

CRIPTANALIZA. REZULTATE ȘI TEHNICI MATEMATICE

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

Vasile PREDA, Emil SIMION și Adrian POPESCU

Ediția a doua 2011

Cuprins

1	INTRODUCERE	15
2	NOȚIUNI GENERALE	19
2.1.	Obiectul criptanalizei	19
2.2.	Criterii și standarde	20
2.2.1.	Beneficii ale standardelor	20
2.2.2.	Organisme de standardizare	21
2.2.3.	Standardele ISO 15408 și FIPS 140-2	22
2.3.	Modelul OSI (Open System Interconectation)	22
2.3.1.	Definirea nivelurilor rețelei	22
2.3.2.	Nivelul fizic	23
2.3.3.	Nivelul legătură date	23
2.3.4.	Nivelul rețea	24
2.3.5.	Nivelul transport	24
2.3.6.	Nivelul sesiune	24
2.3.7.	Nivelul prezentare	24
2.3.8.	Nivelul aplicație	25
2.3.9.	Protocolul TCP/IP	25
2.4.	Testarea sistemelor criptografice	25
2.4.1.	Introducere	25
2.4.2.	Programul de validare a modulelor criptografice	26
2.4.3.	Proceduri de certificare și autorizare	28
2.4.4.	Autoritate de certificare	28
2.5.	Procesul de selectare a modulelor criptografice	29
2.5.1.	Faza de planificare	29
2.5.2.	Faza de definire a cerințelor și specificațiilor de securitate	31
2.5.3.	Faza de achiziție	31
2.5.4.	Faza de operare	32
2.6.	Operații în criptanaliză	32

2.6.1.	Principii criptanalitice	32
2.6.2.	Criterii de evaluare	33
2.6.3.	Patru operații de bază ale criptanalizei	33
2.6.4.	Evaluare și spargere	34
2.7.	Clasificări ale atacurilor criptanalitice	38
2.7.1.	Tipuri de atac asupra algoritmilor de cifrare	38
2.7.2.	Tipuri de atac asupra cheilor	40
2.7.3.	Tipuri de atac asupra protocoalelor de autentificare	41
2.7.4.	Tipuri de atac asupra sistemului	42
2.7.5.	Atacuri hardware asupra modulelor criptografice	42
2.8.	Aplicații	43
3	TEORIA COMPLEXITĂȚII ALGORITMILOR	45
3.1.	Introducere	45
3.2.	Algoritmi și mașini Turing	45
3.3.	Teoria problemelor NP - complete	46
3.4.	Exemple de probleme NP - complete	48
3.5.	Limite actuale ale calculatoarelor	51
3.6.	Aplicații	52
4	ANALIZA STATISTICO-INFORMAȚIONALĂ	53
4.1.	Noțiuni teoretice	53
4.2.	Generatoare și teste statistice	54
4.2.1.	Generatoare uniforme	54
4.2.2.	Conceptul de test statistic	54
4.2.3.	Modele statistice pentru generatoare	56
4.2.4.	Teste elementare de aleatorism statistic	57
4.2.5.	Interpretarea rezultatelor testelor statistice	58
4.3.	Entropia variabilelor aleatoare discrete	59
4.4.	Surse de aleatorism de numere întregi	62
4.4.1.	Formula analitică a probabilității ideale a unei surse de aleatorism de numere întregi	62
4.4.2.	Metoda de calcul efectiv al lui p respectiv q	63
4.5.	Metode de decorelare	64
4.5.1.	Metode deterministe	64
4.5.2.	Metode nedeterministe	64
4.6.	Teste statistice de aleatorism	65
4.6.1.	Algoritmul de implementare al testului frecvenței	65
4.6.2.	Algoritmul de implementare al testului serial	66
4.6.3.	Algoritmul de implementare al testului succesiunilor	67

4.6.4.	Algoritmul de implementare al testului autocorelației temporale	68
4.6.5.	Algoritmul de implementare al testului autocorelațiilor temporale	68
4.6.6.	Algoritmul de implementare al testului autocorelației circulare	69
4.6.7.	Algoritmul de implementare al testului autocorelațiilor circulare	70
4.6.8.	Algoritmul de implementare al testului poker	70
4.6.9.	Algoritmul de implementare al testului <i>CUSUM</i> (sumelor cumulate)	72
4.6.10.	Algoritmul de implementare al testului de aproximare a entropiei	75
4.6.11.	Algoritmul de implementare al testului lui Maurer (1992) și testul entropiei	76
4.6.12.	Algoritmul de implementare al testului χ^2	78
4.6.13.	Algoritmul de implementare al testului Kolmogorov-Smirnov	80
4.6.14.	Testul spectral (transformarea Fourier discretă)	81
4.6.15.	Teste de corelație	83
4.6.16.	Algoritmul de implementare al testului corelațiilor temporale și circulare	83
4.6.17.	Creșterea sensibilității algoritmilor de testare statistică	83
4.7.	Teste de aleatorism algoritmic	85
4.7.1.	Scurt istoric	85
4.7.2.	Măsurarea complexității	86
4.7.3.	Complexitatea segmentului	89
4.7.4.	Complexitatea segmentului ca măsură a aleatorismului	90
4.8.	Teste de necorelare algoritmică	92
4.8.1.	Formularea problemei	92
4.8.2.	Principii de test	92
4.9.	Teste de verificare a jocurilor de tip Casino	93
4.9.1.	Metoda $3-\sigma$ pentru ruletă	94
4.9.2.	Metoda $3-\sigma$ pentru diferențe la ruletă	94
4.9.3.	Metoda X^2 pentru ruletă	94
4.9.4.	Metoda X^2 aplicată diferențelor pentru ruletă	95
4.9.5.	Metoda X^2 pentru jocurile de tip loto	95
4.10.	Aplicații	95
5	CODIFICAREA IN ABSENȚA PERTURBAȚIEI	99
5.1.	Introducere	99
5.2.	Codificarea în absența perturbației	99
5.3.	Codurile Fano și Huffman	102
5.3.1.	Algoritmul de implementare a codului Fano	102
5.3.2.	Algoritmul de implementare a codului Huffman	102

