \leq n-2$.
1.2 calculează $r = a^{n-1} \bmod n$.
1.3 dacă $r \neq 1$ atunci se decide n *este compozit*.
PASUL 2. Se decide: n *este prim*.

Observaţia 9.9.1. Un număr compozit n care este declarat prim de testul Fermat se numeşte *număr pseudoprim Fermat* iar numerele a pe baza cărora se ia decizia de primalitate se numesc *numere false Fermat*.

Testul Solovay-Strassen

Ideea *testului Solovay-Strassen* este aceea de a utiliza *criteriului lui Euler*: dacă n este un număr prim atunci:

$$a^{\frac{n-1}{2}} \equiv (\frac{a}{n}) \bmod n \text{ pentru orice întreg } a \text{ care satisface } (a, n) = 1,$$

unde $(\frac{a}{n})$ este simbolul lui Legendre definit în Anexa I.

Vom prezenta, sub formă de pseudocod, algoritmul de primalitate Solovay-Strassen.

SOLOVAY-STRASSEN(n,t)
Intrare: un număr impar $n \geq 3$ şi un parametru de securitate $t \geq 1$.
Ieşire: decizia asupra primalităţii.
PASUL 1. Pentru $i = 1, ..., t$ se execută:
1.1 se alege aleatoriu a cu $2 \leq a \leq n-2$.
1.2 calculează $r = a^{\frac{n-1}{2}} \bmod n$.
1.3 dacă $r \neq 1$ şi $r \neq n-1$ atunci se decide n *este compozit*.
1.4 se calculează simbolul lui Legendre $s = (\frac{a}{n})$.
1.5 dacă $r \neq s$ atunci se decide n *este compozit*.
PASUL 2. Se decide: n *este prim*.

Observaţia 9.9.2. Un număr compozit n care este declarat prim de testul Euler se numeşte *număr pseudoprim Euler* iar numerele a pe baza cărora se ia decizia de primalitate se numesc *numere false Euler.*

Observaţia 9.9.3. *Rata de eroare* (probabilitatea da un număr compozit să fie declarat prim) a testului Solovay-Strassen este mai mică ca $\frac{1}{2^t}$.

Testul Miller-Rabin

Testul Miller-Rabin este cunoscut şi ca testul tare de primalitate şi se bazează pe următoarea teoremă.

Teorema 9.9.1. Fie n un număr prim impar, şi $n-1 = 2^s r$ unde r este impar. Fie deasemenea un întreg a astfel ca $(a,n) = 1$. Atunci $a^r \equiv 1 \bmod n$ sau $a^{2^j r} \equiv -1 \bmod n$ pentru un j, $0 \leq j \leq s-1$.

Vom prezenta, sub formă de pseudocod, algoritmul de primalitate Miller-Rabin.
MILLER-RABIN(n,t)
Intrare: un număr impar $n \geq 3$ şi un parametru de securitate $t \geq 1$.
Ieşire: decizia asupra primalităţii.
PASUL 1. Se scrie $n-1 = 2^s r$ unde r este impar.
PASUL 2. Pentru $i = 1, ..., t$ se execută:
2.1 se alege aleatoriu a cu $2 \leq a \leq n-2$.
2.2 calculează $y = a^r \bmod n$.
2.3 dacă $y \neq 1$ şi $y \neq n-1$ atunci se execută:
$j = 1$.
Atâta timp cât $j \leq s-1$ şi $y \neq n-1$ se execută:
calculează $y = y^2 \bmod n$.
dacă $y = 1$ atunci se decide n *este compozit.*
$j = j+1$.
dacă $y \neq n-1$ atunci se decide n *este compozit.*
PASUL 2. Se decide: n *este prim.*

Observaţia 9.9.4. Un număr compozit n care este declarat prim de testul Miller-Rabin se numeşte *număr pseudoprim Miller-Rabin* (sau pseudoprime tari) iar numerele a pe baza cărora se ia decizia de primalitate se numesc *numere false Miller-Rabin (tari).*

Observaţia 9.9.5. i) *Rata de eroare* (probabilitatea da un număr compozit să fie declarat prim) a testului Miller-Rabin este mai mică ca $\frac{1}{4^t}$ mult mai mică decât a testului Solovay-Strassen care este $\frac{1}{2^t}$.

ii) Cele trei teste pot fi ordonate descendent din punct de vedere al eficienţei, în sensul că mulţimea numerelor false Miller-Rabin este inclusă în mulţimea numerelor false Euler care la rândul ei este inclusă în mulţimea numerelor false Fermat.

iii) Testul Solovay-Strassen este cel mai complex în ceea ce priveşte implementarea datorită necesităţii evaluării simbolului lui Legendre.

