Introducere în Criptografie

Funcții Criptografice, Fundamente Matematice și Computaţionale

Bogdan Groza

Prefaţă

Prezenta lucrare, extinde lucrarea autorului publicată în 2007 sub titlul *Introducere în criptografia cu cheie publică*[1]. Această extensie, care refoloseşte o bună parte din textul anterior era necesară din două motive: pe de o parte textul anterior cerea îmbunătăţiri şi actualizări, pe de altă parte o carte dedicată strict criptografiei cu cheie publică adresa o nişă pentru care era greu de găsit o audienţă potrivită (cel puţin în cadrul universităţii în care autorul îşi desfăşoară activitatea didactică).

Astfel, prezenta lucrare completează lucrarea anterioară cu detalii privitoare la criptografia cu cheie simetrică, funcţii hash, coduri de autentificare, noţiuni de bază despre curbe eliptice, etc., oferind o imagine mult mai completă asupra criptografiei. Nu în ultimul rând, prezenta carte este văzută de autor ca fiind prima dintr-o serie de trei volume ce formează un compendiu de criptografie şi securitatea informaţiei. Volumul 1 este dedicat funcţiilor criptografice şi fundamentelor matematice şi computaţionale, în timp ce volumul 2 va fi dedicat protocoalelor criptografice şi aplicaţiilor practice iar volumul 3 unor concepte avansate în criptografie. Orizontul de timp în care se doreşte publicarea celorlalte două volume este 5-10 ani.

Trecând mai bine de trei decenii de la intrarea criptografiei în domeniul academic, există multe cărţi de referinţă ale domeniului şi desigur ne putem întreba de ce este utilă încă o carte, aceasta. În contextul actual nu pare fezabil a preda criptografie după niciuna din cărţile mari ale domeniului, în special la nivelul studiilor de licenţă. Poate acest lucru ar deveni fezabil în câţiva ani, sau poate este parţial fezabil şi acum la ciclul master, dar din punctul de vedere al autorului este necesară mult mai multă seriozitate din partea studenţilor care participă. Altfel riscăm a preda criptografie doar pentru 5% din audienţă. Mijloace moderne de predare (computerul şi proiectorul) fac flexibilă predarea şi ne oferă posibilitatea de a modela o programă de studii după cum dorim. Dar foliile de curs în format de proiectare conţin doar informaţie fragmentară, incompletă, punând studentul la a rezolva un puzzle prin citirea lor. Desigur poate fi un puzzle mai greu sau mai simplu în funcţie de priceperea autorului, la fel şi cartea, dar în orice caz cartea oferă informaţii mai fluente, mai uşor de interpretat şi este în general un instrument de învăţare mai bun.

---

[1] Bogdan Groza, Introducere in Criptografia cu Cheie Publica, Editura Politehnica, 2007.

Nu în ultimul rând, o carte oferă viziunea personală a autorului asupra domeniului şi poate chiar modul în care autorul l-a învăţat. Cu riscul de a nu fi cea mai bună cale, în lipsa unui criteriu pentru aceasta, trebuie să ne asumăm puterea de a spune ce avem de spus.

Existenţa unor cărţi în limba română, într-un moment în care limba ştiinţei la nivel mondial este engleza, nu poate fi justificată prin prisma noţiunii de cultură ştiinţifică naţională, deoarece ştiinţa nu are o naţie, ea este în cele din urmă a tuturor (matematica nu diferă între continente). Dar, existenţa unor cărţi în limba naţională, devine peste ani dovada incontestabilă a existenţei unui învăţământ naţional de tradiţie în acel domeniu. Pentru a conserva partea pozitivă din identitatea învăţământului românesc, este bine să avem şi cărţi de studiu în limba română.

Timişoara, 2012

Cuprins

1 INTRODUCERE

"Cryptography is communication in the presence of adversaries" – R. Rivest.

Ce este securitatea informaţiei şi care este rolul criptografiei, acestea sunt două întrebări la care dorim să răspundem. Evitând o explicaţie tip dicţionar care nu ar sluji scopului acestei cărţi, vom încerca să conturăm un răspuns mai larg în cadrul unui context general. Vom face aceasta în cadrul obiectivelor de securitate şi al adversarilor la adresa acestora, un cadru ce poate încadra funcţiile criptografice ca obiecte ce au un scop precis. Continuăm apoi cu o scurtă privire istorică asupra criptografiei şi o taxonomie a funcţiilor criptografice. Nu în ultimul rând, facem o trecere în revistă a literaturii relevante în domeniu.

1.1 SECURITATEA INFORMAŢIEI, DEFINIREA PROBLEMEI

Un instrument trebuie văzut în primul rând în cadrul contextului din care face parte, criptografia reprezintă instrumentul de bază în domeniul mai larg al securităţii informaţiei. Într-un secol în care informaţia este indispensabilă, asigurarea securităţii acesteia devine o preocupare de prim rang. Aceasta se datorează faptului că informaţia este lipsită de valoare atâta timp cât atributele ei de securitate nu sunt asigurate. În mare, securitate înseamnă protecţie în faţa unei potenţiale ameninţări iar în ceea ce priveşte informaţia ameninţările pot varia de la simpla alterare neintenţionată a acesteia până la accesarea de către persoane neautorizate sau distrugerea ei.

Securitatea informaţiei este domeniul care se ocupă de studiul mecanismelor de protecţie a informaţiei, în scopul asigurării unui nivel de încredere în informaţie, şi este corect a spune că nivelul de încredere în informaţie depinde de nivelul mecanismelor de securitate care îi garantează protecţia în faţa riscurilor care apar asupra securităţii ei. Tehnicile de asigurare a securităţii informaţiei adresează informaţie indiferent de natura ei şi este important de remarcat că informaţia are valoare în special atunci când este subiect al schimbului sau procesării, deci tocmai atunci când este vulnerabilă în

faţa unor părţi ce nu pot fi considerate de încredere. Astfel, fără a exclude valoarea informaţiei stocate, valoarea informaţiei creşte evident în acţiunea de schimb sau de procesare şi este evident că o informaţie complet izolată nu conduce la riscuri de securitate foarte mari dar în acelaşi timp nici nu poate aduce foarte multe beneficii având în general valoare scăzută.

În diverse materiale pot fi găsite numeroase definiţii şi denumiri asociate domeniului mai larg al securităţii informaţiei. Este relevantă definiţia din glosarului lucrării [26] unde Securitatea Sistemelor Informatice (INFOSEC – Information System Security) este definită după cum urmează:

Securitatea Sistemelor Informatice (INFOSEC) înseamnă protecţia sistemelor informatice împotriva accesului neautorizat sau a modificării informaţiei, fie stocată, procesată sau în tranzit, şi împotriva refuzului de servicii către utilizatorii autorizaţi sau asigurării de servicii către utilizatorii neautorizaţi, incluzând acele metode necesare detectării, documentării şi respingerii acestor ameninţări.

Securitatea informaţiei este un domeniu care provoacă practicantul la un mod de a gândi. Pentru a defini specificaţiile de securitate ale unui sistem, trebuie să existe o imagine suficient de clară asupra sistemului şi pericolelor ce planează asupra sa. O imagine devine clară pe măsură ce apar cât mai multe întrebări şi desigur răspunsuri. Nu există nici un diagnostic general şi nici o reţetă universală pentru a face securitate. Există totuşi câteva întrebări relevante în faţa cărora trebuie să dăm un răspuns înainte de a proiecta o potenţială soluţie de securitate, iată câteva dintre acestea, conform lucrării [31]: cum ajunge adversarul la sistem? Care sunt obiectivele de securitate ce trebuie asigurate în sistem? Care este nivelul de securitate la care trebuie să răspundă sistemul? Desigur perspectivele deschise de aceste întrebări nu sunt nici pe departe exhaustive şi nici suficient de detaliate. De exemplu, poate că fără a introduce problematici noi, putem nuanţa aceste trei întrebări prin altele cum ar fi [3]: ce trebuie protejat? Care sunt ameninţările şi vulnerabilităţile? Care sunt implicaţiile în cazul distrugerii sau pierderii echipamentului? Care este valoarea echipamentului pentru organizaţie? Ce poate fi făcut pentru a minimiza expunerea la pericole? O expunere succintă a problemelor de evaluare a ameninţării şi riscurilor se găseşte în [3].

Este de remarcat că în cadrul răspunsului la prima întrebare putem distinge două situaţii: situaţia în care adversarul ajunge personal la sistem şi situaţia în care ajunge pe cale electronică la sistem. Pentru prima situaţie soluţiile de tratare se încadrează în categoria soluţiilor de **securitate fizică**. Pentru cea de a doua situaţie vorbim de **securitate electronică**, tehnicile criptografice joacă un rol fundamental în acest context. Este de remarcat că aceste două tipuri distincte de soluţii de securitate pot adesea să se contopească, de exemplu un lacăt ataşat

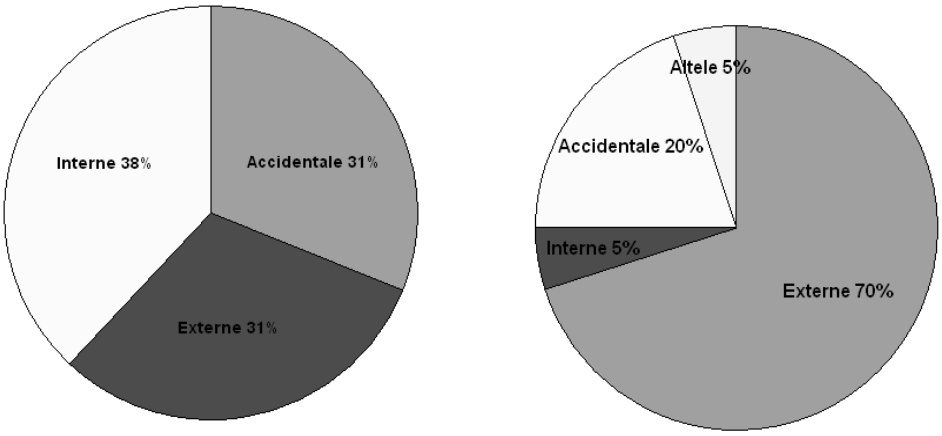
unei uşi este un dispozitiv de securitate fizică, dar unui astfel de lacăt i se poate ataşa un dispozitiv de autentificare bazat pe tehnici criptografice, cum ar fi de exemplu un smart-card. Acest nivel de detaliere este suficient pentru prezentul capitol, dar în practică nu este nici pe departe complet deoarece în cazul securităţii electronice există de asemenea multe alte moduri în care adversarul poate ajunge la sistem şi problema se complică în continuare (de exemplu poate ajunge de pe un suport de date, sau prin intermediului unei reţele etc.).

1.2 INCIDENTUL CA MOTIVAȚIE, EVOLUȚIA CAUZELOR INCIDENTELOR DE LA INTERNE LA EXTERNE

O caracteristică interesantă pentru adoptarea practicilor bune de securitate (şi a dezvoltării tehnicilor criptografice în general) este faptul că aceasta este în general motivată de incidente, de cele mai multe ori incidente cu urmări negative. Într-adevăr, de cele mai multe ori, lumea evită să adopte măsuri de securitate sub pretexte de forma: ”aşa ceva nu se poate întâmpla”. Astfel, din comoditate, lumea evită de exemplu să instaleze un software de securitate pe un smart-phone, software care ar permite în cazul furtului trimiterea unui simplu SMS ce ar porni un sistem de localizare sau ştergerea datelor ce au caracter personal. Odată ce telefonul este furat, posesorul regretă acest lucru. Este nevoie aşadar de incident pentru a conştientiza riscul de securitate. Aici exemplele sunt numeroase, multe lucruri sunt aparent greu crezut, probabil şi înainte de ziua neagră 9/11 puţini ar fi crezut că un zgârie-nori ar putea fi lovit şi doborât de un avion deturnat de teroriști.

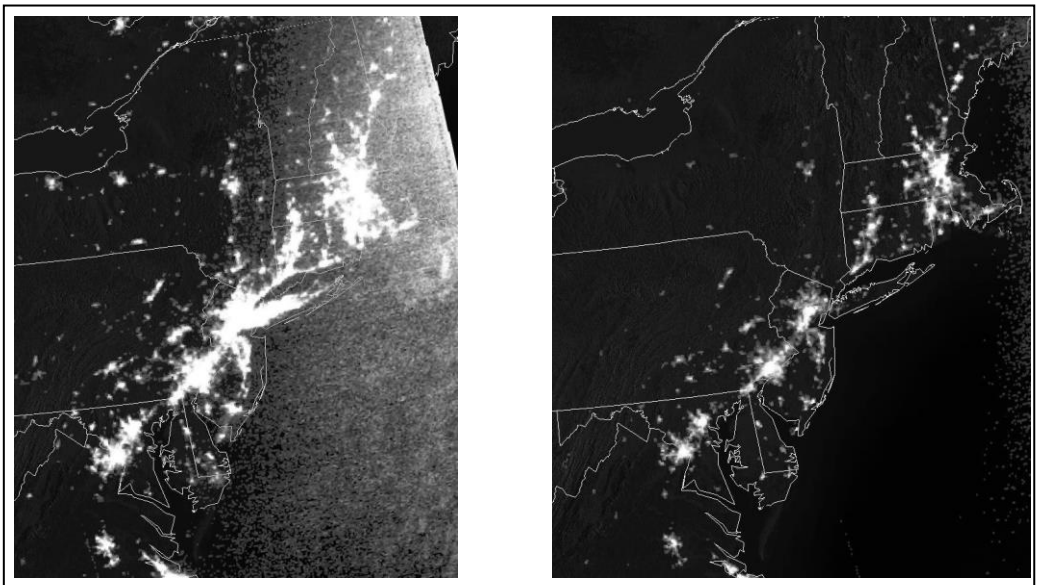
Dacă în perioada 1982-2000 exista un echilibru între sursele incidentelor de securitate, aşa cum este sugerat în Figura 1.1 i), acestea provenind în esenţă din trei direcţii distincte: externe, interne, accidentale; în perioada 2001-2003 pe lângă faptul că numărul de atacuri creşte semnificativ şi acest echilibru este pierdut, marea parte a problemelor de securitate începând să fie datorate factorilor externi, lucru sugerat în Figura 1.1 ii) [19] .

Explicaţia acestui fapt vine exact din deschiderea naturală a sistemelor către utilizarea reţelelor publice şi utilizarea componentelor standard impuse de piaţă. Deschiderea sistemului (adversarul are acces la sistem) este datorată faptului că nu mai există perimetre securizate în sensul de izolare informatică. Chiar deconectarea de la Internet nu garantează acest lucru deoarece un USB drive infectat odată introdus poate infecta sistemul izolat (aceasta pare să fi fost calea de intrare a viermelui Stuxnet în centralele de prelucrare a uraniului din Iran).



i) ii)

FIGURA 1.1. CAUZELE INCIDENTELOR DE SECURITATE I) ÎN PERIOADA 1982-2000 II) ÎN PERIOADA 2001-2003



i) ii)

FIGURA 1.2. CĂDEREA DE ENERGIE ELECTRICĂ DIN AUGUST, 2003: I) ÎNAINTE II) DUPĂ (COPYRIGHT: POZELE APARŢIN DOMENIULUI PUBLIC ŞI SUNT FĂCUTE DE U.S. NATIONAL OCEANIC AND ATHMOSPHERIC ADMINISTRATION)

Dispariţia securităţii prin obscuritate este datorată standardelor (adversarul cunoaşte sistemul). Securitatea prin obscuritate nu este o soluţie deoarece ea expune sistemul în faţa oamenilor din interior care pot fi uşor de corupt sau cumpărat (în cele din urma oamenii din interior sunt angajaţi cu salarii modeste dar care cunosc detalii critice).

Blackout-ul din August 2003 (cădere de energie electrică pe coasta de Est a continentului american) cu toate că nu este rezultatul direct al unui atac informatic este un exemplu clar al pagubelor pe care un astfel de incident poate să le aducă: o arie de 24.000 km^2 a fost afectată, 265 de centrale electrice au fost afectate din care 22 erau nucleare, 50 de milioane de oameni au fost afectaţi din care peste 21 milioane din New York. Figura 1.2 surprinde imaginea din satelit asupra acestui incident. Pierderile totale sunt estimate la 6 miliarde de dolari. Una dintre cauzele căderii a fost o problemă în sistemele informatice care nu au declanşat semnalul de alarmă făcând astfel ca luarea de măsuri pentru a preveni căderea să întârzie până când a fost prea târziu. Desigur, este doar o problemă de timp până când adversari, precum mişcările extremiste de exemplu, ar putea exploata vulnerabilităţi de securitate criptografică pentru a cauza pagube similare. Acest lucru este recunoscut în mod curent în special în Statele Unite în sectoarele de distribuţie a gazului şi electricităţii aşa cum arată lucrări de ultimă oră.

1.3 CADRUL DE LUCRU ŞI ECUAŢIA DE BAZĂ (VULNERABILITATE + ADVERSAR = RISC DE SECURITATE)

Existenţa unei vulnerabilităţi şi a unui potenţial adversar implică existenţa unui **risc de securitate**, aceasta este o lege nescrisă a securităţii informaţiei pe care datorită importanţei o vom scrie ca ecuaţie:

Vulnerabilitate + Adversar = Risc de Securitate

Cadrul general de lucru este ilustrat în Figura 1.3. Entităţile participante la comunicare, numite simplu participanţi, în general notate cu A şi B, schimbă succesiv rolul de emiţător şi receptor. Se distinge pe lângă acestea prezenţa unui adversar, care obturează canale nesigure de comunicare (prin acestea înţelegând canale publice care pot fi accesate de către adversar), şi prezenţa unei părţi de încredere, aceasta fiind utilă în special în scenariile cu chei secrete unde rolul părţii de încredere este de a distribui chei de sesiune între participanţi. De asemenea ne raportăm la un cadru de timp. Este posibilă şi prezenţa unor canale sigure de comunicare, prin acestea înţelegând în general canale cu securitate

fizică dar pentru a nu încărca figura le vom omite. În principiu orice element al imaginii poate să lipsească, mai puţin un participant care este întotdeauna necesar şi adversarul în faţa căruia dorim să asigurăm diverse obiective de securitate.

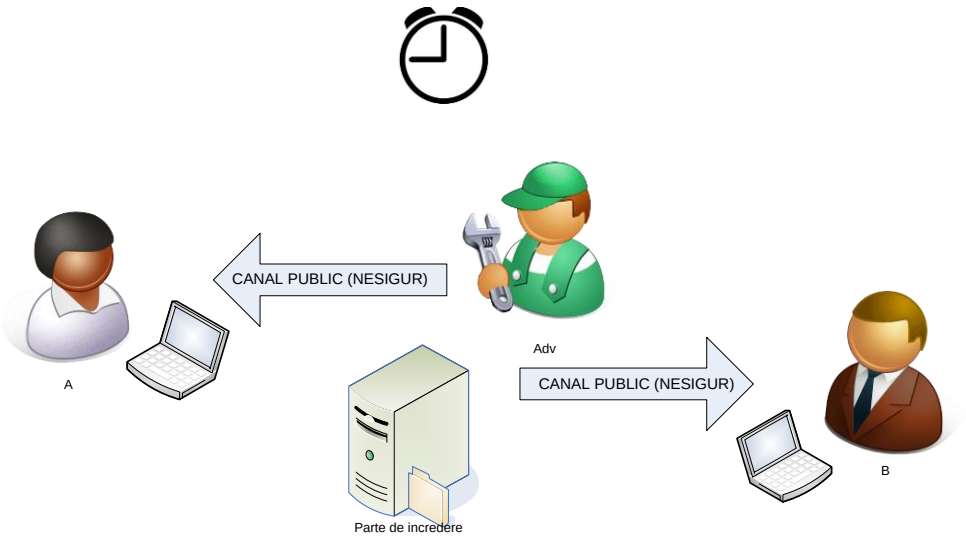


FIGURA 1.3. CADRUL DE LUCRU: CLIENŢI ONEŞTI (A,B), ADVERSAR (ADV), PARTE DE ÎNCREDERE ŞI TIMP.

1.4 ADVERSARI (CU CINE NE CONFRUNTĂM)

Entitatea care comite un atac o vom numi în limbaj de securitate **adversar** (mai rar se foloseşte termenul de atacator, şi mai rar cel de intrus). Lista potenţialilor adversari poate fi un punct de pornire în evaluarea riscurilor. Cu privire la aceştia interesează câteva coordonate cum ar fi: resursele de calcul, resursele financiare, resursele intelectuale, motivaţia, amploarea şi natura pagubelor care le pot cauza. Fără a face o enumerare exhaustivă a acestora şi fără

a insista pe detaliile cu privire la coordonatele anterior amintite, conform [72] aceasta ar fi o potenţială listă:

i) Hackerii dispun de resurse de calcul şi financiare scăzute şi atacă sisteme în general doar motivaţi de dorinţa de a brava sau pentru amuzament.

ii) Clienţii unei reţele au putere de calcul limitată şi sunt motivaţi de interese economice, de exemplu: fraudarea valorii facturate în reţeaua termo-electrică etc.

iii) Comercianţii dispun de putere de calcul modestă şi de resurse financiare apreciabile având interes în aflarea sau manipularea secretelor pentru câştigarea avantajelor financiare.

iv) Crima organizată are putere de calcul modestă şi putere financiară ridicată având interes în alterarea parametrilor la care funcţionează sistemele pentru câştigarea unor avantaje financiare.

v) Teroriştii dispun de putere ridicată de calcul şi financiară fiind motivaţi de raţiuni politico-religioase cu scopul de a răspândii panică şi pagube de ordin financiar.

vi) Guvernele statelor adversare, dispun de putere de calcul şi financiară ridicată, şi mai mult, de operatori pricepuţi şi antrenaţi cu experienţă în domeniu. Scopul acestora este în general de a afecta infrastructurile în atacuri complexe de natură fizică sau electronică.

vii) Oamenii din sistem sunt persoane care deţin informaţii de detaliu şi au acces la elementele cheie în funcţionare, sunt motivaţi în general de interese sau nemulţumiri de ordin salarial/financiar.

viii) O combinaţie a variantelor de mai sus este cel mai periculos tip de adversar; în practică este posibilă orice combinaţie şi aceasta are cele mai mari şanse de succes într-un atac.

1.5 OBIECTIVE DE SECURITATE (ŢINTE ALE ADVERSARULUI)

În ceea ce priveşte obiectivele de securitate care trebuie asigurate în sistem acestea variază într-un diapazon destul de larg. În ceea ce priveşte criptografia în general, şi nu securitatea în ansamblu, putem distinge patru obiective majore de securitate care sunt recunoscute de orice autor în domeniu: confidenţialitate, integritate, autenticitate şi non-repudiere.

Confidenţialitatea înseamnă asigurarea faptului că informaţia rămâne accesibilă doar părţilor autorizate în acest sens. Acesta este cel mai vechi obiectiv al criptologiei. În rândul necunoscătorilor este încă larg răspândită opinia că noţiunea de criptografie este sinonimă cu cea de confidenţialitate. Sigur opinia

este eronată pentru că criptografia se ocupă şi de asigurarea obiectivelor ce sunt enumerate în continuare şi care nu au nici o legătură cu păstrarea secretă a informaţiei. În ceea ce priveşte asigurarea acestui obiectiv prin tehnici criptografice, el este îndeplinit cu ajutorul funcţiilor de criptare. În general datorită eficienţei se folosesc funcţii simetrice (cu cheie secretă), dar scenarii practice conduc în general la orchestrarea acestora cu funcţii asimetrice (cu cheie publică).

Integritatea se referă la asigurarea faptului că informaţia nu a fost alterată pe parcursul transmisiei sau de către un posibil adversar. Funcţiile criptografice utilizate în acest scop sunt funcţiile hash care fac ca modificarea unui singur bit de informaţie să poată fi detectată. Menţionăm că în principiu este greşită utilizarea funcţiilor de criptare simetrice şi asimetrice în acest scop, instrumentul criptografic dedicat fiind funcţiile hash (şi codurile MAC sau semnăturile digitale în contextul mai larg al autentificării). Desigur, există şi funcţii simetrice şi asimetrice de criptare care pot asigura şi integritatea dar alegerea lor pentru acest scop trebuie făcută cu precauţie.

Autentificarea are două coordonate distincte: autentificarea entităţilor şi autentificarea informaţiei. Autentificarea entităţilor se referă la existenţa unei garanţii cu privire la identitatea unei anume entităţi. Autentificarea informaţiei se referă la determinarea sursei de provenienţă a informaţiei – în mod implicit aceasta garantează şi integritatea informaţiei deoarece dacă informaţia nu mai are integritate, deci un potenţial adversar a alterat-o, atunci nici sursa ei de provenienţă nu mai este aceeaşi. Însă doar integritatea informaţiei nu implică şi autenticitatea ei deoarece nu garantează identitatea sursei de provenienţă. Autentificarea se realizează în general prin protocoale care pot avea la bază întreg arsenalul de funcţii criptografice: funcţii hash, MAC, criptări simetrice şi asimetrice, semnături digitale. Autentificarea include de cele mai multe ori şi un factor temporal care implică o garanţie cu privire la momentul de timp la care entitatea de care este legată a depozitat-o (deoarece în absenţa unei garanţii temporale, informaţia putea fi replicată şi depusă de orice altă entitate pierzându-se astfel o valenţă a autentificării).

Non-repudierea (sau nerepudierea) previne o entitate în a nega o acţiune întreprinsă (acţiune materializată desigur în transmisia unei informaţii). Aceasta înseamnă că dacă la un moment dat o entitate neagă ca ar fi emis o anume informaţie, entitatea care a primit informaţia respectivă poate demonstra unei părţi neutre că informaţia provine într-adevăr de la entitatea în cauză. Non-repudierea se realizează prin utilizarea semnăturilor digitale.

Este util de spus, în special din considerente istorice în domeniu, că acum mai bine de 20 de ani în securitatea informaţiei trei obiective au fost considerate

ca fiind fundamentale, acestea formează aşa numita triadă CIA: confidenţialitate, integritate, disponibilitate („CIA" adică Confidentiality, Integrity, Availability). Recent însă, în 2002, Donn Parker a propus extinderea acestora la şase obiective care poartă numele de Hexada Parkeriană: Confidenţialitate, Posesie sau Control, Integritate, Autenticitate, Disponibilitate, Utilitate (Confidentiality, Possession or Control, Integrity, Authenticity, Availability, and Utility). De asemenea, la fel de bine cunoscută şi vehiculată este tetrada PAIN care provine de la Confidenţialitate, Disponibilitate-Autenticitate, Integritate, Non-repudiere (Privacy, Availability-Authentication, Integrity, Non-repudiation).

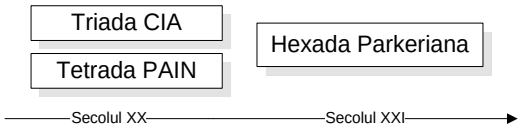


FIGURA 1.4. EVOLUŢIA OBIECTIVELOR DE SECURITATE

Desigur că aceste obiective nu sunt o sinteză completă a obiectivelor de securitate existente în practică. Există desigur şi alte obiective, ceva mai particulare dar la fel de relevante, care trebuie să le amintim. Rămâne o problemă deschisă dacă obiectivele următoare pot sau nu să fie derivate din cele amintite anterior, util pentru această lucrare este să le enumerăm şi să le explicăm succint în cele ce urmează.

Actualitatea informaţiei se referă la asigurarea faptului că informaţia primită este actuală (proaspătă). Acest aspect poate fi interpretat în două moduri: pe de o parte se referă la faptul că informaţia poate expira după o anumită perioadă de timp, pe de altă parte se referă la faptul că un posibil adversar ar putea schimba ordinea în care pachetele cu informaţie ajung la destinaţie (diverse scenarii pot fi imaginate). Se realizează în general prin utilizarea parametrilor varianţi în timp: amprente temporale (time stamps), numere aleatoare (nonce), contoare (counter), etc.

Anonimitatea se referă la împiedicarea identificării identităţii unei entităţi care a solicitat un serviciu. Spre exemplu, poate fi extrem de util în tranzacţii bancare când nu se doreşte identificare persoanei care sau către care se face o plată, sau în servicii de e-mail pentru păstrarea anonimităţii expeditorului, etc. Se realizează fie prin protocoale, fie prin funcţii criptografice adaptate acestui scop. De exemplu există un puternic segment de cercetare în zona funcţiilor de criptare cu renegare (deniable encryption), prin care se poate cripta informaţie al cărei conţinut poate fi schimbat la decriptare, făcând astfel renegabilă orice informaţie criptată (nu există pentru aceasta soluţii eficiente până în prezent).

Autorizarea se referă la controlul accesului şi la prevenirea intrării agenţilor neautorizaţi în sistem. Relaţia între obiectivul autentificării entităţilor şi controlul accesului constă în aceea că cel din urmă obiectiv se construieşte în general pe primul (e normal să fie necesară o metodă de a autentifica entitatea înainte de a-i permite accesul) dar obiectivele sunt totuşi distincte. Aceasta deoarece autorizarea înseamnă utilizarea unui mecanism de autentificare şi a unei politici de securitate pentru a decide dreptul de acces al unor entităţi asupra unor resurse.

Disponibilitatea se referă la asigurarea faptului că un serviciu este accesibil atunci când un utilizator legitim îl solicită. Asigurarea acestui obiectiv presupune că o entitate neautorizată nu poate bloca accesul unei entităţi autorizate la serviciile sistemului. În acest caz însă nu intră în discuţie problemele legate de autorizarea accesului, anterior amintite, ci cele de disponibilitate a resursei în sine. Aceasta presupune a evita problemele de epuizare a resurselor sistemului din cauza utilizării nelegitime a acestora. Atacurile asupra acestui obiectiv sunt cele de tip Denial of Services (DoS) şi cauzează atât pagube economice dar şi de siguranţă în funcţionare.

Protecţia părţilor terţe se referă la evitarea transmiterii pericolului asupra părţilor cu care există o legătură. De exemplu atacul asupra unei anume componente IT nu va defecta şi altă componentă, sau din punct de vedere economic: căderea unei componente datorită unei erori de manipulare nu va duce la discreditarea producătorului, etc.

Revocarea se referă la posibilitatea de a revoca un drept oferit. Poate cel mai relevant exemplu în legătură cu criptografia este posibilitatea de a revoca un certificat de cheie publică de către entitatea care l-a emis.

Trasabilitatea sau urmărirea unui sistem se referă la posibilitatea de a reconstrui istoricul funcţionării sistemului pe baza înregistrărilor, de exemplu înregistrarea comenzilor relevante, a persoanelor care le-au lansat etc. Obiectivul este relevant în determinarea cauzelor eventualelor problemelor de funcţionare, deci este utilizat în diagnosticare.

Nici această listă de obiective nu este completă, aproape fiecare carte în domeniu prezintă liste mai mult sau mai puţin complete, iar prezentarea lor exhaustivă poate fi în sine subiectul unei cărţi. Totuşi aceste obiective pot crea o imagine parţială asupra a ceea ce trebuie protejat în sisteme informatice şi asupra modului în care criptografia contribuie la aceasta.

1.6 VULNERABILITĂȚI (CE EXPLOATEAZĂ ADVERSARUL)

Vulnerabilitatea este un punct slab al unui sistem care poate fi exploatat de un potenţial adversar. Vis-a-vis de vulnerabilităţi, trebuie reţinută regula comun enunţată în cărţile de securitate că un sistem nu este mai sigur decât cea mai vulnerabilă componentă a sa.

Vulnerabilităţile unui sistem informatic pot fi clasificate conform [31] în: vulnerabilităţi de proiectare (de exemplu proiectarea greşită a unui protocol de comunicare), vulnerabilităţi de implementare (de exemplu overflow), vulnerabilităţi fundamentale (de exemplu alegerea unor parole slabe). În [62] sunt evidenţiate câteva vulnerabilităţi ce duc frecvent la căderea protocoalelor de securitate: vulnerabilitatea unei primitive criptografice ce poate fi amplificată de protocolul de securitate (este important de subliniat că multe funcţii criptografice au vulnerabilităţi ascunse, iar pe de altă parte chiar şi folosirea celei mai solide funcţii criptografice în mod necorespunzător poate duce la pierderea totală a securităţii), asumarea unor garanţii de securitate care sunt supra-specificate sau prost înţelese (poate cea mai comună astfel de eroare este utilizarea unei funcţii de criptare şi prezumţia faptului că aceasta va asigura şi faptul că datele nu vor fi alterate), neatenţia în aplicarea unor principii generale aplicabile unei clase mai largi de primitive (de exemplu criptarea folosind un mecanism cu cheie publică a unei informaţii de entropie scăzută care poate fi uşor aflată cu un atac de tip forward-search).

Dacă luăm ca studiu de caz sistemele industriale, sunt relevant exemplele din lucrarea [72] unde între deficienţele de securitate în sisteme SCADA se enumeră următoarele: lipsa definiţiilor formale, lipsa unui buget de securitate incremental, absenţa expertizei necesare în domeniul securităţii, incapacitatea de a instala uşor tehnologia necesară, necesitatea educaţiei cu privire la bunele practici în vederea asigurării securităţii. Sigur toate aceste deficienţe nu sunt uşor de acoperit şi există şi recomandări concrete în vederea asigurării securităţii. De exemplu lucrarea [88] indică 21 de paşi pentru a asigura securitatea sistemelor SCADA, între aceştia mulţi au la bază utilizarea unor tehnici criptografice: de exemplu pasul 5 se referă la eliminarea protocoalelor personalizate (custom) a căror securitate se bazează pe obscuritatea protocolului (singura soluţie care nu se bazează pe obscuritatea protocolului este criptografia) iar pasul 21 se referă la introducerea unor politici de securitate şi cursuri pentru ca personalul să fie conştient de protejarea informaţiei senzitive (inclusiv parolele care sunt un ingredient criptografic). Nu în ultimul rând există şi produse program destinate asigurării securităţii în sisteme SCADA precum IntruShield de la McAfee. În topul primelor 10 deficienţe de securitate ale reţelelor SCADA lucrarea (McAfee, 2007) enumeră: politici de securitate inadecvate, sisteme slabe de control ale reţelelor,

proasta configurare a sistemelor, comunicare wireless neadecvată, autentificare neadecvată la comunicare, lipsa mecanismelor de detecţie şi restricţionare a drepturilor de acces la sisteme de control, absenţa uneltelor pentru detecţia şi raportarea activităţilor anormale, utilizarea reţelei pentru trafic neautorizat, lipsa mecanismelor de detecţie a erorilor ce pot conduce la epuizarea bufferelor, lipsa mecanismelor de control a schimbului de software. Multe dintre aceste ameninţări au ca soluţie utilizarea criptografiei, de exemplu ameninţările legate de autenticitatea comenzilor şi de controlul accesului. Iar potenţialele atacuri încep să se diversifice de la o aplicaţie la alta conducând spre o paletă relativ largă de obiective de securitate ce trebuie asigurate.

1.7 ATACURI ASUPRA CANALULUI DE COMUNICARE (CAPABILITĂŢI ALE ADVERSARULUI)

Un **atac** este o agresiune asupra unui obiectiv de securitate. Există o gamă foarte largă de atacuri şi diverse clasificări ale acestora în funcţie de diverse criterii. De exemplu în [31] acestea sunt clasificate în: **atacuri nedirecţionate** ce constau în infiltrări în sistem în scopul exploatării oricărei vulnerabilităţi găsite şi **atacuri direcţionate** ce au un scop precis care constă în exploatarea unei anumite părţi a sistemului (de exemplu extragerea unei anumite informaţii, furtul unei parole etc.). O clasificare ceva mai clasică a atacurilor, preferată de majoritatea cărţilor de securitate cum ar fi [80] este în: atacuri pasive şi atacuri active.

Atacurile pasive sunt **citirea mesajelor** şi **analiza de trafic**. Diferenţa între cele două atacuri este că la analiza de trafic nu este necesară citirea efectivă a mesajelor transmise ci doar observarea unor modele în comunicare, de exemplu ce natură are informaţia trimisă, e vorba de un download pentru un film, o convorbire telefonica prin Internet, etc. Este important de remarcat că aceste atacuri sunt violări ale obiectivului numit confidenţialitate.

Atacurile active sunt cele de: **modificarea** informaţiei vehiculate între două entităţi care presupune alterarea detectabilă sau nu a informaţiei, **impostura (impersonarea sau mascaradarea)** care înseamnă a pretinde o altă identitate decât cea reală, **retransmisie** care înseamnă a transmite o informaţie care a fost transmisă anterior de un participant şi **întreruperea legăturii** care înseamnă efectiv tăierea canalului de comunicare (adică Denial Of Services - DoS). Este important de remarcat, din punct de vedere al obiectivelor de securitate, că primele trei atacuri sunt o agresiune a obiectivului de autenticitate iar ultimul o agresiune a disponibilităţii.

Toate aceste atacuri se mai numesc şi **atacuri asupra canalului de comunicare** şi pentru combaterea lor se folosesc cu succes în practică tehnici criptografice. Figura 1.5 şi Figura 1.6 prezintă aceste atacuri.

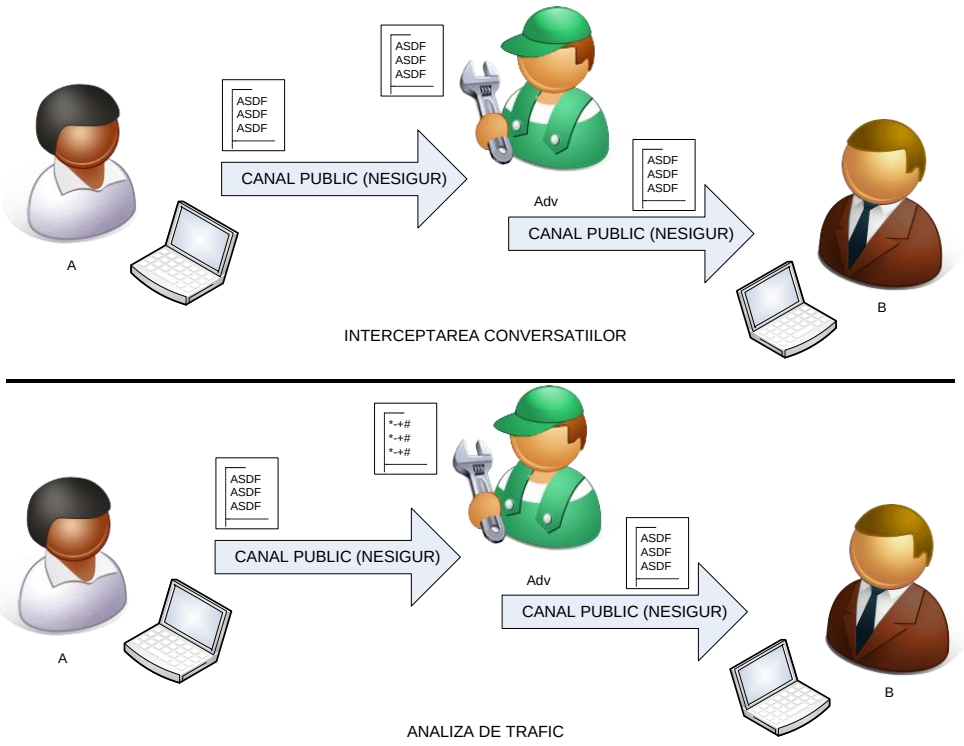


FIGURA 1.5. ATACURI PASIVE ASUPRA CANALULUI DE COMUNICARE

Desigur, există şi alte atacuri asupra securităţii unui sistem pentru a căror contracarare nu se folosesc în general tehnici criptografice, ele fiind contracarate prin alte mijloace. Exemple de cauze ale unor astfel de atacuri sunt: i) viruşi informatici care duc în general la distrugerea software-ului prin infiltrare în fişiere existente şi fac imposibilă funcţionarea sistemului prin execuţia unor acţiuni dăunătoare, ii) programe tip „Cal Troian" sunt programe folosite în scopul obţinerii accesului la anumite informaţii, de exemplu aflarea parolelor în sisteme de operare perimate gen Win98, iii) programe tip „Vierme" care sunt programe care se propagă automat fără controlul unui utilizator şi în general afectează rata de transfer a reţelei (spre deosebire de viruşi nu se infiltrează în fişiere existente).

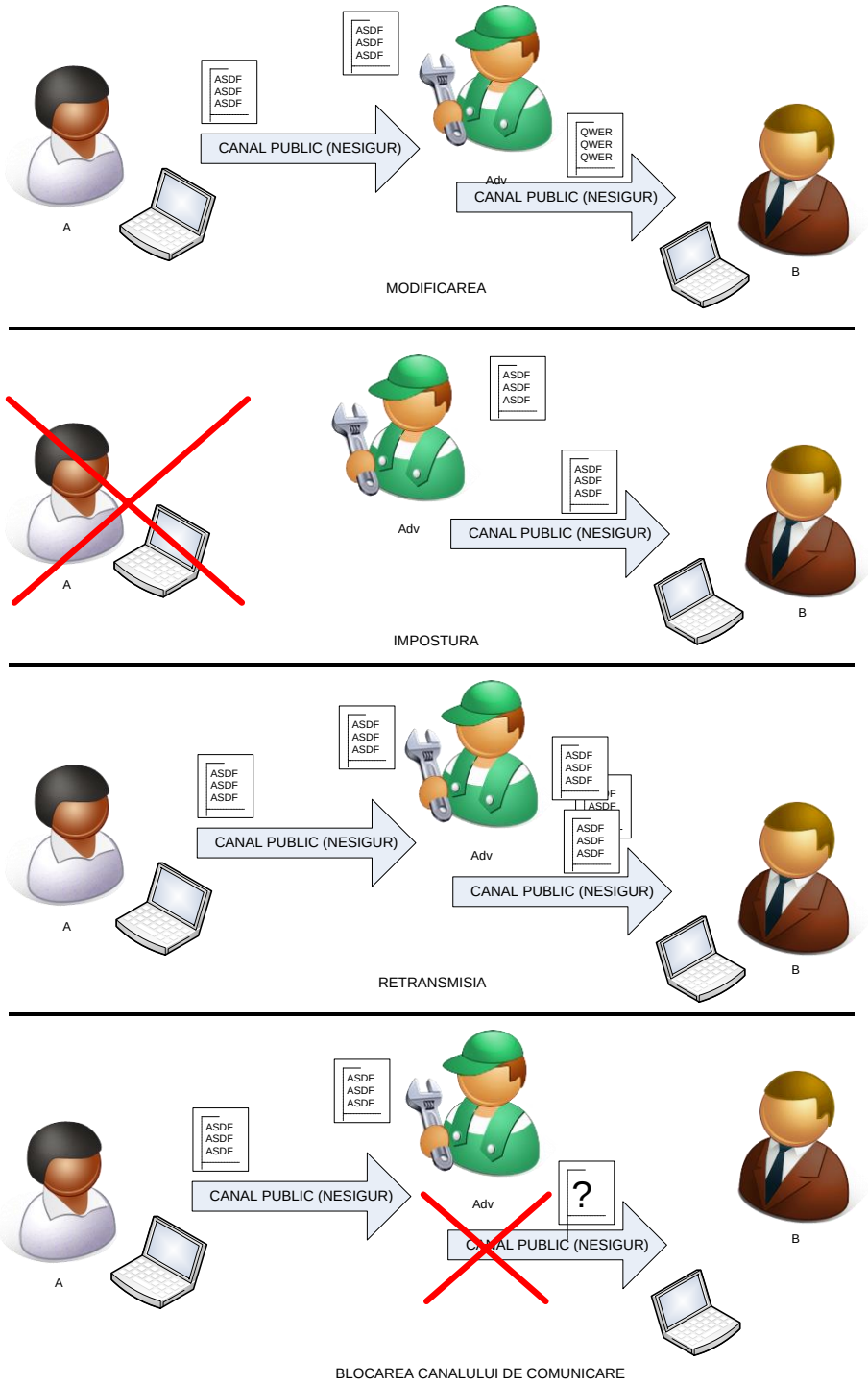


FIGURA 1.6. ATACURI ACTIVE ASUPRA CANALULUI DE COMUNICARE

Trebuie însă menţionat că şi împotriva acestor atacuri, aparent imune la măsuri criptografice, încep să fie utilizate tehnici criptografice. Un bun exemplu de soluţie de ultimă oră în care funcţii criptografice sunt utilizate împotriva viruşilor este utilizarea funcţiilor criptografice hash pentru a construi o amprentă a sistemului de operare (sau individual pentru fiecare fişier) şi verificare la bootarea de pe un dispozitiv extern dacă imaginea curentă a sistemului coincide cu imaginea stocată în scopul detectării unor potenţiale modificări făcute de viruşi.