5.4.	Coduri optime	103
5.5.	Aplicații	104
6	CRIPTANALIZA CIFRURILOR CLASICE	109
6.1.	Substituția simplă și multiplă	109
6.1.1.	Substituția simplă	109
6.1.2.	Substituția multiplă	111
6.2.	Substituția polialfabetică	113
6.2.1.	Caracteristicile și identificarea sistemelor de substituție polialfabetică	113
6.2.2.	Atacul sistemelor polialfabetice	114
6.3.	Soluția unui cifru de substituție	114
6.4.	Transpoziția	115
6.5.	Sisteme mixte	115
6.6.	Proceduri de identificare a sistemului	115
6.6.1.	Funcția Kappa	116
6.6.2.	Funcția Chi	117
6.6.3.	Funcția Psi	118
6.6.4.	Funcția Phi	120
6.7.	Funcții simetrice de frecvență a caracterelor	121
6.8.	Atac stereotip asupra cifrurilor de substituție	122
6.9.	Atac de tip frecvență maximă a cifrurilor de substituție	122
6.10.	Concluzii	123
6.11.	Aplicații	124
7	CRIPTANALIZA CIFRURILOR FLUX	127
7.1.	Atacul generatoarelor pseudoaleatoare	127
7.2.	Criptanaliza liniară	128
7.2.1.	Complexitatea liniară	128
7.2.2.	Algoritmul Berlekamp-Massey. Rezultate teoretice	134
7.2.3.	Implementarea algoritmului Berlekamp-Massey	134
7.2.4.	Testul Berlekamp ca test statistico-informațional	135
7.3.	Metoda corelației	137
7.4.	Metoda corelației rapide	137
7.4.1.	Transformata Walsh-Hadamard	137
7.4.2.	Testul statistic Walsh-Hadamard	141
7.4.3.	Caracterizarea proprietăților criptografice	144
7.5.	Atacul Siegenthaler	148
7.6.	Atacul consistenței liniare	148
7.7.	Metoda sindromului linear	149

7.7.1.	Formularea problemei	149
7.7.2.	Preliminarii teoretice	149
7.7.3.	Algoritmul Sindromului Linear	150
7.7.4.	Numere critice și numere de rafinare	150
7.8.	Corecția iterativă a erorii	154
7.8.1.	Prezentare generală	154
7.8.2.	Prezentarea algoritmilor de corecție iterativă	155
7.8.3.	Rezultate experimentale	157
7.8.4.	Concluzii	157
7.9.	Algoritm de criptanaliză diferențială	159
7.10.	Câteva tehnici de proiectare	160
7.10.1.	Transformarea neliniară feed-forward	160
7.10.2.	Generatorul Geffe	160
7.10.3.	Generatorul Jennings	161
7.10.4.	Generatorare cu tact controlat	162
7.10.5.	Generatoare cu ceasuri multiple	165
7.10.6.	Generatoare autodecimate	166
7.11.	Exemplu de atac criptanalitic	166
7.12.	Aplicații	168
8	CRIPTANALIZA CIFRURILOR BLOC	173
8.1.	Introducere și concepte generale	173
8.2.	Securitatea și complexitatea atacurilor	174
8.3.	Criterii de evaluare a cifrurilor bloc	174
8.4.	Moduri de operare	174
8.4.1.	Modul ECB (electronic code-book)	175
8.4.2.	Modul CBC (cipher-block chaining)	176
8.4.3.	Modul CFB (cipher feedback)	179
8.4.4.	Modul OFB (output feedback)	180
8.4.5.	Modul BC (block chaining)	182
8.4.6.	Modul BC cu sumă de control (BC-checksum)	182
8.4.7.	Modul OFBNLF (output feedback block with a nonlinear function)	182
8.4.8.	Cascade de cifruri și cifrări multiple	183
8.5.	Generarea tabelor de substituție	186
8.6.	Criptanaliza diferențială	187
8.7.	Criptanaliza liniară	187
8.8.	Alte metode	187
8.9.	Implementări și rezultate experimentale	188
8.9.1.	Implementarea standardului de cifrare A.E.S.	188

8.9.2.	Testarea algoritmului AES	189
8.9.3.	Rezultate experimentale	189
8.9.4.	Interpretarea rezultatelor	192
8.10.	Concluzii	192
8.11.	Aplicații	193
9	CRIPTANALIZA CIFRURILOR CU CHEI PUBLICE	197
9.1.	Principii de bază	197
9.1.1.	Introducere	197
9.1.2.	Securitatea algoritmilor cu cheie publică	198
9.1.3.	Comparații ale algoritmilor asimetrice și a algoritmilor simetrici	198
9.2.	Algoritmi de tip rucsac	199
9.2.1.	Algoritmi rucsac supercrescător	199
9.2.2.	Crearea cheii publice din cheia privată	199
9.2.3.	Cifrarea	200
9.2.4.	Descifrarea	200
9.2.5.	Implementarea efectivă	200
9.3.	Algoritmul RSA	200
9.3.1.	Descrierea principiilor de cifrare și descifrare	200
9.3.2.	Viteza algoritmilor tip RSA	201
9.3.3.	Securitatea RSA-ului	203
9.3.4.	Tipuri de atacuri asupra algoritmilor RSA	204
9.3.5.	Trape în generarea cheilor RSA	209
9.4.	Algoritmul Pohlig-Hellman	209
9.5.	Algoritmul Rabin	210
9.6.	Algoritmul ElGamal	211
9.7.	Curbe eliptice	211
9.8.	Aplicații practice	213
9.9.	Teste de primalitate și metode de factorizare	214
9.9.1.	Teste de primalitate	214
9.9.2.	Metode de factorizare	217
9.9.3.	Metode de generare a numerelor prime	217
9.10.	Infrastructura Cheilor Publice (PKI)	218
9.10.1.	Elementele PKI	218
9.10.2.	Ghid de folosire a tehnologiei PKI în rețele deschise	219
9.10.3.	Riscuri ale utilizării tehnologiei PKI	220
9.10.4.	Standarde ale PKI	221
9.11.	Aplicații	221