Observaţia 9.9.6. Dacă $n \equiv 3 \bmod 4$ atunci un număr a este număr fals Euler dacă şi numai dacă acesta este fals Miler-Rabin.

9.9.2. Metode de factorizare

Metodele de factorizare se rulează de regulă după testele de primalitate şi pun în evidenţă factorii primi ce constituie numărul testat. O serie de tehnici de factorizare fac apel la utilizarea filtrelor (de exemplu sita pătratică):

-metoda Dixon;

-metoda Pomerace-Silverman-Montgomery.

9.9.3. Metode de generare a numerelor prime

Acestă secţiune prezintă tehnici de generare a numerelor prime precum şi a numerelor prime tari (vezi observaţia 9.3.1 pentru definiţia acestora). Un algoritm de generat numere prime (prin căutare prin incrementare), între două limite $p_{\min}$ şi $p_{\max}$, care satisface condiţia $gcd(p-1, f) = 1$ (cu f dat de IEEE P1363/D13 [33], pagina 73) este următorul.

GEN_PRIM($p_{\min}$, $p_{\max}$, f)

Intrare. Limitele $p_{\min}$ şi $p_{\max}$ şi f număr natural.

Ieşire. Un număr prim p cuprins între limitele $p_{\min}$ şi $p_{\max}$.

PASUL 1. Se generează aleatoriu un număr impar p cuprins între limitele $p_{\min}$ şi $p_{\max}$.

PASUL 2. Dacă $p > p_{\max}$ atunci $p = p_{\min} + p \bmod (p_{\max} + 1)$ şi se trece la **PASUL 2**.

PASUL 3. Se calculează $d = gcd(p-1, f)$. Dacă $d = 1$, se testează numărul p pentru primalitate. Dacă p este număr prim atunci se returnează numărul prim p. În caz contrar $p = p + 2$ şi se trece la **PASUL 3**.

Următorul algoritm atribuit lui *Gordon* va genera numere prime tari p: $p-1$ are un factor prim mare u, $p+1$ are un factor prim mare v, $u-1$ are un factor prim mare t.

GORDON($p_{\min}$, $p_{\max}$, s, t)

Intrare. Limitele $p_{\min}$ şi $p_{\max}$ şi un parametru de securitate $t \geq 1$ şi s un parametru de fiabilitate.

Ieşire. Un număr prim p tare.

PASUL 1. Se generează aleatoriu două numere prime aleatoare v şi w de aproximativ aceeiaşi lungime (conform algoritmului de generare numere prime prin incrementare).

PASUL 2. Se alege un întreg i_0. Fie $u = 2iw+1$ primul număr prim în progresia $2iw + 1$ pentru $i = i_0, i_0 + 1, ...$

PASUL 3. Se calculează $p_0 = 2(v^{w-2} \bmod u)v - 1$.

PASUL 4. Se alege un întreg j_0. Fie $p = p_0 + 2juv$ primul număr prim în progresia $p_0 + 2juv$ pentru $j = j_0, j_0 + 1, ...$

PASUL 5. Se returnează numărul prim tare p.

9.10. Infrastructura Cheilor Publice (PKI)

9.10.1. Elementele PKI

Algoritmii criptografici asimetrici conferă un grad de securitate comercial iar aceştia se pot folosi, de regulă, în cadrul protocoalelor criptografice referitoare la transferul cheilor de cifrare sau autentificare. În unele aplicaţii se pot folosi şi în cadrul operaţiei de cifrare. Ambele metode (cifrarea simetrică şi cifrarea asimetrică) au punctele lor slabe: managementul cheilor de cifrare respectiv complexitatea algoritmului de cifrare. Una dintre aplicaţiile cele mai frecvente ale sistemelor asimetrice este aceea a PKI (*Public Key Infrastructure*). O astfel de infrastructură se compune din următoarele elemente:

-*autoritatea de certificare* (*CA*) care generează, revocă, publică şi arhivează certificatele digitale;

-*autoritatea de înregistrare* (*RA*) care acceptă şi validează cererile de certificate, trimite cereri de certificate către CA, primeşte certificatele şi *lista certificatelor revocate* (*CRL*), generează cererile de revocare a certificatelor;

-*deţinătorii de certificate* care generează semnături digitale, generează cereri de certificate, cereri de revocare de certificate, cereri de reînregistrare (opţional);

-*clientul PKI* care verifică semnături, obţine certificate şi CRL de la baza de date şi validează calea de certificarea.

Comunicaţia datelor între RA şi CA trebuie realizată în condiţii de deplină siguranţă (off-line) pentru a proteja cheia privată a autorităţii de certificare.

O infrastructură PKI se foloseşte de regulă pentru a realiza *confidenţialitatea, nerepudierea, integritatea şi autentificarea mesajelor* şi/sau *identificarea/autentificarea utilizatorului* (control acces).

Cheia privată pentru semnătura digitală (utilizată în nerepudiere, integritate şi autentificare) trebuie generată de către utilizator şi nu trebuie să fie transmisă unei terţe părţi.

Cheia privată folosită la cifrare (utilizată pentru realizarea confidenţialităţii) trebuie să fie şi în posesia unei terţe părţi pentru activarea serviciului de recuperare a cheilor (de regulă în acest caz cheia privată este generată de către RA şi comunicată la CA). În acest caz perechea de chei publică-privată se foloseşte pentru cifrarea cheii secrete de sesiune. Cifrarea mesajelor sau a comunicaţiei se face de regulă cu un algoritm simetric (bloc sau flux).

Standardul X.509 V3, oferă printre altele, şi *forma certificatelor digitale* care trebuie să conţină: versiunea cerificatului, numărul serial, algoritmul utilizat de semnătură utilizat de emitent, identificarea emitentului, perioada de valabilitate, identificarea posesorului, informaţii despre cheia publică a utilizatorului (cheia publică a utilizatorului: versiunea, modulul de cifrare, exponentul de cifrare, etc.), tipul posesorului (utilizator sau CA) informaţii despre posesor, atribute ale cheii (utilizată la semnătură sau schimbul de chei), informaţii despre politica de securitate a CA, extensii, semnătura (emitentului certificatului) a datelor anterioare.

Cheia privată a utilizatorului este compusă din: versiunea cheii, modulul de cifrare care este produs de două numere prime mari, exponentul de cifrare, exponentul de descifrare, cele două numere prime p şi q al căror produs este modulul de cifrare, $d \bmod (p-1)$, $d \bmod (q-1)$, $q^{-1} \bmod p$ (acestea fiind utilizate pentru optimizarea operaţiilor private cu CRT).

9.10.2. Ghid de folosire a tehnologiei PKI în reţele deschise

Următoarele cinci probleme trebuiesc urmărite înainte de a implementa tehnologia PKI în reţelele deschise (de exemplu reţeaua Internet).

1. *Beneficiile, directe şi indirecte, financiare şi nefinanciare, obiective şi subiective ale utilizării semnăturilor digitale pentru aplicaţia propusă:*

-optimizarea resursei timp;

-optimizarea costurilor;

-disponibilitate marită a tipului de servicii furnizate;

-integritatea datelor;

2. *Costurile a) de a converti un sistem de procesare electronic la un sistem ce foloseşte sistemul de semnătura digitală sau de a converti un proces non-electronic la unul electronic ce foloseşte sistemul de semnătură digitală, b) de funcţionare după operaţia de conversie:*

-nivelul de încredere solicitat;

-integritatea cheilor publice şi a cheilor private;

-cuantificarea consecinţelor potenţialelor riscuri;

-politici, practici şi proceduri;

-conectarea la o infrastructură existentă;

-interoperabilitatea cu alte infrastructuri;