1.8 CRIPTOGRAFIA ŞI ROLUL EI

Concluzia imediată a paragrafului anterior este aceea că riscurile de securitate există ca şi consecinţă a unor vulnerabilităţi în sistem şi a unor potenţiali adversari – acesta fiind un aspect natural şi inevitabil în practică. Riscurile de securitate, ca orice alte riscuri de altfel, trebuie acoperite cu garanţii de securitate. Atunci când obiectul manipulat este informaţia singura, criptografia este una din puţinele garanţii demonstrabile. Deci rolul acesteia este de a oferi garanţii în faţa riscurilor de securitate ale informaţiei.

Criptografia este comun utilizată într-o gamă largă de aplicaţii din zona: instituţiilor medicale, publice, private, bancare şi chiar în cele mai comune aplicaţii pe care le folosim zi de zi: telefonie mobilă, servicii de e-mail, editoare de documente, etc.

Conform lucrării de referinţă în domeniu [62] ***criptografia*** *este definită ca fiind studiul tehnicilor matematice referitoare la aspecte de securitatea informaţiei precum confidenţialitate, integritate, autentificarea entităţilor, autentificarea provenienţei datelor*.

Totuşi o astfel de definiţie nu este completă. Pe de o parte deoarece criptografia nu este în totalitate matematică (chiar dacă marea ei parte este), de exemplu criptarea cuantică face mai mult apel la cunoştinţe de fizică decât de matematică sau implementarea criptografiei ţine mai mult de ştiinţa calculatoarelor decât de matematică. Pe de altă parte pentru că nu ţine cont de fondul problemei. Ron Rivest a făcut o remarcă pe cât de simplă pe atât de profundă în ceea ce priveşte criptografia şi această remarcă poate fi considerată o excelentă definiţie a criptografiei: *criptografia înseamnă comunicare în prezenţa adversarilor*. Orice comentariu la adresa remarcii lui Rivest este superfluu.

Din păcate definiţia din dicţionarul explicativ al limbii române, DEX ediţia a II-a 1998, nu poate fi utilizată pentru clarificare deoarece este complet depăşită,

criptografia fiind definită ca: *scriere secretă cu ajutorul unui cod de semne convenţionale*. Această descriere corespunzând stadiului domeniului de acum câteva secole.

Domeniul criptografiei se ocupă de construcţia funcţiilor criptografice. Diverse funcţii criptografice vor fi descrise în capitolele următoare, iar pe moment putem să ne formăm o imagine intuitivă asupra unei funcţii criptografice ca fiind o funcţie care depinde de un parametru numit **cheie** şi se aplică unui **mesaj** („plaintext") pentru a obţine un mesaj criptat numit **criptotext** („ciphertext")– aceasta este ceea ce în principiu numim funcţie de criptare. Totodată obiectivul criptografiei este şi cel de a construi inversa acestei funcţii cu ajutorul căreia din criptotext alături de cheie se poate recupera mesajul original – aceasta fiind ceea ce numim funcţie de decriptare. Toate acestea sunt sugerate în Figura 1.7. Subliniem că această imagine este doar intuitivă deoarece nu toate funcţiile criptografice au cheie, şi mai mult, nu toate funcţiile criptografice admit o inversă.

Domeniul care are ca obiectiv recuperarea din criptotext a mesajului şi a cheii se numeşte **criptanaliză** şi este ceea ce în limbaj comun numim spargerea funcţiilor criptografice care din punct de vedere intuitiv poate fi văzută ca inversarea acestora. Subliniem însă că în general a sparge o funcţie criptografică presupune acţiuni mult mai simple decât inversarea ei, imaginea creată având din nou doar valoare intuitivă. Criptografia şi criptanaliza sunt cele două ramuri ale domeniului numit **criptologie**.

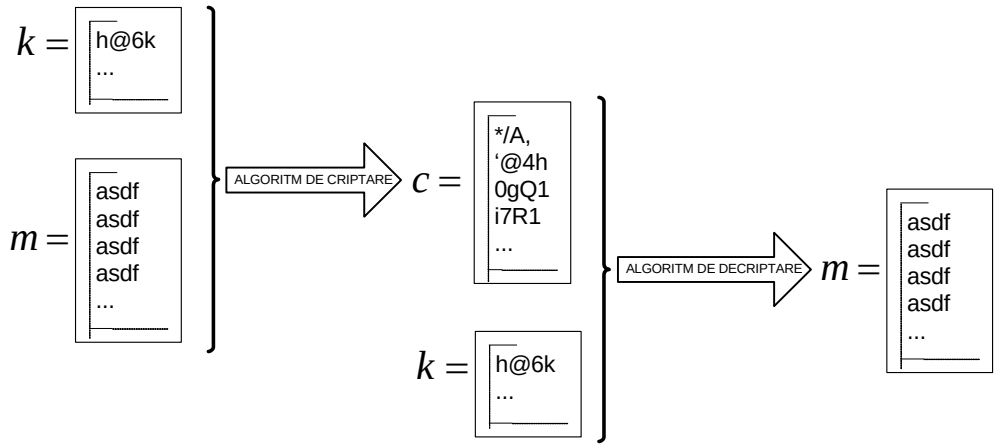


FIGURA 1.7. IMAGINE INTUITIVĂ CU PRIVIRE LA CE ÎNSEAMNĂ CRIPTAREA ŞI DECRIPTAREA – OBIECTIVE CONSTRUCTIVE ALE CRIPTOGRAFIEI.

Evoluţia criptografiei, şi mai mult sau mai puţin a oricărui alt domeniu, poate fi văzută ca urmând următorul drum: Teorie, Practică, Standarde. Fără a

reduce rolul aplicaţiilor practice şi a standardelor care ne fac viaţa mai uşoară este de remarcat că standardele acoperă doar probleme care sunt implementate practic, implementările practice urmează doar probleme fundamentate teoretic în prealabil iar teoria reprezintă baza cea mai solidă şi consistentă a domeniului. Mai mult, soluţiile de vârf ale domeniului se găsesc în teorie şi doar rar în implementările practice, care tratează în general aspecte demult cunoscute (de exemplu curbele eliptice sunt utilizate în practică doar în ultimii ani, în ciuda faptului că au fost propuse de Miller şi Kobliz încă din 1985).

1.9 GARANŢIA OFERITĂ DE CRIPTOGRAFIE: NIVELUL DE SECURITATE

Între întrebările din primul subcapitol enunţam şi o întrebare cu privire la **nivelul de securitate** (uneori numit **nivel de încredere**). Nivelul de securitate cuantifică efortul computaţional necesar spargerii unei funcţii criptografice. Aşa cum remarcă şi autorii din [31] nivelul de securitate al unei primitive criptografice este o funcţie care scade odată cu timpul, aceasta datorându-se atât creşterii nivelului tehnologic cât şi descoperiri unor noi atacuri asupra funcţiei. De exemplu dacă o informaţie criptată folosind algoritmul DES (standard în criptarea simetrică începând din 1976) putea oferi un nivel ridicat de încredere în urmă cu 30 de ani, astăzi o astfel de informaţie poate fi uşor recuperată prin spargerea codului în doar câteva zile pe o maşină de calcul actuală.

Cerinţele curente sunt în general definite prin intermediul standardelor. În particular, pentru cazul tehnicilor criptografice, care sunt frecvent folosite pentru asigurarea obiectivelor de securitate anterior amintite, există standarde pentru marea parte a funcţiilor criptografice şi a dimensiunii cheilor pentru acestea. De exemplu Guvernul USA prin documentaţiile Federal Information Processing Standards (FIPS) impune standarde pentru funcţii hash, criptare simetrică, semnătură digitală; totuşi aceste standarde nu acoperă nici pe departe toate mecanismele. În Tabelul 1.1 sunt prezentate dimensiunile de chei recomandate pentru câţiva algoritmi frecvent utilizaţi în practică. În Tabelul 1.2 sunt prezentate recompensele oferite de RSA-Security pentru spargerea unor chei de RSA. Prin comparaţie între cele două tabele se poate observa că o cheie de RSA de 1024 biţi este recomandată de primul tabel pentru folosire până în 2010, în timp ce RSA-Security oferă o recompensă de 100.000 dolari pentru spargerea unei astfel de chei (în realitate costurile de spargere ale unei astfel de chei fiind de sute sau mii de ori mai mari). Pentru a forma o imagine mai clară asupra dimensiunii acestor numere, iată valoarea numărului RSA-640:

n=3107418240490043721350750035888567930037346022842727545
720161948823206440518081504556346829671723286782437916272838033
415471073108501919548529007337724822783525742386454014691736602
477652346609.

Cei doi factori primi ai săi sunt:

p=1634733645809253848443133883865090859841783670033092312
181110852389333100104508151212118167511579,

q=1900871281664822113126851573935413975471896789968515493
666638539088027103802104498957191261465571.

Nivelul de securitate al funcţiilor criptografice este frecvent clasificat după următoarele categorii: **securitate necondiţionată**, **securitate bazată pe complexitate**, **securitate demonstrată**. Această clasificare nu se referă la o cuantificare numerică a nivelului de securitate ci la garanţia pe care acesta se bazează. Încercăm să explicăm aceste trei noţiuni. Securitate necondiţionată au acele criptosisteme care nu pot fi sparte, indiferent de puterea de calcul de care dispune adversarul (one-time pad este un astfel de sistem). Discutăm de securitate bazată pe complexitate atunci când nivelul de securitate este dat de complexitatea (timpul de calcul ilustrat în număr de paşi) al unui algoritm care poate rezolva problema şi acest algoritm este cea mai bună cale de a rezolva problema. Noţiunea de securitate demonstrabilă (provable security) se foloseşte cu privire la acele criptosisteme despre care se poate demonstra că spargerea lor conduce la rezolvarea altei probleme care se ştie că este greu de rezolvat.

Securitate la nivel de biţi	**AES (dimensiunea cheii)**	**RSA (dimensiunea modulului)**	**Curbe eliptice (valoarea ordinului bazei)**	**Durata de utilizare recomandată**
80	x	1024	160-223	Până în 2010
112	x	2048	224-255	Până în 2030
128	128	3072	256-383	După 2030
192	192	7680	384-511	x
256	256	15360	512+	x

TABELUL 1.1. NIVELE COMPARATIVE DE SECURITATE (CONFORM CU [64]).

De asemenea, în ceea ce priveşte nivelul de securitate se face uneori o diferenţiere între două nivele de securitate distincte numite: **securitate slabă** – este securitatea care răspunde unor garanţii în faţa vulnerabilităţilor ce pot fi exploatate de adversari neinteligenţi (cum ar fi erori introduse de mediu care pot fi corectate prin mecanisme redundante precum codurile de corecţie a erorilor CRC) şi **securitate puternică** care răspunde unor garanţii în faţa unor vulnerabilităţi ce pot fi exploatate de adversari inteligenţi (precum omul, sau agenţii software inteligenţi). Securitatea puternică este în mare măsură asigurată de tehnicile criptografice, dar nu exclusiv, de exemplu viruşii informatici pot fi consideraţi agenţi inteligenţi care exploatează sisteme dar criptologia nu oferă încă soluţii prea bune în acest sens.

RSA-576	**RSA-640**	**RSA-704**	**RSA-768**
10.000$ (factorizat în decembrie 2003)	20.000$ (factorizat în noiembrie 2005)	30.000$	50.000$
RSA-896	**RSA-1024**	**RSA-1536**	**RSA-2048**
75.000$	100.000$	150.000$	200.000$

TABELUL 1.2. RECOMPENSE (EXPIRATE) PENTRU SPARGEREA UNOR CHEI RSA [71].

1.10 SCURT ISTORIC AL CRIPTOGRAFIEI

În opinia autorului (deşi poate nu numai deoarece opinii apropiate transpar şi din alte lucrări), în context istoric evoluţia criptografiei poate fi sistematizată în cadrul a patru etape: i) perioada antichităţii, ii) perioada medievală, iii) perioada premergătoare primului război mondial până după cel de-al doilea război mondial şi iv) perioada criptografiei moderne cu începere după cel de-al doilea război mondial (poate mai exact în jurul anilor 70). Există o separare clară de mentalitate între cele 4 perioade aşa cum vom încerca pe scurt să creionăm în cele ce urmează.

Prima dintre ele, antichitatea, este caracterizată de sisteme criptografice lipsite de fundamente matematice dar şi de dorinţa vreunui automatism (adică o procedură de a crea un model automat, de exemplu mecanic al criptării). Din

această perioadă datează codul Cezar (o simplă permutare a literelor) şi cilindrul grecesc *skytale* pe care era înfăşurată o bandă de hârtie pe care se scria textul (decriptarea necesita desigur înfăşurare pe un cilindru similar ca diametru). Într-adevăr aceste tehnici sunt în mică măsura legate de criptografia modernă.

Cea de a doua perioadă, cea medievală, este distinctă prin utilizarea tehnicilor de substituţie polialfabetică. În acestea o literă este substituită cu orice altă literă fără repetiţie evidentă, nu doar de una ca în cazul substituţilor simple, de exemplu permutarea din cadrul codului Cezar. Primul criptosistem polialfabetic a fost descoperit de Leon Battista Alberti (1404–1472) şi se numeşte codul lui Alberti, aceasta este considerată prima mare descoperire în criptografie de după vremea lui Cezar. Cel mai elocvent exemplu este sistemul Vigenère după numele lui Blaise de Vigenère (1523–1596) deşi el a fost descris prima dată de criptograful italian Giovan Battista Bellaso (1505-?) în 1553 în timp ce Vigenère îl publica în 1586. Tot Bellaso este autorul tabelei de criptare cunoscută sub numele de tabela Porta publicată de Giambattista della Porta (1535? –1615) în 1563 fără a da credit lui Bellaso. În aceeaşi perioadă desfăşoară activităţi în criptografie şi Gerolamo (Geronimo) Cardano (1501–1576) al cărui nume este cunoscut în matematică pentru formulele de rezolvare a ecuaţiilor pătratice, cubice şi cuartice; acesta introduce un sistem de criptare numit grila lui Cardano în 1550. Mai multe detalii despre istoria criptografiei pot fi găsite în cartea lui Kahn [53].

Cea de a treia perioadă, cea a războaielor mondiale şi perioada premergătoare, este remarcabilă prin apariţia unor principii, precum legile lui Auguste Kerckhoffs (1835 – 1903) care sunt actuale chiar şi azi şi mai mult de atât a unor criptosisteme precum codul lui Gilbert Sandford Vernam (1890 – 1960) care până în ziua de azi rămâne singurul cod cu securitate necondiţionată (cunoscut sub numele de codul Vernam, şi în varianta uşor modificată ca one-time-pad).

Primele deziderate în construcţia unui algoritm criptografic au fost enunţate de către Kerckhoffs în secolul XIX. Pe lângă importanţa istorică, acestea au relevanţă chiar şi în zilele de astăzi, iar în acest context considerăm utilă amintirea lor:

i) Sistemul trebuie să fie, dacă nu teoretic imposibil de spart, atunci imposibil de spart în practică.

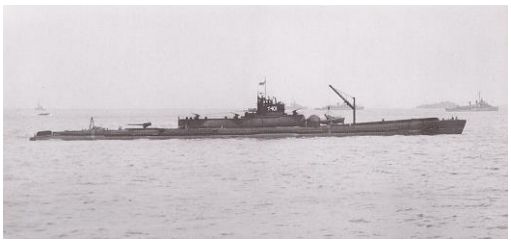
ii) Compromiterea detaliilor cu privire la sistemul criptografic nu trebuie să creeze probleme corespondenţilor.

iii) Cheia trebuie să fie uşor de memorat fără a fi notată şi să fie uşor de schimbat.

iv) Cifrul (mesajul criptat) trebuie să fie uşor de transmis prin telegraf.

v) Aparatul de criptare trebuie să fie uşor de purtat şi operabil de către o singură persoană.

vi) Sistemul trebuie să fie simplu, fără să necesite cunoaşterea unei liste lungi de reguli sau stres intelectual.



(i)

(ii)

FIGURA 1.8. COMUNICAȚII WIRELESS (I) IERI (ÎN TIMPUL CELUI DE AL DOILEA RĂZBOI MONDIAL) ŞI AZI (II) (COPYRIGHT: POZELE DE LA I) AU FOST ALESE DIN DOMENIUL PUBLIC)

În perioada premergătoare celui de-al doilea război mondial apare şi maşina Enigma, un criptosistem despre care se poate spune că a jucat un rol decisiv în unul din cele mai importante evenimente din istoria omenirii: cel de-al doilea război mondial. Această perioadă include contribuţiile fundamentale ale lui Alan Mathison Turing (1912 –1954) constructor al maşinii care a spart Enigma (a

nu se confunda cu maşina Turing, o descoperire fundamentală a acestuia care reprezintă poate prima abstractizare a unei maşini de calcul moderne). Alan Turing rămâne recunoscut ca părinte al ştiinţei calculatoarelor şi inteligenţei artificiale.

(i) (ii)

FIGURA 1.9. MAŞINI CRIPTOGRAFICE: (I) IERI (MAŞINA ENIGMA) ŞI (II) AZI (SMART-CARD)

În unele prezentări istorice se face o separare între perioada premergătoare războiului (perioada lui Kerckhoffs şi Vernam) şi perioada celui de-al doilea război mondial. Aici am decis să le grupăm deoarece descoperirile din ambele par să joace la unison în dezvoltarea criptografiei moderne. Într-adevăr însă există şi raţiuni pentru a le separa, deoarece în mod cert descoperirile din perioada celui de-al doilea război mondial au avut un impact semnificativ asupra evoluţiei omenirii şi nu doar a criptografiei. Nu este de mirare că cel de-al doilea

război mondial a dus la dezvoltarea criptografiei, deoarece rolul de bază în soarta războiului a fost jucat de submarine şi avioane, două arme care depind de comunicare wireless, comunicare uşor de făcut (fără să necesite o infrastructură pentru mediul de transmisie, nu există cablu, ci doar aer) dar care este expusă adversarilor (oricine aude ce se vorbeşte). În Figura 1.8 sugerăm evoluţia comunicaţilor wireless din mediul militar către cel al societăţii de consum. Mediul wireless este unul dintre principalele motoare de dezvoltare a securităţii, evident datorită fragilităţii mediului în faţa adversarilor.

Criptografia modernă începe în opinia multora cu Claude Elwood Shannon (1916 –2001) părinte al domeniului teoriei informaţiei şi care marchează intrarea criptografiei în era criptografiei matematice. Poate un alt început al criptografiei moderne ar putea fi văzut în codul lui Horst Feistel şi naşterea criptografiei cu cheie publică datorată lui Diffie-Hellman-Merkle. Desigur însă vorbim de un continuum, aşa că nu este uşor a trage o linie exactă acolo unde începe criptografia modernă şi poate nici nu este aşa de relevant. Relevant este că odată cu descoperirea criptografiei cu cheie publică, criptografia trece dintr-un domeniu preponderent militar, într-un domeniu academic, al universităţilor şi grupurilor de cercetare.

1.11 ISTORIA CRIPTOGRAFIEI CU CHEIE PUBLICĂ ÎN PARTICULAR

Cu certitudine istoria criptografiei cu cheie publică îşi merită propriul ei capitol. Aceasta deoarece, am spus şi subliniem, începutul ei marchează trecerea criptografiei din domeniul preponderent militar din care făcea parte anterior în domeniul public, academic.

Criptografia cu cheie publică s-a născut ca domeniu din necesitatea de a rezolva două probleme fundamentale în securitate: i) t**ransmisia de informaţie pe un canal nesigur** între doi participanţi care nu au un secret partajat prealabil şi ii) s**emnarea digitală a informaţiei** în scopul asigurării non-repudierii acesteia.

Este foarte simplu de înţeles de ce **criptarea cu cheie secretă**, numită şi **criptare simetrică**, nu poate fi folosită pentru atingerea acestor deziderate. Aşa cum a fost sugerat în Figura 1.7 cu privire la criptarea cu cheie secretă se poate observa că aceeaşi cheie, numită **cheie secretă**, este folosită pentru criptare şi decriptare. Astfel pentru ca doi participanţi să poată comunica în condiţii de securitate este strict necesar ca ei să fii schimbat în prealabil o cheie – lucru care impune existenţa anterioară a unui canal sigur de comunicare. Dar în practică de cele mai multe ori interacţiunile sunt spontane şi avem de a face cu participanţi

care nu au beneficiat în prealabil de un canal sigur de comunicare fiind astfel imposibilă existenţa unui secret partajat prealabil. **Criptarea cu cheie publică**, sau **criptarea asimetrică**, poate rezolva acest neajuns. Aceasta deoarece într-un sistem de criptare cu cheie publică criptarea şi decriptarea se fac cu chei distincte, şi astfel cheia de criptare, numită şi **cheie publică**, poate fi trimisă pe un canal nesigur deoarece intercepţia ei nu poate duce la decriptarea unui mesaj criptat cu aceasta. În acelaşi timp doar posesorul legitim al cheii de decriptare, numită şi **cheie privată,** este cel care poate recupera din criptotext mesajul original. În Figura 1.10 este sugerat mecanismul de criptare şi decriptare cu cheie asimetrică.

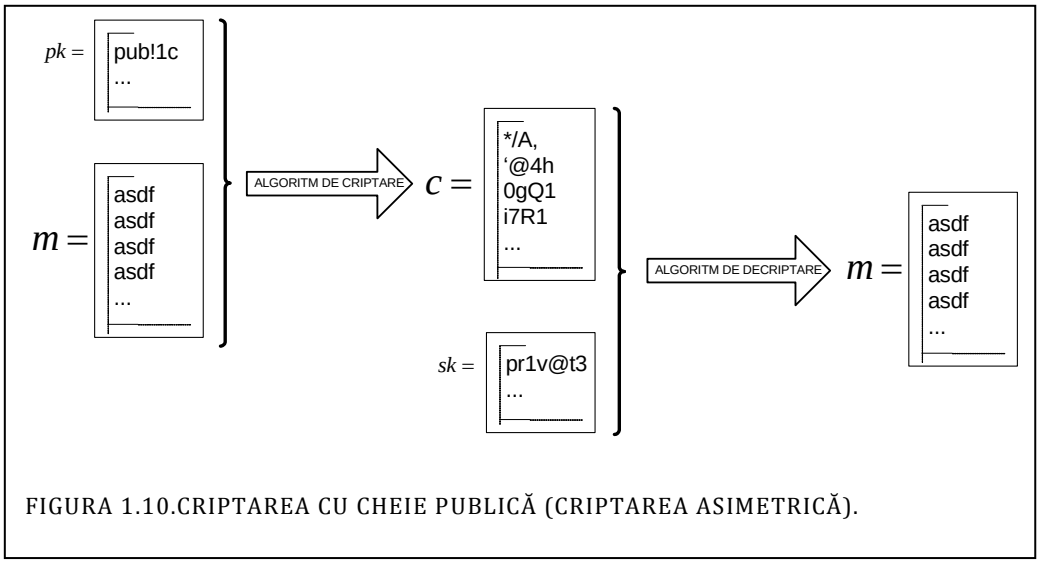


FIGURA 1.10.CRIPTAREA CU CHEIE PUBLICĂ (CRIPTAREA ASIMETRICĂ).

Conceptul de criptosistem cu cheie publică şi utilitatea lui în atingerea acestor obiective a fost publicat de Diffie şi Hellman în 1976. În fapt însă descoperirea conceptului de criptare cu cheie publică este descoperit de Ralph Merkle puţin înainte, dar aşa cum spune Hellman: „Diffie, Rivest, Shamir, Adleman şi eu am avut norocul de a primi rapid recenzii ale lucrărilor noastre, şi astfel au apărut înaintea contribuţiilor fundamentale ale lui Merkle” (în prefaţa de la cartea lui Yan [87]). Diffie şi Hellman însă eşuează în a propune o soluţie pentru semnătura digitală şi singura lor propunere este schimbul de cheie Diffie-Hellman care poate fi utilizat în scopul criptării asimetrice pentru rezolvarea criptării cu cheie publică. Doi ani mai târziu, în 1978, Rivest, Shamir şi Adleman descoperă criptosistemul denumit RSA, după numele autorilor, care poate fi utilizat pentru rezolvarea ambelor probleme: criptare asimetrică şi semnătură digitală. În 1979 Rabin publică un criptosistem similar cu RSA care diferă doar la valoarea

exponentului utilizat şi care are securitatea echivalentă problemei factorizării întregilor (până în prezent o astfel de echivalenţă nu se poate demonstra în ceea ce priveşte RSA-ul). Doar în 1983 ElGamal descoperă că inclusiv logaritmii discreţi pe a căror dificultate se baza securitatea schimbului de cheie propus de Diffie şi Hellman pot fi utilizaţi pentru construcţia unei scheme de semnături digitale. În 1985 Miller şi Koblitz propun utilizarea curbelor eliptice în criptosistemele lui Diffie-Hellman şi ElGamal, practic problema logaritmului discret peste grupuri de întregi este transpusă în problema logaritmului discret peste curbe eliptice – principiul criptosistemelor rămânând însă acelaşi. În 2002, Hellman a propus ca schimbul de cheie descoperit în 1976 să fie numit Diffie-Hellman-Merkle (poate că inventatorul criptografiei cu cheie publică este totuşi Ralph Merkle). Acestea ar fi cele mai semnificative momente din istoria veche a criptografiei cu cheie publică.

Relativ recent, în 1997, a fost făcută publică informaţia că serviciile secrete britanice erau în posesia sistemului RSA cu 5 ani înainte ca el să fie descoperit şi publicat de Rivest et al., în 1973, autorul fiind Clifford Cocks angajat al UK Intelligence Agency. Acelaşi lucru este valabil şi cu privire la schimbul de cheie Diffie-Hellman, descoperit în 1974 de serviciile secrete britanice şi anume de către Malcolm J. Williamson. Acestea sunt însă doar episoade din istoria ascunsă a criptografiei, istorie care este lipsită de o ţinută ştiinţifică şi care nu poate fi decât condamnată având în vedere faptul că în esenţă s-a opus unui progres natural.

În prezent există o paletă largă de funcţii de criptare cu cheie publică, numită şi criptare asimetrică. Acestea pot fi împărţite în două mari categorii după cele două mari probleme pe care se bazează (această clasificare nu este exhaustivă deoarece există şi alte probleme pe care se pot construi criptosisteme cu cheie publică, doar că acestea nu prezintă eficienţa necesară în practică): i) criptosisteme cu cheie publică bazate pe **dificultatea factorizării întregilor** (au ca punct de plecare algoritmul RSA [67]) şi ii) criptosisteme cu cheie publică bazate pe **dificultatea logaritmului discret** (au ca punct de plecare schimbul de cheie Diffie-Hellman [29] şi semnătura digitală ElGamal [33] precum şi extensiile acestora peste grupurile formate de curbe eliptice propuse de Koblitz şi Miller [63]).

Mai există desigur şi alte momente marcante în evoluţia criptografiei cu cheie publică care trebuie amintite deoarece vom discuta construcţii legate de acestea în capitole următoare. Goldwasser şi Micali descriu ideea de securitate semantică în 1982, Dolev, Dwork şi Naor descriu conceptul de non-maleabilitate în 1991, Bellare şi Rogaway introduc modelul oracolelor aleatoare 1995, şi tehnica de padding OAEP pentru a produce criptosisteme rezistente în faţa unor

adversari activi. În 1998 Bellare, Desai, Pointcheval şi Rogaway trasează şi demonstrează primele relaţii între noţiunile de securitate pentru criptosisteme cu cheie publică, Shoup demonstrează în 2001 că OAEP nu are securitatea susţinută de Bellare şi Rogaway, Fujisaki, Okamoto, Pointcheval şi Stern arată că RSA-OAEP este sigur, Canneti, Goldreich şi Halevi arată că modelul ROM este nesigur în 1996 (în 2002 apare versiunea de jurnal a tezei) – rămânând deschisă problema găsirii unui model mai bun. Lista numelor relevante în criptografia cu cheie publică nu este nici pe departe completă şi rămâne desigur deschisă. În secţiunile următoare urmează ca aceste realizări să fie explicate, aici a fost un bun prilej de a le pune succint într-un cadru istoric.

1.12 O PRIVIRE DE ANSAMBLU ASUPRA FUNCŢIILOR CRIPTOGRAFICE

Nu este deloc simplu a fixa o terminologie în acest domeniu străin oarecum de limba română. Vorbim de **funcţii criptografice**, de exemplu funcţii de criptare, funcţii de semnare digitală, sau alternativ în loc de funcţie folosim termenul de algoritm, deci **algoritm criptografic**, algoritm de criptare, etc. În general însă pentru a asigura operaţiile necesare avem nevoie de perechi de funcţii (algoritmi), de exemplu avem nevoie de o funcţie de criptare dar şi de una de decriptare, mai mult chiar şi de o funcţie de generare a cheii. Din acest motiv vorbim de **sisteme criptografice**, prin acestea desemnând o colecţie de una sau mai multe funcţii. De exemplu un sistem criptografic cu cheie secretă este colecţie de trei funcţii: o funcţie de generare a cheii, o funcţie de criptare şi o funcţie de decriptare. În engleză însă, mai frecvent decât termenul de sistem se foloseşte termenul *scheme*, de exemplu vorbim de *symmetric encryption scheme* desemnând colecţia de trei algoritmi anterior menţionaţi. La fel există asymmetric *encryption scheme* şi *digital signature scheme*. Pentru uniformitate cu terminologia în engleză, vom folosi termenul de **schemă criptografică** şi în particular **schemă de criptare simetrică, schemă de criptare asimetrică** şi **schemă de semnătură digitală** pentru a desemna un sistem format din unul sau mai mulţi algoritmi care asigură funcţionalităţile necesare (criptare, decriptare, semnare, etc.). Se foloseşte de asemenea frecvent în criptografie noţiunea de **funcţie one-way** pentru a desemna o funcţie care este greu (imposibil în limite rezonabile de timp) de inversat din punct de vedere computaţional. Un caz particular îl joacă **funcţiile one-way cu trapă**, adică acele funcţii care devin eficient de inversat dacă se cunoaşte o informaţie suplimentară numită trapă. În esenţă toate schemele criptografice se bazează pe funcţii one-way, lucru uşor de dedus de altfel (de exemplu funcţiile one-way cu trapă stau la baza oricărei scheme de criptare).

Definiţii mai riguroase pentru funcţiile one-way cu sau fără trapă se găsesc în anexa acestui volum.

În general, o schemă criptografică şi echivalent un sistem criptografic, sau simplu **criptosistem**, este un ansamblu format din trei algoritmi: un algoritm de generare a cheilor (cheie de criptare şi cheie de decriptare), un algoritm de criptare şi un algoritm de decriptare alături de mulţimile din care provin intrările lor – acest lucru este sugerat în Figura 1.11. Noţiunea de criptosistem aşa cum a fost anterior definită nu este deloc rigidă ea putând de fapt să surprindă toate primitivele criptografice ce vor fi descrise în continuare, fiind un simplu exerciţiu de imaginaţie a gândi scenariile pentru care algoritmul de generare a cheii poate să lipsească sau algoritmii de criptare sau decriptare îndeplinesc alte roluri decât criptarea sau decriptarea efectivă a informaţiei, etc.

Aşa cum am spus, se foloseşte frecvent, fără a pierde din semnificaţii, termenul de funcţie în locul celui de algoritm, de exemplu spunem funcţie de criptare sau funcţie de decriptare cu referire la algoritmii de criptare şi decriptare. Totodată este folosită şi noţiunea de **primitivă criptografică**, aceasta având un sens larg, desemnând orice bloc constructiv.

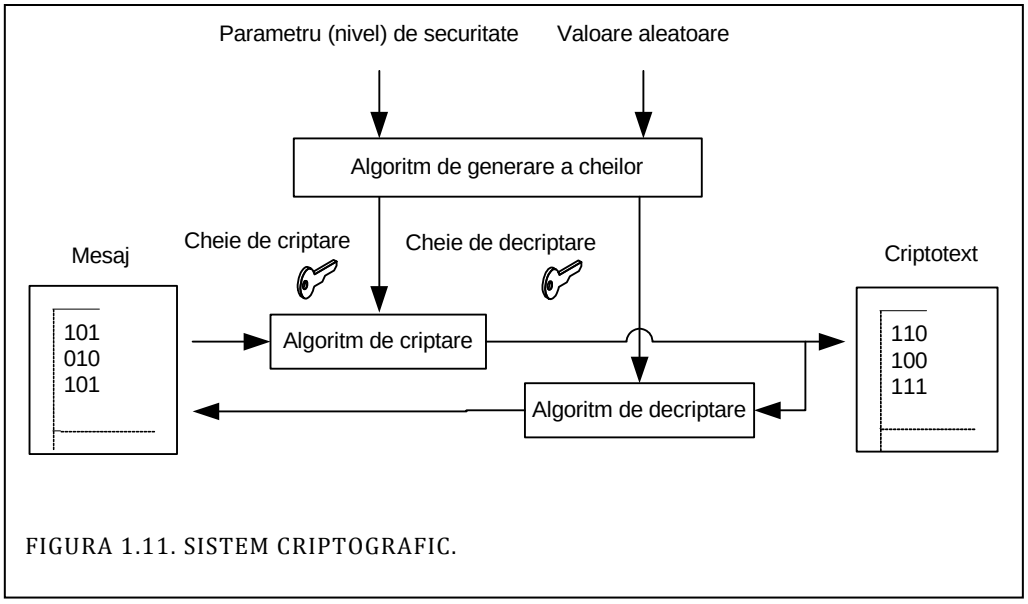


FIGURA 1.11. SISTEM CRIPTOGRAFIC.

Există o paletă largă de funcţii şi sisteme criptografice în literatura de specialitate, acestea pot fi clasificate ca aparţinând la trei mari categorii: criptosisteme cu

cheie secretă (sau **cheie simetrică)**, criptosisteme cu **cheie publică** (sau **cheie asimetrică**) şi criptosisteme **fără cheie**. Criptosistemele cu cheie simetrică utilizează aceeaşi cheie pentru criptare şi decriptare; avantajul lor este că necesită resurse de calcul reduse în timp ce dezavantajul este necesitatea unei chei secrete cunoscute de participanţii la comunicare (deci un secret partajat). Criptosistemele cu cheie asimetrică presupun utilizarea de chei diferite pentru criptare/decriptare; avantajul este posibilitatea de a efectua transmisii pe canale nesigure în absenţa unor secrete partajate iar dezavantajul este puterea de calcul relativ ridicată de care este nevoie. În principiu securitatea nu se poate construi decât în prezenţa funcţiilor din ambele categorii. Pe lângă criptosistemele cu cheie simetrică şi asimetrică mai există şi criptosistemele **fără cheie**. O taxonomie a criptosistemelor este sugerată în Figura 1.12.

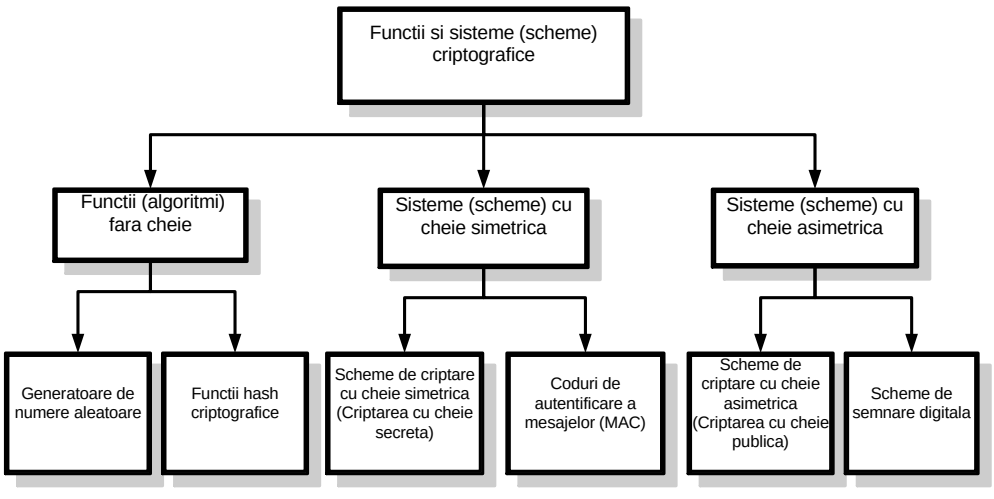


FIGURA 1.12. PRIVIRE DE ANSAMBLU ASUPRA SISTEMELOR CRIPTOGRAFICE.

1.13 DEZIDERATE ÎN SISTEMELE CRIPTOGRAFICE CONTEMPORANE

Este necesară introducerea unei astfel de secţiuni, pe cât de scurtă pe atât de utilă. În cei mai bine de 8 ani de când desfăşor activităţi în acest domeniu în cadrul Universităţii Politehnica din Timişoara, am remarcat o clară dezorientare a

studenţilor (chiar masteranzi sau doctoranzi) în ceea ce priveşte dezideratele constructive ale sistemelor criptografice. Pe scurt spus, lumea crede că obiectivul este construcţia unor sisteme imposibil de spart şi atât. În realitate însă scopul este construcţia unor sisteme imposibil de spart, dar totodată şi eficiente computaţional. Mai mult, eficienţa computaţională este un factor decisiv în alegerea unui sistem criptografic în practică.

Fără o înţelegere adecvată a domeniului, mulţi entuziaşti nu sunt conştienţi că oricând am putea construi un sistem mai greu de spart (de exemplu criptarea multiplă) dar acesta nu ar ajuta cu nimic dacă nu ar fi şi eficient. Codul Vernam, sau one-time-pad, este în continuare cel mai sigur criptosistem dar el nu este folosit pentru ca este ineficient (necesită chei de lungime egală cu mesajul). Deci pe scurt, în ceea ce priveşte funcţiile de criptare scopul ar fi construcţia unor funcţii cât mai rapide şi cu chei cât mai mici.

În prezent, în criptografie interesul este în a găsi sisteme noi, eficiente şi proprietăţi noi de securitate precum şi demonstraţii matematice că aceste proprietăţi sunt atinse.

1.14 LITERATURĂ RELEVANTĂ ÎN DOMENIU

Încerc să fac o trecere în revistă a literaturii care poate fi de interes pentru cititorul neiniţiat în acest domeniu.

Cărţi clasice despre criptografie. Există trei cărţi de bază în criptografia clasică: Menezes et al. [62], Stinson [81] şi Schneier [73]. Denumim aceste cărţi clasice deoarece, cu toate că ele sunt actuale, nu oferă informaţii cu privire la tehnici moderne fără de care domeniul nu mai poate fi conceput în zilele noastre (de exemplu reducţia de securitate sau curbe eliptice). Cartea lui Menezes et al. [62] este extrem de sintetică şi consistentă din punct de vedere al matematicii, iar cartea lui Stinson [81] este de asemenea coerentă din punct de vedere matematic şi conţine formalisme destul de bune cu privire la criptosisteme. Cartea lui Schneier [73] este din punct de vedere al matematicii ceva mai slabă, expunerile fiind mai literare, o recomand în special celor care sunt interesaţi de securitate în general şi nu neapărat de criptografie. Atenţie, toate aceste trei cărţi prezintă soluţii învechite, de exemplu nu sunt abordate noţiunile de IND-CCA2 şi NM-CCA2 vitale pentru criptosistemele cu cheie publică contemporane (vezi capitolul 6 al prezentei lucrări). Pentru cititorul care nu are mult timp de pierdut şi care trebuie să îşi formeze rapid o imagine despre criptografie recomand cartea recentă a lui Fergusson şi Scheneier [35] care are expuneri simple şi de ansamblu asupra domeniului.

Cărţi moderne despre criptografie. Cărţile lui Mao [60] şi Katz şi Lindell [54] sunt cărţi moderne care conţin noţiuni de securitate moderne şi demonstraţii (reducţii) de securitate actuale. Prima prezintă o abordarea bazată pe soluţiile existente în lumea reală şi este cartea pe care o recomand în primul rând. Este excelentă ca lucrare de ansamblu, suficient de scurtă şi totuşi enciclopedică şi modernă, cursul ţinut la MIT de Bellare şi Goldwasser [4]. Acesta poate fi însă greu de înţeles pentru cititorul lipsit de o bază oarecare în domeniu şi este un material extrem de dens. Cărţi solide pentru fundamente ale criptografiei sunt cele două volume publicate de Goldreich [42], [43].

Cărţi despre securitatea informaţiei. Pentru o vedere generală asupra aplicaţiilor de securitate şi parţial a criptografiei este utilă cartea lui Anderson [1] deşi nu are prea mult de a face cu ingineria aşa cum titlul sugerează. Tot orientată spre practică este şi cartea lui Stallings [80].