10	CRIPTANALIZA SEMNĂTURILOR DIGITALE	225
10.1.	Prezentare generală	225
10.2.	Noțiuni preliminare	226
10.3.	Funcții hash	228
10.3.1.	Generalități	228
10.3.2.	Algoritmi hash	229
10.3.3.	Funcții hash bazate pe cifruri bloc	230
10.3.4.	Funcții hash nebazate pe cifruri bloc	230
10.4.	Modalități de realizare a semnăturilor digitale	231
10.4.1.	Aplicarea criptosistemelor simetrice	231
10.4.2.	Aplicarea criptosistemelor asimetrice	232
10.4.3.	Apelarea la funcții hash unidirecționale	232
10.4.4.	Semnături digitale convenționale (normale)	233
10.5.	Alte tipuri de semnături digitale	235
10.5.1.	Semnătura invizibilă	235
10.5.2.	Semnături fail-stop	235
10.6.	Legislația în domeniu	235
10.7.	Aplicații	236
11	CRIPTANALIZA PROTOCOALELOR CRIPTOGRAFICE	237
11.1.	Protocoale elementare	237
11.1.1.	Protocoale de schimb de chei	237
11.1.2.	Protocoale de autentificare	241
11.1.3.	Autentificarea și schimbul de chei	244
11.1.4.	Protocoale de transfer orb	248
11.1.5.	Analiza formală a protocoalelor de autentificare și a protocoalelor de schimb de chei	248
11.1.6.	Divizarea secretului	249
11.1.7.	Partajarea secretului	249
11.2.	Protocoale avansate	249
11.2.1.	Protocol de tip demonstrație convingătoare fără detalii	249
11.2.2.	Protocol de tip dezvăluire minimă	250
11.2.3.	Protocol de tip dezvăluire zero	250
11.2.4.	Protocoale de tip transfer bit și aplicații	250
11.2.5.	Alte protocoale avansate	254
11.3.	Divizarea și partajarea secretelor	254
11.3.1.	Protocol de divizare a secretului	255
11.3.2.	Protocolul de partajare LaGrange	255
11.3.3.	Protocolul de partajare vectorial	256
11.3.4.	Protocolul de partajare Asmuth-Bloom	256

11.3.5. Protocolul de partajare Karnin-Greene-Hellman	256
11.3.6. Atacuri asupra protocoalelor de partajare (divizare) a secretului	257
11.4. Exemple de implementare	257
11.4.1. Scheme de autentificare	257
11.4.2. Algoritmi de schimb al cheilor	260
11.5. Aplicații	264
12 CRIPTANALIZA SISTEMELOR DE CIFRARE ANALOGICE	265
12.1. Formularea problemei	265
12.2. Calcul Operațional	265
12.2.1. Transformata Laplace	266
12.2.2. Transformata Fourier	267
12.2.3. Transformata Fourier Discretă	269
12.2.4. Transformata Cosinus Discretă	271
12.2.5. Transformata Walsh	271
12.2.6. Transformata z	272
12.3. Caracterizarea variabilelor aleatoare	273
12.4. Conversia Analogic/Digital	274
12.4.1. Modulația în puls	274
12.5. Cifrarea Analogică	275
12.5.1. Inversarea spectrului	275
12.5.2. Rotirea spectrului inversat	276
12.5.3. Amestecarea spectrului	277
12.5.4. Multiplexarea în timp	279
12.6. Aplicații	279
13 MANAGEMENTUL CHEILOR CRIPTOGRAFICE	281
13.1. Managementul cheilor	281
13.2. Generarea cheilor	283
13.3. Protecția cheilor criptografice	284
13.3.1. Cheie utilizator	284
13.3.2. Arhivarea cheilor	285
13.3.3. Distrugerea cheilor	285
13.4. Lungimea cheilor criptografice	286
13.5. Aplicații	287
A METODE ȘI TEHNICI DE PROGRAMARE	289
A.1. Structuri de date	289
A.2. Alocarea memoriei	290