-înregistrarea operaţiilor (a managementului);
-conformitatea cu standardele PKI;
-realizarea aplicaţiilor program;
-evaluările părţilor implicate;
-cerinţe de statut adiţionale;
3. *Riscurile asociate cu folosirea cheii publice pentru această aplicaţie:*
-frauda;
-întreruperea serviciului sau disfuncţionalităţi de scurtă durată;
-legibilitate;
4. *Beneficiile determinate la prima problema comparativ cu costurile stabilite la cea de-a doua problemă şi riscurile stabilite la pasul 3:*
-beneficii inerente;
-parte dintr-un proces mult mai complex;
-impactul asupra utilizatorului;
-examinarea riscurilor;
5. *Implementările critice pe care un organism trebuie să le rezolve pentru a implementa şi folosi PKI pentru operaţia de semnătură digitală:*
-realizarea unei politici de certificare şi reguli practice de certificare;
-stabilirea acelor *directory service* care sunt necesare aplicaţiei pe care o utilizează PKI, asigurarea disponibilităţii acestora;
-interoperabilitatea cu alte infrastructuri externe similare (PKI);
-adaptarea aplicaţiilor la structura PKI;
-implementarea structurii PKI şi a aplicaţiilor în mod secvenţial;
-integrarea procesului de înregistrare la PKI în politica de personal;
-identificarea cerintelor softului PKI şi a aplicaţiilor program pentru a opera prin firewall şi rutere;
-tipul de funcţionare on-line si/sau off-line;
-stabilirea personalui ce realizează operaţia de audit;
-validarea semnăturii digitale după realizare.

9.10.3. Riscuri ale utilizării tehnologiei PKI

Vom enumera o serie de riscuri referitoare la folosirea tehnologiei PKI.
1. În cine avem încredere şi pentru ce anume.
2. Protecţia cheii (de semnătură digitală) private a utilizatorului.
3. Securitatea computerului care realizează verificarea.
4. Asocierea cheii publice cu numele destinatarului.
5. Autoritatea de certificare (CA).
6. Politica de securitate a utilizatorului.
7. Unicitatea CA şi dualismul RA.

8. Identificarea de către CA a deţinătorului de certificat.
9. Folosirea adecvată a certificatelor.
10. Utilizarea procesului CA.

9.10.4. Standarde ale PKI

Deoarece o infrastrucură cu chei publice trebuie să fie interoperabilă cu alte structuri similare a apărut ca o necesitate standardizarea formei certificatelor digitale şi a funcţiilor criptografice utilizate. Un astfel de standard este standarul *X.509* pentru certificate şi protocolul *SSL* (*Secure Socket Layer*) pentru primitivele criptografice. Standardul SSL prevede următoarele primitive criptografice:

1. *Cifruri bloc*: DES (standard de cifrare SUA), AES (standard de cifrare SUA), BLOWFISH (Bruce Schneier), CAST (Carlisle Adams şi Stafford Tavares), IDEA (Xuejia Lai şI James Massey), RC2, RC5 (Ron Rivest).
2. *Cifruri flux*: RC4 (Ron Rivest).
3. *Cifruri cu chei publice*: RSA (Rivest, Shamir, Adellman), EC (curbe eliptice), PKCS (firma RSA).
4. *Funcţii hash*: SHA (NIST şi NSA), MD2, MD4, MD5 (Ron Rivest), RIPEMD (Comunitatea Europeană), LHASH (Eric Young).
5. *Funcţii de semnătură digitală*: DSA (NIST), MDC (funcţii de semnătură cu ajutorul cifrurilor bloc), HMAC (funcţii de semnătură cu ajutorul funcţiilor hash bazate pe o cheie secretă).
6. *Protocoale de schimb al cheilor*: Diffie-Hellman.
7. *Protocoale de autentificare*: Kerberos.
8. *Generatoare pseudoaleatoare*: RAND pachet de generatoare de numere pseudoaleatoare.

9.11. Aplicaţii

Exerciţiul 9.11.1. Să se cifreze mesajul $M = 3$, utilizând sistemul *RSA* cu următorii parametrii: $n = 187 = 11 \cdot 17$ (modulul de cifrare), $s = 7$ (cheia publică).

Răspuns. Criptograma este:

$$E = M^s = 3^7 = 2187 = 130 \bmod 187.$$

Deoarece dispunem de factorizarea $n = 11 \cdot 17$ calculăm $\varphi(n) = 16 \cdot 10 = = 160$, $\varphi(\varphi(n)) = 64$.