Cărţi despre Teoria Numerelor. Din punct de vedere al criptografiei cu cheie publică sunt foarte relevante aspectele de teoria numerelor, pentru acestea au devenit referinţă în domeniu cărţile lui Apostol [2] şi a lui Hardy şi Wright [49]. Pentru teoria probabilităţilor trebuie consultată [34]. Abordări ale criptografiei prin prisma matematicii (în special algebră şi teoria numerelor) pot fi găsite în cartea lui Koblitz [56] şi cea a lui Cohen [23]. Este de asemenea o mare realizare în domeniu materialul oferit de Shoup [77] care se focalizează asupra elementelor de calcul necesare (pe Internet este disponibilă şi o bibliotecă de funcţii implementate în C). Tot ca abordare din perspectiva teoriei numerelor recomand şi cartea lui Wagstaff [83]. Pentru cei interesaţi de criptografia pe curbe eliptice (extensie a criptosistemelor lui Diffie-Hellman şi ElGamal), care poate fi văzută în sine ca un domeniu, recomand în primul rând lucrarea lui Cohen et al. [24] apoi lista continuă cu lucrarea lui Washington [84] şi lucrarea lui Hankerson et al. [48] care o recomand în special pentru cei care vor să implementeze, primele două cărţi le recomand celor interesaţi de aspectele de matematică. Cele două cărţi publicate la Cambridge despre curbe eliptice având coautor pe Gadiel Seroussi sunt călduros recomandate [10], [11], personal nu le-am consultat dar am asistat la cursurile lui Serrousi pe această tema de la Bonn, Germania, în septembrie 2007.

Articole de specialitate. În ceea ce priveşte articolele de specialitate din conferinţe şi jurnale aici lista recomandărilor ar fi enormă. Am inclus la bibliografie doar câteva articole de referinţă. În general sunt recomandate lucrările publicate în conferinţele centrale ale criptografiei, organizate de International Association for Cryptologic Research (IACR) [52]: Crypto, Eurocrypt, Asiacrypt, International Workshop on Practice and Theory in Public Key Cryptography (PKC), Fast Software Encryption (FSE) şi mai recent Theory of

Cryptography Conference (TCC). Jurnalul central în domeniu care este Journal of Cryptology publicat tot de IACR (detalii cu privire la toate acestea se găsesc pe pagina IACR www.iacr.org). Este de menţionat şi jurnalul Design Codes and Cryptography, jurnal cu tradiţie în acest domeniu.

Desigur, mai există şi alte publicaţii bune decât cele enumerate aici, lista nu este exhaustivă.

1.15 SĂ NU SUPRAEVALUĂM CRIPTOGRAFIA: SECURITATEA ESTE UN PROCES, NU UN PRODUS

Este bine să încheiem capitolul cu modestie. Bruce Schneier în prefaţa la lucrarea Secrets and Lies îşi recunoaşte greşeala din Apllied Cryptography de a supraevalua rolul criptografiei. Trebuie înţeles că criptografia este o piesă esenţială în securitate, în opinia autorului cea mai importantă, dar nu e singura. "Security is a process, not a product" este o afirmaţie comună în securitatea informaţiei. Trebuie înţeles că securitatea în sine nu este un simplu produs pe care îl cumpărăm şi apoi totul decurge de la sine. Considerăm cazul simplu al unui antivirus, îl cumpărăm şi instalăm. Va funcţiona? Pentru aceasta este necesar şi ca el să poată face update, şi ca echipa care îl produce să lucreze la aceste update-uri, etc. Nu este vorba deci de o simplă cumpărare. Altfel, putem instala securitate de cel mai înalt nivel la un serviciu on-line, putem constrânge utilizatorii să folosească parole de 14 caractere corect alese. Dar dacă utilizatorii îşi spun parola mai departe, este sistemul sigur? Astfel securitatea trebuie văzută ca un proces, ca un sistem în ansamblu, în care fiecare componentă joacă un rol. Nu putem avea securitatea personală prin simpla cumpărare a unui produs, este necesară şi implicarea personală care presupune în primul rând educarea utilizatorului. Din aceste motive securitatea nu este simplu de asigurat în cadrul unui sistem.

2 SCHEME DE CRIPTARE CU CHEIE SIMETRICĂ (CRIPTAREA CU CHEIE SECRETĂ)

Istoria criptografiei începe cu construcţia funcţiilor de criptare cu cheie secretă. Începem cu expunerea unor procedee constructive de bază şi continuăm cu detalii despre criptosisteme contemporane precum DES şi AES. Nu în ultimul rând discuţia se axează şi asupra modurilor de operare a acestor criptosisteme, moduri esenţiale pentru securitate în practică.

2.1 DEFINIŢIE ŞI PROPRIETĂŢI

Criptarea simetrică a mesajului m cu cheia secretă K o vom nota cu $E_K(m)$, unde E reprezintă algoritmul (funcţia) de criptare şi este întotdeauna cunoscut public. Securitatea criptosistemului (schemei de criptare simetrică) depinde doar de păstrarea secretă a cheii K (în nici un caz de vreun detaliu al algoritmului care este public). A cripta înseamnă a aplica algoritmul E_K asupra mesajului m pentru a obţine criptotextul $c = E_K(m)$ iar a decripta înseamnă a aplica algoritmul $E_{K^{-1}}$ (adeseori notat şi D_K de la decriptare sau E_K^{-1} simbolizând inversa funcţiei de criptare) asupra criptotextului c pentru a obţine mesajul m. Un sistem criptografic simetric îl definim după cum urmează.

Definiţia 2.1. (schemă de criptare cu cheie simetrică) *O schemă (sistem) de criptare cu cheie simetrică (secretă) constă în trei algoritmi: algoritmul de generare a cheii* $K \leftarrow Sym.Gen(1^k)$ *care primeşte ca intrare nivelul de securitate* k *şi returnează cheia* K*, algoritmul de criptare* $E_K(m) \leftarrow Sym.Enc(m, K)$ *care primeşte ca intrare mesajul* m *şi cheia* k *returnând mesajul criptat* $c = E_K(m)$ *şi algoritmul de decriptare* $m \leftarrow Sym.Dec(c, K)$ *care primeşte ca intrare un criptotext* $c = E_K(m)$ *şi cheia* K *şi returnează mesajul aferent* m*. Toate acestea alături de spaţiile datelor de intrare asociate acestora care fără a pierde din generalitate le considerăm spaţii ale stringurilor binare.*

Prin 1^k notăm prin convenţie stringuri binare de k, biţi iar acest parametru al algoritmului de generare are rolul de a fixa dimensiunea cheii.

Pentru algoritmii simetrici, prin convenţie, presupunem $K = K^{-1}$ prin aceasta înţelegând că putem calcula uşor cheia de decriptare din cea de criptare. Subliniem faptul că aceasta nu înseamnă că aceste chei sunt identice (şi nici nu sunt în cele mai multe cazuri practice) ci doar că una este uşor de dedus din cealaltă. Prin antiteză, pentru algoritmii asimetrici ce vor fi introduşi într-un capitol următor avem $K \neq K^{-1}$, ceea ce înseamnă că nu este fezabil a calcula cheia de decriptare din cea de criptare.

Figurile 2.1 şi 2.2 sunt ilustrative pentru algoritmii de criptare şi decriptare iar în Figura 2.3 sunt utilizaţi algoritmii din figurile 2.1 şi 2.2 pentru a construi schema de ansamblu a unui sistem de criptare cu cheie simetrică.

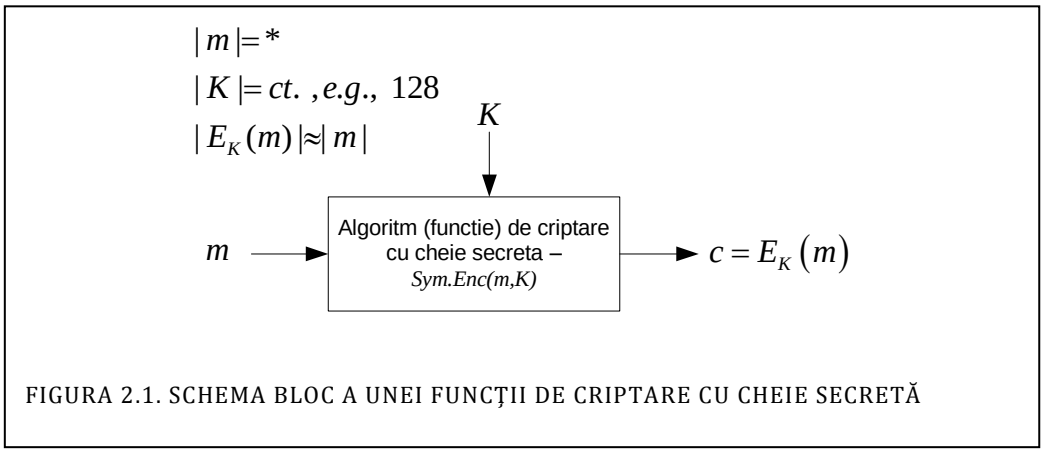


FIGURA 2.1. SCHEMA BLOC A UNEI FUNCŢII DE CRIPTARE CU CHEIE SECRETĂ

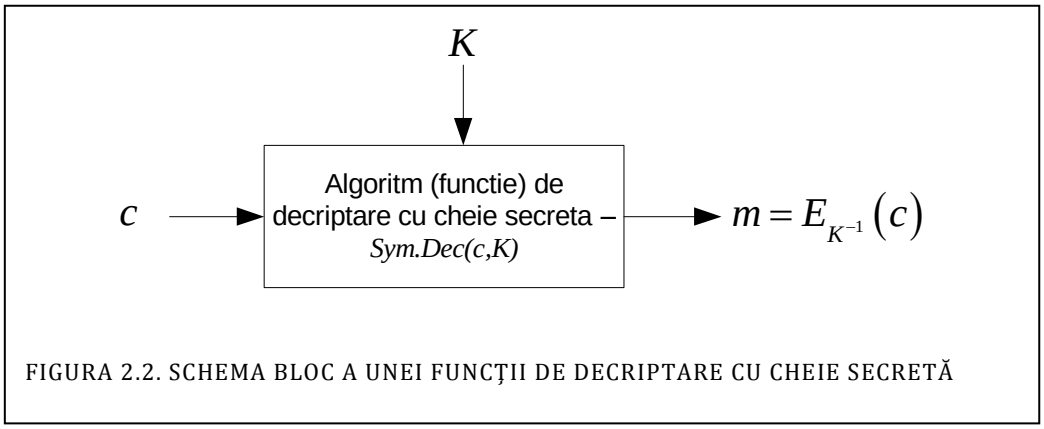


FIGURA 2.2. SCHEMA BLOC A UNEI FUNCŢII DE DECRIPTARE CU CHEIE SECRETĂ

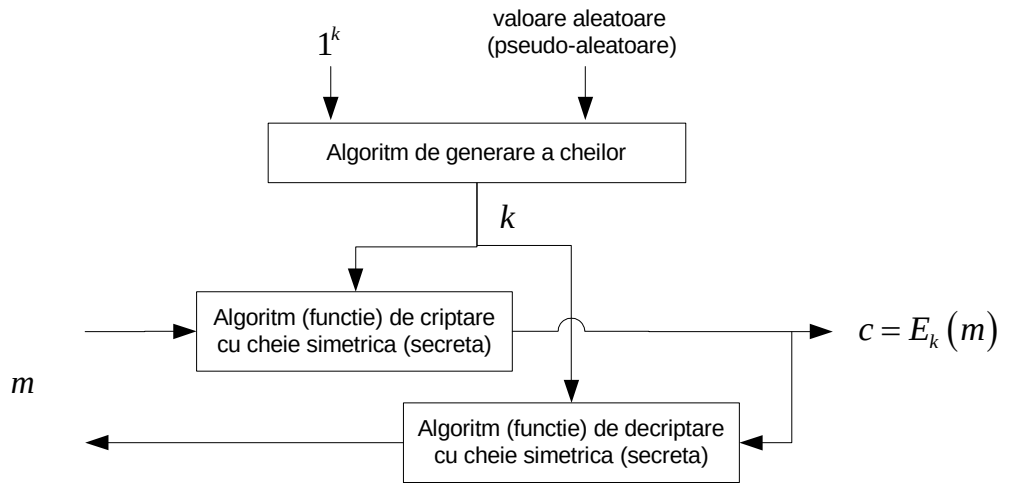


FIGURA 2.3. SCHEMA DE CRIPTARE CU CHEIE SIMETRICĂ

.

Două proprietăţi sunt esenţial de subliniat în cazul schemelor simetrice:

i) Fără cheia K din criptotextul $c = E_K(m)$ nu se poate afla nimic despre mesajul m.

ii) Criptotextul $c = E_K(m)$ alături de orice informaţie cu privire la mesajul m nu aduce nici o informaţie cu privire la K.

Nu sunt însă nicidecum singurele proprietăţi ale unei scheme de criptare simetrică. Acestora li se pot asocia şi proprietăţi avansate, precum imposibilitatea adversarului de a modifica mesajul din interiorul unui criptotext, adică fără cunoaşterea cheii K, chiar dacă m este cunoscut, un adversar nu poate altera $c = E_K(m)$ fără ca receptorul mesajului să poată detecta acest lucru la momentul decriptării. Această proprietate poartă numele de **non-maleabilitate** (non-maleability). O altă proprietate este imposibilitatea adversarului de a distinge dacă este criptat un bit de valoare 0 sau un bit de valoare 1. Adică, având $c = E_K(0)$ şi $c = E_K(1)$, fără cunoaşterea cheii K un adversar nu poate spune

care este criptarea lui 1 şi care a lui 0. Această proprietate se numeşte **imperceptibilitatea sau nedistingerea criptotextelor** (indisinguishability of encryptions).

Avantajul algoritmilor simetrici constă în viteza ridicată de criptare şi decriptare. În comparaţie cu cei asimetrici aceştia sunt mai rapizi cu câteva ordine de magnitudine, de 10, 100 sau chiar 1000 de ori. În ceea ce priveşte deficienţele, prima deficienţă a algoritmilor simetrici este aceea că nu permit efectuarea unei comunicări pe un canal nesecurizat în prealabil (necesită partajarea prealabilă a unui secret). În acelaşi context, al cheilor partajate, deficienţa algoritmilor simetrici constă în faptul că odată cu creşterea numărului de participanţi la comunicare creşte şi numărul de chei secrete care trebuie cunoscute; de exemplu pentru comunicarea între n entităţi avem nevoie de $n\cdot(n-1)/2$ chei secrete. O alternativă ar fi utilizarea unui server de autentificare (parte de încredere) care stochează chei criptografice partajate cu fiecare entitate (aceste chei fiind valabile pe termen lung de unde şi denumirea de *long-term key*) şi le foloseşte pentru a transmite chei valabile pe termen scurt pentru comunicarea între entităţi care nu au avut un contact prealabil. Într-un astfel de scenariu numărul de chei este egal cu cel de participanţi.

2.2 CLASIFICARE: CODURI BLOC ŞI CODURI STREAM

Codurile de criptare cu cheie simetrică pot fi împărţite în două mari categorii după modul în care prelucrează mesajul ce urmează a fi criptat:

i) **Coduri bloc**: sunt algoritmi de criptare care operează asupra mesajului ce trebuie criptat bloc cu bloc, fiecare bloc având dimensiune fixă (deci criptează un bloc la un moment dată).

ii) **Coduri stream**: sunt algoritmi de criptare care criptează „bit cu bit” (practic lungimea blocului este 1) şi transformarea de criptare se poate modifica pentru fiecare caracter în parte.

Codurile stream sunt în special utile când există erori de transmisie (deoarece nu au erori de propagare) şi atunci când datele trebuie procesate pe măsură ce ajung la destinaţie (nu există spaţiu de stocare). Modurile de funcţionare ce urmează a fi discutate în acest capitol permit transformarea unui cod bloc în cod stream. În practică se folosesc cu precădere coduri bloc, dar nu sunt de neglijat nici cele stream, de exemplu în transmisii wireless. În general la codurile stream operaţia de criptare constă într-o operaţie simplă (de exemplu un

simplu XOR cu cheia curentă) iar generarea cheii curente este mai intensă computaţional.

2.3 PRINCIPII CONSTRUCTIVE: SUBTITUŢIA ŞI TRANSPOZIŢIA, CIFRU PRODUS

În construcţia funcţiilor criptografice simetrice există două operaţii universal utilizate:

i) **Substituţia** înseamnă înlocuirea unor simboluri sau grupuri de simboluri prin alte simboluri sau grupuri de simboluri şi creează **confuzie**.

ii) **Transpoziţia (sau permutarea)** – înseamnă permutarea (amestecarea) simbolurilor din cadrul unui bloc şi creează **difuzie** (numită uneori şi difuziune).

Aceste două noţiuni au fost introduse pentru prima dată de Claude Shannon în 1949 şi reprezintă concepte fundamentale în construcţia schemelor moderne de criptare simetrică. Toate schemele moderne conţin casete de substituţie S-Box şi permutare P-box pentru a crea confuzie şi difuzie.

Se foloseşte termenul de **cifru produs** (product cipher) pentru a desemna un cifru obţinut din combinarea a două sau mai multe proceduri care conduc la un criptosistem mai sigur decât utilizarea individuală a acestora. Astfel, o reţea de substituţii (S) şi permutări (P), denumită în general **reţea-SP** (SP-network), este unul dintre cele mai simple cazuri de cifruri produs.

2.4 MAŞINA ENIGMA

Maşina Enigma şi componentele sale sunt ilustrate în Figura 2.4. Textul ce urma a fi criptat era introdus de la tastatură şi la fiecare tastă apăsată se aprindea becul aferent criptotextului în tabela de becuri (rezultatul fiind notat pe hârtie de un operator). Au existat şi variante la care rezultatul era tipărit pe hârtie dar acestea au fost mai puţin frecvente (costuri suplimentare şi greutate). Varianta standard avea 3 rotoare alese dintru set de 5 şi o tabelă de ştechere unde maxim 13 ştechere se puteau folosi pentru a lega câte 2 litere. Variante mai evoluate au avut şi 4 rotoare sau mai mult. Pe scurt principiul de funcţionare era următorul: la apăsarea unei taste se închidea un circuit ce permitea curentului electric să treacă prin tabela de ştechere (unde o literă putea fi sau nu legată de altă literă) apoi

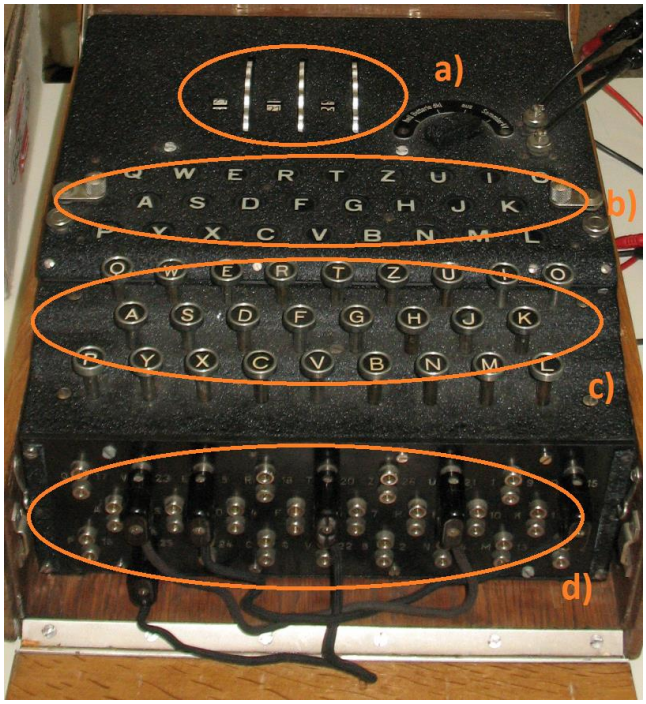


FIGURA 2.4. COMPONENTELE MAŞINII ENIGMA: A) CELE 3 ROTOARE, B) TABELA CU BECURI, C) TASTATURA, D) TABELA DE ŞTECHERE

prin cele 3 rotoare care aveau şi ele cablaje diferite iar din cel de-al patrulea rotor (numit reflector) curentul se întorcea prin cele 3 rotoare, revenea în tabela de ştechere şi aprindea unul din becurile din panou (care era rezultatul literei criptate).

Pe scurt, cheia maşinii Enigma avea la bază următoarele setări: ordinea rotoarelor (alese ca 3 rotoare din 5 variante şi puse în orice ordine), poziţia iniţială a rotoarelor (fiecare are 26 de poziţii), setarea inelului (pe fiecare rotor era un inel care indica unde se acţionează mişcarea următorului rotor) şi setarea tabelului de ştechere (unde maxim 13 ştechere se puteau folosi pentru a conecta cele 26 de intrări). Dacă considerăm doar cele 13 ştechere care leagă 26 de poziţii şi faptul că avem 3 rotoare din care fiecare poate fi pus în 26 de poziţii deja depăşim spaţiul de chei al DES-ului (aproape de 2 ori!) deoarece avem: $26^3 \times \frac{26!}{13! \times 2^{13}} = 138953282533065000$ variante de cheie. Aceasta nu ţine cont de setarea inelului pe fiecare rotor şi de ordinea rotoarelor şi tot este mai mult decât cei 56 de biţi de cheie ai DES, criptosistem care a fost sigur până la nivelul anilor 90.

Astfel, spaţiul din care provine cheia este impresionant, mult mai mare decât al DES-ului şi chiar mai mare decât în cazul AES dacă Enigma este folosită la

puterea ei maximă. Dar nici această dimensiune a spaţiului din care provine cheia nu este suficientă pentru securitate, maşina Enigma realizează doar o simplă substituţie polialfabetică. Se poate observa o deficienţă constructivă chiar şi după această descriere sumară: prin apăsarea unei litere circuitul electric nu se putea închide sub acea literă şi deci o literă nu se putea cripta în ea însăşi. Acest lucru este o deficienţă majoră deoarece o bună parte din mesajele sursă pot fi eliminate din start (în special în război, variantele de mesaj sursă sunt puţine deoarece în general reprezentau ordine militare scurte, etc.).

2.5 REŢEAUA FEISTEL

Maşina Enigma este încă un sistem fără prea multe legături cu sistemele contemporane. **Reţeaua Feistel**, sau codul Feistel, este primul criptosistem simetric modern. Aceasta a fost construită în anii 70 de către Horst Feistel şi a apărut în primul criptosistem comercial numit Lucipher construit de Horst Feistel şi Don Coppersmith la IBM în 1973.

Marea parte a criptosistemelor simetrice contemporane urmează principiile codului Feistel. Schema de principiu a reţelei Feistel se găseşte în Figura 2.5. În reţelele Feistel se aplică următoarele transformări asupra mesajului: permutări (P-boxes) pentru a crea difuzie, substituţii, pentru a crea confuzie (S-boxes) şi operaţii pe biţi (XOR). În general este necesar un minim de 16 runde pentru securitate adecvată.

Principii constructive generale în reţeaua Feistel sunt: cu cât dimensiunea blocului, a cheii şi numărul de runde creşte, creşte şi securitatea respectiv scade viteza de criptare/decriptare, iar dacă scad, scade şi securitatea respectiv creşte viteza de criptare/decriptare.

Marele avantaj este că criptarea şi decriptarea se fac parcurgând aceeaşi reţea în sens invers, deci este foarte eficientă ca şi cost de implementare (în special hardware). În Figura 2.5 de asemenea se pot distinge cheile $K_0, K_1, ..., K_{l-1}$ ale celor l runde de criptare şi funcţia F care este funcţia rundei de criptare. În principiu această funcţie poate fi oricât de complexă şi nu trebuie să fie o funcţie reversibilă. Un bun exemplu pentru utilizarea reţelei Feistel este criptosistemul DES ce îl discutăm în continuare.

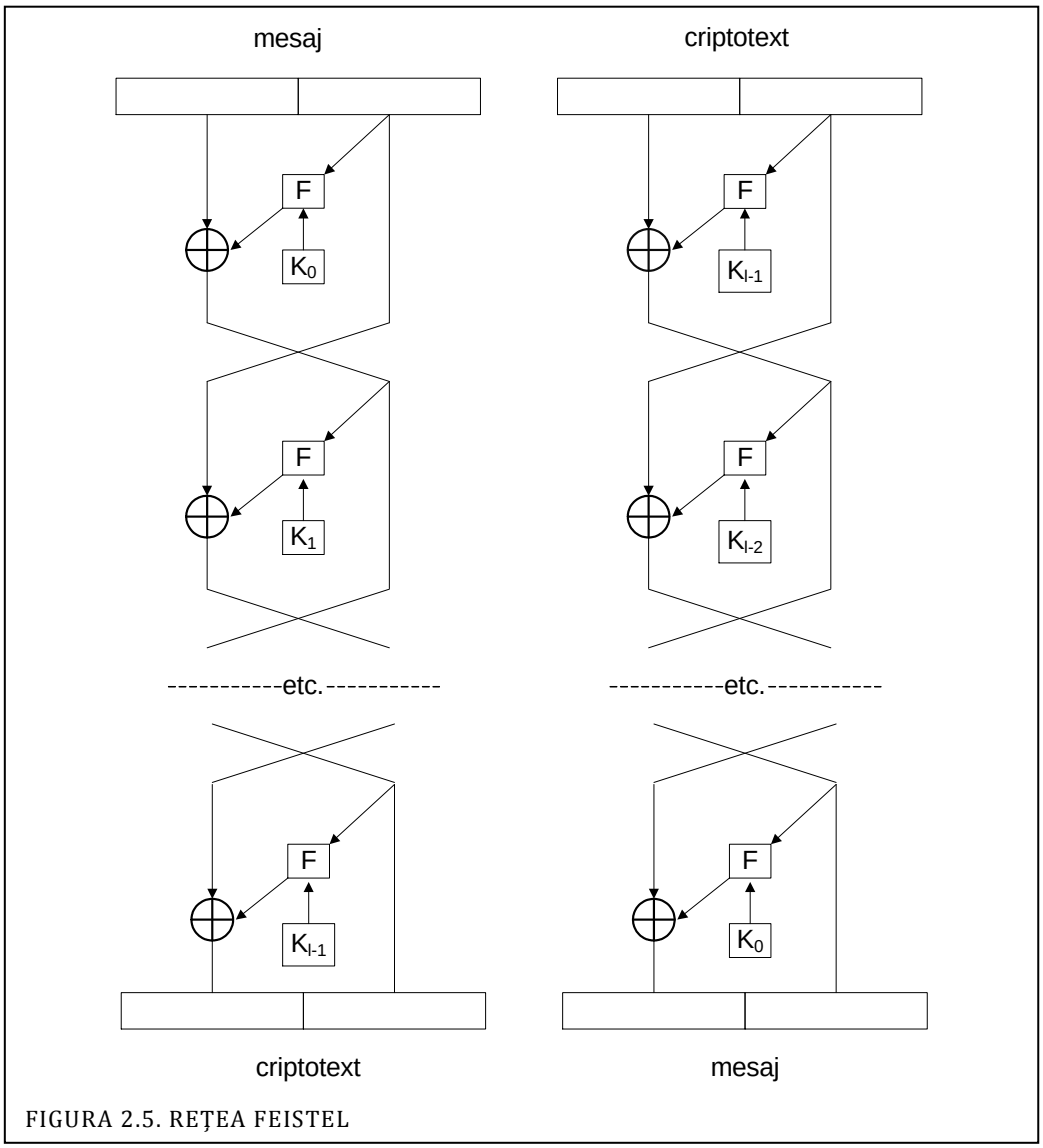


FIGURA 2.5. REŢEA FEISTEL

2.6 CRIPTOSISTEMUL DES

Standardul în criptarea simetrică, valabil până în 2001, a fost **DES (Data Encryption Standard)** [36]. DES este un cod construit pe o reţea Feistel cu 16 runde care transformă mesaje de 64 de biţi în criptotext de 64 de biţi.

Cheia DES are însă doar 56 de biţi şi nu 64 (confuzie comună datorată poate faptului că la fiecare 7 biţi de cheie există 1 bit de paritate ceea ce duce la o dimensiune totală de 64 de biţi). Înainte de intrarea în reţeaua Feistel şi după ieşirea din aceasta, blocul de 64 de biţi este trecut printr-o permutare respectiv prin inversa acesteia. Acest lucru este sugerat în Figura 2.6 (i). Rolul permutării este desigur de a crea difuzie. Permutarea, denumită IP (Initial Permutation), şi inversa acesteia sunt ilustrate în Tabelul 2.1. Se observă (ca exemplu de permutare) pe poziţia 1 apare valoarea 58, adică bitul de pe poziţia 58 vine pe poziţia 1, iar în permutarea inversă, pe poziţia 58 apare valoarea 1, adică bitul de pe poziţia 58 vine pe poziţia 1, etc.

Funcţia de rundă, aşa cum este descrisă în cadrul reţelei Feistel, are ca parametru de intrare blocul drept şi cheia de rundă. Dat fiind că blocul de intrare este de 64 de biţi, blocul drept (la fel ca şi cel stâng) are 32 de biţi. Acesta este însă trecut printr-o funcţie de expansiune E care îl aduce la 48 de biţi pentru a face XOR cu cheia de rundă tot de 48 de biţi. Rezultatul este folosit ca intrare pentru 8 casete de substituţie S-box. Fiecare casetă are la ieşire 4 biţi de unde rezultă un total de 32 de biţi care sunt din nou trecuţi printr-o permutare P. Funcţia de expansiune E şi permutarea P sunt prezentate în Tabelul 2.1.

Casetele de substituţie S-box lucrează în felul următor: sunt 8 casete la intrarea fiecăreia venind cât 6 biţi din blocul de la intrare (6x8=48). Din cei 6 biţi 4 au rolul de a selecta coloana şi 2 de a selecta linia din S-box iar la ieşire oferă valoarea de pe poziţia respectivă. Deoarece la intrarea unui S-box sunt 48 de biţi iar la ieşire 32, S-box este de fapt o compresie (ne-bijectivă). Aceasta nu influenţează corectitudinea decriptării. În Tabelul 2.1 sunt ilustrate casetele S1 şi S2, mai există 6 casete, care nu le prezentăm deoarece nu oferă informaţii foarte relevante (standardul poate fi consultat pentru valorile acestora).

Rămâne de detaliat cum se calculează cheia de rundă. Aşa cum se observă în Figura 2.6 (iii) aceasta trece printr-o permutare numită PC1 apoi la rândul ei este spartă în două blocuri care pentru fiecare rundă sunt shiftate la stânga. Permutarea PC1 este împărţită în două componente, prim şi secund, fiecare corespunzând la câte 28 de biţi de cheie (cheia ne amintim are 56 de biţi). Numărul de shiftări este 1 pentru rundele 1, 2, 9 şi 16, respective 2 pentru celelalte runde. La final cheia de rundă trece printr-o nouă permutare numită PC2. Acest calcul din care provine cheia se mai numeşte şi **program de cheie** (key schedule).

$$IP = \begin{pmatrix} 58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 \\ 60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 \\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 \\ 64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 \\ 57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 \\ 59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 \\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 & 7 \end{pmatrix} \quad IP^{-1} = \begin{pmatrix} 40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 \\ 39 & 7 & 47 & 15 & 55 & 23 & 63 & 31 \\ 38 & 6 & 46 & 14 & 54 & 22 & 62 & 30 \\ 37 & 5 & 45 & 13 & 53 & 21 & 61 & 29 \\ 36 & 4 & 44 & 12 & 52 & 20 & 60 & 28 \\ 35 & 3 & 43 & 11 & 51 & 19 & 59 & 27 \\ 34 & 2 & 42 & 10 & 50 & 18 & 58 & 26 \\ 33 & 1 & 41 & 9 & 49 & 17 & 57 & 25 \end{pmatrix}$$

$$E = \begin{pmatrix} 32 & 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 1 \end{pmatrix} \quad S_1 = \begin{pmatrix} 14 & 4 & 13 & 1 & 2 & 15 & 11 & 8 & 3 & 10 & 6 & 12 & 5 & 9 & 0 & 7 \\ 0 & 15 & 7 & 4 & 14 & 2 & 13 & 1 & 10 & 6 & 12 & 11 & 9 & 5 & 3 & 8 \\ 4 & 1 & 14 & 8 & 13 & 6 & 2 & 11 & 15 & 12 & 9 & 7 & 3 & 10 & 5 & 0 \\ 15 & 12 & 8 & 2 & 4 & 9 & 1 & 7 & 5 & 11 & 3 & 14 & 10 & 0 & 6 & 13 \end{pmatrix}$$

$$P = \begin{pmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{pmatrix} \quad S_2 = \begin{pmatrix} 15 & 1 & 8 & 14 & 6 & 11 & 3 & 4 & 9 & 7 & 2 & 13 & 12 & 0 & 5 & 10 \\ 3 & 13 & 4 & 7 & 15 & 2 & 8 & 14 & 12 & 0 & 1 & 10 & 6 & 9 & 11 & 5 \\ 0 & 14 & 7 & 11 & 10 & 4 & 13 & 1 & 5 & 8 & 12 & 6 & 9 & 3 & 2 & 15 \\ 13 & 8 & 10 & 1 & 3 & 15 & 4 & 2 & 11 & 6 & 7 & 12 & 0 & 5 & 14 & 9 \end{pmatrix}$$

$$PC1' = \begin{pmatrix} 57 & 49 & 41 & 33 & 25 & 17 & 9 \\ 1 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 2 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 3 & 60 & 52 & 44 & 36 \end{pmatrix}$$

$$PC1'' = \begin{pmatrix} 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 7 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 6 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 5 & 28 & 20 & 12 & 4 \end{pmatrix}$$

$$PC2 = \begin{pmatrix} 14 & 17 & 11 & 24 & 1 & 5 \\ 3 & 28 & 15 & 6 & 21 & 10 \\ 23 & 19 & 12 & 4 & 26 & 8 \\ 16 & 7 & 27 & 20 & 13 & 2 \\ 41 & 52 & 31 & 37 & 47 & 55 \\ 30 & 40 & 51 & 45 & 33 & 48 \\ 44 & 49 & 39 & 56 & 34 & 53 \\ 46 & 42 & 50 & 36 & 29 & 32 \end{pmatrix}$$

TABELUL 2.1. ELEMENTE CONSTRUCTIVE DES CONFORM FIPS46-2: PERMUTAREA IP (INITIAL PERMUTATION) ŞI INVERSA EI, FUNCŢIA DE EXPANSIUNE E ŞI PERMUTAREA P, CASETELE DE SUBSTITUŢIE S1 ŞI S2 (MAI SUNT ÎNCĂ 6 CASETE), PERMUTĂRILE PENTRU PROGRAMUL DE CHEIE PC1 SI PC2

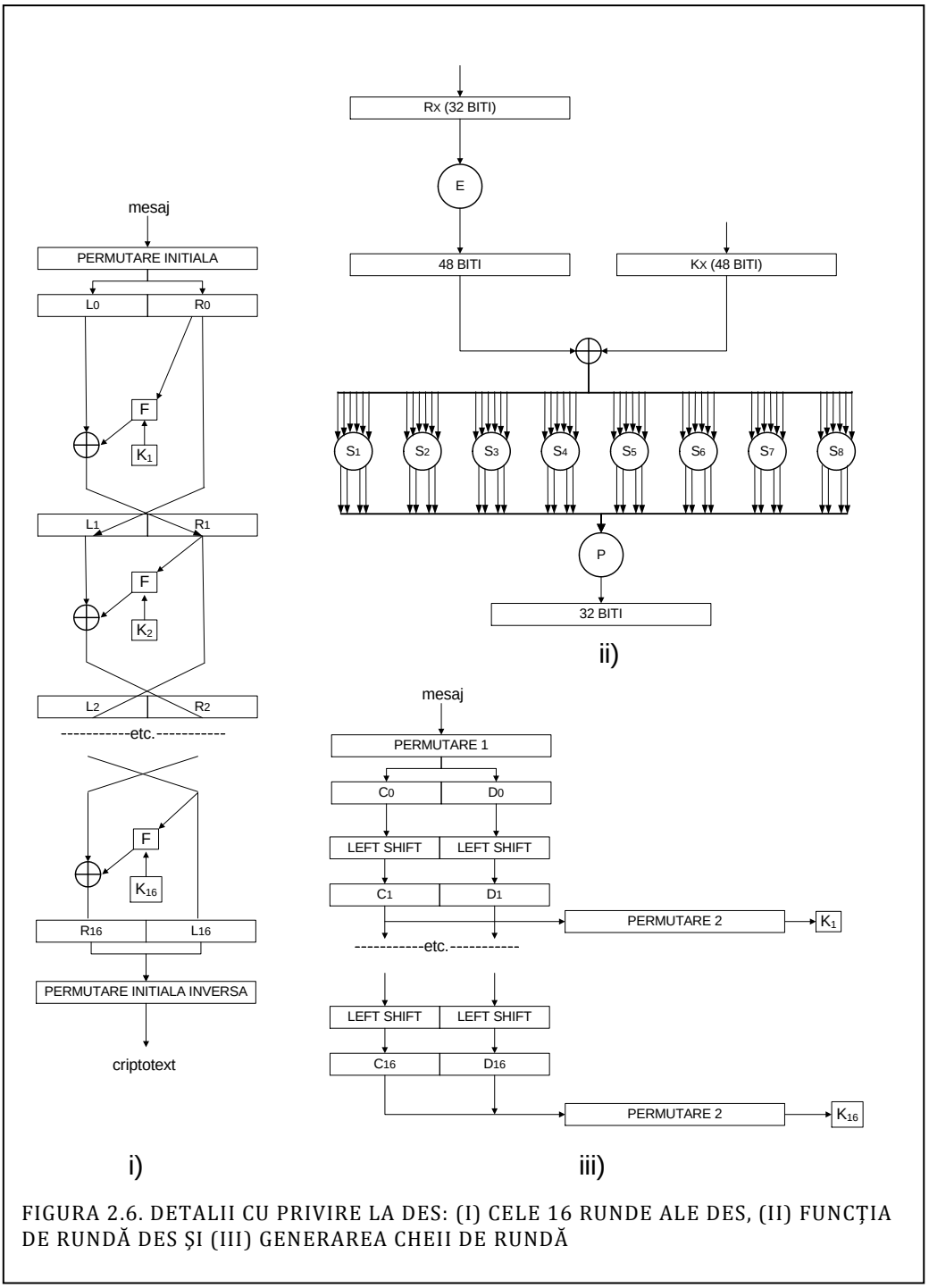


FIGURA 2.6. DETALII CU PRIVIRE LA DES: (I) CELE 16 RUNDE ALE DES, (II) FUNCŢIA DE RUNDĂ DES ŞI (III) GENERAREA CHEII DE RUNDĂ

2.7 3DES

De mulţi ani DES nu mai oferă securitate corespunzătoare, putând fi spart în câteva zile pe o maşină de calcul dedicată (de exemplu COPACOBANA sparge DES în medie de 3.5 zile). DES supravieţuieşte însă sub forma 3DES (recomandat încă din 1999) oferind un nivel de securitate suficient de bun în zilele de azi. Acesta constă în aplicarea transformării DES de 3 ori după cum urmează:

$$c = E_{K3}\left(D_{K2}\left(E_{K1}(m)\right)\right), \qquad m = D_{K1}\left(E_{K2}\left(D_{K3}(c)\right)\right)$$

Conform standardului, există următoarele variante de utilizare a cheilor: opţiunea 1 cu 3 chei independente, opţiunea 2 cu K_1 şi K_2 independene iar $K_1 = K_3$ şi opţiunea 3 cu o singură cheie independentă $K_1 = K_2 = K_3$. Astfel cheia de la 3DES poate fi pe 56, 112 şi 168 de biţi.

Dat fiind că 3DES este mai lent decât AES şi nu oferă securitate mai bună, nu există motive serioase pentru a fi folosit în practică astăzi. Poate un motiv ar fi faptul că există implementări hardware ce pot fi refolosite şi în acest fel se mai reduc din costuri, dar nu există raţiuni de securitate.

2.8 CRIPTOSISTEMUL AES

La nivelul anilor 2001 DES nu mai oferă securitatea necesară (de fapt încă din anii 90 sunt consemnate atacuri de succes asupra DES), pentru care, pe bază de concurs se alege un nou standard **AES (Advanced Encryption Standard)**.

Standardul curent este candidatul la AES numit **Rijndael** [38] ales din cei 5 finalişti: Rijndael, Serpent, Twofish, RC6 şi MARS. AES este un cod bloc disponibil în trei variante de dimensiuni pentru cheie 128, 192, 256. Chiar şi cheia de 128 de biţi este considerată destul de sigură pentru cerinţele din ziua de azi. Necesită doar 10-14 runde în funcţie de dimensiunea cheii, este sigur şi este cel mai rapid dintre candidaţi. Deoarece AES este mai rapid decât alte coduri simetrice, chiar şi decât 3DES, şi oferă cel puţin acelaşi nivel de securitate nu există nici un motiv de a utiliza altceva decât AES în arhitecturi de securitate contemporane.

AES nu foloseşte structura Feistel, are meritul de a fi un criptosistem inovator. El procesează matrici de 4x4 bytes prin intermediul a 4 transformări:

- **AddRoundKey** – se adună cheia de rundă printr-un simplu XOR,

- **SubBytes** – se substituie fiecare byte prin intermediul unei tabele de look-up (substituţie neliniară),
- **ShiftRows** – se shiftează circular (rotire) fiecare linie astfel: prima linie e neatinsă, a 2-a linie 1 la stânga, a 3-a cu 2 la stânga şi a 4 cu 3 la stânga,
- **MixColumns** – se amestecă coloanele prin aplicarea unei transformări de această dată liniară şi reversibilă (de fapt este vorba de multiplicare matricială).