A.3. Recursivitate	290
A.4. Metoda backtracking	290
A.5. Tehnica Divide et Impera	291
A.6. Tehnica branch and bound	291
A.7. Programarea dinamică	292
A.8. Tehnica greedy	292
A.9. Aplicații	293
B ELEMENTE DE TEORIA PROBABILITĂȚILOR	295
B.1. Caracteristici ale variabilelor aleatoare	295
C STATISTICĂ DESCRIPTIVĂ	297
C.1. Momentele unei variabile aleatoare	297
C.2. Teoria selecției	298
C.3. Aplicații	299
D TEORIA ESTIMAȚIEI	301
D.1. Tipuri de estimatori	301
D.2. Margini inferioare ale estimatorilor	302
D.3. Estimația de verosimilitate maximă	304
D.4. Estimația Bayesiană	304
E REPATIȚII STATISTICE	307
E.1. Repartiții continue	307
E.1.1. Repartiția normală	307
E.1.2. Repartiția lognormală	308
E.1.3. Repartiția uniformă	308
E.1.4. Repartiția exponențială	309
E.1.5. Repartiția gama	310
E.1.6. Repartiția beta	312
E.1.7. Repartiția Cauchy	313
E.2. Distribuții discrete	313
E.2.1. Distribuția Bernoulli	313
E.2.2. Distribuția binomială	313
E.2.3. Distribuția Poisson	314
E.2.4. Distribuția hipergeometrică	314
E.2.5. Distribuția geometrică	314
E.3. Calculul numeric al cuantilelor	314
E.3.1. Cuantila repartiției normale	315
E.3.2. Cuantilele repartiției chi-pătrat	315

F	SERII DINAMICE STAȚIONARE	317
F.1.	Exemple de serii dinamice	317
F.2.	Procese stochastice	317
F.3.	Staționaritate și strict staționaritate	319
F.3.1.	Relația dintre Staționaritate și Strict Staționaritate	320
F.4.	Estimarea și eliminarea componentelor tendință și sezoniere	322
F.4.1.	Eliminarea tendinței în absența componenetei sezoniere	323
F.4.2.	Eliminarea simultană a componentelor tendință și sezoniere	325
F.5.	Funcția de autocovarianță a unui proces staționar	327
F.5.1.	Funcția de autocovarianță de selecție	329
G	MODELUL AUTOREGRESIV-MEDIE MOBILĂ	331
G.1.	Modelul autoregresiv $AR(p)$	331
G.2.	Modelul medie mobilă $MA(q)$	332
G.3.	Modelul $ARMA(p,q)$	332
G.4.	Modelul $ARIMA(p,d,q)$	333
G.5.	Probleme puse proceselor $ARIMA(p,d,q)$	333
H	SIMULAREA VARIABILELOR ALEATOARE	335
H.1.	Tehnici de simulare	335
H.2.	Legea tare a numerelor mari	336
I	ELEMENTE DE TEORIA NUMERELOR	339
I.1.	Teste de primalitate	339
I.2.	Lema chinezescă a resturilor	340
I.3.	Numărul de numere prime	341
I.4.	Simbolurile lui Legendre și Jacobi	341
	BIBLIOGRAFIE	343

Capitolul 9

CRIPTANALIZA CIFRURILOR CU CHEI PUBLICHE

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

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

De obicei criptografia simetrică și cea asimetrică (cu chei publice) duc la construcția sistemelor criptografice hibride: se folosește un algoritm simetric (bazat pe o cheie secretă aleatoare) pentru a cifra mesaje și se utilizează un algoritm asimetric (bazat pe o cheie publică și 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 computațională a următoarelor probleme:

- a problemelor NP -complete ca de exemplu *problema rucsacului*: dându-se mulțimea $\{M_1, \dots, M_n\} \subset \mathbf{N}^*$ și 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 ecuației: $y = x^2 \pmod{n}$.

-a logaritmului discret (algoritmul *ElGamal*): să se găsească numărul natural $x < p$ care îndeplinește relația $y = g^x \pmod{p}$ unde p este număr prim și $g < p$;

9.1.3. Comparații ale algoritmilor asimetrici și a algoritmilor simetrici

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

9.2. Algoritmi de tip rucsac

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

9.2.1. Algoritmi rucsac supercrescător

Definiția 9.2.1. Un șir de ponderi $\{M_1, \dots, M_n\}$ se numește *supercrescător* dacă:

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

Problema rucsacului supercrescător este ușor de rezolvat folosind următoarea schemă:

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

Algoritmii de tip rucsac care nu sunt supercrescători nu sunt ușor de rezolvat și nu există nici un algoritm rapid care să rezolve problema. Singura modalitate cunoscută de a determina dacă $b_i = 1$ constă în testarea tuturor soluțiilor. Cei mai rapizi algoritmi de testare au o complexitate exponențială. Algoritmul Merke-Hellman se bazează pe această proprietate: *cheia privată* este șirul ponderilor pentru un rucsac supercrescător iar *cheia publică* este șirul ponderilor pentru un rucsac care are aceeași soluție. *Merkle* și *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 dispoziție o problemă de tip rucsac supercrescător (cheia privată) cu ponderile $\{M_1, \dots, M_n\}$ atunci aceasta se transformă într-o problemă de tip rucsac normală (cheia publică) cu șirul ponderilor

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

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

9.2.3. Cifrarea

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

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

9.2.4. Descifrarea

Destinatarul mesajului cunoaște cheia privată: ponderile originale și valorile lui m și 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 puțin 250 de ponderi. Ponderile sunt cu lungimi între 200 și 400 biți, iar modulul are o lungime între 100 și 200 biți. În cazul implementărilor reale atât ponderile cât și numere m și n sunt generate aleator. Atacul brut nu este fezabil la valorile prezentate anterior. *Shamir* a indicat o serie de condiții în care se poate sparge un astfel de algoritm.

9.3. Algoritmul RSA

Algoritmul *RSA* a fost inventat¹ de către *Ron Rivest*, *Adi Shamir* și *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] și [42] pentru o introducere în domeniu). Cheia publică și cheia privată sunt funcție de o pereche de numere prime mari (de 200 de cifre sau chiar mai mari). Recuperarea textului clar din cheia publică și textul cifrat este chivalent cu factorizarea produsului a două numere prime.