Exponentul de descifrare va fi:

$$t = s^{\varphi(\varphi(n))-1} = 7^{63} = (7^9)^7 = (40353607)^7 = 7^7 = 823543 = 23 \bmod 160.$$

Descifrarea mesajului E va fi:

$$E^t = 130^{23} = 3 = M \bmod 187.$$

Exerciţiul 9.11.2. Să se contruiască cheia publică pentru un algoritmul Merkle-Hellman reprezentat de cheia privată $\{2, 3, 6, 13, 27, 52\}$, modulul $p = 105$ şi multiplicatorul $m = 31$.

Exerciţiul 9.11.3. Să se cifreze mesajul binar 0110001101011011 10 cu ajutorul cheii publice setate la exerciţiul 9.11.2.

Exerciţiul 9.11.4. Să se descifreze mesajul $\{174, 280, 333\}$ cu ajutorul cheii private de la exerciţiul 9.11.2.

Exerciţiul 9.11.5. Se poate da o interpretare fizică a sistemelor de cifrare cu cheie publică?

Răspuns. Cheia publică poate fi privită ca un lacăt desfăcut ce se poate închie fără a utiliza cheia, iar pentru a deschide acest lacăt este nevoie de cheia privată.

Exerciţiul 9.11.6. Să se dea o interpretare fizică a protocolului Diffie-Hellman, dacă se adoptă modelul dat în soluţia problemei 9.11.5.

Exerciţiul 9.11.7. Să se specifice pentru algoritmul ElGamal cu parametrii $p = 17,\ g = 14,\ x = 2$ cheia publică şi cheia privată.

Exerciţiul 9.11.8. Folosind algoritmul stat la problema precedentă 9.11.7 să se cifreze mesajul $M = 4$.

Exerciţiul 9.11.9. Să se descifreze mesajul $(12, 15)$ cu ajutorul algoritmului setat la problema 9.11.7.

Exerciţiul 9.11.10. Să se cifreze mesajul $(10, 9)$ utilizând curba eliptică (publică) $E : y^2 = x^3 + x + 6$ pe $\mathbf{Z}_{11}$ cu ajutorul algoritmului ElGamal.

Răspuns. Pentru a calcula punctele curbei eliptice se calculează valorile $z = x^3+x+6 \bmod 11$, se vede care din aceste valori sunt reziduri pătratice cu ajutorul teoremei lui Euler (z este reziduu pătratic dacă şi numai dacă $z^{\frac{p-1}{2}} \equiv 1 \bmod p$) iar apoi

se calculează rădăcinile pătrate ale acestor reziduri prin formula $y = \pm z^{\frac{p+1}{2}} \bmod p$). Punctele curbei eliptice vor fi:

$$\{(2,7),(2,4),(3,5),(3,6),(5,2),(5,9),(7,2),(7,9),(8,3),(8,8),(10,2),(10,9),O\}.$$

Grupul E este grup ciclic (numărul de elemente este al grupului este număr prim) şi se ia ca generator pentru acesta elementul (public) $\alpha = (2,7)$. Cheia privată de descifrare, notată prin d, este o valoare între 1 şi numărul de puncte de pe o curbă eliptică -1. Cheia publică, notată prin β, se obţine din α şi exponentul secret d prin formula $\beta = d\alpha$.

Operaţia de cifrare a mesajul M cu ajutorul cheii (secrete) k este:

$$E(M,k) = (k\alpha, M + k\beta).$$

Operaţia de descifrare pentru a obţine M este:

$$D_k(y_1, y_2) = y_2 - dy_1.$$

Exerciţiul 9.11.11. Implementaţi un algoritm de generare a cheilor RSA care să conţină trape prin:
-ascunderea exponenţilor privaţi mici;
-ascunderea exponenţilor publici primi;
-ascunderea factorilor primi.

Exerciţiul 9.11.12. Prin ce se caracterizează criptografia asimetrică faţă de criptografia simetrică?

Exerciţiul 9.11.13. Să se factorizeze numărul $n = 97343$ ştiind că acesta este produsul a două numere prime apropiate.