Macrostructura rundei AES este sugerată în Figura 2.7. Fiecare rundă constă în aplicarea tuturor celor 4 transformări şi există 10 runde la chei 128 de biţi, 12 la 192 şi 14 la 256. Runda iniţială constă doar în adăugarea cheii de rundă iar cea finală nu are transformarea MixColumns. Pentru detalii poate fi consultat FIPS-197.

```
AES_Encrypt_Round(State, Key)
{
    SubBytes(State) ;
    ShiftRows(State);
    MixColumns(State);
    AddRoundKey(State, Key);
}
```

```
AES_Decrypt_Round(State, Key)
{
    AddRoundKey-1(State, Key);
    MixColumns-1 (State);
    ShiftRows-1 (State);
    SubBytes-1 (State) ;
}
```

FIGURA 2.7. STRUCTURAINTUITIVĂ A UNEI RUNDE DE CRIPTARE ŞI DECRIPTARE AES

Totuşi trebuie să precizăm că singura suspiciune cu privire la securitatea AES-ului este faptul că foloseşte un design destul de non-conformist, spre deosebire de schemele simetrice clasice, care se construiesc pe reţea Feistel. Acest design nu a fost sub atenţia comunităţii criptologilor decât în ultimii ani, de la propunerea AES-ului. În mod spectaculos, transformarea AES (Rijndael) este echivalentă cu o ecuaţie algebrică destul de simplă (comparativ cu alte coduri) faţă de care există suspiciunea că ar putea duce în viitor la o serie de atacuri. Până în prezent nu a apărut însă nici un atac spectaculos asupra acestui procedeu

constructiv! Deci orice suspiciune nu are un fundament ştiinţific momentan. Pe de altă parte o reţea Feistel este o reţea bine studiată şi un cod construit pe reţea Feistel este puţin probabil să aducă surprize în ceea ce priveşte securitatea. Ca alternativă la Rijndael, poate fi utilizat oricare alt candidat la AES, dar sunt necesare motive serioase pentru a utiliza altceva în practică. Un bun contracandidat este codul Serpent, care nu este acoperit de nici un patent şi poate fi utilizat gratuit în soluţii contemporane de securitate (desigur acesta este mai lent decât AES) şi este construit pe structură Feistel.

2.9 MODURI DE OPERARE A CRIPTĂRILOR SIMETRICE: ECB, CBC CM ŞI ALTELE

Simpla existenţă a unui cod bloc, oricât de sigur, nu este o garanţie suficientă pentru securitatea unui criptotext. Aici intră în discuţie modurile de funcţionare, adică cum se aplică transformarea (codul bloc) asupra blocurilor de mesaj.

Cel mai simplu mod de aplicare al criptării simetrice, **numit electronic codebook (ECB)**, este nesigur. Acesta constă aşa cum sugerează şi Figura 2.8 şi Figura 2.9 în extinderea mesajului prin **padding** la un număr de biţi multiplu de dimensiunea blocului de criptare şi prin aplicarea transformării bloc cu bloc (cele trei careuri colorate ale ultimului bloc de mesaj sugerează paddingul). Chiar dacă funcţia de criptare a blocului este sigură, rezultatul poate fi în anumite cazuri o criptare nesigură. Aceasta deoarece acelaşi bloc de text se va cripta întotdeauna în acelaşi bloc de criptotext, rezultatul fiind aşadar predictibil.

Pentru a evita această problemă de securitate, modul de funcţionare **cipher block chaining (CBC)** foloseşte ieşirea blocului anterior la intrarea blocului ce urmează a fi criptat făcând un XOR cu acesta. Astfel, rezultatul fiecărui bloc se propagă şi influenţează până la ultimul bloc rezultatul criptării. Avantajul este că criptotextul va arăta perfect aleator, fără repetiţii cauzate de repetarea intrării. Dezavantajul este că pierderea unui bloc duce la imposibilitatea de a decripta blocurile următoare, chiar dacă acestea sunt corect recepţionate. Pentru criptarea primului bloc se face XOR cu un **vector de iniţializare IV (initialization vector)** care este o valoare random dar nu secretă ce joacă rol de bloc criptat anterior primului bloc. Criptarea şi decriptarea CBC sunt sugerate în Figurile 2.10 şi 2.11.

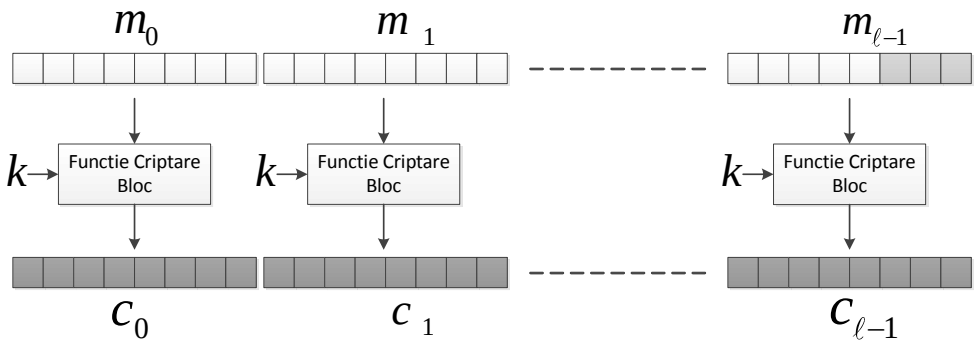


FIGURA 2.8.CRIPTAREA ÎN MODUL DE FUNCŢIONARE ELECTRONIC CODE BOOK (ECB)

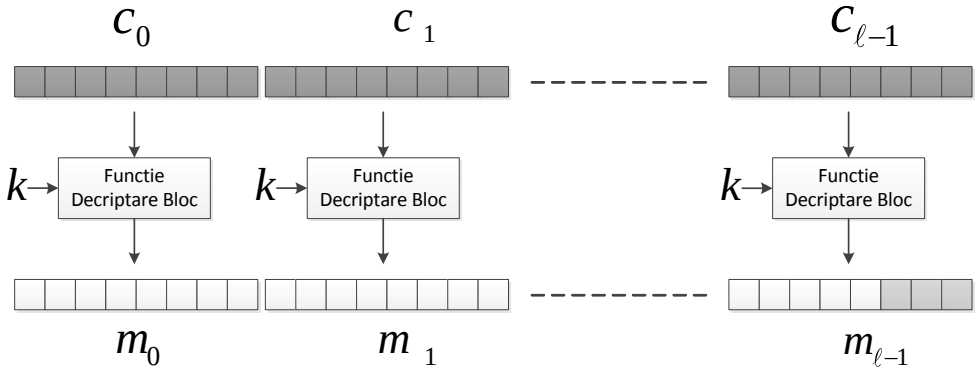


FIGURA 2.9. DECRIPTAREA ÎN MODUL DE FUNCŢIONARE ELECTRONIC CODE BOOK (ECB)

Există diverse variaţiuni pentru înlănţuirea blocurilor. Dintre acestea, **cipher feed-back (CFB)** criptează IV-ul în loc de bloc de mesaj cu care se face XOR şi se propagă rezultatul mai departe. În mod similar lucrează **output feed-back (OFB)** cu diferenţa că se propagă mai departe doar rezultatul IV-ului criptat nu rezultatul criptării în sine. Modurile CFB şi OFB au avantajul de a transforma codul bloc într-un cod stream unde procesarea criptării se face bit cu bit prin operatorul XOR. Avantajul lui OFB faţă de CFB este faptul că permite procesarea stream-ului de cheie chiar dacă mesajul nu este încă disponibil, deci permite

procesarea în avans ceea ce oferă avantaje computaţionale. OFB are şi avantajul că permite decriptarea chiar dacă se pierd din blocurile criptate. O altă variantă este **Propagating Cipher Block Chaining (PCBC)** care propagă mai departe atât textul de la intrare cât şi criptotextul rezultat din fiecare bloc. Criptarea OFB şi CFB sunt sugerate în Figurile 2.12 şi 2.13 iar criptarea şi decriptarea PCBC în Figurile 2.14 şi 2.15.

Foarte întâlnit în practică şi cu un nivel de securitate crescut (fiind tolerant la pierderea blocurilor intermediare) este **Counter Mode (CM)** care foloseşte un **counter**, de obicei concatenat cu o valoare aleatoare numită **salt**. Counterul este criptat şi apoi se face XOR cu textul ce se doreşte criptat. Dezavantajul este necesitatea de a păstra un counter sincron de partea celor care efectuează criptarea respectiv decriptarea. Criptarea în modul counter este sugerată în Figura 2.16.

De amintit şi varianta **cipher-text stealing** permite construirea unui criptotext care are lungime egală cu mesajul criptat (cu condiţia ca acesta să necesite minim 2 blocuri). Această tehnică este utilă deoarece de cele mai multe ori textul nu are lungime multiplu de dimensiunea blocului criptat şi în unele medii este necesară economisirea pe cât se poate a volumului de date transmis (de exemplu în reţele de senzori).

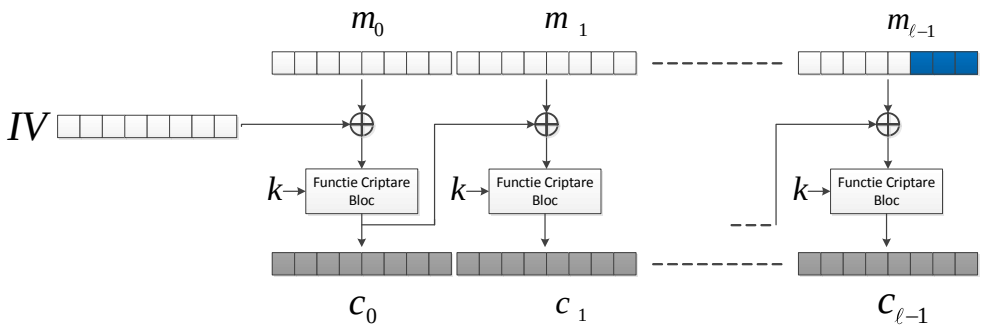


FIGURA 2.10. CRIPTAREA ÎN MODUL DE FUNCŢIONARE CIPHER BLOCK CHAINING (CBC)

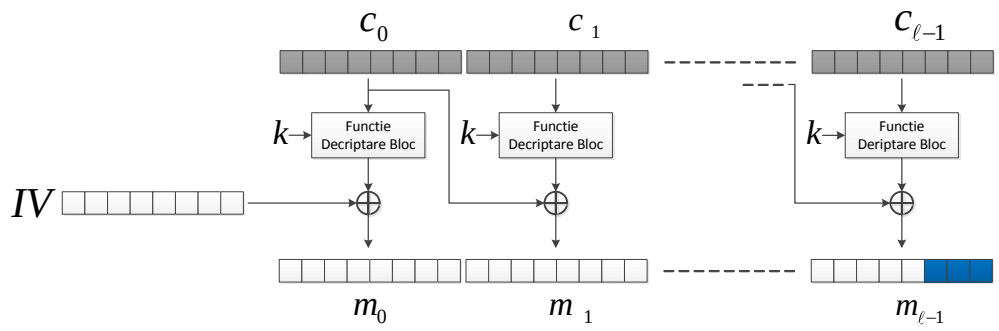


FIGURA 2.11. DECRIPTAREA ÎN MODUL DE FUNCŢIONARE CIPHER BLOCK CHAINING (CBC)

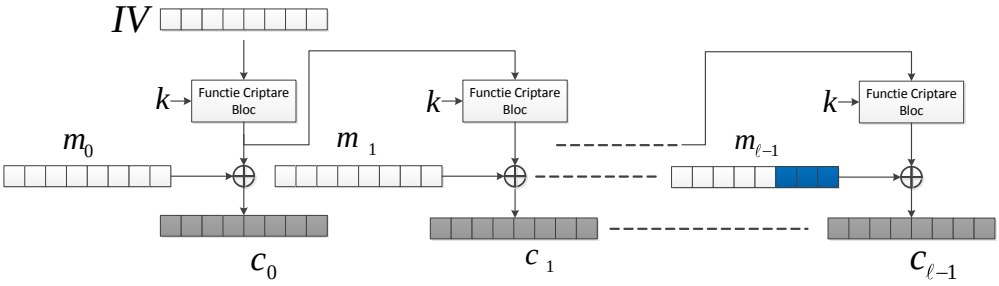


FIGURA 2.12. CRIPTAREA ÎN MODUL DE OUTPUT FEEDBACK BLOCK CHAINING (OFB)

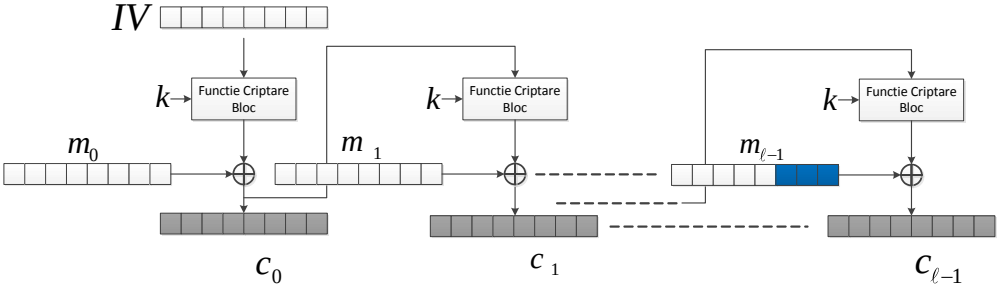


FIGURA 2.13. CRIPTAREA ÎN MODUL DE FUNCȚIONARE CIPHER FEEDBACK (CFB)

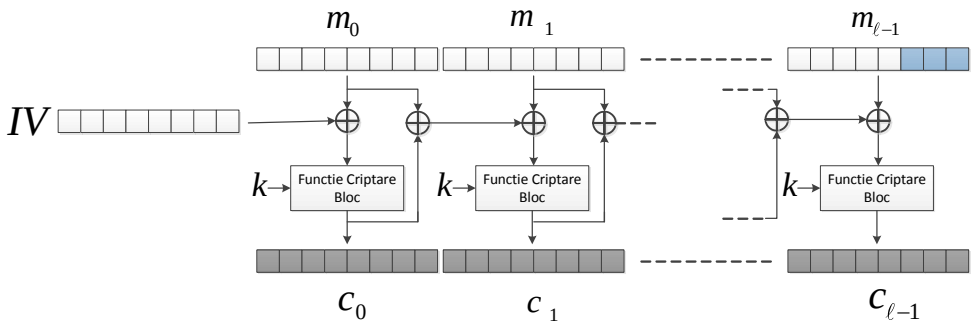


FIGURA 2.14. CRIPTAREA ÎN MODUL DE FUNCŢIONARE PROPAGATING CIPHER BLOCK CHAINING (PCBC)

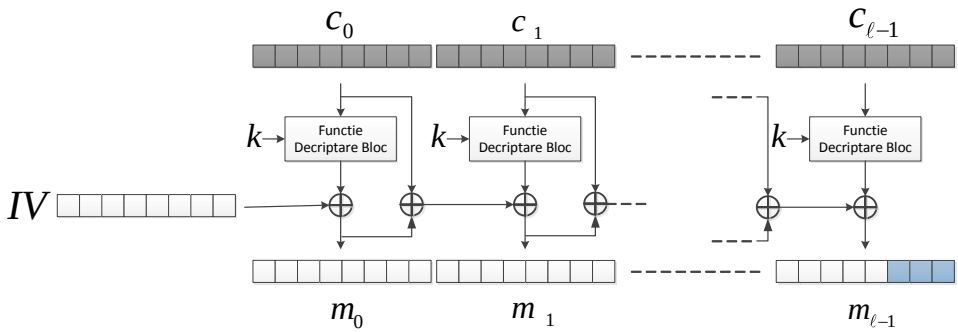


FIGURA 2.15. DECRIPTAREA ÎN MODUL DE FUNCŢIONARE PROPAGATING CIPHER BLOCK CHAINING (PCBC)

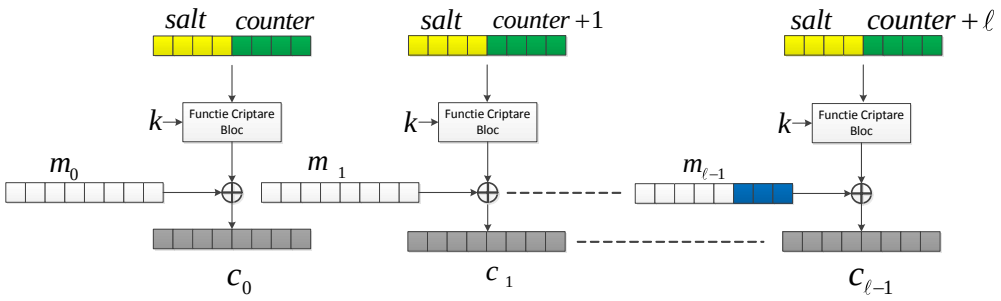


FIGURA 2.16. CRIPTAREA ÎN MODUL DE FUNCŢIONARE COUNTER MODE (CM)

2.10 TIPURI DE ATAC ASUPRA FUNCȚIILOR DE CRIPTARE

Am stabilit că criptanaliza este domeniul care se ocupă de studiul atacurilor asupra funcţiilor criptografice. În principiu atacul unei funcţii criptografice are la bază exploatarea unei vulnerabilităţi ce se datorează în general unei proprietăţi matematice a codului care nu a fost luată în calcul în procesul de proiectare.

Este în general utilă remarca că printr-un atac criptanalitic se urmăreşte fie recuperarea unui mesaj criptat fie a cheii utilizate (subliniem că algoritmul de criptare este bine cunoscut de adversar, în prezent iese din discuţie securitatea obţinută prin obscuritatea algoritmului). Această remarcă oferă însă mai degrabă o perspectivă idealizată asupra unui atac criptanalitic. A recupera întregul mesaj criptat este o condiţie mult prea restrictivă, în realitate criptosistemul poate fi spart mult mai uşor. Astfel, este unanim recunoscut că în cele mai multe cazuri practice recuperarea unui singur bit din mesaj poate avea consecinţe dezastroase asupra securităţii. În acest context este necesară definirea unor obiective de securitate şi atacuri avansate, mult mai complexe decât recuperarea unei chei sau a unui mesaj (acest lucru va fi făcut în relaţie cu criptosistemele asimetrice într-un capitol următor).

Totodată atacurile trebuie gândite ca fiind cauzate de adversari care au acces la maşina de criptare sau decriptare, deoarece în practică aceasta este situaţia generală. De exemplu, în cazul criptosistemelor cu cheie publică oricine are acces la cheia de criptare (maşina de criptare fiind disponibilă în mod nerestrictiv) iar semnarea digitală se face pe baza cheii secrete (astfel maşina de decriptare este expusă prin orice semnătură efectuată). Lucrurile nu stau diferit nici în cazul criptosistemelor simetrice, deoarece în cazul maşinii Enigma războiul a oferit prilejul de a captura maşini de criptare sau decriptare, accesul fiind astfel posibil.

Convenim astfel să clasificăm atacurile după cum urmează în funcţie de accesul la maşina de criptare sau decriptare:

i) **Mesaj ales** (chosen-plaintext attack) – adversarul are acces pe o perioadă fixă de timp la maşina de criptare care acceptă să cripteze orice mesaj (până la momentul la care primeşte un anume criptotext pe care trebuie să îl spargă).

ii) **Mesaj ales adaptiv** (adaptive chosen-plaintext) - adversarul are acces nelimitat la maşina de criptare care acceptă să cripteze orice mesaj.

iii) **Criptotext ales** (chosen cipheretext) – adversarul are acces pe o perioadă fixă de timp la maşina de decriptare care acceptă să de cripteze orice mesaj (până la momentul la care primeşte un anume criptotext pe care trebuie să îl spargă).

iv) **Criptotext ales adaptiv** (Adaptive chosen-chipertext) - adversarul are acces nelimitat la maşina de decriptare care acceptă să cripteze orice mesaj.

În atacurile adaptive se presupune că adversarul are acces nelimitat la maşina criptografică cu o singură restricţie: atât adversarul cât şi maşina cunosc un anume criptotext ţintă, iar maşina criptografică refuză să opereze asupra acestuia, în schimb este dispusă să lucreze pentru adversar asupra oricărui alt criptotext.

Atacurile în care adversarul nu are acces la maşina de criptare/decriptare, sunt şi ele de două tipuri: **doar criptotext cunoscut** (ciphertext only), în care adversarul are la dispoziţie doar criptotextul, şi **mesaj cunoscut** (known-plaintext) în care adversarul cunoaşte şi criptotextul şi mesajul şi doreşte recuperarea cheii. Acestea sunt considerate modele de atac învechite deoarece în practică un adversar se bucură de premize mai bune de atât. Rezistenţa în faţa acestora este o cerinţă elementară.

Importanţa atacurilor cu acces la maşina de decriptare va fi scoasă în lumină prin discuţiile cu privire la adversari activi ai criptosistemelor asimetrice într-un capitol următor.

3 FUNCŢII CRIPTOGRAFICE FĂRĂ CHEIE: FUNCŢII HASH

Discutăm în acest capitol funcţiile hash, un obiect criptografic extrem de simplu şi totuşi greu de definit şi construit. Într-adevăr nu demult acestea au fost numite „gaura neagră” a criptografiei deoarece nu era foarte clar (şi poate nici nu este încă) ce proprietăţi trebuie să îndeplinească şi cum se construiesc corect. Trebuie spus că o bună parte din îngrijorarea cauzată de funcţiile hash a venit datorită atacurilor lui Wang asupra MD5 şi SHA1 [85].

3.1 DEFINIŢIE ŞI PROPRIETĂŢI

O **funcţie hash** este o funcţie care primeşte ca intrare mesaje de dimensiune variabilă şi returnează un mesaj de lungime fixă din care mesajul iniţial nu poate fi recuperat. Funcţiile hash nu folosesc nici un fel de cheie, şi le vom nota cu $H(m)$ reprezentând funcţia hash aplicată mesajului m. Figura 3.1 prezintă schema bloc a unei funcţii hash. Ieşirea unei funcţii hash se mai numeşte şi **etichetă** (**tag**).

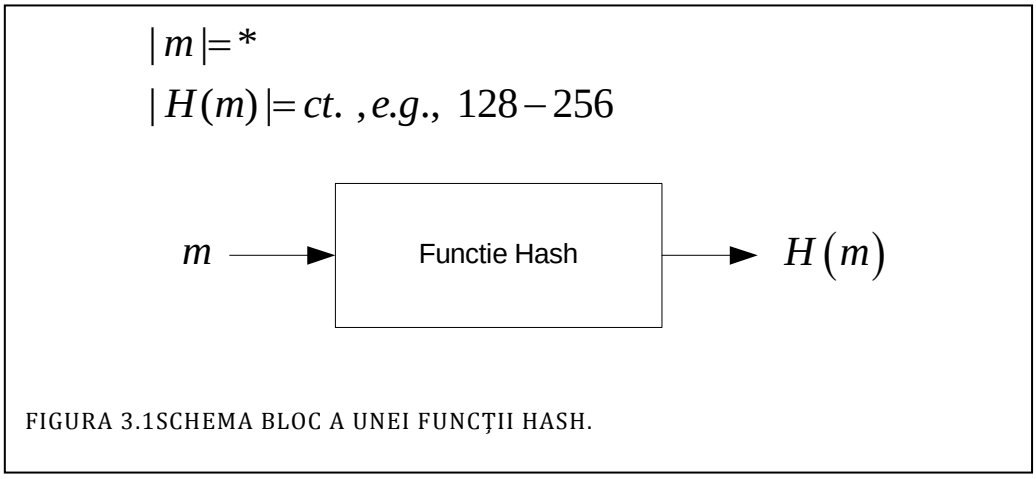


FIGURA 3.1SCHEMA BLOC A UNEI FUNCŢII HASH.

O funcţie hash trebuie să răspundă la următoarele **proprietăţi de securitate**: i) **rezistenţa imaginii** (preimage resistance): având y o ieşire a funcţiei nu se poate găsi x astfel încât $y = H(x)$, ii) **rezistenţă secundară a imaginii** (secondary preimage resistance) având $x, H(x)$ nu se poate găsi x' astfel încât $H(x) = H(x')$ şi iii) **rezistenţă la coliziune** (collision resistance) nu se poate găsi o pereche x, x' astfel încât $H(x) = H(x')$.

Din punct de vedere computaţional, proprietatea de bază a unei funcţii hash este eficienţa. Scopul este de a construi funcţii cât mai simplu de implementat (cod compact) şi cât mai rapide. În timp ce dimensiunea intrării poate fi oricât, la ieşire tagul are in general 128-256 biţi, mai rar până în 512 biţi.

3.2 FUNCŢII HASH FRECVENT UTILIZATE ÎN PRACTICĂ: MD5 ŞI FAMILIA SHA

Standardul curent (încă) şi probabil cea mai utilizată gamă de funcţii hash este familia SHA2 (Secure Hash Algorithm) [37] pentru care dimensiunea ieşirii este 224, 256, 384, 512 biţi indiferent de dimensiunea datelor de intrare. Varianta mai veche de funcţie din familia SHA, şi încă folosită, este SHA-1. Trebuie spus că în practică este încă foarte frecventă funcţia MD5 (posibil chiar mai utilizată decât SHA2).

Menţionăm însă faptul că funcţiile hash MD5 [68] şi SHA-1 nu mai oferă rezistenţă secundară a imaginii şi sunt deci nesigure (în ciuda acestui fapt ele sunt încă folosite în multe aplicaţii datorită eficienţei în special). O discuţie relevantă cu privire la ce implicaţii are pierderea rezistenţei secundare a imaginii poate fi găsită în [59]. De remarcat că nu orice categorie de aplicaţii este pusă în pericol de aceasta şi utilizarea lor este încă posibilă dar trebuie făcută cu precauţie. Atacuri asupra SHA au fost anunţate pentru prima oară în [85] două articole non-tehnice ale lui Schneier cu privire la atacurile asupra funcţiilor hash sunt în [74], [75]. Pentru implementări contemporane se recomandă desigur folosirea SHA-256 şi nicidecum a MD5 sau SHA-1.

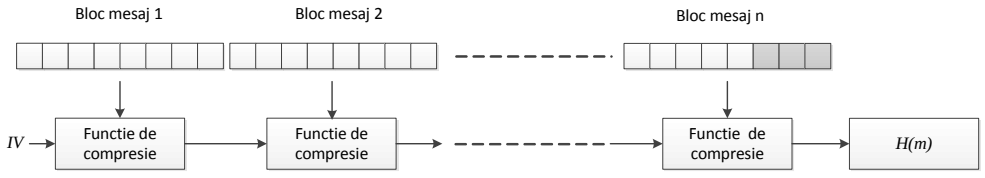


FIGURA 3.2. CONSTRUCTIA MERKLE-DAMGARD A UNEI FUNCTII HASH

În practică, pentru construcţia funcţiilor hash se foloseşte conceptul de funcţie hash iterată. Aceasta presupune spargerea intrării în blocuri de dimensiune fixă care sunt trecute printr-o funcţie de compresie în cadrul căreia se efectuează operaţii specifice. Una dintre metodele specifice de construcţie este construcţia Merkle-Damgard (descrisă de Merkle în 1979) pe care o prezentăm în Figura 3.2.

MD5 a fost construit de Ron Rivest în 1991 şi urmează construcţia Merkle-Damgard. MD5 operează cu blocuri de mesaj de câte 512 biţi. Mesajul iniţial se concatenează cu un bit de 1 şi apoi cu numărul necesar de 0-uri. Ultimii 64 de biţi din mesajul preprocesat reprezintă lungimea mesajului iniţial.

Algoritmul MD5 constă în 64 de iteraţii, grupate în 4 runde de câte 16 iteraţii, în fiecare rundă folosindu-se una din 4 funcţii neliniare de mai jos (4x16=64). Figura 3.3 descrie această structură. Fiecare rundă se efectuează de 16 ori deoarece blocul procesat este de 512 biţi, în timp ce funcţia de rundă procesează 32 de biţi la un moment dat (16x32=512). În figură se observă 4 blocuri de stare (*A*, *B*, *C*, *D*), acestea au câte 32 de biţi. Cele 4 constante predefinite de câte 32 de biţi joacă rol de vector de iniţializare (IV) şi sunt:

A= 0x67452301,
B= 0xefcdab89,
C= 0x98badcfe,
D= 0x10325476.

Funcţia de rundă are la bază una din cele 4 funcţii definite după cum urmează:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z),$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z),$$
$$H(X,Y,Z) = X \oplus Y \oplus Z,$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z).$$

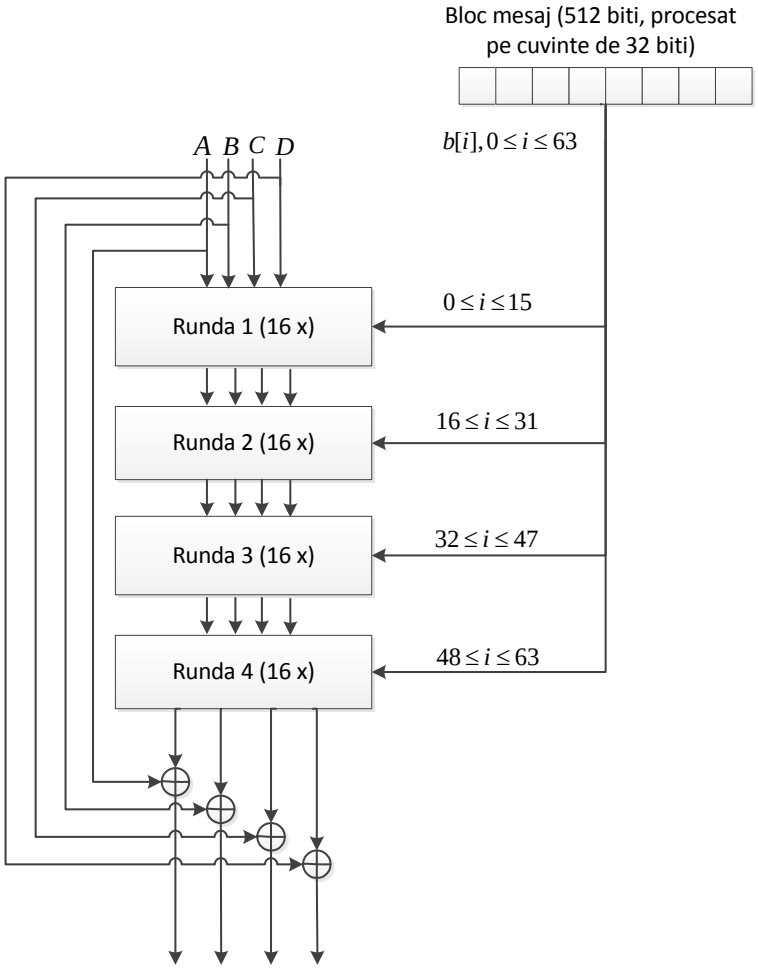


FIGURA 3.3. STRUCTURA ITERATĂ A LUI MD5

Toate aceste funcţii sunt neliniare şi se bazează pe operaţii simple la nivel de bit. În cadrul fiecărei runde, folosind una din cele 4 funcţii anterior definite se efectuează aceeaşi operaţie de 16 ori, operaţie ce constă în: $B+((A+FR(B,C,D)+M+K)<<<s)$ şi rezultatul se depune în *B*. Valorile de stare la sfârşitul fiecărei se interschimbă după cum urmează:

$$\begin{aligned} &D \leftarrow C,\\ &C \leftarrow B,\\ &B \leftarrow B+((A+FR(B,C,D)+M+K)<<<S),\\ &A \leftarrow D. \end{aligned}$$

Prin FR am desemnat funcţia de rundă (care este *F* în runda 1, *G* în runda 2, *H* în runda 3 şi *I* în runda 4), M este un bloc de 32 de biţi al mesajului (observăm că funcţia de rundă e pe 32 de biţi) iar *K* şi *S* sunt valori numerice predefinite (se poate consulta RFC1321 pentru aceste valori, aici interesează doar la nivel conceptual construcţia lui MD5). În Figura 3.4 se prezintă câţiva vectori de test pentru MD5 conform cu RFC 1321 (se observă că rezultatul are întotdeauna 128 de biţi).

SHA2 este construit pe principii similare. În acest cazul există tot 4 funcţii neliniare cu 3 intrări, dar de data aceasta sunt mai complexe şi se folosesc 8 blocuri de stare (*A, B, C, D, E, F, G, H*) de câte 32 de biţi. În mod concret, pentru SHA-256 cele 4 funcţii neliniare sunt:

$$\begin{aligned} &Ch(E,F,G)=(E\wedge F)\oplus(\neg E\wedge G),\\ &Ma(A,B,C)=(A\wedge B)\oplus(A\wedge C)\oplus(B\wedge C),\\ &\Sigma_0(A)=(A>>>2)\oplus(A>>>13)\oplus(A>>>22),\\ &\Sigma_1(E)=(E>>>6)\oplus(E>>>11)\oplus(E>>>25). \end{aligned}$$

```
MD5 ("")           = d41d8cd98f00b204e9800998ecf8427e
MD5 ("a")          = 0cc175b9c0f1b6a831c399e269772661
MD5 ("abc")        = 900150983cd24fb0d6963f7d28e17f72
MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0
MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b
MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789")
                                   = d174ab98d277d9f5a5611c2c9f419d9f
MD5 ("12345678901234567890123456789012345678901234567890123456...234567890")
                                   = 57edf4a22be3c955ac49da2e2107b67a
```

FIGURA 3.4. EXEMPLE DE VECTORI DE TEST PENTRU MD5 CONFORM RFC 1321

SHA2 foloseşte 64 de runde în cazul lui SHA-256 şi SHA-224 şi 80 de runde în cazul lui SHA-384 şi SHA-512 (din acest motiv se poate observa experimental că SHA-384 şi SHA-512 sunt la fel rapide). Detaliile se găsesc în standardul FIPS 180-2.

3.3 NOUA GENERAŢIE DE FUNCŢII HASH SHA3

În anul 2008 a fost deschis concursul pentru SHA3. În prezent există 5 finalişti: BLAKE, Groestl, Skein, Keccak şi JH. Este greu de prezis câştigătorul, dar se pot face câteva remarci despre acestea. JH este deja subiect al unor atacuri şi deci nu va avea şanse de câştig. BLAKE este se pare cea mai eficientă computaţional şi uşor de implementat. Varianta cu output de 256 biţi operează pe 32 de biţi iar cea cu output de 512 biţi pe 64. Permite ca output dimensiuni de 224, 256, 384 sau 512 biţi la fel ca SHA2. Groestl foloseşte aceeaşi casetă de substituţie (S-Box) ca şi AES. Are un output de 256 sau 512 biţi. Keccak pare să fie în implementare hardware cel mai rapid dintre finalişti, între designeri este şi John Daemen (autor al AES). Aceleaşi dimensiuni de output ca în cazul SHA2 sunt valabile. Skein se bazează pe codul bloc Threefish şi permite dimensiune arbitrară la ieşire. Între autorii lui Skein este şi Bruce Schneier.

4 CODURI DE AUTENTIFICARE A MESAJELOR (HASH-URI CU CHEIE)

O gamă aparte de sisteme criptografice sunt codurile de autentificare a mesajelor MAC. Partea interesantă vis-a-vis de acestea este că ele pot fi construite atât folosind funcţii hash, opţiunea mai uzuală, dar şi folosind scheme de criptare simetrică.

4.1 DEFINIŢIE ŞI PROPRIETĂŢI

Codurile de autentificare a mesajelor MAC (Message Authentication Codes), le notăm cu $MAC_K(m)$ ceea ce înseamnă cod de autentificare a mesajului m calculat cu cheia K. Indiferent de dimensiunea mesajului dimensiunea ieşirii funcţiei este constantă (de obicei dimensiunea cheii este egală cu dimensiunea ieşirii funcţiei). Codurile MAC se construiesc pe baza unei funcţii hash, în general se foloseşte MD5 sau SHA1, cu toate că ambele au un nivel de securitate destul de scăzut.

Rolul codurilor MAC este de a testa autenticitatea unei informaţii, deci pentru a verifica sursa de provenienţă a informaţiei, implicând astfel şi o garanţie asupra integrităţii. În practică se folosesc construcţiile, unanim recunoscute ca eficienţă şi securitate, HMAC şi NMAC propuse de Mihir Bellare, Ran Canetti şi Hugo Krawczyk în lucrarea [8]. Un cod de autentificare al mesajelor (MAC) îl definim după cum urmează:

Definiţia 4.1. (Cod de autentificare a mesajelor MAC) *Un cod de autentificare a mesajelor* MAC *constă în trei algoritmi: algoritmul de generare a cheii* $K \leftarrow MAC.Gen(1^k)$ *care primeşte ca intrare nivelul de securitate* k *şi returnează cheia* K*, algoritmul de etichetare* $MAC_K(m) \leftarrow MAC.Tag(m, K)$ *care primeşte mesajul* m *şi cheia* K *returnând eticheta* $\tau = MAC_K(m)$ *şi algoritmul de verificare* $MAC.Ver(m, K, \tau)$ *care primeşte mesajul, cheia şi eticheta returnând o valoare binară care este 1 dacă şi numai dacă eticheta corespunde perechii cheie-mesaj. Toate acestea alături de spaţiile din care provin datele de intrare ale acestora (care fără a pierde generalitatea sunt spaţii ale stringurilor binare).*

Figura 4.1 ilustrează schema bloc a algoritmului de generare a cheii care primeşte ca intrare parametrul de securitate 1^k (care reprezintă de fapt dimensiunea cheii şi deci nivelul de securitate) şi o valoare aleatoare (sau pseudo-aleatoare). Figura 4.2 ilustrează algoritmul de calcul pentru valoarea codului MAC (eticheta) aferent mesajului, valoare notată cu eticheta τ, iar figura 4.3 ilustrează schema bloc a algoritmului de verificare a acestuia, verificarea făcându-se prin compararea etichetei τ cu codul nou generat asupra mesajului $MAC_K(m)$. În Figura 4.4 sunt utilizaţi algoritmii din figurile 4.1, 4.2 şi 4.3 pentru a construi schema unui sistem criptografic de autentificare a mesajelor.

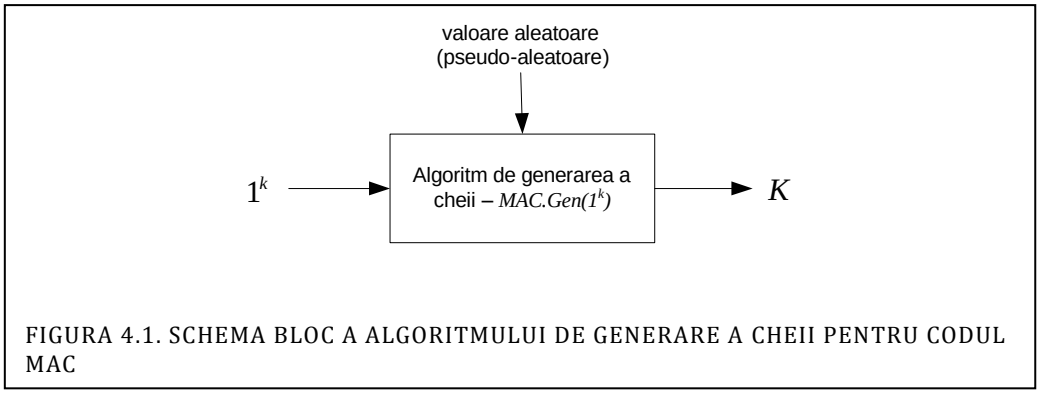


FIGURA 4.1. SCHEMA BLOC A ALGORITMULUI DE GENERARE A CHEII PENTRU CODUL MAC

.

$|m| = *$

$|K| = ct., e.g., 128$

$|MAC_K(M)| = ct., e.g., 128 - 256$

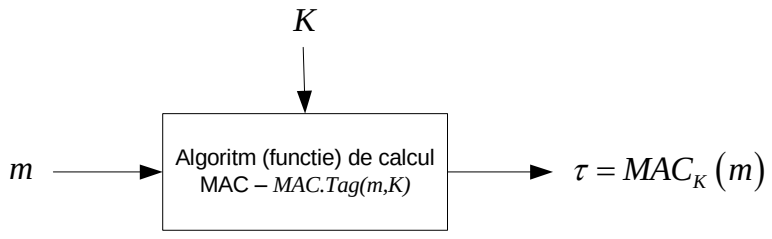


FIGURA 4.2. SCHEMA BLOC A UNUI ALGORITM DE CALCUL MAC.

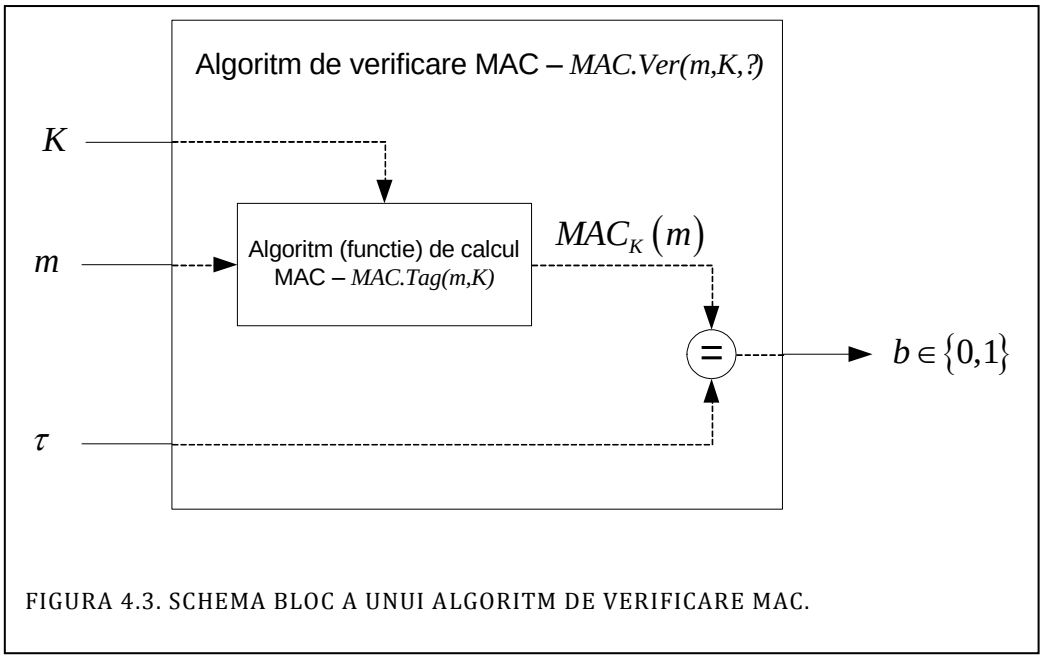


FIGURA 4.3. SCHEMA BLOC A UNUI ALGORITM DE VERIFICARE MAC.