9.3.1. Descrierea principiilor de cifrare și descifrare

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

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

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

$$ed \equiv 1 \pmod{(p-1)(q-1)}.$$

Cu alte cuvinte

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

Remarcăm faptul că d și n sunt relativ prime. Numerele e și n constituie cheia publică iar d este cheia privată. Cele două numere p și 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 și 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 operația de padding cu zero. Mesajul cifrat c se va obține prin concatenarea mesajelor c_i care au aproximativ aceeași lungime. Formula de cifrare va fi:

$$c_i \equiv m_i^e \pmod{n}.$$

Pentru a descifra un mesaj se calculează:

$$m_i \equiv c_i^d \pmod{n},$$

deoarece

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

Observația 9.3.1. Pentru a evita metodele de factorizare cunoscute numerele p și q trebuie să fie numere *prime tari*. Un număr prim p se numește 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 biți modulul) este de 64 kilobiți 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*) și standardul PKCS#1 (*Public Key Cryptographic System*)), 17 și $2^{16} + 1$ (recomandat de standardul de *certificate digitale* X.509 și 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 aceiași valoare pentru parametru e . Viteza operațiile private se poate mări prin utilizarea *lemei chinezești a resturilor* (CRT) dacă se stochează valorile lui p , q , $d \bmod (p-1)$, $d \bmod (q-1)$ și $q^{-1} \bmod p$. Utilizarea lemei chinezești a resturilor înlesnește aplicarea atacurilor de tip timp (*timing attacks*: în funcție de timpul de execuție se pot deduce anumiți biți din cheie) sau atacuri de tip eroare hardware.

Utilizarea CRT

Operația de semnare a unui mesaj M se realizează prin exponențierea amprenteii digitale a documentului $H(M)$ cu ajutorul cheii private: $s = H(M)^d \bmod n$. Verificarea semnăturii se realizează prin comparația 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 diferiți între operațiile private (descifrare/semnare) și cele publice (cifrare/verificare semnătură).

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

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

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

În faza de calcul se execută:

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

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

9.3.3. Securitatea RSA-ului

Securitatea algoritmului RSA este direct proporțională 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 evidență faptul că recuperarea unor biți de informație 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 dispoziție cheia publică e și 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 biți). Metoda de atac brut este mai puțin eficientă decât tehnica factorizării.

Cei mai mulți algoritmi folosesc tehnici probabiliste pentru a calcula numere prime p și q . Se pune, în mod natural, întrebarea dacă algoritmi de testare a primalității detectează numerele compozite. O serie de algoritmi de testare a primalității 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 conștientizat faptul că folosirea algoritmului RSA nu este suficientă iar detaliile sunt de maximă importanță. Vom prezenta trei cazuri de atac cu text cifrat ales.

Cazul 1. Interceptorul pasiv E , va monitoriza comunicațiile lui A și va stoca toate mesajele c cifrate cu ajutorul cheii publice a lui A . Interceptorul dorește 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 și va calcula:

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

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

$$u \equiv y^d \pmod{n}.$$

Interceptorul E va calcula:

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

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

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

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

Slăbiciunea ce a fost exploatată a fost aceea că exponențierea produsului este produsul exponențierilor:

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

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

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

Interceptorul E va forța pe A să semneze mesajele m_1 și 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 folosiți niciodată algoritmul RSA pentru semnarea unui document necunoscut. Folosiți apriori o funcție hash. Formatul *ISO9796* previne acest tip de atac.

Atac cu ajutorul modulelor comune

O posibilă implementare a RSA dă aceeași valoare n , dar valori diferite pentru exponenții e și d . Cea mai evidentă problemă este aceea că dacă același mesaj este cifrat cu doi exponenți diferiți (dar având aceleași module), iar acei exponenți sunt numere relativ prime, atunci textul clar poate fi descoperit fără a cunoaște exponenți de descifrare.

Fie m mesajul în clar. Cele două chei publice de cifrare sunt e_1 și 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 cunoaște n, e_1, e_2, c_1 și c_2 . În continuare se prezintă cum se calculează mesajul clar m .

Întrucât e_1 și e_2 sunt relativ prime, algoritmul extins al lui Euclid poate găsi r și 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,

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

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

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

Atac asupra exponenților de cifrare de dimensiuni mici

Cifrarea și verificarea semnăturii RSA sunt mai rapide dacă se folosește 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 aceeași valoare a lui e , atunci există un atac împotriva sistemului. Dacă sunt mai puțin 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 obține prin calcul direct $m = c^{\frac{1}{e}}$). Cea mai simplă soluție este să se completeze mesajele cu valori aleatoare independente. Aceasta ne asigură de faptul că $m^e \pmod{n} \neq m^e$. Implementările RSA din produsele PEM și PGP realizează acest lucru.

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

Atac asupra exponenților 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 și cheia publică e este mai mică decât modul n . Acest lucru se întâmplă mai rar dacă e și d sunt alese aleator și 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 și q sunt suficient de apropiate următoarea observație 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 și 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 operațiilor private (de exemplu pentru realizarea semnăturii), factorizarea modului n . Pentru fixarea ideilor fie M mesajul care urmează a fi semnat cu cheia privată d . Semnătura acestui mesaj $E \equiv M^d \pmod{n}$ este calculată ca

$$E \equiv aE_1 + bE_2 \pmod{n},$$

unde

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

și

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

Reținem faptul că factorizarea $n = p \cdot q$ este trapa secretă a sistemului. Presupunem acum că dispunem de semnătura E a mesajului M și de o semnătură greșită a aceluiași mesaj (a apărut *numai o eroare hardware* în calculul lui E_1 sau E_2). Presupunem, fără restrângerea generalității că E_1 a fost calculat eronat ca fiind \hat{E}_1 și 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 hardware (ca urmare a erorilor latente, tranziente sau induse) pot fi aplicate algoritmului de semnătură a lui Rabin, protocolului de identificare Fiat-Shamir precum și protocolului de identificare Schnorr.