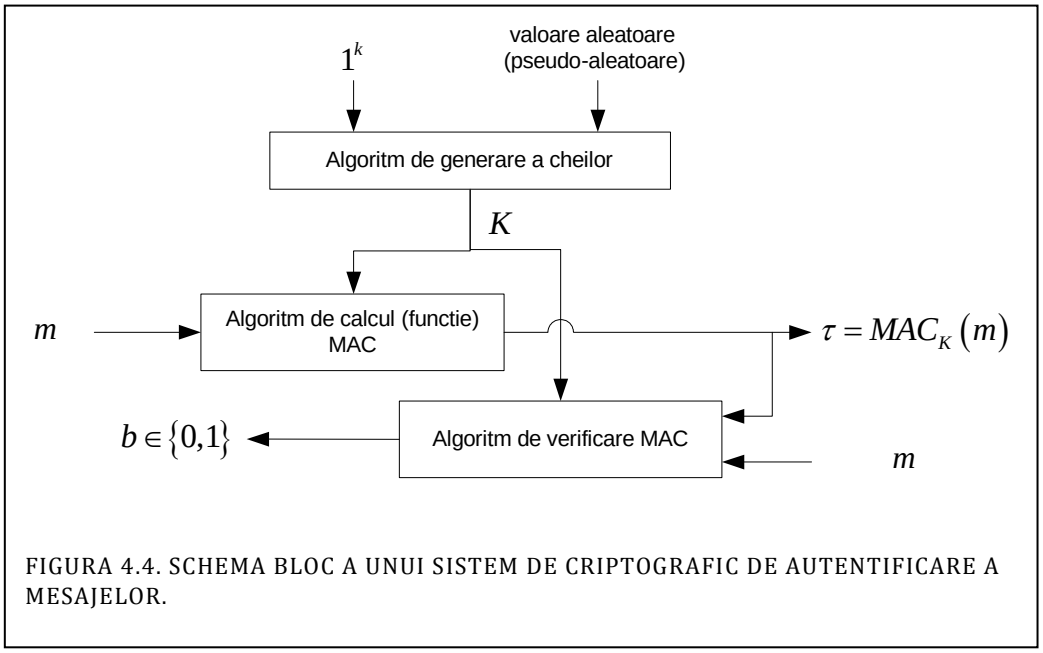


FIGURA 4.4. SCHEMA BLOC A UNUI SISTEM DE CRIPTOGRAFIC DE AUTENTIFICARE A MESAJELOR.

4.2 CODURI MAC ÎN PRACTICĂ: CBC-MAC ŞI H-MAC

Două construcţii de coduri MAC sunt de mare relevanţă practică. CBC-MAC permite construirea unui MAC folosind un simplu cod bloc. H-MAC se bazează pe o funcţie hash şi se remarcă prin simplitate şi eficienţă, lucru pentru care este cel mai comun în practică.

CBC-MAC se construieşte în manieră similară cu criptarea în modul CBC (Cipher Block Chaining) cu următoarele modificări: vectorul de iniţializare IV este setat pe 0 şi singura ieşire este blocul final al criptării (nu există ieşire pentru fiecare bloc de plain-text ca în cazul criptării). Figura 4.5 ilustrează această construcţie. De remarcat că aceasta construcţie este sigură doar pentru mesaje de dimensiune fixă. Pentru a remedia acest lucru şi a face schema sigură pentru mesaje de orice dimensiune, o soluţie este ca la începutul mesajului să fie concatenată dimensiunea sa, deci $m = |m| \,||\, m$.

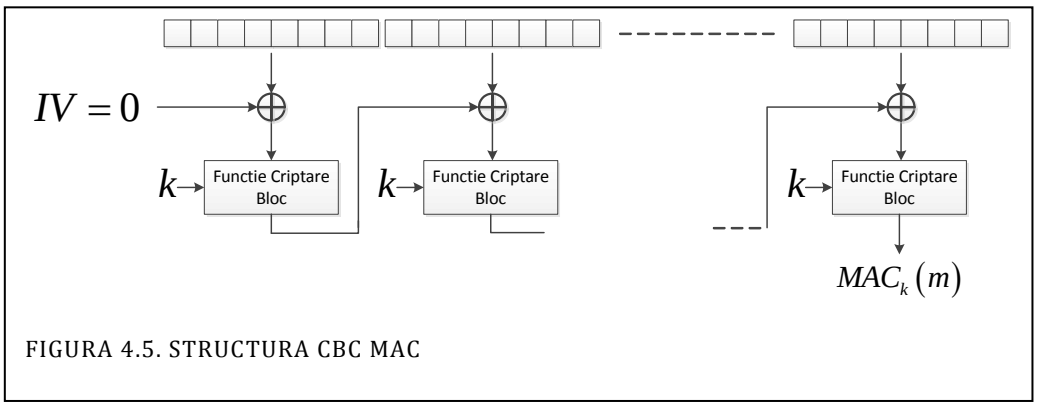


FIGURA 4.5. STRUCTURA CBC MAC

H-MAC constă în aplicarea de două ori a unei funcţii hash folosind de fiecare dată alt padding. Primul padding se numeşte **ipad (inner-padding)** şi cel de-al doilea **opad (outer-padding)** şi reprezintă valori numerice predefinite de dimensiunea blocului care îl procesează. Mai exact ipad este repetarea lui 0x36 şi opad a lui 0x5C de B-ori, unde B este dimensiunea în bytes a blocului procesat (de exemplu B=64 în cazul lui MD5 deoarece MD5 procesează blocuri de 512 biţi aşa cum am discutat anterior) . Practic HMAC constă în următoarea transformare:

$$HMAC(K, m) = H((K \oplus \text{opad}) \,||\, H((K \oplus \text{ipad}) \,||\, m)) .$$

O transformare simplă, frumoasă şi sigură. De remarcat faptul că intuiţia de a construi un cod MAC folosind un simplu hash cu cheie adică utilizarea $H(K \| m)$ conduce la o problemă de securitate datorată modului de calculare a hashului. Dacă este un hash iterat, aşa cum se întâmplă cu mai toate construcţiile practice, se pot falsifica MAC-uri prin concatenarea la mesajul iniţial de valori arbitrare (acest atac din nou nu funcţionează dacă toate mesajele are avea aceeaşi lungime).

Funcţia hash din cadrul H-MAC poate fi orice funcţie hash din practică. Din acest motiv în mediul de programare .NET unde avem funcţiile de hash SHA-256, SHA-384 şi SHA-512, vom avea şi funcţiile H-MAC aferente H-MAC-SHA-256, H-MAC-SHA-384 şi H-MAC-SHA-512, etc.

Desigur, mai există şi alte construcţii. De exemplu **N-MAC (Nested MAC)** care este la fel de eficient ca şi HMAC dar necesită modificarea IV-ului pentru funcţia hash, lucru care, deşi nu costisitor, îl face totuşi mai complicat decât H-MAC şi deci mai puţin frecvent în practică.

5 GENERATOARE DE NUMERE ALEATOARE ŞI PSEUDO-ALEATOARE

Generatoarele de numere aleatoare sunt un bloc constructiv indispensabil pentru criptografie deoarece securitatea unui criptosistem depinde în primul rând de calitatea cheilor cu care se efectuează criptarea (chei alese întotdeauna plecând de la valori aleatoare). Un generator de numere aleatoare este un dispozitiv hard sau soft care generează o secvenţă de numere care nu urmează nici un fel de model şi deci nu pot fi prezise.

În această secţiune nu dorim o prezentare de detaliu a unor astfel de funcţii, dar încercăm să prezentăm câteva alternative constructive. Generatoare de numere pseudo-aleatoare se găsesc implementate în orice mediu de programare dar ele trebuie atent alese pentru a asigura un nivel de securitate real pentru criptosistemele implementate. Dacă generatorul de numere aleatoare este slab, indiferent de rezistenţa teoretică a criptosistemului care utilizează cheile acesta va putea fi cu uşurinţă spart în practică (un caz tipic este generatorul utilizat de Netscape în primele variante de SSL).

Dificultatea practică nu stă neapărat în construcţia unor astfel de generatoare (se cunosc alternative eficiente şi sigure) ci mai degrabă în a defini ce proprietăţi trebuie să îndeplinească acestea. Mai exact, având un black-box care generează numere, cum putem stabili dacă acesta este sau nu un generator de numere aleatoare. De exemplu, având secvenţele „1, 7, 3" şi „3, 3, 3" care din cele două secvenţe este mai aleatoare? Ei bine, dacă în grabă cineva ar putea zice că prima este mai aleatoare ca a doua, de fapt, la ieşirea unui generator de numere aleatoare ambele pot fi obţinute iar o catalogare după doar 3 ieşiri este lipsită de fundament. Amintim în acest context şi paradoxul numit „eroarea jucătorului" (gambler's fallacy). Acesta poate fi ilustrat prin următoarea întrebare: dacă un jucător a dat cu zarul de 3 ori consecutiv 6, probabilitatea ca a 4-a oara să dea 6 este 1/6 sau mai mică? Răspunsul corect este 1/6 deoarece evenimentul celei de a 4-a aruncări cu zarul este independent de celelalte şi deci are tot probabilitatea 1/6. Sunt mulţi însă cei care cred că probabilitatea este mai mică a patra oară.

Există teste implementate pentru stabilirea calităţii unui generator de numere aleatoare. Cea mai cunoscută suită este DieHarder (http://www.phy.duke.edu/~rgb/General/dieharder.php). Pentru a stabili calitatea unui generator sunt necesare fişiere de zeci, sute de mega-bytes sau chiar mai mult. Problema reală este însă faptul că este posibil a construi un

generator de numere aleatoare care să treacă orice test şi totuşi să poată fi predictibil!

Există două tipuri de generatoare: hard şi soft. Generatoarele hard se numesc în general generatoare de numere aleatoare iar cele soft se numesc pseudo-aleatoare. Cele hardware se bazează pe dificultatea modelării (sau necunoaşterea exactă a modelului) procesului fizic din dispozitivul hard, ceea ce face ca ieşirea să nu poată fi prezisă. Cele software, sunt programe care pe baza unei valori de iniţializare (numită „seed") produc secvenţe de numere aparent aleatoare. Pentru scopuri practice, un generator software poate fi la fel de bun ca unul hardware! În general cele software sunt ieftine şi rapide, iar cele hardware mai lente dar mai sigure. Din punct de vedere algoritmic, dezideratul constructiv este de a construi un generator de numere aleatoare astfel încât să nu existe nici un algoritm în timp polinomial care având ieşirile generatorului de numere pseudo-aleatoare şi un generator de numere aleatoare să poată decide care este generatorul de numere pseudo-aleatoare cu probabilitate mai mare de ½ (proprietate care numeşte „computational indistinguishability").

5.1 GENERATORUL LINEAR CONGRUENŢIAL

Cel mai simplu şi eficient generator de numere aleatoare este **generatorul linear congruenţial** (linear congruential generator). Acesta se bazează pe generarea şirului recurent:

$$X_{i+1} = aX_i + c \bmod n$$

Unde a, c, n, sunt parametrii fixaţi şi X_0 este valoarea de iniţializare (seed). Acest generator este simplu de implementat dar nu este sigur din punct de vedere criptografic! Adică, prin calcule matematice pot fi aflaţi parametrii, având la dispoziţie suficiente valori de ieşire. Este însă un generator suficient de bun pentru scopuri non-criptografice. Perioada maximă a acestuia este n.

Un caz particular se obţine prin setarea lui $c = 0$ ceea ce duce la $X_{i+1} = aX_i \bmod n$. Acest generator se numeşte generatorul Lehmer (uneori referit şi ca generatorul Park-Miller). Alegeri uzuale pentru n sunt numere prime de forma $2^k - 1$ deoarece acestea conduc la o perioadă maximă (desigur dacă şi

ceilalţi parametrii sunt bine aleşi). Knuth oferă o analiză detaliată a acestui generator în [55].

5.2 GENERATOARE LFSR (FIBONACCI ŞI GALOIS)

Registrele de deplasare liniară **LFSR (Linear Feedback Shift Register)** reprezintă una dintre cele mai eficiente metode de generare a unor secvenţe pseudo-aleatoare. Atenţie, nici acestea nu sunt destul de sigure pentru scopuri criptografice. Dar, sunt simplu de implementat şi utile în diverse alte aplicaţii. Două exemple sunt codurile CRC şi, în criptografie, codurile de criptare stream.

Există două tipuri de registre: **LFSR standard** (numite şi **Fibonacci**) unde valorile din registre se însumează modulo 2 (XOR) pentru a obţine feed-backul şi **LFSR Galois** unde valorile se însumează modulo 2 (XOR) succesiv pentru a obţine valoarea din fiecare registru în parte. Figura 5.1 ilustrează cele două tipuri de registru. Cele două construcţii sunt echivalente, dar varianta Galois are avantajul că poate fi paralelizată deoarece se execută XOR independent la fiecare registru în timp ce în celălalt caz se execută XOR asupra tuturor valorilor.

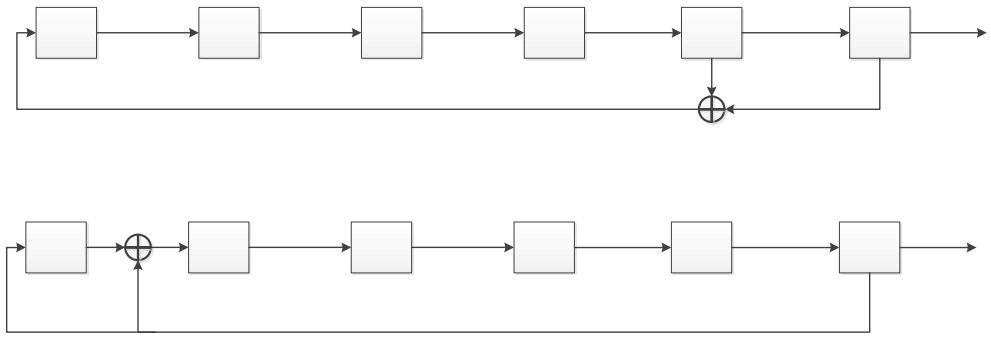


FIGURA 5.1. IMPLEMENTARE LFSR FIBONACCI (SUS) ŞI GALOIS (JOS) A POLINOMULUI $X^6 + X^5 + 1$

Aceste registre lucrează de fapt în câmpul finit F_{2^n} (câmp Galois, a se vedea capitolul de fundamente matematice) iar operaţiile efectuate pot fi interpretate ca operaţii binare asupra coeficienţilor unui polinom. Coeficienţii polinomului sunt 1 acolo unde există cablaj către XOR şi 0 altfel, vezi de exemplu Figura 5.1. pentru polinomul $x^6 + x^5 + 1$. Uşor de imaginat, dacă mai exista un termen în x^3 de exemplu mai apărea un XOR la două blocuri după cel aferent termenului în x^5.

În mod clar perioada maximă a unui astfel de generator este $2^n - 1$ şi această perioadă este atinsă pentru anumite polinoame numite **polinoame primitive** (polinoame ale căror rădăcini sunt generatori ai câmpului) indiferent de valoarea de iniţializare cu condiţia ca ea să fie nenulă. De exemplu pentru $x^4 + x^3 + 1$ perioada este 15 iar pentru $x^9 + x^5 + 1$ perioada este 511, etc.

Aşa cum am spus, LFSR reprezintă o soluţie nesigură din punct de vedere criptografic. Aceasta deoarece sunt algoritmi care pot afla coeficienţii polinomului pe baza ieşirii registrului şi deci pot în acest fel „sparge" generatorul RNG. O soluţie eficientă pentru aflarea coeficienţilor este **algoritmul Berlekamp-Massey**.

5.3 GENERATORUL BLUM-BLUM-SHUB

Cum am spus, generatorul anterior nu este sigur din punct de vedere criptografic. În mod clar utilizarea lui pentru chei criptografice conduce la sisteme nesigure. Un deziderat constructiv este construirea unui generator cu privire la care să se poată demonstra că prezicerea informaţiei de la ieşire este echivalentă cu rezolvarea unei probleme despre care se ştie că este imposibil de rezolvat în practică (adică care să deţină securitate demonstrabilă în sensul descris în capitolul introductiv).

Un exemplu ilustrativ în acest sens, şi totodată ilustrativ pentru utilizarea funcţiilor criptografice în construcţia de generatoare pseudo-aleatoare, este **generatorul Blum-Blum-Shub** (BBS) [14], [15] a cărui securitate este echivalentă cu dificultatea problemei factorizării întregilor, şi care utilizează funcţia ridicare la pătrat utilizată în multe criptosisteme cu cheie publică (de exemplu Rabin, vezi şi [46], [69] pentru aplicaţii).

Generatorul BBS se bazează pe calcularea şirului recurent:

$$X_i = X_{i-1}^{\,2} \bmod n,$$

Aici $n = p \cdot q$ este un întreg suficient de mare astfel încât factorizarea lui să nu poate fi uşor calculată şi X_0 este o valoarea de iniţializare aleasă aleator. La ieşirea generatorului nu se returnează valoarea X_i ci bitul de paritate al acesteia.

Ca exemplu artificial, cu numere foarte mici, pentru $n = 13 \cdot 17 = 221$, $X_0 = 100$ avem $X_1 = 55$, $X_2 = 152$, $X_3 = 120$, etc., ieşirea generatorului BBS este $paritate(X_0) = 1$, $paritate(X_1) = 1$, $paritate(X_2) = 1$, $paritate(X_3) = 0$, etc. Acest generator este lent dar este suficient de sigur pentru a fi utilizabil în scopuri criptografice.

5.4 GENERATOARE HARDWARE

Există suficient de multe surse hardware din care se pot extrage secvenţe aleatoare. Exemple relevante constituie: zgomotul unei diode, drift-ul oscilatoarelor, zgomotul termic, zgomotul atmosferic, etc. Uşor de implementat sunt primele două alternative, cea bazată pe zgomotul unui diode şi cea bazată pe drift-ul a două oscilatoare. Alternativa preferată de practică este zgomotul diodei deoarece conduce la performanţe mai bune. În prezenta carte suntem interesaţi de probleme fundamentale şi deci detaliile constructive ale unor astfel de generatoare nu prezintă interes, le-am amintit doar pentru a aduce o imagine completă în faţa cititorului.

6 SCHEME DE CRIPTARE CU CHEIE ASIMETRICĂ (CRIPTAREA CU CHEIE PUBLICĂ)

> „A private conversation can therefore be held between any two individuals regardless of whether they have ever communicated before. Each one sends messages to the other enciphered in the receiver public enciphering key and deciphers the messages he receives using his own secret deciphering key. We propose some techniques for developing public key cryptosystems, but the problem is still largely open (…) We note that neither public cryptosystems nor one-way authentication systems can unconditionally secure because the public information always determines the secret information uniquely among members of a finite set. With unlimited computation, problem could therefore be solved by a straightforward touch.” W. Diffie & M. Hellman[2].

Discutăm acum schemele de criptare cu cheie publică. Aceste construcţii au jucat rolul fundamental în apariţia criptografiei moderne, bazată pe o comunitate academică activă şi standarde deschise publicului larg. În continuare vom discuta cele mai relevante criptosisteme de la schimbul de cheie Diffie-Hellman şi RSA până la criptarea folosind curbe eliptice. Nu în ultimul rând discutăm şi atacuri ale adversarilor activi precum şi tehnici moderne de padding ce pot fi folosite pentru a spori securitatea acestor criptosisteme.

6.1 DEFINIŢIE ŞI PROPRIETĂŢI

În introducere a fost prezentat un scurt istoric al criptosistemelor cu cheie publică. Dorim acum să introducem formalismul necesar descrierii unui astfel de criptosistem. Criptarea asimetrică, o notăm similar cu cea simetrică, pentru

[2] Din lucrarea care lansează criptografia cu cheie publică „New directions in cryptography”.

claritate însă schimbăm cheia k cu PK (Public Key), astfel avem $c = E_{PK}(m)$ (aceasta semnificând criptarea cu cheia PK a mesajului. Decriptarea se face folosind cheia privată ca $m = D_{SK}(c)$ (în general se foloseşte noţiunea de cheie privată şi nu secretă dar pentru a evita redundanţa notaţiei vom scrie SK care trimite către Secret Key).

Criptarea cu cheie publică comparativ cu cea cu cheie secretă are ca dezavantaj viteza şi prezintă două avantaje majore: i) nu necesită schimbul prealabil de chei secrete, deci comunicaţia poate fi efectuată şi pe un canal nesigur fără să existe secrete partajate şi ii) numărul de chei partajate la comunicarea între n entităţi este minim (o cheie publică şi o cheie privată pentru fiecare entitate). Un criptosistem cu cheie publică îl definim după cum urmează:

Definiţia 6.1. (Schemă de criptare cu cheie publică)*. O schemă de criptare cu cheie publică (criptosistem cu cheie publică) constă în trei algoritmi: algoritmul de generare a cheilor* $(PK, SK) \leftarrow PKE.Gen(1^k)$ *care primeşte ca parametru nivelul de securitate* k *şi returnează perechea cheie publică-privată* (PK, SK)*, algoritmul de criptare* $PKE.Enc(m, PK)$ *care primeşte mesajul* m *şi cheia publică* PK *şi returnează criptotextul* $c = E_{PK}(m)$ *şi algoritmul de decriptare* $PKE.Dec(c, SK)$ *care primeşte criptotextul* c *şi cheia privată* SK *şi returnează mesajul* m*. Toate acestea alături de spaţiile din care provin datele de intrare ale acestora (care fără a pierde generalitatea sunt spaţii ale stringurilor binare).*

Pentru corectitudinea criptosistemului, se impune ca pentru fiecare pereche cheie publică-privată $\forall (PK, SK) \leftarrow PKE.Gen(1^k)$ şi pentru orice mesaj m, decriptarea aplicată criptării conduce tot timpul la mesajul original, adică $m \leftarrow PKE.Dec(PKE.Enc(m, PK), SK)$. Aceasta ne asigură că criptosistemul furnizează rezultate corecte şi întotdeauna decriptarea criptării unui mesaj returnează acelaşi mesaj.

Algoritmul de generare a cheii diferă de cel de la primitivele cu cheie simetrică prin faptul că returnează două chei distincte, acest lucru este sugerat în Figura 6.1. Figura 6.2 şi Figura 6.3 sunt ilustrative pentru algoritmii de criptare şi decriptare iar în Figura 6.4 sunt utilizaţi algoritmii din Figura 6.1, Figura 6.2 şi Figura 6.3 pentru a construi schema unui sistem de criptare cu cheie asimetrică.

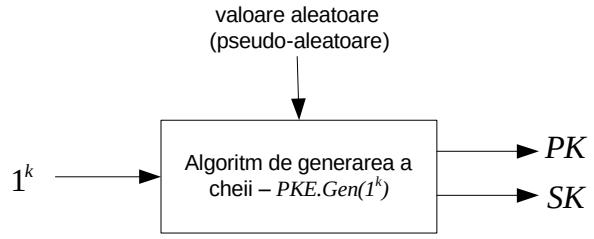


FIGURA 6.1. SCHEMA BLOC A ALGORITMULUI DE GENERARE A CHEII PENTRU CRIPTAREA ASIMETRICĂ.

Uzual:

$|m| < |PK|$

$PK = ct.\ , e.g.,\ 1024 - 4096$ (ECC de la 160)

$|E_{PK}(m)| = PK$

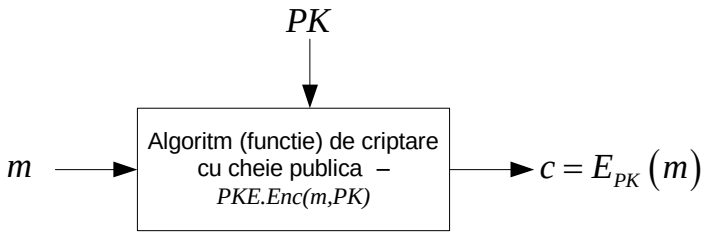


FIGURA 6.2. SCHEMA BLOC A UNEI FUNCŢII DE CRIPTARE CU CHEIE ASIMETRICĂ.

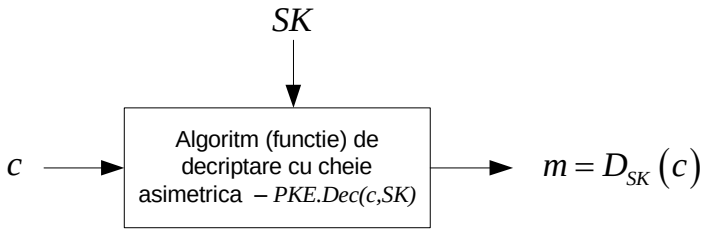


FIGURA 6.3. SCHEMA BLOC A UNEI FUNCŢII DE DECRIPTARE CU CHEIE ASIMETRICĂ.

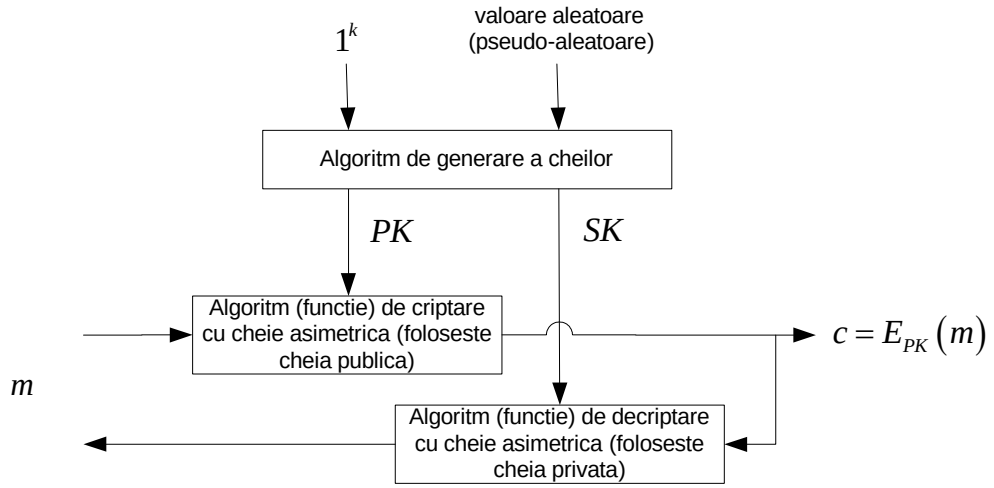


FIGURA 6.4. SCHEMA DE PRINCIPIU A UNUI SISTEM DE CRIPTARE CU CHEIE ASIMETRICĂ.

6.2 TAXONOMIE A CRIPTOSISTEMELOR CU CHEIE PUBLICĂ

Înainte de a trece la descrierea celor mai esenţiale criptosisteme cu cheie publică, în scopul creării unei imagini de ansamblu, considerăm utilă prezentarea unei taxonomii simplificate a acestora în Figura 6.5. Sistemele au fost clasificate în funcţie de problema de teoria numerelor pe baza căreia au fost dezvoltate. Subliniem că deşi criptografia cu cheie publică are mai bine de 30 de ani de existenţă, toate criptosistemele cu cheie publică relevante în practică se bazează fie pe problema factorizării întregilor fie pe problema logaritmului discret. Singura noutate este că aceasta din urmă a început să fie mai frecvent utilizată în ultimul deceniu pe grupurile curbelor eliptice. În taxonomia din Figura 6.5 linia continuă semnifică echivalenţa problemei cu problema de bază (factorizare sau logaritm discret) iar linia punctată semnifică faptul că o astfel de echivalenţă nu există sau nu a fost încă demonstrată.

Într-adevăr, există şi alte probleme pe care se pot construi criptosisteme cu cheie publică. Una dintre acestea, relevantă din punct de vedere istoric, este problema de optimizare combinatorică cunoscută sub numele de **suma unei submulţimi** (knapsack problem) sau problema rucsacului. Un criptosistem construit din aceasta a existat încă din 1978 fiind propus de Merkle şi Hellman, dar criptosistemul a fost prea ineficient pentru a avea interes practic. În ultimul deceniu, par a fi promiţătoare criptosistemele bazate pe latici. Sunt probleme pe care laticile le pot rezolva mai eficient decât alte construcţii, precum criptarea complet homomorfică (fully homomorphic encryption). Este posibil ca laticile să ofere criptosistemele viitorului. Momentan însă peisajul este dominat de criptosistemele bazate pe factorizare şi logaritmi discreţi.

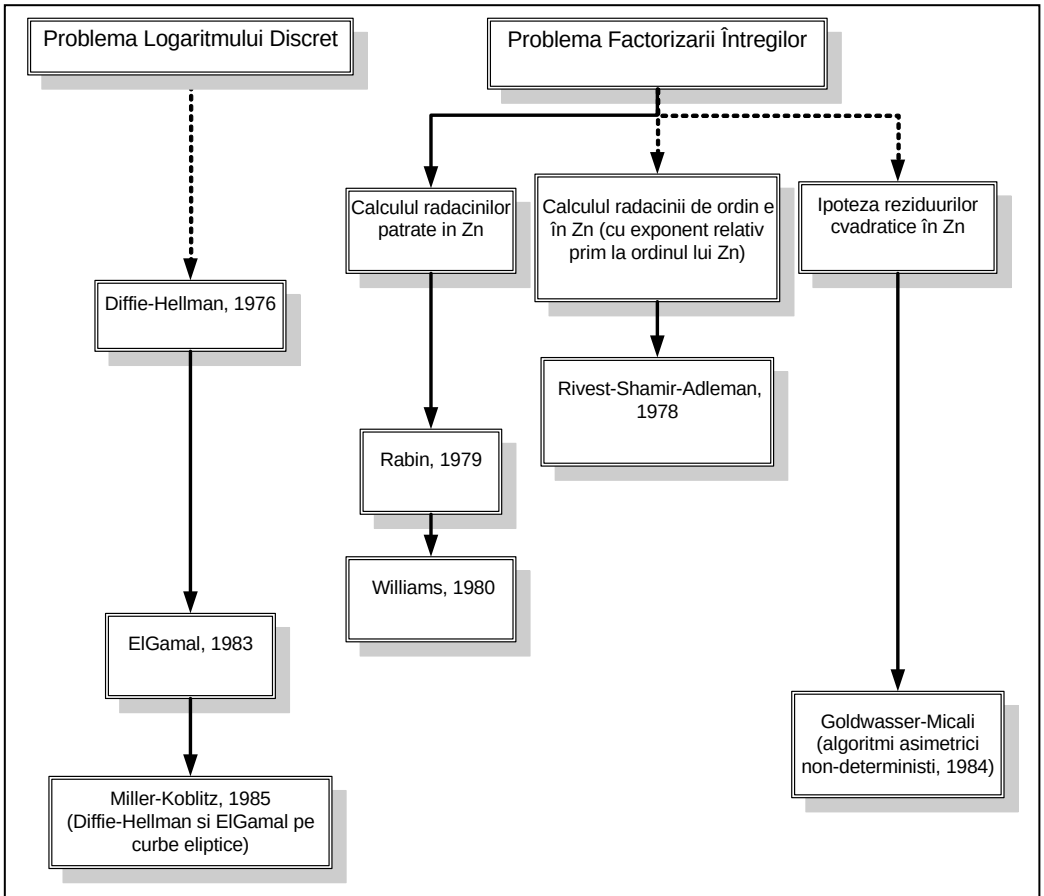


FIGURA 6.5. O TAXONOMIE A ALGORITMILOR DE CRIPTARE CU CHEIE PUBLICĂ.

6.3 SCHIMBUL DE CHEIE DIFFIE-HELLMAN-MERKLE

Schimbul de cheie Diffie-Hellman marchează începutul criptografiei cu cheie publică. Prin schimbarea bazată pe informaţii asimetrice a unei chei secrete care poate fi apoi utilizată pentru criptarea unei informaţii, se realizează în esenţă funcţionalitatea unui criptosistem cu cheie publică. De fapt criptosistemul ElGamal foloseşte exact paşii acestui schimb, aceeaşi idee urmând să apară sub formă de criptosistem în schema ElGamal discutată într-o secţiune următoare. De asemenea am adăugat numele lui Merkle în titlu, conform recomandării lui Hellman, bazat pe faptul că ideile lui Merkle ca pionier al criptografiei cu cheie publică au dus la construcţia acestui protocol.

Ideea pe care se bazează este faptul că operaţia de ridicare la putere în Z_p este comutativă, i.e. $\left(x^y\right)^z \equiv \left(x^z\right)^y \bmod p$, în timp ce **extragerea logaritmului discret** (care presupune găsirea lui y astfel încât $x^y \equiv a \bmod p$) nu este rezolvabilă eficient pentru elemente ale grupului de ordin foarte mare. Ce înseamnă ordinului unui element vom discuta detaliat în capitolul de fundamente matematice. Pentru moment, trebuie să ştim doar că **ordinul unui element** x este cel mai mic număr t pentru care $x^t \bmod p = 1$. În plus, spunem că x este **generator** al lui Z_p dacă ordinul său este $p-1$.

Considerăm pentru scenariul nostru doi participanţi A şi B care aleg un număr prim p şi un generator g al grupului Z_p^*, ambele informaţii fiind publice. Totodată A alege un umăr aleator a pe care îl păstrează ca informaţie privată (fără să îl anunţe pe B) respectiv B face acelaşi lucru alegând un număr aleator b. Schimbul de cheie constă în paşii ilustraţi în Protocolul 6-1.

La sfârşitul celor două sesiuni ambii participanţi pot calcula valoarea lui $g^{ab} \bmod p$. În mod evident A foloseşte valoarea lui $g^b \bmod p$ primită de la *B* şi valoarea lui a pe care o cunoaşte deoarece el a ales-o şi calculează $\left(g^b \bmod p\right)^a \bmod p$. Similar, B foloseşte $g^a \bmod p$, b şi calculează $\left(g^a \bmod p\right)^b \bmod p$. Din moment ce logaritmul discret nu poate fi extras, un potenţial adversar nu poate calcula nici pe a şi nici pe b fiind în imposibilitatea de a extrage cheia comună $g^{ab} \bmod p$. Subliniem că această cheie este aşadar secretă şi a fost obţinută doar pe baza unui schimb de informaţie publică. Ulterior

cheia secretă poate fi utilizată pentru orice altă operaţie criptografică, de exemplu pentru criptarea simetrică.

$DH.Setup\left(1^k\right)$: Fixează un număr prim p de k biţi şi un generator g al grupului Z_p^* ca parametri publici ai protocolului.

$DH.Exchange$: *A: alege un număr aleator a şi trimite*

1. $A \to B: g^a \bmod p$

B: alege un număr aleator b şi trimite

2. $B \to A: g^b \bmod p$

A şi B: calculează cheia comună ca $g^{ab} \bmod p$

PROTOCOLUL 6-1. SCHIMBUL DE CHEIE DIFFIE-HELLMAN

6.4 SECURITATEA SCHIMBULUI DE CHEIE DIFFIE-HELLMAN-MERKLE

Pe lângă **problema logaritmului discret** amintită anterior mai există două probleme relevante în strânsă legătură cu aceasta şi cu securitatea schimbului de cheie Diffie-Hellman. Le amintim în cele ce urmează.

Definiţia 6.2. (DDH - Decisional Diffie-Hellman). *Problema decizională Diffie-Hellman întreabă următoarele: având un număr prim p, un generator g al*

grupului Z_p^* *şi valorile* $g^a \bmod p$, $g^b \bmod p$ *respectiv un număr* r *să se decidă dacă* r *este sau nu chiar* $g^{ab} \bmod p$.

Definiţia 6.3. (CDH – Computational Diffie-Hellman). P*roblema computaţională Diffie-Hellman întreabă următoarele: având un număr prim* p *un generator* g *al grupului* Z_p^* *şi valorile* $g^a \bmod p$ *respectiv* $g^b \bmod p$ *să se calculeze* $g^{ab} \bmod p$.

Şi problema decizională (DDH) şi cea computaţională Diffie-Hellman (CDH) pot fi generalizate la fel ca problema logaritmului discret (DLP) introdusă pe diverse grupuri. Relaţia între cele trei probleme este următoarea:

$$DDH <_P CDH <_P DLP$$

Prin $P <_P Q$ notăm faptul că problema P se reduce la Q în timp polinomial, adică, dacă avem un algoritm care rezolvă P putem rezolva şi Q. Aşa cum se observă, problemele nu sunt echivalente. Problema echivalenţei între cele trei probleme rămâne în continuare deschisă (pe unele grupuri este demonstrată). În ceea ce priveşte securitatea schimbului de cheie Diffie-Hellman aceasta este echivalentă cu problema computaţională CDH şi nu cu cea a logaritmului discret.

Subliniem faptul că acest schimb de cheie este neautentificat şi un adversar poate interveni în schimb introducând alte valori. Aşadar, acest schimb nu este rezistent în faţa unui atac de tip **man-in-the-middle** (în care un adversar se interpune între cei doi participanţi) aşa cum se poate uşor observa în protocolul următor:

1. $A \to ...: g^a \bmod p$

   1". Adversarul interceptează şi trimite lui B $g^{adv} \bmod p$

   2. $B \to ...: g^b \bmod p$

      2". Adversarul interceptează şi trimite lui A $g^{adv} \bmod p$

La finalul protocolului *A* a schimbat cu adversarul cheia $g^{adv\cdot a} \bmod p$ iar *B* a schimbat cu adversarul cheia $g^{adv\cdot b} \bmod p$. În realitate deci, între *A* şi *B* nu este partajată nici un fel de cheie secretă. Astfel, utilizarea directă a acestui schimb de cheie nu oferă un canal sigur în practică.

6.5 CRIPTAREA ASIMETRICĂ RSA

RSA este prima realizare concretă de algoritm de criptare asimetrică şi semnătura digitală. Acest algoritm se bazează pe utilizarea pentru criptare a funcţiei $f(x) = x^{\varepsilon} \bmod n$ unde n este un întreg compozit produs a două numere prime iar ε este un exponent întreg care respectă $c.m.m.d.c.(\varepsilon, \phi(n)) = 1$. Această funcţie este o bijecţie şi admite ca inversă funcţia $f^{-1}(x) = x^{\delta} \bmod n$ care va fi utilizată la decriptare, δ este un întreg care satisface relaţia $\varepsilon\delta \equiv 1 \bmod \phi(n)$. Desigur că inversarea acestei funcţii este posibilă dacă se cunoaşte factorizarea lui n. Schema de principiu este ilustrată în Figura 6.6. iar Sistemul 6-1 este descrierea acestui criptosistem.Corectitudinea algoritmului poate fi uşor demonstrată observând că $\varepsilon\delta \equiv 1 \bmod \phi(n)$ implică $c^{\delta} \equiv m^{1+k\phi(n)} \equiv m \bmod n$, deoarece $m^{k\phi(n)} \equiv \left(m^{\phi(n)}\right)^{k} \equiv 1 \bmod n$.

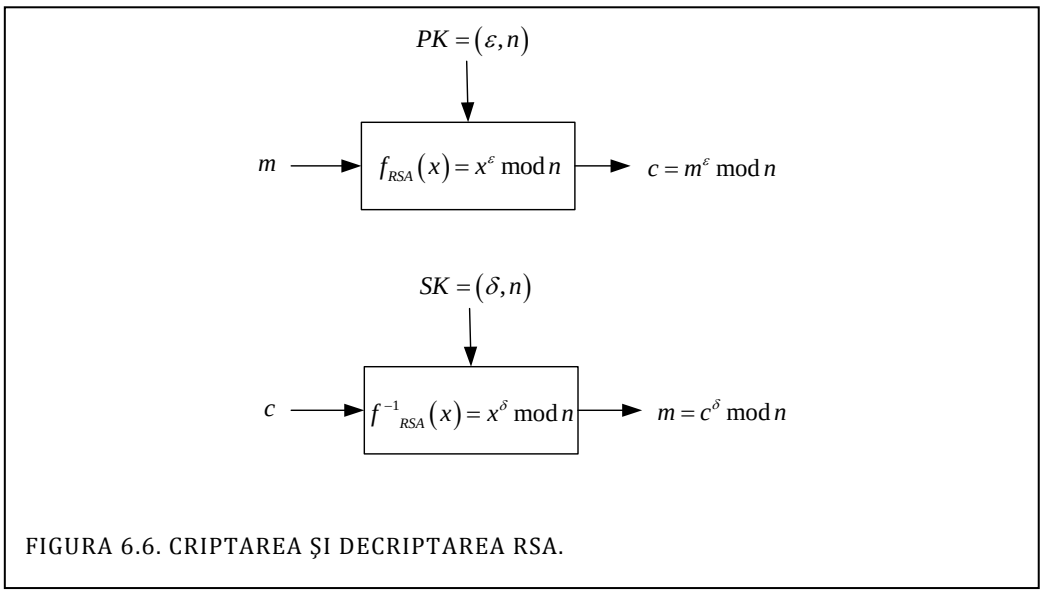


FIGURA 6.6. CRIPTAREA ŞI DECRIPTAREA RSA.

$RSA.Gen\left(1^k\right)$: *Generează două numere prime aleatoare* p, q *şi calculează* $n = pq$ *respectiv* $\phi(n) = (p-1)(q-1)$ *(presupunem că numerele* p *şi* q *au fost generate în aşa fel încât* n *are* k *biţi). Generează un întreg* ε *astfel încât* $c.m.m.d.c.\left(\varepsilon, \phi(n)\right) = 1$, *calculează* δ *astfel încât* $\varepsilon\delta \equiv 1 \bmod \phi(n)$. *Cheia publică este* $PK \leftarrow (n, \varepsilon)$ *iar cea privată este* $SK \leftarrow (n, \delta)$.

$RSA.Enc(m, PK)$: *Se obţine cheia publică a entităţii aferente* $PK \leftarrow (n, \varepsilon)$ *şi se reprezintă mesajul ca întreg în intervalul* $(1, n)$. *Se calculează criptotextul ca* $c \leftarrow m^{\varepsilon} \bmod n$.

$RSA.Dec(c, SK)$: *Decriptează mesajul ca* $m \leftarrow c^{\delta} \bmod n$.

SISTEMUL 6-1. SCHEMA DE CRIPTARE CU CHEIE PUBLICĂ RSA

6.6 SECURITATEA CRIPTOSISTEMULUI RSA

Singura cale cunoscută de a sparge complet sistemul RSA este factorizarea modulului, reamintim însă că nu există nici o demonstraţie că aceasta este singura metodă de a sparge complet RSA-ul. Adică, nu există nici o demonstraţie cu privire la echivalenţa dintre RSA şi problema factorizării întregilor IFP. Mai mult, recent s-a instalat mult scepticism cu privire la echivalenţa între securitatea RSA şi problema factorizării întregilor odată cu apariţia articolului lui Boneh şi Venkatesan [16].

Este însă demonstrat că a calcula o pereche de chei RSA (cheie publică şi cheie privată) este echivalent cu problema factorizării întregilor. În acest sens următoarele două relaţii sunt relevante cu privire la securitatea RSA-ului:

$$RSA.Dec <_P IFP$$

$$RSA.Gen \Leftrightarrow PFI$$

Prima relaţie ilustrează faptul că decriptarea RSA se reduce polinomial la factorizare (desemnată ca PFI de la Problema Factorizării Întregi) şi este evidentă, calculul cheii de decriptare RSA făcându-se pe baza factorilor modulului.

Cea de-a doua relaţie ilustrează faptul că a calcula o pereche de chei RSA este o problemă echivalentă factorizării şi poate fi demonstrată după cum urmează. În mod evident dacă cunoaştem factorizarea modulului putem calcula perechea de chei, rămâne deci de arătat doar că o pereche de chei poate fi utilizată pentru a factoriza modulul. Presupunem că se cunosc ε şi δ astfel încât $\varepsilon\delta \equiv 1 \bmod \phi(n), n = pq$ şi dorim aflarea lui p şi q. Enumerăm două metode care pot fi folosite în acest scop:

i)[3] Se observă că $\varepsilon\delta \equiv 1 \bmod \phi(n) \quad \Rightarrow \varepsilon\delta = 1 + k(p-1)(q-1)$ $\Rightarrow \varepsilon\delta - 1 = k(n+1-p-q)$ deoarece discutăm în contextul criptografiei cu cheie publică şi numărul n are o magnitudine foarte mare raportat la ceilalţi membrii ai ecuaţiei în mod cert vom avea $\Rightarrow k = \left\lfloor \frac{\varepsilon\delta - 1}{n} \right\rfloor$. Folosind această valoare pentru k avem $p + q = \left(\left\lfloor \frac{\varepsilon\delta - 1}{n} \right\rfloor (n+1) - \varepsilon\delta + 1 \right) \Big/ \left\lfloor \frac{\varepsilon\delta - 1}{n} \right\rfloor$. Din moment ce cunoaştem atât suma factorilor cât şi produsul, care este chiar valoarea lui n, putem extrage cele două numere ca rădăcini ale unei ecuaţii de gradul 2. Calculul este banal în continuare şi nu are sens să fie detaliat.

[3] Metoda este preluată de la cursul de „Security reductions for asymmetric systems" susţinut de Pascal Paillier la Bonn, Germania în septembrie 2007 http://www.b-it-center.de/Wob/en/view/class211_id867.html.

ii)[4] Se observă de asemenea că pentru $\forall x$ avem $x^{\varepsilon\delta} \equiv x \bmod n \Rightarrow x^{\varepsilon\delta-1} \equiv 1 \bmod n$ şi prin împărţire succesivă a exponentului la 2 vom ajunge la un moment dat la $x^{(\varepsilon\delta-1)/2^i} \neq \pm 1 \bmod n$ în timp ce $x^{(\varepsilon\delta-1)/2^{i-1}} \equiv 1 \bmod n$ care implică $\left(x^{(\varepsilon\delta-1)/2^i}+1\right)\left(x^{(\varepsilon\delta-1)/2^i}-1\right) \equiv 0 \bmod n$ ceea ce înseamnă că membrii produsului din partea stângă a ecuaţiei ascund factori distincţi ai lui n şi aceştia pot fi extraşi uşor cu de calcul al celui mai mare divizor comun (vezi secţiunile de fundamente matematice şi computaţionale).

Alegerea corectă a parametrilor din stadiul de iniţializare a cheii este extrem de importantă. Pentru a evita atacuri prin factorizarea modulului n se recomandă utilizarea unui modul de 1024-4096 biţi pentru o securitate pe termen lung (vezi tabelul din secţiunea introductivă). Alegerea celor două numere prime p şi q este critică pentru securitate. Este recomandabil ca numerele să fie alese astfel încât să aibă acelaşi număr de biţi iar $p-q$ să fie suficient de mare pentru a preveni un atac exhaustiv prin căutarea unui factor mai mic decât $\sqrt{n}$. Pentru respectarea primei condiţii numerele sunt alese aleator ca având fiecare $\frac{1}{2}\log_2 n$ biţi iar o alegere aleatoare garantează şi respectarea celei de-a doua condiţii cu o probabilitate suficient de mare.

Pentru a spori rezistenţa în faţa atacurilor prin factorizare a fost propusă şi varianta RSA nebalansat în care factorii nu au număr egal de biţi. Aceasta este mult mai rezistentă la factorizare dar are o vulnerabilitate fatală în faţa unui atac de tip criptotext ales care v-a fi discutată mai jos.

De asemenea exponentul public poate fi ales ca având forme speciale pentru a face criptarea mai eficientă. Sunt preferaţi exponenţii care au cât mai puţini biţi de 1, aceasta datorită algoritmului de exponenţiere care consumă mai mult timp când bitul exponentului este 1. Din acest motiv între exponenţii preferaţi pentru criptarea RSA sunt numerele 3, 17 şi 65537.

În mod cert RSA este criptosistemul cu cheie publică cel mai intens studiat (nu în ultimul rând acest lucru se datorează simplităţii sale). Astfel, de-a lungul timpului o gamă relativ largă de atacuri şi vulnerabilităţi ale RSA au fost publicate. În primul rând sunt relevante **atacurile pasive**, care au ca sursă un adversar care încearcă off-line să spargă criptosistemul, dintre acestea amintim:

---

[4] Metoda este preluată din cartea lui Menezes et al. [62].

- **Atacul prin factorizare** presupune factorizarea întregului n - cu ajutorul algoritmilor cunoscuţi. În prezent acest lucru nu este posibil pentru valori suficient de mari ale lui n.

- **Atacul prin căutarea directă a mesajului (forward-search)** este un atac general, fezabil asupra oricărui criptosistem cu cheie publică. Deoarece cheia publică este prin definiţie publică un adversar, având un criptotext capturat, poate face o căutare exhaustivă în cazul în care spaţiul mesajului este redus. Pentru a evita acest lucru se folosesc tehnici de padding precum OAEP sau PKCS prezentate la sfârşitul acestui capitol (acestea fac atacul imposibil în sens practic).

În al doilea rând şi mult mai periculoase sunt atacurile active pentru cazul în care un adversar are acces la maşina de decriptare (fiind criptare cu cheie publică accesul la maşina de criptare este evident). Dintre acestea amintim:

- **Atacul prin temporizare**. În mod cert cantitatea de timp necesară decriptării poate conduce la informaţii suplimentare despre exponentul utilizat. Paul Kocher este cel care a adus în discuţie acest tip de atac. O abordare temeinică asupra unui astfel de atac împotriva OpenSSL este în lucrarea [18] – concluzia lucrării este că astfel de atacuri sunt posibile şi deci trebuie contracarate. Soluţia împotriva acestor atacuri este fie de a fixa un timp fix de calcul indiferent de dimensiunea exponentului, fie de a efectua operaţii suplimentare în mod aleator pentru a deruta adversarul, detalii se găsesc în lucrarea [18].

- **Adaptive chosen chipertext-attack** – se discută în contextul adversarilor activi la sfârşitul acestui capitol.

Există câteva vulnerabilităţi ale RSA-ului comun cunoscute şi care sunt necesare de amintit, în practică astfel de scenarii trebuind evitate cu orice preţ:

- **Exponenţii de criptare sau decriptare relativ mici**. Atacurile asupra exponenţilor mici de criptare au fost introduse de [50] iar cele asupra exponenţilor mici de decriptare de Wiener. Atacul asupra unui exponent mic de criptare presupune existenţa unei relaţii între mesajele criptate şi se contracarează prin extinderea mesajului cu biţi aleatori. Exponenţii mici de decriptare trebuie evitaţi în practică (evitarea decurge în mod natural deoarece în practică se utilizează exponenţi de criptare mici sau cu forme speciale şi aceştia fac ca exponenţii de decriptare să fie mari). Tot în categoria exponenţilor de criptare mici intră şi **atacul bazat pe teorema chineză a resturilor** (vezi secţiunea de fundamente matematice) deoarece în cazul criptării cu exponent mic a aceluiaşi mesaj folosind mai multe module diferite, mesajul ar putea fi extras ca rădăcină reală a

criptotextului (calculul rădăcinii reale este fezabil în timp polinomial, a se vedea capitolul de fundamente matematice pentru teorema chineză a resturilor).

- **Problema modulului comun**. Utilizarea aceluiaşi modul de către mai mulţi participanţi nu este posibilă deoarece cunoaşterea unei perechi de chei publică-privată duce iminent la factorizarea modulului şi pierderea securităţii între participanţi. Alte scheme asimetrice precum ElGamal permit utilizarea unui modul comun fără a avea această deficienţă.
- **RSA balansat şi RSA nebalansat**. Varianta de RSA în care cele două numere prime sunt alese ca având aproximativ aceeaşi dimensiune poartă numele de RSA balansat şi este varianta recomandată şi utilizată în practică. Shamir a propus şi utilizarea unei variante de RSA numite RSA nebalansat care are rezistenţă mult mai mare decât RSA balansat împotriva factorizării şi are aceeaşi viteză de criptare/decriptare [4, p. 276]. Aceasta presupune utilizarea unui număr prim p relativ mic (câteva sute de biţi) şi a unui număr prim q relativ mare (câteva mii de biţi) iar apoi decriptarea se va face modulo p pentru a câştiga timp (evident pentru mesaje mai mici decât p). Această schemă are însă un dezavantaj major – nu rezistă în faţa unui atac de tip criptotext ales. Explicaţia este următoarea: presupunem că un adversar criptează un mesaj $m' > p$ iar valoarea criptată este $c' = m'^{\varepsilon} \bmod n$ dacă maşina de decriptare oferă ca răspuns $m'' = c'^{\delta} \bmod p$ în mod evident $m'' \neq m'$ şi totodată $m'' \equiv m' \bmod p$ deci $c.m.m.d.c.\left(m''-m', n\right) = p$.

Şi nu în ultimul rând câteva proprietăţi ale criptării RSA sunt bine de menţionat cu specificaţia că aceste proprietăţi s-au dovedit de-a lungul timpului a fi pe de o parte avantajoase pentru că au permis dezvoltarea unor soluţii de securitate exotice (precum semnătura blind introdusă de Chaum ce va fi discutată în capitolul următor) şi pe de altă parte dezavantajoase deoarece au fost sursa unor noi atacuri asupra RSA:

- **Ciclicitatea criptării**. Funcţiile definite pe mulţimi finite conduc la cicluri prin compoziţia lor succesivă. Astfel dacă avem un mesaj criptat $c = m^e \bmod n$ şi continuăm să îl criptăm de un număr finit de ori se va ajunge ca după k runde sa obţinem $c^{e^k} \equiv c \bmod n$. În cele din urmă acest atac devine chiar o metodă de factorizare (vezi metodele bazate pe coliziuni din capitolul de fundamente matematice). Cum factorizarea nu este fezabilă, ciclicitatea criptării nu reprezintă un dezavantaj în practică.

- **Criptarea identică** se poate remarca că există şi mesaje care în urma criptării rămân neschimbate. Aceste mesaje sunt acele numere care satisfac ecuaţia $m^{\varepsilon} \equiv m \bmod n \Leftrightarrow m^{\varepsilon-1} \equiv 1 \bmod n$. Ne propunem să determinăm numărul de mesaje care satisfac această condiţie. Datorită izomorfismului descris de teorema chineză a resturilor este suficient să lucrăm modulo p şi modulo q. Astfel în Z_p ecuaţia $m^{\varepsilon-1} \equiv 1 \bmod p$ are exact $\lambda = c.m.m.d.c.(\varepsilon-1, p-1)$ rădăcini. Analog există $c.m.m.d.c.(\varepsilon-1, q-1)$ soluţii ale ecuaţiei în Z_q iar din teorema chineză a resturilor rezultă că în Z_n^* există $c.m.m.d.c.(\varepsilon-1, p-1)c.m.m.d.c.(\varepsilon-1, q-1)$ soluţii. Numărul de mesaje care verifică această ecuaţie este deci redus şi nu poate afecta securitatea schemei de criptare RSA.
- **Proprietatea multiplicativă** constă în faptul că pentru două mesaje criptate $c_1 = m_1^{\varepsilon} \bmod n$ şi $c_2 = m_2^{\varepsilon} \bmod n$ avem $c_1 c_2 = (m_1 m_2)^{\varepsilon} \bmod n$. Această proprietate a dus la multe construcţii interesante bazate pe RSA dar din nefericire şi la atacul de tip criptotext ales adaptiv care va fi prezentat la sfârşitul capitolului.

6.7 CRIPTAREA ASIMETRICĂ RABIN

Se bazează pe utilizarea funcţiei $f(x) = x^2 \bmod n$ unde $n = pq$ este un întreg compozit produs a două numere prime exact ca la RSA. Evident schema Rabin nu este un caz particular al RSA deoarece RSA pretinde ca $c.m.m.d.c.(\varepsilon, \phi(n)) = 1$ în timp ce $c.m.m.d.c.(2, \phi(n)) = 2$. Mai mult, funcţia $f(x) = x^2 \bmod n$ nu este o permutare a elementelor din Z_n ca în cazul RSA fiind clar că aceasta transformă Z_n^* în Q_n şi $|Q_n| = \frac{(p-1)(q-1)}{4}$ (se va consulta capitolul de fundamente matematice pentru a lămuri de ce este aşa). Pentru cazul în care *n* este întreg Blum (adică $p \equiv q \equiv 3 \bmod 4$) şi domeniul de definiţie se schimbă în $f : Q_n \to Q_n$, funcţia Rabin devine o permutare, deoarece fiecare reziduu cvadratic are exact patru rădăcini din care exact una este reziduu cvadratic. Criptosistemul este ilustrat în Figura 6.7 iar descrierea sa formală este în Sistemul 6-2.

$Rabin.Gen\left(1^k\right)$: *Generează două numere prime aleatoare* p, q *şi calculează* $n = pq$ *(presupunem că numerele* p *şi* q *au fost generate în aşa fel încât* n *are* k *biţi). Cheia publică este* $PK \leftarrow (n)$ *iar cea privată este* $SK \leftarrow (p, q)$.

$Rabin.Enc(m, PK)$: *Se obţine cheia publică a entităţii aferente* $PK \leftarrow (n)$ *şi se reprezintă mesajul ca întreg în intervalul* $(1, n)$. *Se calculează criptotextul ca* $c \leftarrow m^2 \bmod n$.

$Rabin.Dec(c, SK)$: *Decriptează mesajul ca* $m \leftarrow \sqrt{c} \bmod n$ *(calculul rădăcinii pătrate este descris în capitolul de fundamente matematice).*

SISTEMUL 6-2. SCHEMA DE CRIPTARE CU CHEIE PUBLICĂ RABIN

Se observă că este necesară prezenţa unui element de redundanţă pentru a putea distinge între mesajul original şi celelalte trei rădăcini (reamintim că fiecare reziduu cvadratic are exact 4 rădăcini dacă $n = pq$). În practică, introducerea unui astfel de element de redundanţă nu este o problemă - fiind suficientă setarea într-o formă prestabilită a câtorva biţi sau transmisia pe lângă mesajul criptat a câtorva biţi redundanţi. Din nou, tehnicile de padding ce vor fi discutate la finalul acestui capitol rezolvă şi această problemă, pe lângă atacurile active la care răspund.

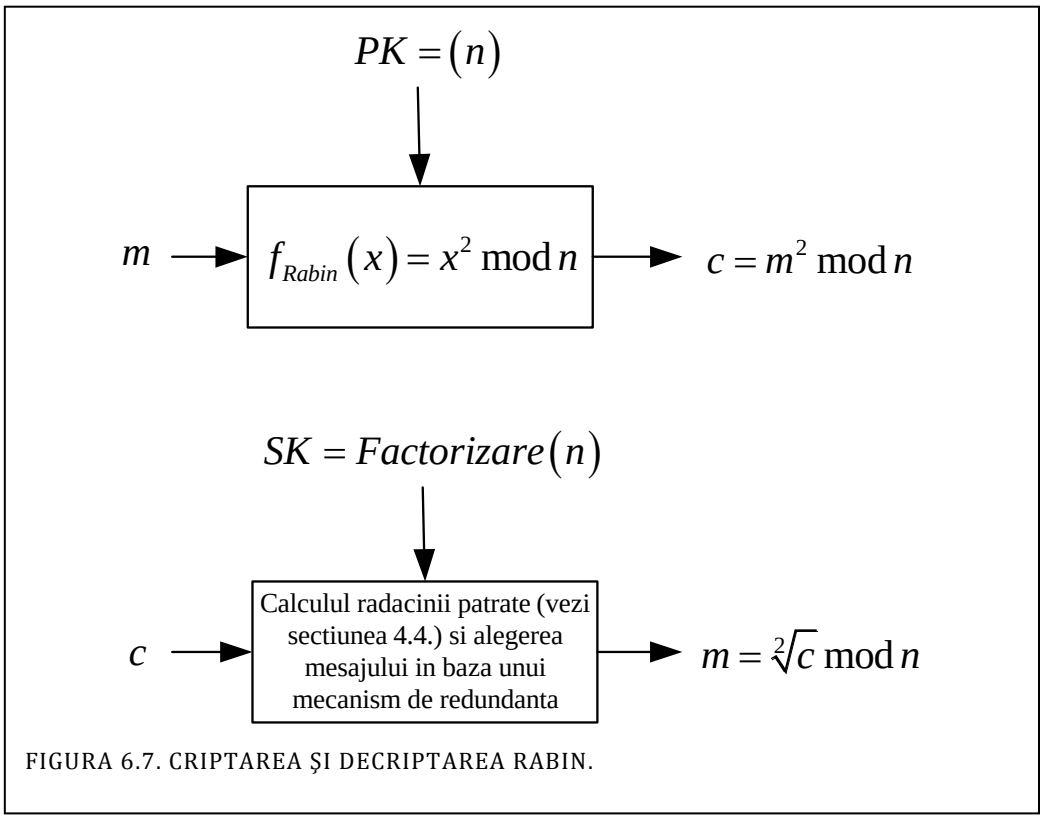


FIGURA 6.7. CRIPTAREA ŞI DECRIPTAREA RABIN.

6.8 SECURITATEA CRIPTOSISTEMULUI RABIN

Ne propunem să enumerăm câteva atacuri clasice asupra algoritmului Rabin. Atacul de tip **criptotext ales** este probabil cauza care a dus la marginalizarea acestui algoritm de criptare. Posibilitatea de a găsi toate rădăcinile ecuaţiei $x^2 \equiv a \bmod n$ duce uşor la factorizarea întregului n. În acest context dacă un adversar are acces la maşina de decriptare şi poate obţine decriptarea unui mesaj ales arbitrar cu o probabilitate de 50% acesta va reuşi să factorizeze întregul n. Folosirea riguroasă a redundanţei poate duce la evitarea acestui tip de atac. Atacul prin **căutarea directă a mesajului** este valid în acelaşi context ca la RSA pentru orice criptosistem cu cheie publică determinist. Atacul bazat pe **teorema chineză a** este posibil şi aici de această dată exponentul de criptare fiind 2.

Spre deosebire de RSA, în cazul căruia nu există nici o demonstraţie că securitatea sa este echivalentă cu factorizarea, pentru algoritmul Rabin o astfel de

demonstraţie este uşor de făcut. De fapt demonstraţia este directă deoarece posibilitatea de a calcula reziduuri cvadratice conduce la factorizare aşa cum poate fi uşor desprins din secţiunea de fundamente matematice. Următoarea echivalenţă este adevărată cu privire la schema Rabin:

$$Rabin.Dec \Leftrightarrow PFI$$

6.9 CRIPTAREA ASIMETRICĂ EL-GAMAL

Lucrarea lui Diffie şi Hellman [29] propunea ideea de criptare asimetrică şi un protocol de schimb de cheie asimetric. Şapte ani mai târziu, El-Gamal utilizează ideile propuse de Diffie-Hellman pentru a construi un algoritm de criptare asimetrică şi o semnătură digitală [33].

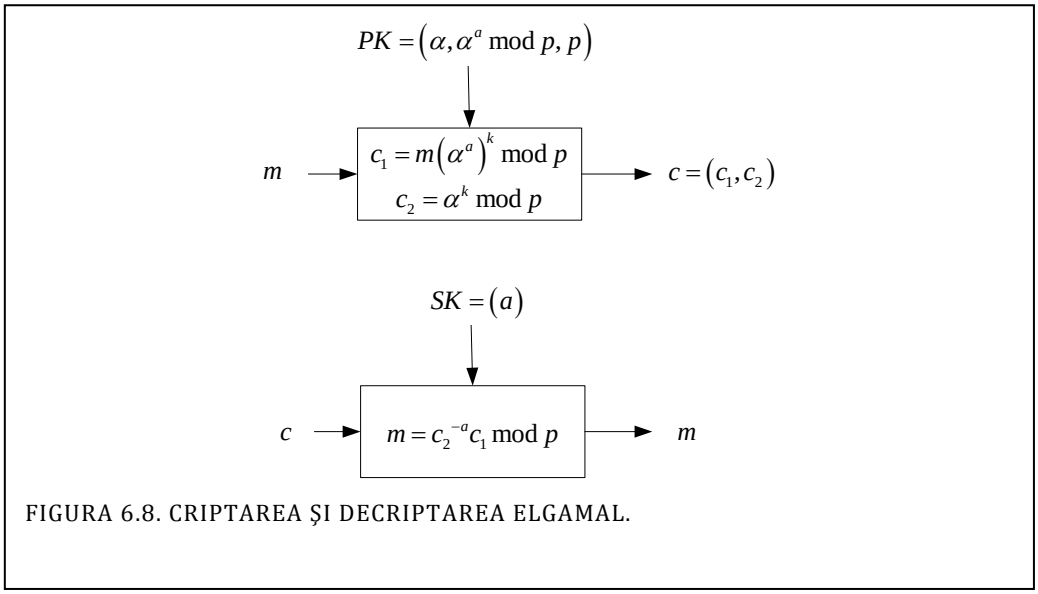


FIGURA 6.8. CRIPTAREA ŞI DECRIPTAREA ELGAMAL.

Principiul de funcţionare al criptării asimetrice este identic cu cel al protocolului de schimb de cheie Diffie-Hellman cu menţiunea că cheia simetrică este apoi utilizată pentru criptarea unui mesaj prin efectuarea unei multiplicări modulare între aceasta şi mesajul propriu-zis. Sistemul 6-3 oferă descrierea schemei ElGamal, la fel şi Figura 6.8.

$ElGamal.Gen(1^k)$: *Generează un număr prim* p *de* k *biţi şi alege un generator* g *al grupului* Z_p. *Generează un întreg aleator* $a \in (1, p-2)$ *şi calculează* $g^a \bmod p$. *Cheia publică este* $PK \leftarrow (\alpha, \alpha^a \bmod p, p)$ *iar cea privată este* $SK \leftarrow (a)$.

$ElGamal.Enc(m, PK)$: *Se obţine cheia publică a entităţii* $PK \leftarrow (\alpha, \alpha^a \bmod p, p)$ *şi se reprezintă mesajul ca întreg în intervalul* $(1, p)$. *Se generează un întreg aleator* $1 < k < p-2$, *se calculează criptotextul ca* $c \leftarrow (\gamma = \alpha^k \bmod p, \delta = m \cdot (\alpha^a)^k \bmod p)$.

$ElGamal.Dec(c, SK)$: *Decriptează mesajul ca* $m = \delta \gamma^{-a} \bmod p$.

SISTEMUL 6-3. SCHEMA DE CRIPTARE CU CHEIE PUBLICĂ ELGAMAL

Se observă că dimensiunea mesajului transmis este dublă în cazul criptării ElGamal deoarece trebuie transmisă perechea γ, δ.

6.10 SECURITATEA CRIPTOSISTEMULUI ELGAMAL

Algoritmul El-Gamal este un algoritm de criptare non-determinist datorită alegerii parametrului aleator k. Adică, un mesaj va rezulta în criptări diferite la fiecare rulare a schemei. Se observă că utilizarea aceluiaşi exponent k nu este posibilă, deoarece dacă acelaşi k este utilizat pentru a cripta mesaje diferite atunci este valabilă egalitatea $\frac{\gamma_1}{\gamma_2} = \frac{m_1}{m_2}$ ce duce la posibilitatea de a calcula m_1 ca funcţie de m_2 şi reciproc (deci algoritmul nu rezistă în faţa unui atac de tip criptotext ales).

Primul studiu serios asupra securităţii criptosistemului ElGamal se găseşte în [82]. Securitatea criptosistemului ElGamal este echivalentă cu problema computaţională Diffie-Hellman (CDH) şi deci impune ca şi problema logaritmului discret (DLP) să nu poată fi eficient rezolvată pentru a fi sigur. Următoarea relaţie este adevărată:

$$ElGamal.Dec \Leftrightarrow CDH$$

6.11 CRIPTOSISTEMUL GOLDWASSER-MICALI

Criptosistemele RSA şi Rabin aşa cum au fost anterior prezentate sunt criptosisteme deterministe. Marele dezavantaj al criptării deterministe este că un anume mesaj corespunde aceleiaşi valori de criptotext întotdeauna, lucru care face dificilă transmiterea aceluiaşi mesaj către o anume entitate, deoarece un adversar poate observa cel puţin repetiţia aceleiaşi informaţii şi deci obţine o informaţie parţială cu privire la mesajele vehiculate. Un alt dezavantaj este şi acela că unele mecanisme de criptare deterministe permit recuperarea unor biţi individuali de informaţie din criptotext (într-un exemplu din secţiunea de fundamentare teoretică arătăm că simbolul Jacobi al mesajului poate fi recuperat din criptotextul RSA). Tehnicile de padding discutate la finalul capitolului rezolvă acest neajuns, dar ele au venit la mai bine de un deceniu după schemele iniţiale. Criptosistemul Goldwasser-Micali a apărut mult mai devreme fiind revoluţionar pentru vremea la care a apărut şi rezolvând această problemă. Acesta este avangardist ca procedeu constructiv şi principii introduse în securitate, dar nu ca şi eficienţă (el este ineficient din punct de vedere computaţional şi absent în practică). Sistemul este un sistem criptografic non-determinist, ceea ce înseamnă că rezultatul are o valoare randomizată (acelaşi mesaj criptat de mai multe ori va rezulta în criptotexte diferite).

Principiul pe care se bazează criptosistemul este imposibilitatea de a distinge **reziduurile cvadratice** de **pseudo-reziduurile cvadratice** din Z_n^* fără a cunoaşte factorizarea lui n - acest lucru mai se numeşte şi **problema reziduurilor cvadratice** (QRP – Quadratic Residuosity Problem). Ce trebuie să ştim despre aceasta înainte de a citi capitolul de fundamente matematice, este că reziduurile cvadratice sunt numerele pătrate perfecte din Z_n^* iar pseudo-reziduurile sunt numere care nu sunt pătrate perfecte dar nu se cunosc algoritmi care pot face diferenţa dintre acestea şi pătratele perfecte (decât dacă se cunoaşte

factorizarea lui n). Sistemul 6-4 este descrierea schemei de criptare asimetrică Goldwasser-Micali.

$GM.Gen\left(1^k\right)$: *Generează două numere prime aleatoare* p, q *şi calculează* $n = pq$. *Alege* $y \in Z_n^*$ *astfel încât* y *este un pseudo-reziduu cvadratic, i.e.* $y \in Q_n$ *(vezi Definiţia 9.44). Cheia publică este* $PK \leftarrow (n, y)$ *iar cea privată este* $SK \leftarrow (p, q)$.

$GM.Enc(m, PK)$: *Se obţine cheia publică a entităţii aferente* $PK \leftarrow (n, y)$ *şi se reprezintă mesajul* m *în binar ca* $m = m_0 m_1 m_2 ... m_t$. *Pentru* $i = \overline{1,t}$ *alege un întreg aleator* $x \in Z_n^*$ *dacă* $m_i = 1$ *atunci* $c_i = yx^2 \bmod n$ *altfel* $c_i = x^2 \bmod n$. *Mesajul criptat este* $(c_0 c_1 c_2 ... c_t)$.

$GM.Dec(c, SK)$: *Pentru* $i = \overline{1,t}$ *calculează simbolul Legendre* $\left(\frac{c_i}{p}\right)$ *şi dacă* $\left(\frac{c_i}{p}\right) = 1$ *atunci* $m_i = 0$ *altfel* $m_i = 1$. *Decriptează mesajul ca* $m \leftarrow c^\delta \bmod n$.

SISTEMUL 6-4. SCHEMA DE CRIPTARE CU CHEIE PUBLICĂ GOLDWASSER-MICALI

Aşa cum se observă, entitatea care generează cheia cunoaşte şi factorizarea lui n şi astfel poate face diferenţa între reziduuri şi pseudo-reziduuri. Pentru criptare, fiecare bit este criptat într-un reziduu sau pseudo-reziduu după cum este 1 sau 0 folosind un pseudo-reziduu publicat de cel care a generat cheia. Evident prin înmulţirea a unui reziduu cu un pseudo-reziduu se obţine tot un pseudo-reziduu, astfel $c_i = yx^2 \bmod n$ este un pseudo-reziduu pentru că y este pseudo-reziduu în timp ce $c_i = x^2 \bmod n$ este un reziduu.

Securitatea schemei este echivalentă cu problema calculului apartenenţei la mulţimea reziduurilor cvadratice, deci:

$$GM.Dec \Leftrightarrow QRP$$

Nu securitatea sau eficienţa acestei scheme este partea care o face relevantă, ci faptul că această schemă atinge un obiectiv important de securitate (imperceptibilitatea criptotextelor) care va fi discutat într-un capitol următor.

6.12 SCHIMBUL DE CHEIE DIFFIE-HELLMAN FOLOSIND CURBE ELIPTICE (ECDH)

Curbele eliptice erau cunoscute de criptografi încă din anii 80. Cu toate acestea, ele au apărut în practică doar după anii 2000. Dintre motivele întârzierii, cel puţin două par clare: faptul că sunt mai dificil de implementat şi faptul că matematica pe care se bazează este mai complexă (nu toate proprietăţile curbelor sunt perfect înţelese).

Deloc surprinzător, schimbul de cheie Diffie-Hellman poate fi implementat şi folosind curbe eliptice. Într-o secţiune următoare discutăm ce sunt acestea, pentru moment ele pot fi interpretate ca un simplu obiect matematic. Această implementare este comun referită în practică ca ECDH adică Elliptical-Curve Diffie-Hellman. Poate ar fi corect, la fel cum protocolul Diffie-Hellman a fost propus să se numească Diffie-Hellman-Merkle, ca acest protocol să se numească Koblitz-Miller-Diffie-Hellman-Merkle (KMDHM) deoarece Koblitz şi Miller au fost pionierii curbelor eliptice în criptografie.

Spre deosebire de criptosisteme precum RSA, unde parametrii erau generaţi aleator şi păstraţi secret, aici parametrii curbei sunt publici şi pot fi (re)folosiţi de oricâte ori. Mai mult NIST recomandă anumite curbe în FIPS 186-3 care sunt larg folosite de toate implementările practice. Curbele recomandate de NIST folosesc unul din cele 5 câmpuri prime generate de numere prime p de 192, 224, 256, 384 şi 512 biţi sau unul din cele 5 câmpuri binare de 163, 233, 283, 409 sau 571 biţi. Pentru acestea din urmă sunt recomandate atât o curbă simplă cât şi o curbă Koblitz (care permite cu uşurinţă calcularea numărului de puncte), deci în

total sunt 5 curbe în câmpuri prime şi 10 curbe în câmpuri binare recomandate de NIST. Protocolul 6-2 descrie schimbul de cheie ECDH.

$ECDH.Gen\left(1^k\right)$: *Alege un număr prim* p *de* k *biţi, coeficienţii* a *şi* b *ai unei curbe eliptice* $E: y^2 = x^3 + ax + b \bmod p$ *şi un generator* P. *Toţi aceşti parametrii sunt publici.*

$ECDH.Exchange$: A: *alege un număr aleator* a *şi trimite*

1. $A \to B$: aP

B: *alege un număr aleator* b *şi trimite*

2. $B \to A$: bP

A *şi* B: *calculează cheia comună ca* abP

PROTOCOLUL 6-2. SCHIMBUL DE CHEIE DIFFIE-HELLMAN FOLOSIND CURBE ELIPTICE

6.13 LIPSA SECURITĂŢII ÎN VARIANTELE „TEXT-BOOK" ALE ALGORITMILOR DE CRIPTARE

> "Of concern with both of these schemes is that there is no compelling reason to believe that x is as hard to compute from f(rx) as rx is hard to compute from f(rx) let alone that all interesting properties of x are well-hidden by f(rx). Indeed whether or not [22, 15] "work" depends on aspects of f beyond its being one-way, insofar as it is easy to show that if there exists a trapdoor permutation then there exists one for which encryption as above is completely insecure." M. Bellare & P. Rogaway[5].

În trecut criptosistemele cu cheie publică erau construite pentru a atinge obiective de securitate rudimentare sau chiar vag definite, de exemplu un adversar să nu poată afla mesajul criptat – ceea ce era uzual asimilat cu inversarea totală a funcţiei one-way pe care se baza criptosistemul. Un astfel de deziderat de securitate este însă ineficient în practică şi este evident că folosirea schemelor criptografice în variantă „text-book" face posibilă aflarea unor informaţii parţiale despre textul criptat. De exemplu criptarea RSA nu modifică simbolul Jacobi al mesajului criptat, astfel, un adversar poate oricând face distincţie între criptările a două mesaje cu simboluri Jacobi diferite.

Totodată, în trecut securitatea era gândită în faţa unor adversari pasivi care în principiu analizau criptotextul în vederea găsirii mesajului sau a cheii. În lumea reală adversarii nu sunt pasivi ci activi, având acces la maşinile de criptare şi decriptare. În faţa atacurilor active sistemele în variantă text-book anterior prezentate sunt nesigure.

Pentru claritatea expunerii vom considera un simplu atac de tip criptotext ales adaptiv asupra schemei RSA. Un atac de tip criptotext ales adaptiv (CCA2)

[5] Din lucrarea „Optimal asymmetric encryption - How to encrypt with RSA".

presupune un adversar care nu cunoaşte cheia privată dar are acces la "maşina" de decriptare, la fel ca cel sugerat în Figura 6.9, iar faptul că criptotextul este ales adaptiv înseamnă că alegerea criptotextelor de către adversar se face pe baza unui criptotext ţintă care adversarul vrea să îl decripteze, maşina de decriptare fiind dispusă să decripteze orice criptotext mai puţin criptotextul ţintă. Adversarul cu acces adaptiv este sugerat în Figura 6.10.

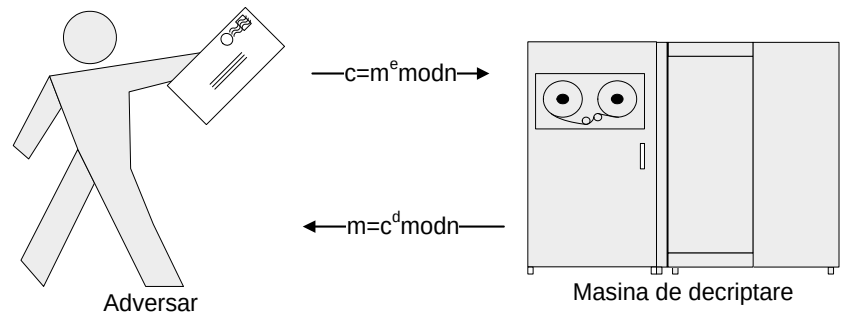


FIGURA 6.9. ADVERSAR CU ACCES LA MAŞINA DE DECRIPTARE.

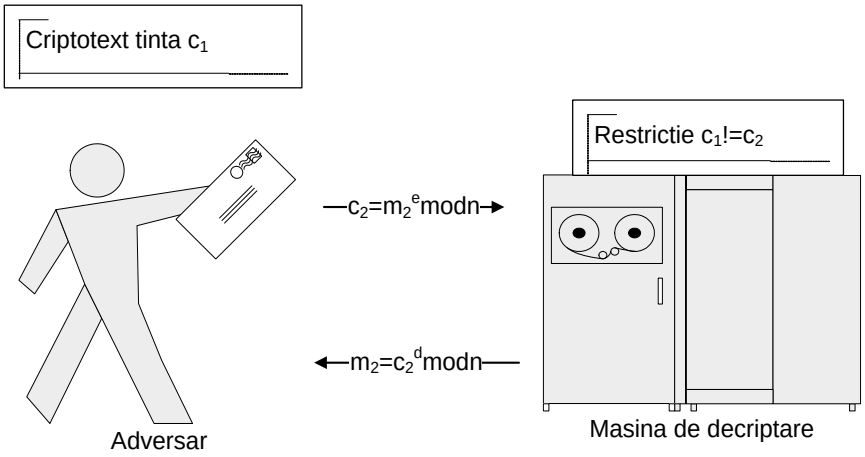


FIGURA 6.10. ADVERSAR CU ACCES ADAPTIV LA MAŞINA DE DECRIPTARE.

În acest scenariu este simplu de observat că adversarul poate ascunde intenţia de a decripta un mesaj $c_1 = m_1^{\varepsilon} \bmod n$ calculând $c_2 \equiv c_1 \cdot m_2^{\varepsilon} \bmod n$ pentru o valoare oarecare m_2, iar apoi obţine de la maşina de decriptare $m_2' \equiv c_2^{\delta} \bmod n$ de unde poate calcula $m_1 = m_2^{-1} m_2' \bmod n$. Atacul, sugerat şi în Figura 6.12, este pe cât de simplu pe atât de fatal (criptotextul fiind spart). Subliniem că toate criptosistemele anterior introduse sunt vulnerabile în faţa unui astfel de atac. Pentru a înlătura această limitare vom introduce în paragraful următor câteva obiective moderne de securitate şi câteva conversii pentru a construi criptosisteme rezistente în faţa unor astfel de adversari.

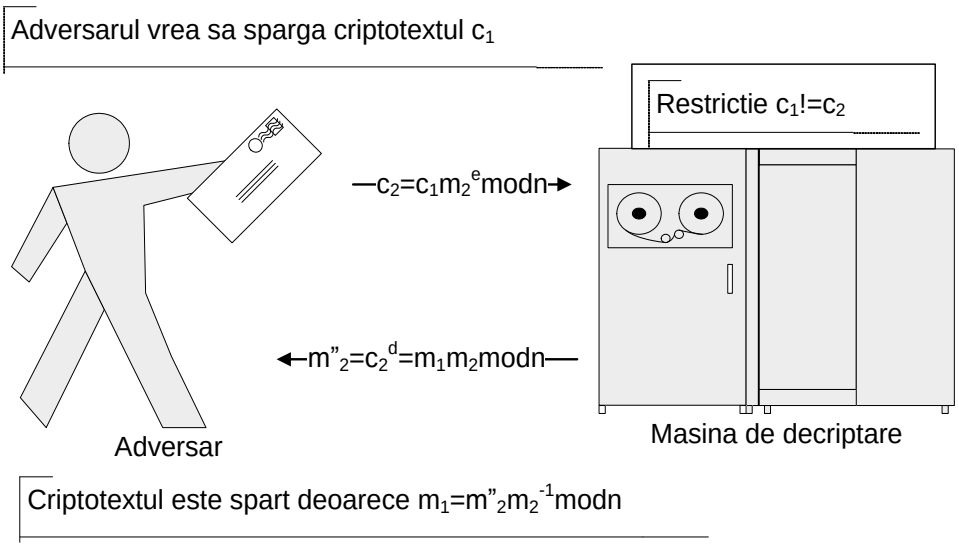


FIGURA 6.11. ATAC ADAPTIV ASUPRA MAŞINII DE DECRIPTARE RSA FINALIZAT CU SUCCES.

Nici criptosistemul ElGamal nu este rezistent în faţa unui atac CCA2. Din nou, un adversar, având un criptotext ţintă $c_1 = \left(\gamma_1 = \alpha^{k_1} \bmod p, \delta_1 = m_1 \cdot (\alpha^a)^{k_1} \bmod p\right)$ poate altera a doua valoare din criptotext prin înmulţirea ei cu o valoare arbitrară şi obţine $\delta_2 = m_2 \cdot \delta_1 \bmod p$. Acum criptotextul nou creat $c_2 = \left(\gamma_1, \delta_2\right)$ este oferit maşinii de decriptare şi

răspunsul acesteia $m_2'' = \delta_2 \cdot \gamma_1^{-a} \bmod p$ poate fi folosit pentru a sparge criptotextul iniţial pentru că într-adevăr $m_1 = m_2'' m_2^{-1} \bmod p$. Atacul este sugerat şi în Figura 6.12.

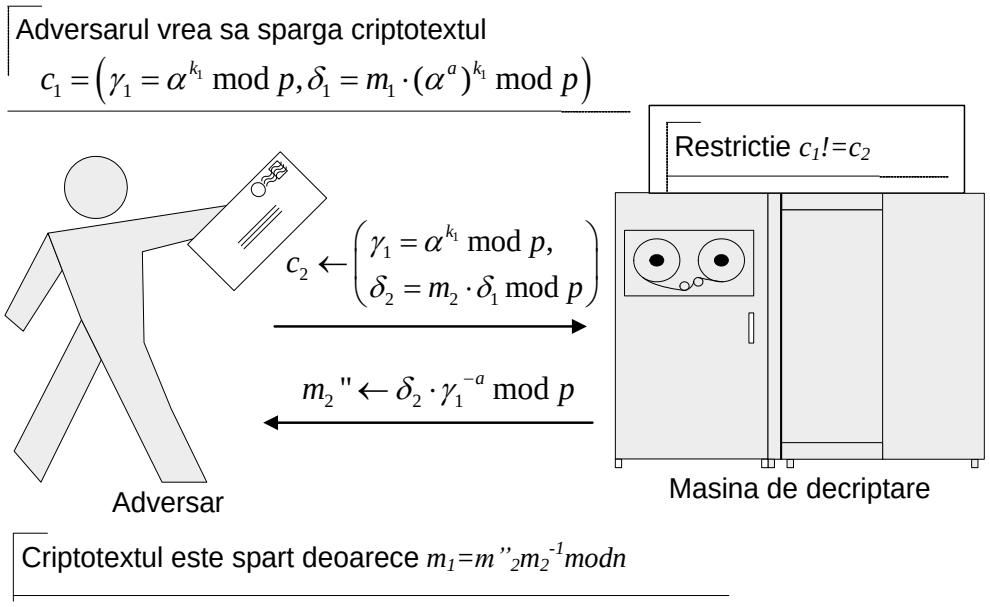


FIGURA 6.12. ATAC ADAPTIV ASUPRA MAŞINII DE DECRIPTARE ELGAMAL FINALIZAT CU SUCCES.