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

Atacul asupra cifrării și semnării cu algoritmul RSA

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

Fie utilizatorul A care dorește 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 și 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 situație B poate pretinde că a primit mesajul m' de la A . Funcțiile hash nu rezolvă problema. Se poate opta pentru utilizarea unui exponent de cifrare fixat.

Condiții necesare dar nu și suficiente pentru asigurarea securității criptografice a RSA

O serie de restricții folosite în implementarea RSA se bazează pe succesul atacurilor menționate:

- cunoașterea unei perechi de exponenți cifrare/descifrare pentru un modul dat permite factorizarea modulului;
- cunoașterea unei perechi de exponenți cifrare/descifrare pentru un modul dat permite generarea de noi exponenți 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 exponenților de cifrare mici, precum și atacul de tip eroare hardware (în cazul utilizării CRT pentru optimizarea operațiilor 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 și 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. Evidențierea existenței acestor trape conduce la compromite imaginea producătorului produsului criptografic. Vom exemplifica o serie de astfel de trape și anume:

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

-ascunderea exponentului public mic e , precum și a unei informații referitoare la exponentul privat d . Acest lucru se face prin utilizarea unei funcții 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 trapelelor menționate mai sus trebuie dezvoltate teste de detecție a acestora.

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

Ascunderea exponentului privat mic d

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

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

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 cunoașterea 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 Pohlig-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 ușor una din cealaltă, atât cheia de cifrare cât și cheia de descifrare trebuie să fie secrete. La fel

ca în algoritmul RSA:

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

unde

$$ed \equiv 1 \pmod{\text{un număr complicat}}.$$

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 și n atunci putem calcula pe d . Fără cunoașterea lui e sau d interceptorul trebuie să rezolve problema (dificilă din punct de vedere computațional):

$$e \equiv \log_P C \pmod{n}.$$

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 și q ambele congruente cu 3 mod 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

$$C \equiv M^2 \pmod{n}.$$

Deoarece receptorul cunoaște cele două numere p și q acesta va rezolva două dintre congruențe cu ajutorul lemei chinezești a resturilor (vezi Anexa I). Calculăm:

$$\begin{aligned} m_1 &\equiv C^{\frac{p+1}{4}} \pmod{p} \\ m_2 &\equiv (p - C^{\frac{p+1}{4}}) \pmod{p} \\ m_3 &\equiv C^{\frac{q+1}{4}} \pmod{q} \\ m_4 &\equiv (q - C^{\frac{q+1}{4}}) \pmod{q} \end{aligned}$$

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

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

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 secvență aleatoare (o cheie sau o semnătură digitală) atunci nu putem să indicăm nici o variantă. Soluția 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 și 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 și două numere g și x ambele mai mici decât p . Vom calcula:

$$y \equiv g^x \pmod{p}.$$

Cheia publică este y, g, p iar cheia privată este x . Ambele numere g și 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:

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

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

Pentru a descifra (a, b) calculăm:

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

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

9.7. Curbe eliptice

Folosirea curbelor eliptice în criptografie a fost propusă pentru prima oară în 1985 de *Victor Miller* (IBM) și, 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 eliptice au fost folosite pentru conceperea unor algoritmi eficienți de factorizare a întregilor și pentru demonstrarea primalității. Ele au fost utilizate în construirea criptosistemelor cu chei publice, în construcția generatoarelor pseudoaleatoare

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

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

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

(unde a și b sunt constante astfel încât $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, iar p este un număr prim) împreună cu un elemente singular, notat O și numit *punctul de la infinit*. Acest punct poate fi privit ca fiind punctul din vârful și de la baza oricărei linii verticale.

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

Fiind date două puncte pe E , $P_1(x_1, y_1)$ și $P_2(x_2, y_2)$, avem următoarele cazuri:

-dacă $x_2 = x_1$ și $y_2 = -y_1$ atunci $P_1 + P_2 = O$.

-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

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

Operația de adunare pe o curbă eliptică este corespondenta operației de înmulțire în sistemele cu chei publice obișnuite, iar adunarea multiplă este corespondența exponențierii modulare din acestea.

Deși 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 .

Corespondența problemei logaritmilor discreți 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) și 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

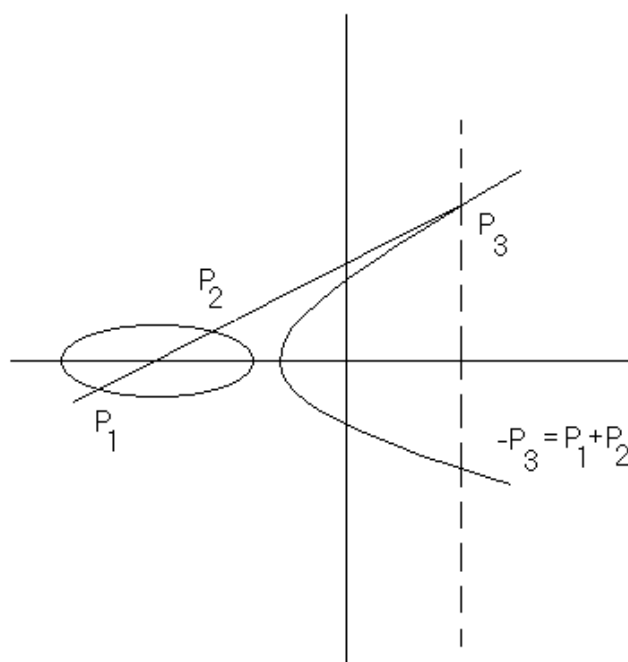


Figura 9.1: Operația de adunare pe o curbă eliptică.