6.14 VARIANTE CONTEMPORANE ALE CRIPTOSISTEMOR CLASICE DE CRIPTARE CU CHEIE PUBLICĂ (REZISTENŢA IND/NM-CCA2)

Datorită nivelului slab de securitate oferit de noţiunea de „securitate de tip totul sau nimic”, criptosistemele cu cheie publică contemporane trebuie să atingă obiective de securitate avansate precum **ne-distingerea sau imperceptibilitatea criptotextelor IND** (**indistinguishability of encryptions**) şi **non-maleabilitatea criptotextelor NM** (**non-malleability**). Prin IND, noţiune

introdusă de Goldwasser şi Micali [44], se înţelege faptul că un adversar nu poate afla nici un fel de informaţie cu privire la un mesaj având doar valoarea criptotextului aferent – în mod ideal aceasta înseamnă că ceea ce un adversar ştie având criptotextul, ştie şi fără criptotext. Prin NM, noţiune introdusă în [30], se înţelege faptul că un adversar nu poate construi un criptotext având un criptotext dat la care nu cunoaşte mesajul aferent astfel încât între mesajele aferente să existe o legătura cunoscută de către adversar. Noţiunea de IND a apărut sub diverse forme în literatura de specialitate, dintre acestea amintim: securitate semantică şi securitate polinomială (toate noţiunile au aceeaşi semnificaţie în securitate).

Mai mult, toate aceste obiective trebuie să fie atinse în prezenţa unor adversari activi care pot fi împărţiţi în trei categorii:

i) **CPA** (chosen plaintext attack) desemnează adversari care au acces la maşina de criptare. Este evident că orice criptosistem cu cheie publică trebuie să fie rezistent CPA deoarece orice adversar are acces la maşina de criptare cheia de criptare fiind în mod evident publică. În acest sens nu se discută niciodată de rezistenţa CPA a unui criptosistem cu cheie publică, aceasta fiind o cerinţă mai mult decât evidentă.

ii) **CCA1** (non-adaptive chosen ciphertext attack) desemnează adversari care au acces neadaptiv la maşina de decriptare. Mai exact adversarul are acces la maşina de decriptare până la momentul la care primeşte valoarea criptotextului care trebuie atacat, moment la care pierde accesul la maşina de decriptare.

iii) **CCA2** (adaptive chosen ciphertext attack) desemnează adversarii care au acces adaptiv la maşina de decriptare, adică accesul la maşină rămâne valabil chiar şi după primirea valorii criptotextului care trebuie atacat. Deoarece atacul CCA1 este considerat oarecum perimat, adeseori în literatura de specialitate se vorbeşte doar de atac CCA prin acesta înţelegându-se de fapt atacul de tip CCA2.

Grupând cele 3 obiective de securitate {IND, NM} cu cele 3 tipuri de adversari {CPA, CCA1, CCA2} obţinem 6 noţiuni de securitate în cazul criptosistemelor cu cheie publică, acestea sunt:

IND-CPA, IND-CCA1, IND-CCA2, NM-CPA, NM-CCA1, NM-CCA2

Lucrarea [9] este cea care a oferit prima abordare unitară a acestor noţiuni de securitate definind legăturile între ele. În Figura 6.13 sunt sintetizate aceste legături, figura este unanim acceptată în domeniu.

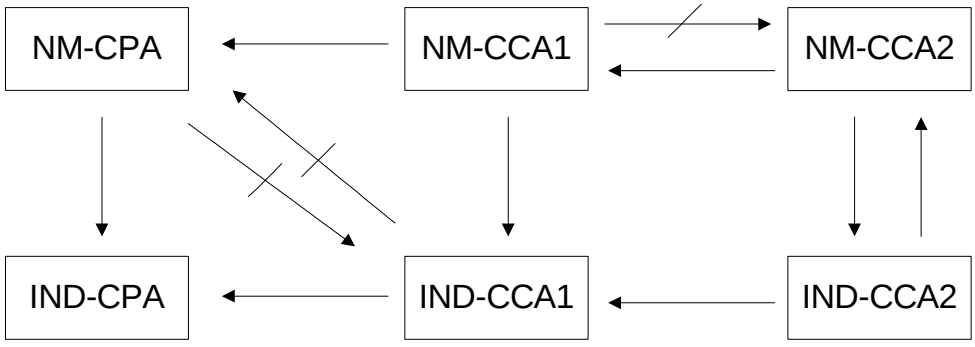


FIGURA 6.13. RELAŢII ÎNTRE NOŢIUNI DE SECURITATE PENTRU CRIPTOSISTEME CU CHEIE PUBLICĂ (DEMONSTRATE ÎN [9]).

Cel mai esenţial aspect în aceste relaţii este echivalenţa $IND-CCA2 \Leftrightarrow NM-CCA2$ care înseamnă că un criptosistem pentru care un adversar cu acces adaptiv la maşina de criptare nu poate asocia un criptotext unui mesaj, chiar dacă i se oferă mesajul original şi un alt mesaj, cu o probabilitate mai mare de ½ atunci acest criptosistem este şi non-maleabil, adică un adversar având un criptotext nu poate construi un alt criptotext astfel încât mesajele aferente să aibă vreo legătură cunoscută de adversar.

Din punct de vedere formal rezistenţa IND-CCA2 poate fi definită după cum urmează (preluăm definiţia originală din lucrarea în care au fost demonstrate aceste noţiuni [9]):

Definiţia 6.1. (Rezistenţa IND-CPA, IND-CCA1, IND-CCA2) Fie criptosistemul $\Xi = \{K, E, D\}$, $\mathrm{A} = \{\mathrm{A}_1, \mathrm{A}_2\}$ un adversar al criptosistemului $at \in \{cpa, cca1, cca2\}$ şi k un parametru de securitate. Definim avantajul IND al adversarului împotriva criptosistemului ca fiind:

$$Adv_{\mathrm{A},\Xi}^{ind-atl}(k) = = 2 \cdot \Pr\left[\begin{array}{l}(pk, sk) \leftarrow K(1^k), (x_0, x_1, s) \leftarrow \mathrm{A}_1^{O_1}(pk), \\ b \leftarrow (0,1), y \leftarrow E_{pk}(x_b) : \mathrm{A}_2^{O_2}(x_0, x_1, s, y) = b\end{array}\right] - 1$$

Având următoarele instanţe pentru at, O_1 şi O_2:

- Dacă $at = cpa$ atunci $O_1(\cdot) = \perp$ şi $O_2(\cdot) = \perp$
- Dacă $at = cca1$ atunci $O_1(\cdot) = D_{sk}(\cdot)$ şi $O_2(\cdot) = \perp$
- Dacă $at = cca2$ atunci $O_1(\cdot) = D_{sk}(\cdot)$ şi $O_2(\cdot) = D_{sk}(\cdot)$

În paragraful următor, dedicat criptosistemelor cu cheie publică rezistente CCA2 vom lucra cu cazul $at = cca2$. Lucrarea de faţă nedorindu-se a fi exhaustivă, nu introducem definiţia pentru proprietatea NM datorită echivalenţei $IND - CCA2 \Leftrightarrow NM - CCA2$ (practic pentru criptosistemele următoare odată demonstrată rezistenţa IND-CCA2 aceasta implică şi rezistenţă NM-CCA2).

6.15 FUNCŢIA DE PADDING OAEP

Propunerile generice ale lui Bellare si Rogaway sunt primele propuneri cu securitate demonstrată. Aceste propuneri, pe lângă importanţa istorică, fiind primele propuneri cu securitate demonstrată, sunt esenţiale deoarece şi propuneri ulterioare (de exemplu RSA-KEM din [79]) chiar dacă mult mai elaborate şi aparent mai complicate sunt extrem de apropiate ca tehnică constructivă de mecanismele iniţiale propuse de Bellare şi Rogaway.

În [5] a fost introdusă prima euristică în baza căreia se poate demonstra securitatea în faţa adversarilor activi. Metoda are la bază utilizarea unui model numit **modelul oracolului aleatoare** ROM (Random Oracle Model) şi are ca bază simularea comportamentului funcţiilor hash ca funcţii perfect aleatoare (altfel spus modelul presupune că un potenţial adversar nu poate face diferenţa între ieşirea unei funcţii aleatoare şi ieşirea unei funcţii hash). Modelul ROM a adus şi criticism din partea unor nume puternice în domeniu [20] dar în cele din urmă este singura metodă la momentul actual de a demonstra securitatea unui criptosistem. Criticismul are la bază faptul că desigur, în cele din urmă, o funcţie hash nu este o funcţie aleatoare, deci Oracole Aleatoare nu există în lumea reală, dar în cele din urmă modelul este acceptat ca fiind cel puţin un compromis şi reprezintă cel puţin un test necesar pentru criptosisteme. Un criptosistem care nu este sigur în ROM nu trebuie pus sub nici o formă în practică în timp ce un criptosistem care rezistă în ROM are şanse bune ca şi în practică să nu poată fi eficient atacat. Totuşi trebuie menţionat că există criptosisteme care pot fi demonstrate ca fiind sigure în ROM şi totuşi pot fi sparte – deci problema de bază a modelului ROM este incompletitudinea. Două tehnici de criptare sunt introduse în [6]:

- Criptarea $E(x) = f(r) \| G(r) \oplus x$ are securitate IND în faţa adversarilor CCA1.
- Criptarea $E(x) = f(r) \| G(r) \oplus x \| H(rx)$ are securitate IND şi NM în faţa adversarilor CCA2 (la data publicării lucrării echivalenţa $IND - CCA2 \Leftrightarrow NM - CCA2$ nu era încă demonstrată aşa că în [6] se găsesc demonstraţii separate pentru cele două proprietăţi de securitate).

În criptosistemele de mai sus G este un generator de numere aleatoare, H este o funcţie hash, iar f este o funcţie de criptare oarecare (în practică G şi H pot fi instanţiate cu o funcţie hash datorită comportamentului funcţiilor hash similar cu funcţiile aleatoare).

Nu este de mirare, că tot Bellare şi Rogaway introduc prima tehnică de criptare pe bază de RSA care este rezistentă CCA2. Aceasta este binecunoscuta tehnică de formatare (padding) a mesajului **OAEP** (Optimal Asymmetric Encryption Padding) folosită sub funcţia RSA, ansamblu cunoscut sub numele de RSA-OAEP [6]. În primul rând sunt importante câteva menţiuni istorice cu privire la OAEP. Bellare şi Rogaway au introdus OAEP în [6] susţinând că OAEP este o

tehnică de padding care funcţionează pentru orice funcţie one-way trapdoor făcând-o rezistentă în faţa atacurilor CCA2. Ulterior Shoup [78] demonstrează că OAEP nu poate să garanteze acest lucru pentru orice funcţie, aducând un contra-exemplu cu o funcţie XOR-maleabilă. Deficienţa descoperită de Shoup ridică mari semne de întrebare cu privire la RSA-OAEP şi sunt Fujisaki, Okamoto, Pointcheval şi Stern [40] cei care reuşesc să demonstreze că RSA-OAEP este rezistentă CCA2. În concluzie RSA-OAEP este un mecanism eficient şi sigur de criptare folosind RSA. Continuăm cu descrierea acestui criptosistem. Funcţia de criptare *f-OAEP* defineşte criptarea ca: $E(x) = f\left(x \oplus G(r) \| r \oplus H\left(x \oplus G(r)\right)\right)$. Se poate uşor observa că este vorba de includerea unei reţele Feistel sub o funcţie one-way cu trapă şi acest lucru se observă uşor în desenul din Figura 6.14.

OAEP este deci o tehnică de padding a cărei securitate este demonstrată la nivelul utilizării funcţiei cu trapă RSA. Desigur există însă şi alte funcţii cu trapă decât RSA-ul. În acest context dezvoltarea unor metode cât mai variate de padding a devenit necesară. Probabil cea mai bună propunere este cea a lui **Fujisaki si Okamoto**. Pe scurt propunerea acestora este următoarea: criptarea $E(x) = E_{pk}\left((x \| r) \| H(x,r)\right)$, unde r este o valoare aleatoare, H este o funcţie hash. În modelul ROM Fujisaki şi Okamoto au demonstrat că o astfel de criptare este rezistentă IND-CCA2 cu condiţia ca funcţia de criptare să fie sigură IND-CPA. Între exemplele oferite de autori se află aplicarea unui astfel de padding asupra schemelor Blum-Goldwasser, El-Gamal şi Okamoto-Uchiyama [39]. Această schemă de padding mai este cunoscută şi sub numele de **Enhanced Probabilistic Encryption**.

Înainte de a încheia trebuie amintite şi criptosistemele de criptare hibridă rezistente IND/NM-CCA2 introduse de Shoup şi Cramer în [27]. O soluţie analoagă pentru construcţia de criptosisteme hibride rezistente CCA2 este în [57] (ulterior, [51] arată că o componentă a criptosistemului din [54] nu este rezistentă CCA2) .

Este simplu de verificat şi rămâne ca exerciţiu pentru cititor că folosind aceste tehnici de padding atacurile anterior arătate nu mai funcţionează. Desigur prezentarea din acest capitol nu este completă din două motive: primul este că nu a fost prezentată nici o demonstraţie formală cu privire la rezistenţa acestor criptosisteme şi al doilea este că nu au fost oferite detalii complete cu privire la parametrii de securitate care trebui utilizaţi în practică (dimensiunile cheilor, ale valorilor aleatoare etc.). Tot ce am dorit în acest capitol a fost să trasăm ideea de rezistenţă IND/NM-CCA2 şi să schiţăm câteva soluţii pentru aceasta, studiul în continuare rămânând deschis pentru cititor. În mediile de programare moderne,

de exemplu .NET, schema de criptare RSA este disponibilă alături de paddingul OAEP (de altfel în .NET nici nu este posibilă folosirea ei fără padding).

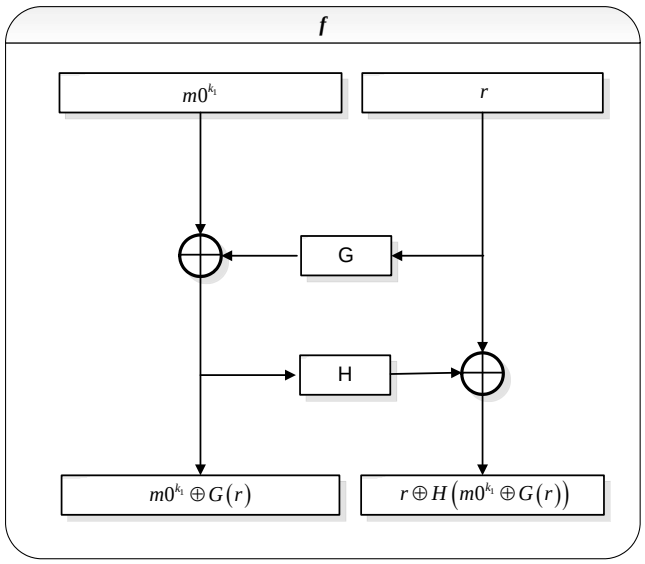


FIGURA 6.14. SCHEMA BLOC A FUNCŢIEI F-OAEP

6.16 FUNCŢIA DE PADDING PKCS

Deşi nu oferă un nivel de securitate la fel de bun ca OAEP, standardele oferite de RSA Laboratories prescriu o schemă de padding cunoscută şi sub numele de PKCS padding. Aceasta apare în diverse implementări practice, de exemplu în .NET.

Conform PKCS#1 (Public-Key Cryptography Standards) [70] acest padding constă în concatenarea unui octet cu valoarea 0, urmat de un octet cu valoarea 2, urmat de un număr aleator de $k-d-3$ bytes (unde d este dimensiunea mesajului) urmat de un octet cu valoarea 0. Astfel criptarea se efectuează ca: $\left(00\ldots00 \,||\, 00\ldots10 \,||\, random \,||\, 00\ldots00 \,||\, m\right)^e \bmod n$.

Utilizarea acestui padding este sigură în faţa atacurilor asupra variantei text-book indicate anterior şi permite recuperarea ne-ambiguă a mesajului criptat (totuşi, acest padding nu are o demonstraţie de securitate în faţa unui adversar activ CCA2 aşa cum are OAEP deci trebuie considerat mai slab ca securitate).

7 SCHEME DE SEMNARE DIGITALĂ

Semnăturile digitale reprezintă echivalentul electronic al semnăturilor de mână, acest concept fiind introdus ca funcţionalitate adiţională a criptosistemelor cu cheie publică de către Diffie şi Hellman în 1976 dar în absenţa unei scheme criptografice pentru acest scop. Obiectivul principal de securitate pe care îl asigură semnăturile digitale îl reprezintă non-repudierea, şi anume faptul că o entitate odată ce a semnat o informaţie nu poate nega că a emis acea informaţie şi orice altă entitate neutră poate verifica acest lucru.

7.1 PROPRIETĂŢI ŞI CLASIFICARE: SEMNĂTURI CU APENDICE ŞI CU RECUPERAREA MESAJULUI

Semnăturile digitale reprezintă deci o valoare numerică care leagă conţinutul unui mesaj de identitatea unei entităţi (mai exact de o cheie privată cu care s-a efectuat semnătura). În cele mai multe cazuri, orice algoritm asimetric poate fi utilizat pentru crearea unei semnături digitale prin inversarea rolului cheii publice cu cea privată, iar primele propuneri de semnături digitale se găsesc în lucrările lui Rivest, Rabin şi ElGamal [67], [65], [33]. Subliniem că şi folosind algoritmi simetrici se pot crea semnături digitale (numite semnături one-time [12], [13], [32], [61]) dar acestea sunt rar utilizate în practică şi nu sunt relevante în contextul prezentei lucrări. Definiţia unei scheme de semnătură digitală este următoarea:

***Definiţia 7.1 (Schemă de semnătură digitală)**. O schemă de semnătură digitală constă într-un triplet de algoritmi: algoritmul generare a cheii* $(PK, SK) \leftarrow Sig.Gen(1^k)$ *care primeşte ca parametru nivelul de securitate* k *şi returnează perechea cheie publică-privată* (PK, SK)*, algoritmul de semnare* $Sig_{SK}(m) \leftarrow Sig.Sign(m, SK)$ *care primeşte mesajul* m *şi cheia privată* SK *şi returnează semnătura* $sig = Sig_{SK}(m)$ *şi algoritmul de verificare* $Sig.Ver(sig, m, PK)$ *care primeşte semnătura* sig*, cheia privată* SK *şi mesajul* m *şi returnează 1 doar dacă semnătura este validă. Toate acestea alături de spaţiile din care provin datele de intrare ale acestora (care fără a pierde generalitatea sunt spaţii ale stringurilor binare).*

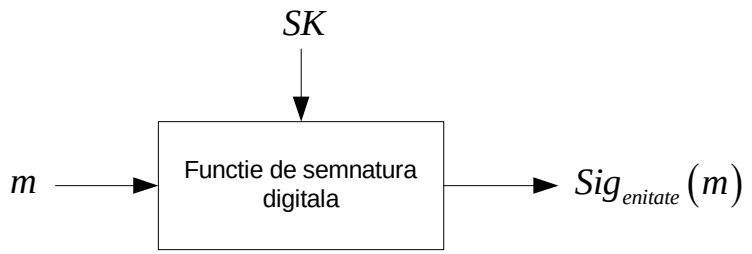
Pentru o semnătură digitală folosim în general notaţia $Sig_A(m)$ prin aceasta înţelegând semnătura entităţii A asupra mesajului m. Semnătură se efectuează întotdeauna folosind cheia privată, din acest motiv putem folosi notaţia $Sig_{SK_A}(m)$ cu sau fără nominalizare entităţii dacă nu este necesară (uneori se foloseşte chiar şi notaţia $E_{SK_A}(m)$, adică criptare cu cheia secretă). În Figura 7.1 se prezintă schema bloc a unei funcţii de semnătură digitală.

Câteva proprietăţi computaţionale ale semnăturilor digitale trebuie amintite: i) trebuie să fie **uşor de produs** doar de către cel care semnează mesajul (funcţia de semnare trebuie să fie uşor de calculat), ii) trebuie să fie **uşor de verificat** de către oricine (funcţia de verificare trebuie să fie uşor de calculat), iii) trebuie să deţină o **durată de viaţă corespunzătoare** (adică semnătura să nu poată fi falsificată până când nu mai este necesară scopului în care a fost creată).

Există două mari categorii distincte de semnături digitale:

i) **Semnături digitale cu recuperarea mesajului** – mesajul poate fi recuperat direct din semnătura digitală. Cel mai simplu exemplu de construcţie este prin inversarea rolului cheii publice şi private în cazul schemei RSA [67]. Pentru a evita fraudarea lor este obligatorie folosirea unei funcţii de redundanţă asupra mesajului după care semnătura propriu-zisă se aplică asupra mesajului redundant. Acest lucru este sugerat în Figura 7.2.

ii) **Semnături digitale cu apendice** (sau cu anexă) – semnături digitale din care mesajul nu poate fi recuperat, drept care este trimis adiţional ca anexă la semnătura digitală. Se pot construi uşor prin aplicarea unei funcţii hash asupra mesajului şi semnarea hash-ului obţinut, vezi Figura 7.3. Datorită eficienţei computaţionale în semnarea mesajelor de dimensiuni mari (deoarece se semnează efectiv doar hash-ul mesajului care are în jur de o sută, două sute de biţi indiferent de lungimea mesajului), aceste semnături sunt cele mai utilizate în practică. Totodată orice schemă de semnătură digitală cu recuperarea mesajului poate fi uşor convertită în semnătură cu apendice.

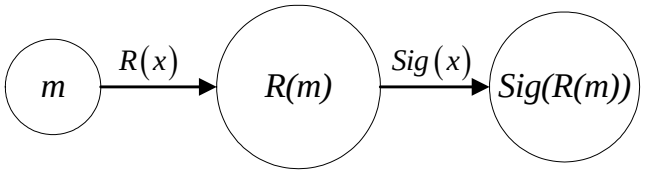


$$|m| = *$$
$$|SK| = ct.\ , e.g.,\ 1024 - 4096 \qquad \text{(ECC de la 160)}$$
$$|Sig_A(m)| \approx |SK|., e.g., 1024 - 4096$$

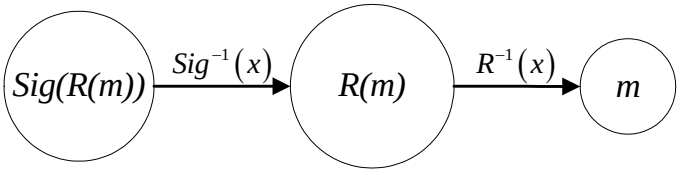
FIGURA 7.1. SCHEMA BLOC A UNEI FUNCŢII DE SEMNĂTURĂ DIGITALĂ CU ANEXĂ.

De asemenea semnăturile digitale pot fi clasificate în **semnături deterministe** respectiv **semnături randomizate (non-deterministe)** după cum algoritmul de semnare foloseşte valori aleatoare şi returnează sau nu aceeaşi semnătură pentru acelaşi mesaj de fiecare dată.

$$|m| < |R(m)| \leq |Sig(R(m))|$$



A) Mecanismul de semnare cu recuperarea mesajului



B) Mecanismul de verificare a unei semnaturi cu recuperarea mesajului
(mesajul este recuperat din semnatura si pentru validitatea semnaturii
se verifica daca mesajul contine bitii corespunzatori functiei de redundanta)

FIGURA 7.2. SCHEMA DE PRINCIPIU A UNEI SEMNĂTURI CU RECUPERAREA MESAJULUI.

O altă clasificare a algoritmilor de semnătură mai poate fi în **scheme de semnătură de unică folosinţă (one-time)** sau pentru folosire multiplă (multiple-time). Menţionăm că toţi algoritmii prezentaţi în acest capitol sunt algoritmi cu folosire multiplă, cei pentru semnături de unică folosinţă nefiind prezentaţi în cadrul acestei lucrări în special datorită absenţei lor în practică.

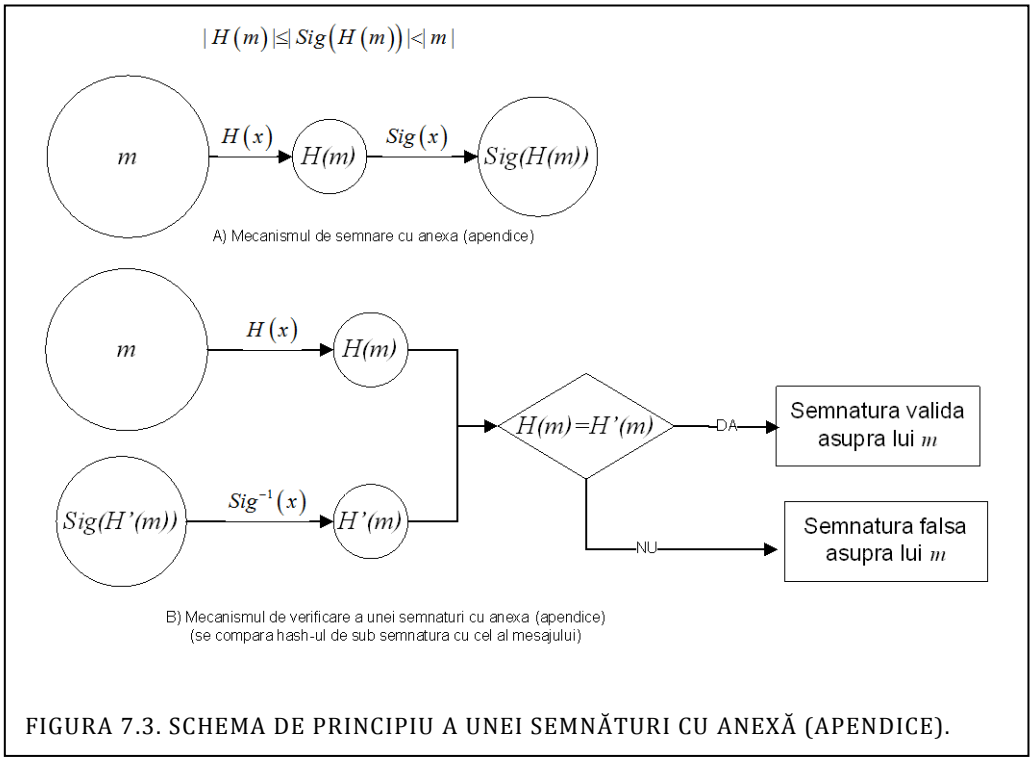


FIGURA 7.3. SCHEMA DE PRINCIPIU A UNEI SEMNĂTURI CU ANEXĂ (APENDICE).

7.2 SEMNĂTURA DIGITALĂ RSA (VARIANTA TEXT-BOOK)

Principiul constructiv care stă la baza semnăturii digitale RSA este inversarea rolului cheii publice şi private pentru a transforma algoritmul de criptare în algoritm de semnătură digitală.

Este mai utilă în scop didactic prezentarea semnăturii RSA cu recuperarea mesajului. Subliniem că generarea cheii se face identic ca în cazul criptării asimetrice urmând ca diferenţele să apară la semnare unde se utilizează funcţia $f^{-1}(x) = x^{\delta} \bmod n$ precum şi funcţia de redundanţă şi prin analogie la verificare se utilizează $f(x) = x^{\varepsilon} \bmod n$ apoi din nou funcţia de redundanţă. Rolul principal al funcţiei de redundanţă este de a preveni atacuri, precum ar fi cele de tip fals existenţial.

$RSA.Gen(1^k)$: *Generează două numere prime aleatoare* p, q *şi calculează* $n = pq$ *respectiv* $\phi(n) = (p-1)(q-1)$ *(presupunem că numerele* p *şi* q *au fost generate în aşa fel încât* n *are* k *biţi). Generează un întreg* ε *astfel încât* $c.m.m.d.c.(\varepsilon, \phi(n)) = 1$, *calculează* δ *astfel încât* $\varepsilon\delta \equiv 1 \bmod \phi(n)$. *Cheia publică este* $PK \leftarrow (n, \varepsilon)$ *iar cea privată este* $SK \leftarrow (n, \delta)$.

$RSA.Sign(m, SK)$: *Calculează* $\overline{m} = R(m)$ *şi* $s = \overline{m}^{\delta} \bmod n$. *Returnează* $s \leftarrow \overline{m}^{\delta} \bmod n$.

$RSA.Ver(s, PK)$: *Se obţine cheia publică* n, ε *a entităţii în cauză şi se calculează* $\overline{m} = s^{\varepsilon} \bmod n$ *şi verifică dacă mesajul aparţine spaţiului de redundanţă, returnează 0 în caz contrar altfel returnează 1 şi recuperează mesajul ca* $m = R^{-1}(\overline{m})$.

SISTEMUL 7-1. SCHEMA DE SEMNARE DIGITALĂ RSA CU RECUPERAREA MESAJULUI.

Subliniem că în practică nu se utilizează semnătura digitală RSA cu recuperarea mesajului ci cea cu anexă care se construieşte peste o funcţie hash. În general se preferă semnăturile cu anexă în practică deoarece sunt mult mai eficiente din punct de vedere al timpului de calcul (evident în special atunci când mesajul semnat are dimensiuni mai mari decât modulul aşa cum este în general cazul).

$RSA.Gen\left(1^k\right)$: *Identic cu cazul semnăturii cu recuperarea mesajului.*

$RSA.Sign\left(m, SK\right)$: *Calculează* $\overline{m} = H\left(m\right)$ *şi* $s = \overline{m}^{\delta} \bmod n$. *Returnează* $s \leftarrow \overline{m}^{\delta} \bmod n$.

$RSA.Ver\left(s, m, PK\right)$: *Se obţine cheia publică* n, ε *a entităţii în cauză, se calculează* $\overline{m} = s^{\varepsilon} \bmod n$ *şi se verifică dacă* $\overline{m} = H\left(m\right)$. *În caz afirmativ returnează 0 iar în caz contrar returnează 1.*

SISTEMUL 7-2. SCHEMA DE SEMNARE DIGITALĂ RSA CU APENDICE

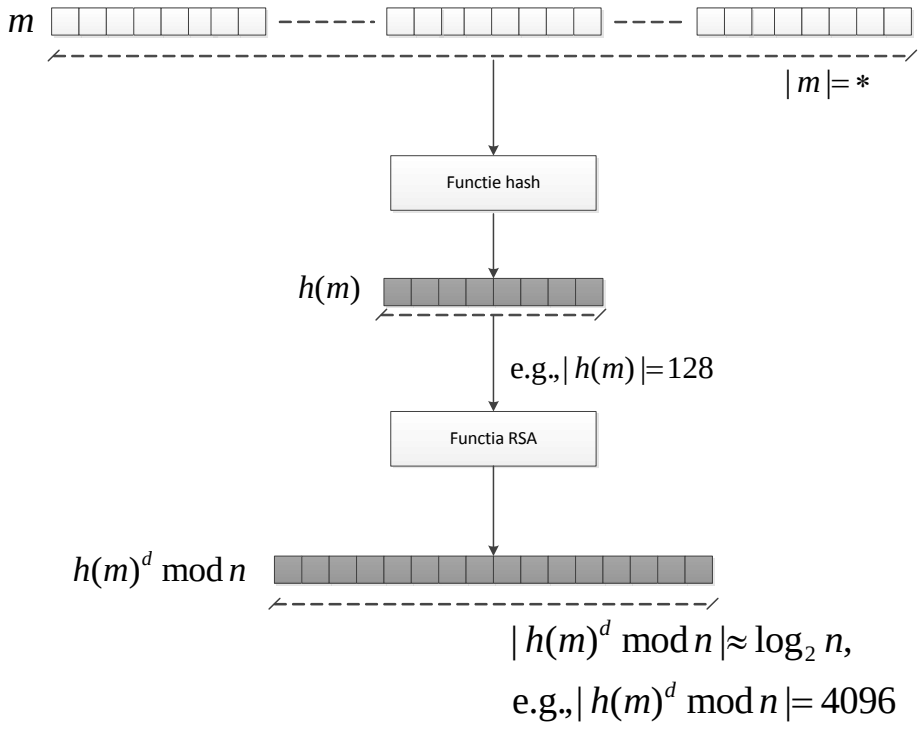


FIGURA 7.4. PARADIGMA HASH-THEN-SIGN

Această paradigmă, care presupune calcularea hashului şi semnarea acestuia, se numeşte **paradigma hash-then-sign**. Pentru RSA schema este ilustrată în Figura 7.4. Sistemul 7.1 descrie schema de semnare RSA cu recuperarea mesajului iar Sistemul 7.2 schema de semnare RSA cu apendice.

7.3 VARIANTE SIGURE ÎN PRACTICĂ PENTRU SEMNĂTURĂ RSA: FDH, PSS, PKCS V.1.5

Aşa cum în cazul criptării RSA era doar padding-ul OAEP cel care făcea criptarea să fie sigură, este şi cazul semnăturii RSA care devine sigură doar în momentul în care este corect folosită. Există diverse metode prelucrare a mesajului înaintea aplicării exponenţierii folosind exponentul privat RSA, trei dintre aceste variante sunt de mare relevanţă practică şi totodată sigure: **Full Domain Hash (FDH)**, **Probabilistic Standard Signature (PSS)** şi **PKCS v.1.5**.

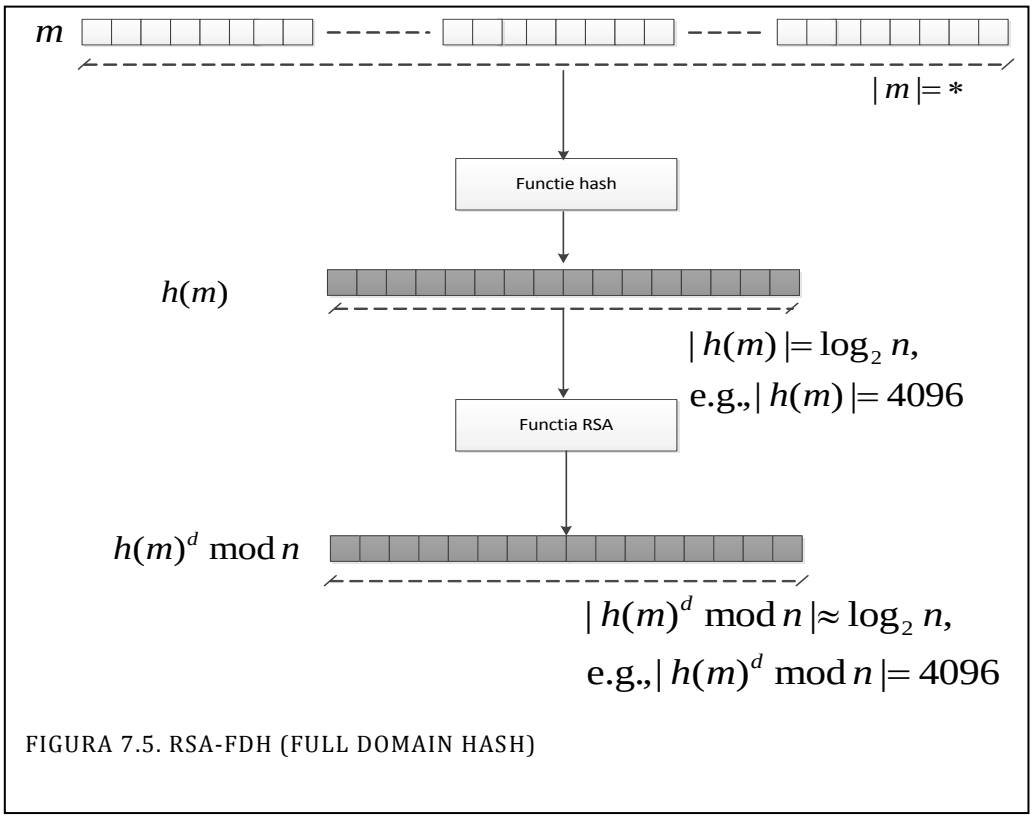


FIGURA 7.5. RSA-FDH (FULL DOMAIN HASH)

Cea mai sigură dintre acestea este PSS, varianta din PKCS1 discutată în continuare este o adaptare a variantei iniţiale propuse de Bellare şi Rogaway, iar cele mai simple FDH şi PKCS v.1.5 dar mai puţin sigure.

Varianta FDH se bazează pe utilizarea unei funcţii hash care creează un hash al mesajului de dimensiune egală cu modulul, apoi se continuă cu semnarea standard după cum sugerează Figura 7.5.

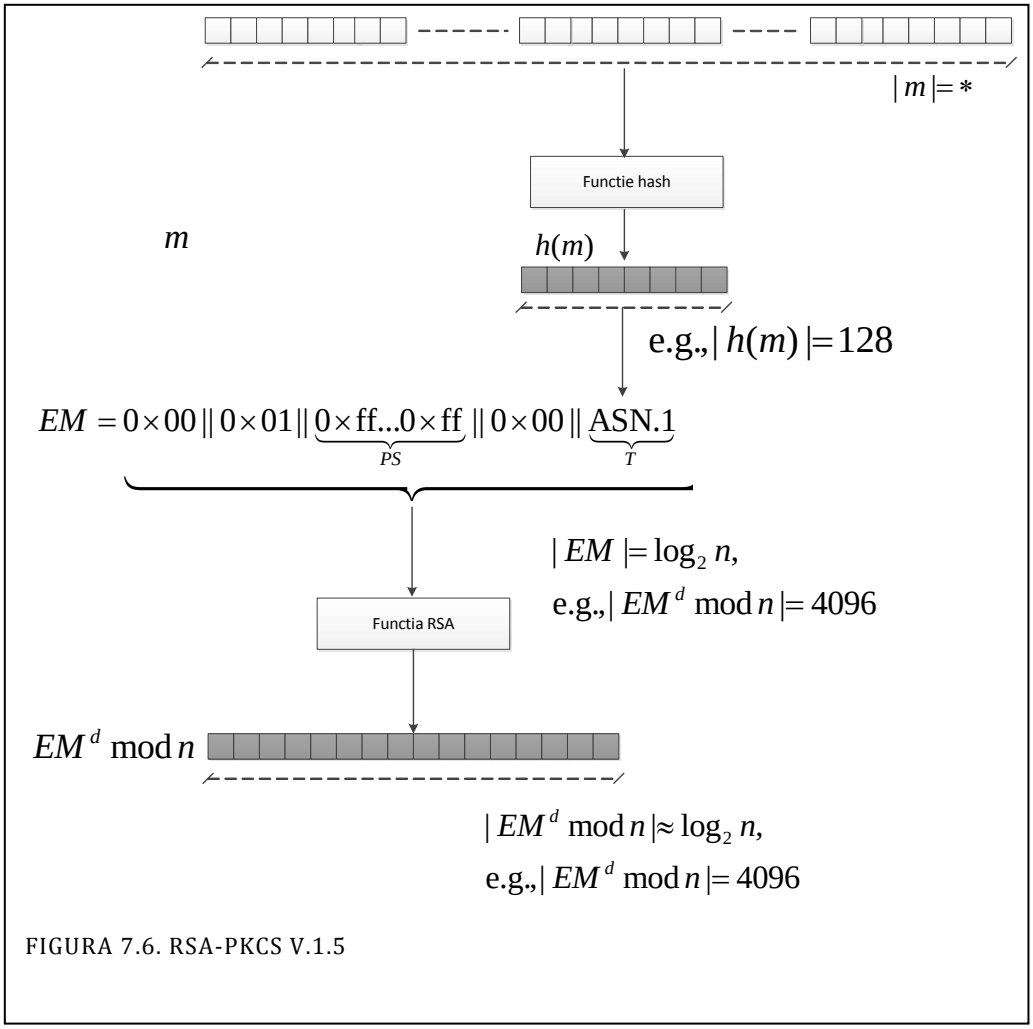


FIGURA 7.6. RSA-PKCS V.1.5

Varianta PKCS v.1.5 foloseşte un hash standard şi adaugă mesajului un simplu padding, după cum sugerează Figura 7.6. Conform standardului PKCS1, PSS şi PKCS v.1.5 aplică transformarea RSA (decriptare) asupra mesajului codat (Encoded Message) notat cu EM. Sunt relevant de remarcat diferenţele dar şi

similitudinile cu paddingul PKCS ilustrat în secţiunea anterioară. Astfel la criptare valoarea lui PS este random în timp ce la semnare are o valoare fixă (biţi de 1), mai diferă şi cea de a doua constantă a paddingului (0x02 la criptare vs. 0x01 la semnare). Structura celor două paddinguri (pentru criptare şi semnare) este însă identică. Valoarea hashului de mesaj este transformată într-o valoare **ASN.1 (Abstract Syntax Notation)**, aceasta este doar o simplă codificare standard care pe lângă valoarea hashului include si denumirea algoritmului cu care s-a făcut semnarea (MD2, MD5, SHA-1, etc.); deci poate fi interpretată ca fiind concatenarea hashului cu numele algoritmului de hashing (pentru detalii poate fi consultat standardul PKCS aferent).

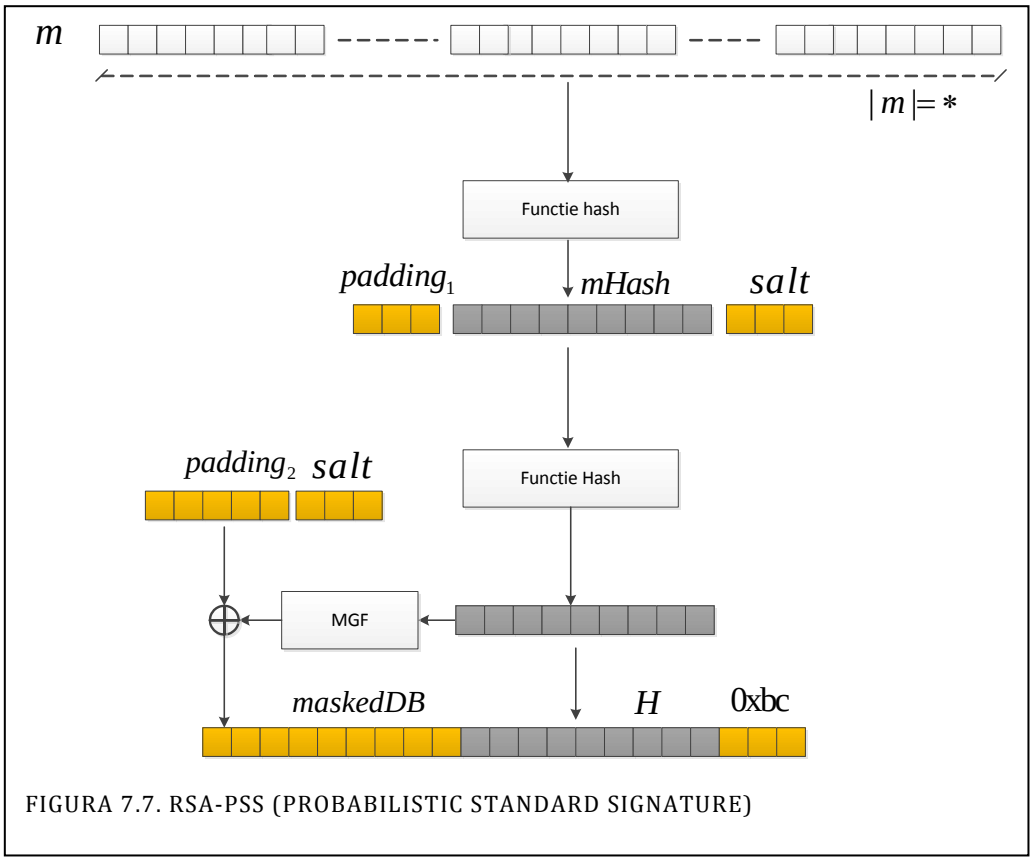


FIGURA 7.7. RSA-PSS (PROBABILISTIC STANDARD SIGNATURE)

PSS aplică de două ori o funcţie hash, de două ori padding şi mai foloseşte şi o funcţie de generare mască (MGF – Mask Generation Function) aşa cum este sugerat în Figura 7.6. Funcţia MGF este implementată tot pe o funcţie hash. Valoarea de salt este aleasă random ceea ce face semnătura să fie non-

deterministă, în timp ce valorile de padding sunt fixe. Varianta de PSS conform standardului PKCS pe care am prezentat-o este o modificare minoră a originalului PSS propus de Bellare şi Rogaway [7].