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

9.8. Aplicații practice

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

- aplicații software cum ar fi programul de protecție a poștei 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) și a semnării mesajului (cu ajutorul cheii private a emitentului);

-smart-carduri folosite în sistemul de plăți electronice;

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

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

Direcții ulterioare de investigare ar consta în identificarea vulnerabilităților ce pot apare în cazul utilizării problemelor computaționale de *calculul a rădăcinii pătrate modulo un număr compozit* (algoritmul *Rabin*) și 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 și metode de factorizare

9.9.1. Teste de primalitate

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

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

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

Teorema lui Wilson. *Numărul p este prim dacă și numai dacă avem congruența $(p-1)! \equiv -1 \pmod{p}$.*

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

Pentru valori mari ale lui p testul lui Wilson și 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 *pseudoprime* cu baza b dacă $b^{p-1} \equiv 1 \pmod{p}$. 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 numește *număr Carmichael* dacă este pseudoprime 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 depășește 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 exponențiere modulară (ridicare la pătrat urmată de stabilirea restului modulo p), stabilește 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 $n \geq 3$ și un parametru de securitate $t \geq 1$.

Ieșire: decizia asupra primalității.

PASUL 1. Pentru $i = 1, \dots, t$ 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.

Observația 9.9.1. Un număr compozit n care este declarat prim de testul Fermat se numește *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:

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

unde $\left(\frac{a}{n}\right)$ 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 $n \geq 3$ și un parametru de securitate $t \geq 1$.

Ieșire: decizia asupra primalității.

PASUL 1. Pentru $i = 1, \dots, t$ 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 $r \neq 1$ și $r \neq n - 1$ atunci se decide n este compozit.

1.4 se calculează simbolul lui Legendre $s = \left(\frac{a}{n}\right)$.

1.5 dacă $r \neq s$ atunci se decide n este compozit.

PASUL 2. Se decide: n este prim.

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

Observația 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 și ca testul tare de primalitate și se bazează pe următoarea teoremă.

Teorema 9.9.1. Fie n un număr prim impar, și $n - 1 = 2^s r$ unde r este impar. Fie deasemenea un întreg a astfel ca $(a, n) = 1$. Atunci $a^r \equiv 1 \pmod{n}$ sau $a^{2^j r} \equiv -1 \pmod{n}$ 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 $n \geq 3$ și un parametru de securitate $t \geq 1$.

Ieșire: decizia asupra primalității.

PASUL 1. Se scrie $n - 1 = 2^s r$ unde r este impar.

PASUL 2. Pentru $i = 1, \dots, t$ se execută:

2.1 se alege aleatoriu a cu $2 \leq a \leq n - 2$.

2.2 calculează $y = a^r \pmod{n}$.

2.3 dacă $y \neq 1$ și $y \neq n - 1$ atunci se execută:

$j = 1$.

Atâta timp cât $j \leq s - 1$ și $y \neq n - 1$ se execută:

calculează $y = y^2 \pmod{n}$.

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*.

Observația 9.9.4. Un număr compozit n care este declarat prim de testul Miller-Rabin se numește *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)*.

Observația 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 eficienței, în sensul că mulțimea numerelor false Miller-Rabin este inclusă în mulțimea numerelor false Euler care la rândul ei este inclusă în mulțimea numerelor false Fermat.

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

Observația 9.9.6. Dacă $n \equiv 3 \pmod{4}$ atunci un număr a este număr fals Euler dacă și numai dacă acesta este fals Miler-Rabin.

9.9.2. Metode de factorizare

Metodele de factorizare se rulează de regulă după testele de primalitate și pun în evidență 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

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

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

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

Ieșire. Un număr prim p cuprins între limitele p_{\min} și p_{\max} .

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

PASUL 2. Dacă $p > p_{\max}$ atunci $p = p_{\min} + p \pmod{p_{\max} + 1}$ și 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 $p = p + 2$ și 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} și p_{\max} și un parametru de securitate $t \geq 1$ și s un parametru de fiabilitate.

Ieșire. Un număr prim p tare.

PASUL 1. Se generează aleatoriu două numere prime aleatoare v și w de aproximativ aceeași 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 $i = i_0, i_0 + 1, \dots$

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

PASUL 4. Se alege un întreg j_0 . Fie $p = p_0 + 2j_0v$ primul număr prim în progresia $p_0 + 2j_0v$ pentru $j = j_0, j_0 + 1, \dots$

PASUL 5. Se returnează numărul prim tare p .

9.10. Infrastructura Cheilor Publice (PKI)

9.10.1. Elementele PKI

Algoritmii criptografici asimetrice conferă un grad de securitate comercial iar aceștia se pot folosi, de regulă, în cadrul protocoalelor criptografice referitoare la transferul cheilor de cifrare sau autentificare. În unele aplicații se pot folosi și în cadrul operației de cifrare. Ambele metode (cifrarea simetrică și cifrarea asimetrică) au punctele lor slabe: managementul cheilor de cifrare respectiv complexitatea algoritmului de cifrare. Una dintre aplicațiile 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 și arhivează certificatele digitale;

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

- *deținătorii de certificate* care generează semnături digitale, generează cereri de certificate, cereri de revocare de certificate, cereri de reînregistrare (opțional);

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

Comunicația datelor între RA și CA trebuie realizată în condiții de deplină siguranță (off-line) pentru a proteja cheia privată a autorității de certificare.

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

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

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

Standardul X.509 V3, oferă printre altele, și *forma certificatelor digitale* care trebuie să conțină: versiunea certificatului, numărul serial, algoritmul utilizat de semnătură utilizat de emitent, identificarea emitentului, perioada de valabilitate, identificarea posesorului, informații despre cheia publică a utilizatorului (cheia publică a utilizatorului: versiunea, modulul de cifrare, exponentul de cifrare, etc.), tipul posesorului (utilizator sau CA) informații despre posesor, atribute ale cheii (utilizată la semnătură sau schimbul de chei), informații 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 și 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 operațiilor private cu CRT).

9.10.2. Ghid de folosire a tehnologiei PKI în rețele deschise

Următoarele cinci probleme trebuie urmărite înainte de a implementa tehnologia PKI în rețelele deschise (de exemplu rețeaua Internet).

1. *Beneficiile, directe și indirecte, financiare și nefinanciare, obiective și subiective ale utilizării semnăturilor digitale pentru aplicația 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 folosește sistemul de semnătură digitală sau de a converti un proces non-electronic la unul electronic ce folosește sistemul de semnătură digitală, b) de funcționare după operația de conversie:*

- nivelul de încredere solicitat;
- integritatea cheilor publice și a cheilor private;
- cuantificarea consecințelor potențialelor riscuri;
- politici, practici și proceduri;
- conectarea la o infrastructură existentă;
- interoperabilitatea cu alte infrastructuri;

- înregistrarea operațiilor (a managementului);

- conformitatea cu standardele PKI;

- realizarea aplicațiilor program;

- evaluările părților implicate;

- cerințe de statut adiționale;

3. *Riscurile asociate cu folosirea cheii publice pentru această aplicație:*

- frauda;

- întreruperea serviciului sau disfuncționalități de scurtă durată;

- legibilitate;

4. *Beneficiile determinate la prima problema comparativ cu costurile stabilite la cea de-a doua problemă și 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 și folosi PKI pentru operația de semnătură digitală:*

- realizarea unei politici de certificare și reguli practice de certificare;

- stabilirea acelor *directory service* care sunt necesare aplicației pe care o utilizează

PKI, asigurarea disponibilității acestora;

- interoperabilitatea cu alte infrastructuri externe similare (PKI);

- adaptarea aplicațiilor la structura PKI;

- implementarea structurii PKI și a aplicațiilor în mod secvențial;

- întegrarea procesului de înregistrare la PKI în politica de personal;

- identificarea cerintelor softului PKI și a aplicațiilor program pentru a opera prin firewall și rutere;

- tipul de funcționare on-line și/sau off-line;

- stabilirea personalului ce realizează operația 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 și pentru ce anume.

2. Protecția 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 și dualismul RA.

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

9.10.4. Standarde ale PKI

Deoarece o infrastructură cu chei publice trebuie să fie interoperabilă cu alte structuri similare a apărut ca o necesitate standardizarea formei certificatelor digitale și a funcțiilor criptografice utilizate. Un astfel de standard este standarul *X.509* pentru certificate și 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 și Stafford Tavares), IDEA (Xuejia Lai și James Massey), RC2, RC5 (Ron Rivest).
2. *Cifruri flux*: RC4 (Ron Rivest).
3. *Cifruri cu chei publice*: RSA (Rivest, Shamir, Adelman), EC (curbe eliptice), PKCS (firma RSA).
4. *Funcții hash*: SHA (NIST și NSA), MD2, MD4, MD5 (Ron Rivest), RIPEMD (Comunitatea Europeană), LHASH (Eric Young).
5. *Funcții de semnătură digitală*: DSA (NIST), MDC (funcții de semnătură cu ajutorul cifrurilor bloc), HMAC (funcții de semnătură cu ajutorul funcțiilor 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. Aplicații

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

Răspuns. Criptograma este:

$$E = M^s = 3^7 = 2187 = 130 \pmod{187}.$$

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

Exponentul de descifrare va fi:

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

Descifrarea mesajului E va fi:

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

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

Exercițiul 9.11.3. Să se cifreze mesajul binar 011000110101101110 cu ajutorul cheii publice setate la exercițiul 9.11.2.

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

Exercițiul 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 închide fără a utiliza cheia, iar pentru a deschide acest lacăt este nevoie de cheia privată.

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

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

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

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

Exercițiul 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 $z = x^3 + x + 6 \pmod{11}$, se vede care din aceste valori sunt reziduri pătratice cu ajutorul teoremei lui Euler (z este reziduu pătratic dacă și numai dacă $z^{\frac{p-1}{2}} \equiv 1 \pmod{p}$) iar apoi

se calculează rădăcinile pătrate ale acestor reziduri prin formula $y = \pm z^{\frac{p+1}{2}} \pmod{p}$. 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) și se ia ca generator pentru acesta elementul (public) $\alpha = (2, 7)$. Cheia privată de descifrare, notată prin d , este o valoare între 1 și numărul de puncte de pe o curbă eliptică -1 . Cheia publică, notată prin β , se obține din α și exponentul secret d prin formula $\beta = d\alpha$.

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

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

Operația de descifrare pentru a obține M este:

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

Exercițiul 9.11.11. Implementați un algoritm de generare a cheilor RSA care să conțină trape prin:

- ascunderea exponenților privați mici;
- ascunderea exponenților publici primi;
- ascunderea factorilor primi.

Exercițiul 9.11.12. Prin ce se caracterizează criptografia asimetrică față de criptografia simetrică?

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