Toate aceste variante de padding se pot aplica şi în cazul semnăturii Rabin sau variantei Rabin-Williams ce este discutată în continuare.

7.4 SEMNĂTURA DIGITALĂ RABIN

Posibilitatea de a calcula rădăcini pătrate în Z_n poate fi utilizată pentru construcţia unui algoritm de semnătură digitală. Descrierea schemei de semnătură digitală Rabin este în Sistemul 7.3.

Rabin.Gen$\left(1^k\right)$*: Generează două numere prime aleatoare* p*,* q *şi calculează* $n = pq$*. Cheia publică este* $PK \leftarrow (n)$ *iar cea privată este* $SK \leftarrow (p,q)$*.*

Rabin.Sign(m, SK)*: Calculează* $\overline{m} = R(m)$ *şi o rădăcină pătrată a sa* s*, i.e.* $\overline{m} \leftarrow s^2 \bmod n$*. Returnează* s*.*

Rabin.Ver(s, PK)*: Se obţine cheia publică a entităţii în cauză şi se calculează* $\overline{m} = s^2 \bmod n$ *şi verifică dacă mesajul aparţine spaţiului de redundanţă, i.e.* $\overline{m} \in M_R$*, returnează 0 în caz contrar altfel returnează 1 şi recuperează mesajul ca* $m = R^{-1}(\overline{m})$*.*

SISTEMUL 7-3. SCHEMA DE SEMNARE DIGITALĂ RABIN

Semnătura digitală Rabin are o mare problemă la pasul de semnare şi anume aceea că mesajul pe care se aplică semnătura trebuie să fie un reziduu

cvadratic (pătrat perfect). Rezolvarea acestei probleme nu este foarte elegantă singura soluţie fiind modificarea aleatoare a unor biţi până când mesajul devine reziduu cvadratic (astfel, pentru funcţionare corectă, funcţia de redundanţă trebuie să aibă caracter aleator iar în schemă să se permită generarea mesajului redundant până când acesta este pătrat perfect). Pentru a înlătura această deficienţă Williams a introdus o schemă similară uşor modificată care funcţionează în mod determinist. Sistemul 7.4 descrie această schemă.

$RabinWilliams.Gen\left(1^k\right)$: *Generează două numere prime aleatoare* $p \equiv 3 \bmod 7$, $q \equiv 3 \bmod 8$ *şi calculează* $n = pq$, $d = \frac{n - p - q + 5}{8}$. *Cheia publică este* $PK \leftarrow \left(n\right)$ *iar cea privată este* $SK \leftarrow \left(n, d\right)$.

$RabinWilliams.Sign\left(m, SK\right)$: *Calculează* $\overline{m} = 16m + 6$, *se observă că s-a ales* $R\left(m\right) = 16m + 6$, *şi* $J = \left(\frac{\overline{m}}{n}\right)$. *Dacă* $J = 1$ *atunci calculează* $s = \overline{m}^d \bmod n$ *altfel calculează* $s = \left(\frac{\overline{m}}{2}\right)^d \bmod n$. *Returnează* s.

$RabinWilliams.Ver\left(s, PK\right)$: *Se obţine cheia publică a entităţii în cauză şi se calculează* $m' = s^2 \bmod n$. *În funcţie de congruenţa modulo 8 a acestei valori se calculează: 1.* $m' \equiv 6 \bmod 8$ *atunci* $\overline{m} = m'$ *2.* $m' \equiv 3 \bmod 8$ *atunci* $\overline{m} = 2m'$ *3.* $m' \equiv 7 \bmod 8$ *atunci* $\overline{m} = n - m'$ *4.* $m' \equiv 2 \bmod 8$ *atunci* $\overline{m} = 2\left(n - m'\right)$ *şi verifică dacă mesajul aparţine spaţiului de redundanţă, i.e.* $\overline{m} \in M_R$. *Recuperează mesajul* m *ca* $m = R^{-1}\left(\overline{m}\right) = \frac{\overline{m} - 6}{16}$.

SISTEMUL 7-4. SCHEMA DE SEMNARE DIGITALĂ RABIN-WILLIAMS

7.5 SEMNĂTURA „BLIND" BAZATĂ PE RSA

Conceptul de **blind-signature** (preferăm utilizarea termenului în limba engleză în locul traducerii de „semnătură oarbă") a fost introdus de Chaum în lucrarea [21] şi de asemenea utilizat mai târziu în [22]. Ideea care stă în spatele acestui concept este foarte simplă şi utilă. Scenariul urmărit este următorul: o entitate doreşte să obţină semnătura altei entităţi, desigur cu consimţământul acesteia, asupra unui mesaj fără ca să facă cunoscut mesajul sau semnătura către entitatea în cauză (posesoare a cheii private).

Realizarea unui astfel de mecanism este posibilă dacă pot fi găsite două funcţii λ şi μ astfel încât având algoritmul de semnătură, pe care aici preferăm să îl vedem ca pe o funcţie aplicată unui mesaj $S(m)$ să satisfacă următoarea relaţie $\mu\left(S\left(\lambda(m)\right)\right)=S(m)$. Funcţia λ poartă denumirea de blinding-function iar funcţia μ de unblinding-function, denumirile indicând foarte clar rolul acestora. Pentru cazul în care se utilizează semnătura digitală RSA putem alege $\lambda(m)=m\cdot k^{\varepsilon}$ şi $\mu(m)=m\cdot k^{-1}$ iar funcţia de semnare RSA fiind $S(m)=m^{\delta}$ este evident că $\mu\left(S\left(\lambda(m)\right)\right)=S(m)$ va fi satisfăcută. Protocolul de blind-signature care rezultă în acest fel între două entităţi A şi B este următorul (preferăm descrierea sub formă de protocol şi nu de semnătură digitală conform formalismului anterior introdus deoarece este mai simplu de urmărit):

1. *A: Alege o valoare aleatoare* k *şi calculează* $m^{*}=m\cdot k^{\varepsilon} \bmod n$
2. $A\rightarrow B:\ m^{*}$
3. $B\rightarrow A:\ s^{*}=\left(m^{*}\right)^{\delta} \bmod n$ *(practic B semnează* m^{*} *folosind RSA)*
4. *A: Calculează* $s=s^{*}\cdot k^{-1} \bmod n$ *care este de fapt semnătura originală* s *asupra lui* m*.*

Cititorul poate uşor imagina scenarii practice în care o astfel de schemă de semnătură digitală poate fi utilă iar funcţionarea mecanismului prezentat este evidentă şi nu necesită explicaţii suplimentare.

7.6 SEMNĂTURA DIGITALĂ ELGAMAL

Tot dificultatea de a calcula logaritmi discreţi conduce la posibilitatea construcţiei unui algoritm de semnătură digitală aşa cum a observat ElGamal. Este interesant însă că au trecut câţiva ani de la publicarea schimbului de cheie Diffie-Hellman până când această semnătură bazată tot pe logaritmi discreţi să fie descoperită. Explicaţia este faptul că schema de semnare nu este foarte intuitivă aşa cum se poate observa chiar din descrierea ei. Sistemul 7.5 descrie schema de semnare digitală ElGamal.

$ElGamal.Gen\left(1^k\right)$: *Generează un număr prim* p, *un generator* g *al grupului* Z_p, *un întreg aleator* $1<a<p-2$ *şi calculează* $g^a \bmod p$. *Cheia publică este* $PK \leftarrow \left(g, y=g^a \bmod p, p\right)$ *iar cea privată este* $SK \leftarrow (a)$.

$ElGamal.Sign\left(m, SK\right)$: *Generează un întreg aleator* $1<k<p-2$ *cu* $c.m.m.d.c.\left(k, p-1\right)=1$, *calculează* $r=g^k \bmod p$, $k^{-1} \bmod (p-1)$ *şi* $s=k^{-1}\left\{h(m)-ar\right\} \bmod (p-1)$. *Semnătura digitală a mesajului este perechea* (r,s). *Returnează* $sig \leftarrow (r,s)$.

$ElGamal.Ver\left(s, m, PK\right)$: *Se obţine cheia publică a entităţii* $PK \leftarrow \left(g, g^a \bmod p, p\right)$. *Verifică dacă* $1<r<p-1$ *şi în caz contrar respinge semnătura. Calculează* $v_1 = y^r r^s \bmod p$, $v_2 = g^{h(m)} \bmod p$ *şi acceptă semnătura dacă şi numai dacă* $v_1 = v_2$.

SISTEMUL 7-5. SCHEMA DE SEMNARE DIGITALĂ ELGAMAL

7.7 SEMNĂTURA DIGITALĂ DSA

Digital Signature Algorithm (DSA) este standardul de semnătură digitală propus de NIST în 1991 prin FIPS 186 Digital Signature Standard (DSS). În fapt DSA este o variantă a semnăturii ElGamal. Se diferenţiază însă prin câteva restricţii care conduc cel puţin la dimensiune mai redusă a semnăturii. Iniţial modulul p, număr prim, era restricţionat ca valoare între 512 şi 1024 de biţi (în NET de exemplu, din acest motiv în timp ce cu RSA se pot face semnături şi la 4096 de biţi, DSA nu permite decât până în 1024).

$DSA.Gen\left(1^k\right)$: *Generează un număr prim* p *în intervalul* $\left(2^{1023}, 2^{1024}\right)$ *astfel încât* $p-1$ *să fie divizibil cu un număr prim* q *în intervalul* $\left(2^{159}, 2^{160}\right)$. *Alege un număr aleator* g *al cărui ordin în* Z_p *este* q, *un număr aleator* a *în intervalul* $(0,q)$ *şi calculează* $g^a \bmod p$. *Cheia publică este* $PK \leftarrow \left(p,q,g,g^a\right)$ *iar cea privată este* $SK \leftarrow (a)$.

$DSA.Sign(m, SK)$: *Generează un întreg aleator* $1<k<q$, *calculează* $r = g^k \bmod p \bmod q$ *şi* $s = k^{-1}\{h(m) - ar\} \bmod q$. *Semnătura digitală a mesajului este perechea* (r,s). *Returnează* $sig \leftarrow (r,s)$.

$DSA.Ver(sig, m, PK)$: *Se obţine cheia publică a entităţii* $PK \leftarrow \left(p,q,g,g^a\right)$. *Verifică dacă* $1<r<q$ *şi* $1<s<q$ *iar în caz contrar respinge semnătura. Calculează* $w = s^{-1} \bmod q$, $u_1 = H(m) \cdot w \bmod q$, $u_2 = r \cdot w \bmod q$, $v = \left(\left(g^{u_1} \cdot y^{u_2}\right) \bmod p\right) \bmod q$ *şi acceptă semnătura dacă şi numai dacă* $v = r$.

Modulul p este ales astfel încât $p-1$ are un divizor prim q care conform variantei FIPS 186-3 are 160 de biţi dacă modulul este de 1024 biţi, respectiv 224 pentru module de 2048 de biţi şi 256 pentru module de 2048 sau 3072 de biţi. Mai departe, nu se foloseşte un generator al lui Z_p ci un număr care are ordinul

q acest lucru conducând la o semnătură mai compactă (din moment ce generatorul are ordinul de 160-256 de biţi, componentele semnăturii vor avea tot atâţia biţi şi nu numărul de biţi ai lui p ca în cazul semnăturii ElGamal). Descrierea schemei DSA, care este similară cu algoritmul ElGamal pentru cazul unui modul de 1024 de biţi, este în Sistemul 7.6.

Conform standardului funcţia h trebuie să fie o funcţie hash aprobată pentru semnături digitale, de exemplu SHA-1 care este una dintre cele mai comune alegeri. Parametrii pot fi partajaţi între mai multe entităţi în sensul că tripletul (p,q,g) poate fi folosit de mai mulţi participanţi în timp ce doar x este secret. Se observă că spre deosebire de semnătura ElGamal, valorile sunt reduse modulo q, în rest schema este similară.

$ECDSA.Gen(1^k)$: *Alege un număr prim* p, *coeficienţii* a *şi* b *ai unei curbe eliptice* $E: y^2 = x^3 + ax + b \bmod p$. *Alege un punct* P *al cărui ordin este* q, *un număr aleator* d *în intervalul* $(0,q)$ *şi calculează* dP. *Cheia publică este* $PK \leftarrow (p,a,b,xP)$ *iar cea privată este* $SK \leftarrow (d)$.

$ECDSA.Sign(m, SK)$: *Generează un întreg aleator* $1 < k < q$, *calculează* $r = kP$ *şi* $s = k^{-1}\{h(m) - dr\} \bmod q$. *Semnătura digitală a mesajului este perechea* (r,s). *Returnează* $sig \leftarrow (r,s)$.

$ECDSA.Ver(sig, m, PK)$: *Se obţine cheia publică a entităţii* $PK \leftarrow (p,a,b,dP)$. *Verifică dacă* $1 < r < q$ *şi* $1 < s < q$ *iar în caz contrar respinge semnătura. Calculează* $w = s^{-1} \bmod q$, $u_1 = H(m) \cdot w \bmod q$, $u_2 = r \cdot w \bmod q$, $v = u_1 P + u_2 dP$ *şi acceptă semnătura dacă şi numai dacă coordonata* x *a lui* r *este egală cu coordonata* x *a lui* v.

7.8 SEMNĂTURA DIGITALĂ FOLOSIND CURBE ELIPTICE (ECDSA)

Eliptic Curve Digital Signature Algorithm (ECDSA), Sistemul 7.7, este varianta lui DSA care foloseşte curbe eliptice. Avantajul este că oferă dimensiuni ale cheii publice mult mai mici. De exemplu dacă la DSA cheia publică era de 1024 de biţi, folosind ECDSA cheia scade până la 160 de biţi fără să schimbe nivelul de securitate. Descrierea algoritmului este identică cu DSA cu excepţia că acum se generează parametrii curbei, la fel cum am arătat în cadrul schimbului de cheie Diffie-Hellman pe curbe eliptice (ECDH), iar semnarea se face prin operaţii asupra punctelor curbei.

7.9 TIPURI DE ATAC ASUPRA SEMNĂTURILOR DIGITALE

Discuţia devine şi mai interesantă în momentul în care discutăm despre atacuri asupra semnăturilor digitale. În primul rând raportat la acestea este important să distingem între modurile în care semnăturile pot fi fraudate. Nivelele de fraudare sunt următoarele iar gravitatea lor creşte de la i la iv:

i) **Fals existenţial** – un adversar poate falsifica cel puţin o semnătură dar nu are control total (sau parţial) asupra mesajului semnat.

ii) **Fals selectiv** – un adversar poate falsifica semnătura asupra unui anumit tip de mesaje.

iii) **Fals universal** – adversarul poate calcula semnături digitale asupra oricărui mesaj (totuşi nu cunoaşte cheia cu care mesajele sunt semnate).

iv) **Spargere totală** - adversarul este în posesia cheii private şi poate falsifica orice semnătură.

În ceea ce priveşte atacul efectiv asupra sistemului de semnare digitală, distingem următoarele cazuri în funcţie de posibilităţile de acces ale adversarului la maşina de semnare digitală:

i) **Atacuri bazate doar pe cheie** – este atacul în care adversarul are acces doar la cheia publică, de verificare, a semnăturii digitale şi nu are acces la maşina de semnare.

ii) **Mesaj cunoscut** – adversarul are acces la mesaje semnate de posesorul cheii private, dar aceste mesaje nu sunt la alegerea lui.

iii) **Mesaj ales** – adversarul poate obţine semnătura pentru un anumit număr de mesaje la alegerea sa.

iv) **Mesaj ales adaptiv** – adversarul obţine semnături digitale la alegerea sa pentru un anume set de mesaje, iar acest lucru se desfăşoară interactiv.

Şi în cazul atacului iv), la fel ca în cazul atacurilor de tip criptotext ales adaptiv, posesorul cheii private nu vrea să semneze un anume mesaj ţintă cunoscut atât de el cât şi de adversar, dar este dispus să semneze alte mesaje pe baza cărora adversarul trebuie să construiască semnătura asupra mesajului ţintă. Atacurile ii), iii) şi iv) sunt atacuri în care adversarul are acces la maşina de semnare digitală. Prima metodă de semnare digitală rezistentă în faţa unor atacuri active a fost introdusă în [45].

8 FUNDAMENTE MATEMATICE

> "Nowadays, when a Number Theorist applies for a grant, he says that Number Theory is used in cryptography, and so doing Number Theory is good for National Security. Back then, since it was before the discovery of America, they said Number Theory is used in music. But I won't comment on the progress of civilization since then." – H.W. Lenstra[6].

8.1 ELEMENTE DE TEORIA PROBABILITĂŢILOR

Unul dintre domeniile la care se face frecvent apel în descrierea criptosistemelor este teoria probabilităţilor, cartea de referinţă în domeniu este cea a lui Feller [34]. Aşa cum s-a observat anterior, până şi definiţia unei funcţii one-way, care este o noţiune de bază a criptografiei, utiliza noţiunea de probabilitate. În practică nu avem de a face cu criptosisteme care nu pot fi sparte, ci cu criptosisteme despre care se poate spune pe drept cuvânt că probabilitatea de a fi sparte este neglijabilă.

Definim un eveniment E ca fiind rezultatul unui experiment S şi notăm probabilitatea ca evenimentul E să aibă loc cu $\Pr(E)$. Putem clasifica două evenimente E_1 şi E_2 în:

i) **Independente** dacă evenimentele nu sunt legate unul de altul, deci apariţia unuia nu influenţează apariţia celuilalt, i.e. $\Pr(E_1 \cap E_2) = \Pr(E_1) \cdot \Pr(E_2)$ (se mai numeşte şi regula produs pentru **evenimente simultane**).

---

[6] Preluat de la http://www.matem.unam.mx/~magidin/lenstra.html.

ii) **Mutual exclusive** dacă evenimentele nu pot avea loc simultan, adică apariţia unuia face imposibilă apariţia celuilalt, i.e. $\Pr(E_1 \cap E_2) = 0$.

iii) **Complementare** dacă unul nu apare implică apariţia celuilalt şi invers, i.e. $\Pr(E_1) + \Pr(E_2) = 1$.

Se poate uşor observa că pentru două evenimente E_1 şi E_2 probabilitatea ca oricare sau amândouă să apară este egală cu $\Pr(E_1 \cup E_2 \cup (E_1 \cap E_2)) = \Pr(E_1) + \Pr(E_2) - \Pr(E_1) \cdot \Pr(E_2)$. Pentru cazul în care valoarea lui $\Pr(E_1) \cdot \Pr(E_2)$ este neglijabilă (sau nulă în cazul evenimentelor mutual exclusive) se poate aproxima $\Pr(E_1 \cup E_2) = \Pr(E_1) + \Pr(E_2)$.

Relevantă este şi definirea probabilităţii condiţionale între evenimente.

Definiţia 8.1. *(Probabilitate condiţională) Pentru două evenimente* E_1 *şi* E_2 *definim probabilitatea condiţională, adică probabilitatea ca* E_1 *să fie produs de* E_2 *ca fiind* $\Pr(E_1 | E_2) = \dfrac{\Pr(E_1 \cap E_2)}{\Pr(E_2)}$.

Un rezultat imediat şi foarte relevant legat de probabilitatea condiţională a două evenimente este Teorema lui Bayes.

Teorema 8.1. *(Teorema lui Bayes) Pentru două evenimente* E_1 *şi* E_2 *avem* $\Pr(E_1 | E_2) = \dfrac{\Pr(E_1) \Pr(E_2 | E_1)}{\Pr(E_2)}$.

Aplicarea teoremei lui Bayes este imediată în probleme care întreabă care este probabilitatea unui eveniment dat fiind că s-a întâmplat un anume eveniment. Una dintre aplicaţiile frecvente în practică sunt filtrele anti-spam.

Trebuie avută atenţie sporită deoarece de multe ori rezultatul este contra-intuitiv. Un bun exemplu este **paradoxul cutiilor** a lui Bertrand. Având trei cutii fiecare conţinând 2 monede după cum urmează: prima două de argint, a doua două de aur şi a treia una de aur şi una de argint, dacă am ales o cutie şi în ea este o monedă de aur, care este probabilitatea ca şi cealaltă să fie tot de aur? Un răspuns pripit este ½ plecând de la faptul că dacă am o monedă de aur atunci: fie este cutia aur-argint fie aur-aur şi deci sunt în una din cele 2 situaţii ce conduce la probabilitate ½. Răspunsul este însă greşit, deoarece probabilitatea este 2/3 dat fiind că în momentul în care am ales aur mă aflu în una din cele 3 variante: prima

monedă din cutia aur-aur, a doua monedă din cutia aur-aur, moneda aur din cutia aur-argint. Două din cele trei variante anterioare sunt câştigătoare şi deci 2/3 este răspunsul corect. Acest rezultat poate fi explicat şi prin prisma teoremei lui Bayes. Avem de calculat: $\Pr\left(2\times Aur \middle| 1\times Aur\right) = \dfrac{\Pr\left(2\times Aur\right)\Pr\left(1\times Aur \middle| 2\times Aur\right)}{\Pr\left(1\times Aur\right)}$.

Desigur $\Pr\left(2\times Aur\right)=1/3$, $\Pr\left(1\times Aur \middle| 2\times Aur\right)=1$ în timp ce $\Pr\left(1\times Aur\right)=1/2$ deoarece sunt 6 variante de monedă (2 monede x 3 cutii) din care 3 sunt de aur (deci 3/6). Înlocuind în relaţie conduce la 2/3 aşa cum am arătat şi anterior. Rezultatul se poate explica identic şi folosind definiţia probabilităţii condiţionale.

Cele mai frecvente experimente sunt cele care au ca rezultat doar două evenimente complementare. Să presupunem că avem un generator de numere aleatoare la ieşirea căruia putem avea fie 0 fie 1, evident e vorba de două evenimente complementare. Pentru cazul în care generatorul este perfect echilibrat probabilitatea de a da 0 sau 1 este egală cu ½. Dacă presupunem că generăm o secvenţă de 12 biţi care este probabilitatea ca exact 7 biţi să fie 1 şi exact 5 să fie 0 indiferent de ordinea în care apar? Răspunsul la această întrebare poate fi uşor dat pe baza distribuţiei binomiale.

Teorema 8.2. *(Distribuţia binomială) Dacă un experiment se soldează cu două evenimente complementare* E_1 *şi* E_2 *prin repetarea experimentului de* n *ori probabilitatea ca evenimentul* E_1 *să fi rezultat de* k *ori (bineînţeles că* E_2 *va rezulta de* $n-k$ *fiind complementar) este* $C_n^k \Pr\left(E_1\right)^k \Pr\left(E_2\right)^{n-k}$.

Rămâne ca exerciţiu pentru cititor să răspundă la întrebarea anterioară, şi să imagineze diverse scenarii pentru cazul în care generatorul nu este perfect echilibrat şi probabilitatea ca ieşirea să fie 0 nu este egală cu cea ca ieşirea să fie 1.

O altă aplicaţie imediată a probabilităţilor în programare este construcţia tabelelor de hashuri (hashtable) în care mai multe valori sunt reţinute indexat după chei de k biţi. Fiecare cheie este generată aleator (sau are la baza un hash al valorii pe care o indexează). Întrebarea este după câte valori introduse în tabelă ne putem aştepta la o coliziune cu probabilitate mai mare de ½ ? Întrebarea este relevantă deoarece de exemplu $k=20$ conduce la peste 1.000.000 de chei posibile şi aparent probabilitatea de coliziune este mică. Totuşi teoria ne arată că nu este aşa, probabilitatea de coliziune fiind în mare ½ odată ce s-au atins aproximativ $\sqrt{2^k}$ elemente. Deci 1000 de elemente sunt suficiente pentru o coliziune cu probabilitate destul de mare. Acest fapt este cunoscut sub numele de **paradoxul zilelor de naştere** deoarece, în mod oarecum surprinzător,

probabilitatea ca într-o camera cu 23 de persoane să existe cel puţin 2 persoane născute în aceeaşi zi este aprox. ½ iar intr-o camera cu 100 de persoane 0.999999.

Problema poate fi abstractizată considerând o urnă cu m bile, şi întrebând care este probabilitatea ca după n extrageri (cu reintroducere) să exista cel puţin 1 coliziune. Evident, există $m^{(n)} = \frac{m!}{n!} = m(m-1)(m-2)...(m-n+1)$ variante de a extrage n bile fără coliziune dintr-un total de m^n variante de extragere. Deci, probabilitatea să nu existe nici o coliziune este $1-\frac{m^{(n)}}{m^n}$. Se poate demonstra că dacă $m \to \infty$ atunci numărul de extrageri până la o coliziune este în medie de $\sqrt{\frac{\pi m}{2}}$.

8.2 ELEMENTE DE ALGEBRĂ: GRUPURI, INELE ŞI CÂMPURI

Pentru a elimina confuzii de limbaj, re-amintim definiţiile câtorva obiecte matematice comun folosite în criptografie: grupuri, câmpuri şi spaţii vectoriale.

Definiţia 8.2. *(Grup şi grup abelian) Un grup este format de o mulţime G alături de o operaţie binară notată $\bullet$ cu proprietăţile: $\bullet$ este asociativă adică $\forall a,b,c \in G, a \bullet (b \bullet c) = (a \bullet b) \bullet c$, există un element identitate notat cu 1 astfel încât $\forall a \in G, a \bullet 1 = a$ şi fiecare element din G are un invers,* i.e., $\forall a \in G, \exists a^{-1}$ astfel încât $a \bullet a^{-1} = 1$. *Dacă este comutativ, grupul se numeşte grup abelian, în acest caz $\forall a,b \in G, a \bullet b = b \bullet a$.*

Definiţia 8.3. *(Inel) Un inel este format de o mulţime I şi două operaţii binare $+$ şi $\times$* cu proprietăţile: *$(I,+)$ formează un grup abelian cu elementul identitate notat cu 0, iar $\times$ este asociativă, admite element identitate şi este distributivă faţă de $+$ adică $\forall a,b,c \in I, a \times (b+c) = a \times b + a \times c$.*

Definiţia 8.4. *(Câmp) Un câmp este un inel comutativ, adică $\forall a,b \in I, a \times b = b \times a$, în care fiecare element cu excepţia elementului 0 acceptă invers faţă de $\times$.*

Definiţia 8.5. *(Câmp finit şi ordinul şi caracteristica unui câmp) Un câmp finit este un câmp cu număr finit de elemente iar numărul de elemente se numeşte ordinul câmpului. Caracteristica unui câmp este cel mai mic întreg* m pentru care $\underbrace{1+1+1+\ldots+1}_{m}=0$ *iar dacă un astfel de întreg nu există atunci caracteristica sa este 0.*

8.3 ELEMENTE DE TEORIA NUMERELOR

Domeniul care furnizează cele mai multe elemente necesare pentru construcţia algoritmilor criptografici asimetrici este Teoria Numerelor, de altfel putem spune că criptografia asimetrică este în fond teoria numerelor aplicată. Toţi algoritmii de criptare cu cheie publică utilizaţi în practică au securitatea bazată pe probleme de Teoria Numerelor cu privire la care nu se cunoaşte o rezolvare eficientă (prin rezolvare eficientă înţelegem un algoritm în timp polinomial). Elementele de Teoria Numerelor se folosesc în general în reducţii de securitate pentru a atinge nivelul de **securitate demonstrabilă** (provable security) ceea ce presupune, aşa cum s-a spus, a demonstra că pentru a sparge un criptosistem este necesară rezolvarea unei probleme de teoria numerelor despre care se ştie că nu poate fi rezolvată eficient. Este oarecum contradictorie noţiunea de securitate demonstrabilă în acest context deoarece de cele mai multe ori cu privire la problema de teoria numerelor nu există în cele din urmă nici o demonstraţie că nu ar putea fi rezolvată în timp polinomial, ci pur şi simplu la nivelul de cunoştinţe actual nu se cunoaşte o astfel de rezolvare. Încercăm în cele ce urmează să sintetizăm cele mai relevante noţiuni din teoria numerelor cu aplicabilitate în criptologie, vor fi prezente demonstraţii pentru teoremele cele mai relevante, iar acolo unde aceste demonstraţii lipsesc, recomandăm cititorului interesat să consulte orice carte de referinţă [2], [49] sau curs în teoria numerelor [86].

8.3.1 MULŢIMEA DE RESTURI ZN

Reamintim, deşi probabil nu este necesar, că un număr prim este un număr care este divizibil doar cu 1 şi cu sine, iar un număr ne-prim îl vom numi compozit, precum şi faptul că există o infinitate de numere prime respectiv de numere compozite. Prin factorizarea unui număr înţelegem descompunerea acestuia în factori primi, factorizarea unui întreg fiind unică până la rearanjarea

termenilor. Vom nota cu $a \equiv b \bmod n$ şi vom citi *„a este congruent cu b modulo n”* dacă resturile obţinute prin împărţirea la *n* a întregilor *a* şi *b* sunt egale.

Definiţia 8.6. *Definim* $Z_n = \{0,1,2,3,\ldots,n-1\} = \{x \in N \mid 0 \le x < n\}$ *ca fiind mulţimea resturilor modulo* n.

Definiţia 8.7. *Definim* $Z_n^* = \{x \in Z_n \mid c.m.m.d.c.(x,n) = 1\}$ *ca fiind mulţimea întregilor din* Z_n *relativ primi la* n.

Se observă că în particular pentru un număr prim p avem $Z_p^* = \{1,2,3\ldots,p-1\}$ deci toate elementele din Z_p mai puţin elementul neutru la adunare.

Remarca 8.1. $(Z_n^*,\cdot)$ formează un grup abelian (simbolul $\cdot$ denotă înmulţire) deoarece următoarele proprietăţi sunt satisfăcute:

i) Legea $\cdot$ este asociativă pentru că $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c \bmod n$.

ii) Legea $\cdot$ este comutativă pentru că $a \cdot b \equiv b \cdot a \bmod n$.

iii) Există element neutru deoarece $a \cdot 1 \equiv 1 \cdot a \equiv a \bmod n$.

iv) Fiecare element are un invers multiplicativ $\forall a \exists a^{-1}$ astfel încât $a \cdot a^{-1} \equiv 1 \bmod n$.

Inversul multiplicativ poate fi eficient calculat cu ajutorul Algoritmului Euclidian Extins care va fi discutat în capitolul 4.

8.3.2 TEOREMA CHINEZĂ A RESTURILOR

Pentru doi întregi oarecare a şi b sunt bine cunoscute noţiunile de cel mai mare divizor comun, ca fiind cel mai mare întreg care divide atât pe a cât şi pe b, respectiv cea de cel mai mic multiplu comun, ca fiind cel mai mic întreg divizibil atât cu a cât şi cu b. Este util să introducem însă şi relaţiile matematice care definesc aceste numere:

Definiţia 8.8. (c.m.m.d.c. şi c.m.m.m.c). *Pentru doi întregi* a,b *a căror factorizare este* $a = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ *respectiv* $b = p_1^{f_1} p_2^{f_2} \ldots p_k^{f_k}$ *cu* $e_i \ge 0,\ f_i \ge 0$ *(subliniem că am introdus în mod forţat în factorizarea celor doi întregi aceleaşi numere prime, permiţând ca exponentul să fie 0) sunt adevărate egalităţile:*

$$c.m.m.d.c.(a,b) = p_1^{\min(e_1,f_1)} p_2^{\min(e_2,f_2)} \ldots p_k^{\min(e_k,f_k)},$$

$$c.m.m.m.c.(a,b) = p_1^{\max(e_1,f_1)} p_2^{\max(e_2,f_2)} \ldots p_k^{\max(e_k,f_k)}.$$

Teorema chineză a resturilor este un rezultat frecvent utilizat în criptografie şi criptanaliză.

Teorema 8.3. *(Teorema Chineză a Resturilor): Fie întregii* $n_1, n_2, \ldots, n_k$ *astfel încât* $c.m.m.d.c.(n_i, n_j) = 1, \forall i \neq j, 0 < i \leq k, 0 < j \leq k$. *Atunci următorul sistem de congruenţe are o soluţie unică* x *în* Z_n *unde* $n = n_1 n_2 \ldots n_k$ *:*

$$\begin{cases} x \equiv a_1 \bmod n_1 \\ x \equiv a_2 \bmod n_2 \\ \quad \ldots\ldots \\ x \equiv a_k \bmod n_k \end{cases}$$

Mai mult, soluţia acestui sistem poate fi calculată, folosind algoritmul lui Gauss, ca fiind $x = \sum_{i=1}^{k} a_i N_i M_i \bmod n$ *unde* $N_i = \frac{n}{n_i}$ *iar* $M_i = N_i^{-1} \bmod n_i$.

Posibilitatea de a calcula inverse multiplicative folosind Algoritmul Euclidian Extins (vezi capitolul 4) duce la posibilitatea rezolvării în timp polinomial a sistemului de congruenţe din această teoremă, deoarece aşa cum se poate observa soluţia necesită calcularea de inverse multiplicative în $Z_{n_i}, i = \overline{1,k}$.

Pentru a sublinia importanţa acestui rezultat să considerăm că o entitate A doreşte să trimită un mesaj criptat către trei entităţi care utilizează pentru criptare trei module diferite n_1, n_2, n_3 şi acelaşi exponent de criptare $\varepsilon = 3$ folosind funcţia RSA (chiar dacă criptosistemul nu a fost încă introdus funcţia de criptare RSA a fost prezentată în 2.2). Mesajele criptate transmise sunt următoarele $c_1 = m^3 \bmod n_1$, $c_2 = m^3 \bmod n_2$, $c_3 = m^3 \bmod n_3$. Potenţialul adversar cunoaşte cheile publice, deci $n_1, n_2, n_3, e = 3$, iar după interceptarea mesajelor criptate are posibilitatea de a construi sistemul:

$$\begin{cases} x \equiv c_1 \bmod n_1 \\ x \equiv c_2 \bmod n_2 \\ x \equiv c_3 \bmod n_3 \end{cases}$$

Conform Teoremei Chineze a Resturilor acesta are o soluţie unică în Z_n cu $n = n_1 n_2 n_3$ şi din moment ce $m < n_1, m < n_2, m < n_3$ înseamnă că $m < \sqrt[3]{n_1 n_2 n_3}$ şi deci $m^3 < n_1 n_2 n_3$. Deci evident soluţia sistemului este chiar $x = m^3$ de unde mesajul original m poate fi extras ca rădăcină cubică a lui x în mulţimea numerelor reale. Deci securitatea transmisiei este compromisă.

8.3.3 TEOREMELE LUI FERMAT ŞI EULER

Mica teoremă a lui Fermat şi dezvoltarea acesteia în teorema lui Euler cu privire la funcţia Euler ϕ reprezintă un rezultat matematic pe baza căruia a fost construit criptosistemul RSA precum şi alte criptosisteme a căror securitate este bazată pe problema factorizării întregilor.

Definiţia 8.9. *Pentru un întreg n definim funcţia Euler Fi $\phi(n)$ ca fiind numărul de întregi pozitivi mai mici decât n şi relativ primi la acesta.*

Din definiţie rezultă că $\phi(n) = |Z_n^*|$ (deci $\phi(n)$ este chiar ordinul grupului Z_n^*). Se poate uşor observa că pentru un număr prim p avem $\phi(p) = p - 1$. De asemenea pentru un exponent întreg pozitiv e avem $\phi(p^e) = p^e\left(1 - \frac{1}{p}\right)$, această relaţie fiind uşor de dedus deoarece din cei p^e întregi mai mici decât p^e este evident că avem exact p^{e-1} întregi care nu sunt relativ primi la p deci $\phi(p^e) = p^e - p^{e-1} = p^e\left(1 - \frac{1}{p}\right)$.

Teorema 8.4. *Pentru orice pereche de întregi pozitivi n, m dacă $c.m.m.d.c.(n, m) = 1$ atunci $\phi(nm) = \phi(n)\phi(m)$.*

Teorema poate fi uşor demonstrată plecând de la Teorema Chineză a Resturilor. Prin definiţie există $\phi(m)$ întregi x astfel încât $0 \le x < m$ cu $c.m.m.d.c.(x, m) = 1$ şi $\phi(n)$ întregi y astfel încât $0 \le y < n$ cu $c.m.m.d.c.(y, n) = 1$. Atunci conform Teoremei Chineze a resturilor există exact un întreg $0 \le z < mn$ astfel încât $z \equiv x \bmod m$ şi $z \equiv y \bmod n$; de asemenea z va fi şi el prim relativ la m şi n. Urmează imediat că numărul de întregi z relativ primi

la m respectiv n care pot fi deduşi în acest fel prin Teorema Chineză a Resturilor este chiar $\phi(m)\phi(n)$.

Următoarea teoremă stabileşte relaţia după care $\phi(n)$ poate fi calculat şi pentru întregi compoziţi.

Teorema 8.5. *Pentru un întreg compozit având factorizarea* $n=\prod_{i=1}^{r} p_i^{e_i}$ *funcţia lui Euler* $\phi(n)$ *poate fi calculată după relaţia:* $\phi(n)=n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\ldots\left(1-\frac{1}{p_r}\right)$.

Demonstraţia acestei teoreme este directă deoarece bazându-ne pe faptul că $\phi(nm)=\phi(n)\phi(m)$ dacă $c.m.m.d.c.(n,m)=1$ precum şi pe faptul că $\phi(p^e)=p^e\left(1-\frac{1}{p}\right)$ din moment ce $n=\prod_{i=1}^{r} p_i^{e_i}$ rezultă imediat că

$$\Rightarrow \phi(n)=\phi\left(\prod_{i=1}^{r} p_i^{e_i}\right)=\prod_{i=1}^{r}\phi\left(p_i^{e_i}\right)=\prod_{i=1}^{r} p_i^{e_i}\left(1-\frac{1}{p_i}\right)=n\prod_{i=1}^{r}\left(1-\frac{1}{p_i}\right).$$

Teorema 8.6. *(Mica Teoremă a lui Fermat). Fie* p *un număr prim şi* a *un întreg astfel încât* $c.m.m.d.c.(a,p)=1$ *atunci* $a^{p-1}\equiv 1 \bmod p$.

Demonstraţie: Poate cea mai simplă demonstraţie poate fi făcută plecând de la observaţia că $x^p\equiv(x-1+1)^p\equiv(x-1)^p+1 \bmod p$ aceasta deoarece toţi coeficienţii binomiali sunt divizibili cu p, fiind de forma C_p^k, mai puţin coeficienţii termenilor de rang p. Dar $(x-1)^p$ poate fi scris din nou ca $(x-2+1)^p\equiv(x-2)^p+1 \bmod p$ deci $x^p\equiv(x-2)^p+1+1 \bmod p$. Acelaşi lucru putem să îl facem şi cu $(x-2)^p$ şi procedeul se poate repeta până ajungem la 1^p. Se poate uşor observa că vom ajunge la $x^p\equiv\underbrace{1^p+\ldots+1+1}_{x-ori} \bmod p$ - ceea ce trebuia demonstrat.

O consecinţă imediată a teoremei 8.6 este faptul că dacă $r\equiv s \bmod(p-1)$ atunci $a^r\equiv a^s \bmod p$, deci când se lucrează modulo p exponenţii pot fi reduşi modulo $p-1$ acest fapt fiind de mare importanţă în criptografie.

Teorema este un caz particular al următoarei teoreme introduse de Euler (în acest sens putând fi utilizată şi demonstraţia teoremei 8.7 pentru a demonstra Teorema 8.6).

Teorema 8.7. *(Teorema lui Euler). Fie un întreg oarecare* $n \geq 2$ *şi* $a \in Z_n^*$ *atunci* $a^{\phi(n)} \equiv 1 \bmod n$. *Remarca referitoare la reducerea exponenţilor poate fi extinsă şi pentru acest caz acum exponenţii putând fi reduşi modulo* $\phi(n)$.

O demonstraţie elegantă a acestei teoreme se poate face după cum urmează: definim funcţia $f : Z_n^* \to Z_n^*, f(x) = x$ şi funcţia $g : Z_n^* \to Z_n^*, g(x) = \alpha x$ pentru orice parametru $\alpha \in Z_n^*$. Este evident că $\prod_{x \in Z_n^*} f(x) \equiv \prod_{x \in Z_n^*} g(x) \bmod n$ deoarece imaginile ambelor funcţii conţin toate elementele din Z_n^* dar $\prod_{x \in Z_n^*} f(x) = \prod_{x \in Z_n^*} x$ iar $\prod_{x \in Z_n^*} g(x) = \prod_{x \in Z_n^*} \alpha x = \alpha^{\phi(n)} \prod_{x \in Z_n^*} x$ deci $\prod_{x \in Z_n^*} x \bmod n = \alpha^{\phi(n)} \prod_{x \in Z_n^*} x \bmod n$ de unde rezultă $\Rightarrow \alpha^{\phi(n)} \equiv 1 \bmod n$.

Această teoremă la rândul ei poate fi văzută ca fiind un caz particular al unei teoreme a lui Lagrange care spune că ordinul oricărui element dintr-un grup finit (vezi Definiţia 8.10 pentru ordinul unui element din Z_n^*) divide ordinul grupului.

8.3.4 ORDINUL UNUI ELEMENT ŞI GENERATORI ÎN ZN

Definiţia 8.10. *Pentru un întreg* $a \in Z_n^*$ *definim ordinul elementului* a *ca fiind cel mai mic întreg* t *astfel încât* $a^t \equiv 1 \bmod n$.

Teorema 8.8. *Fie* t *ordinul lui* $a \in Z_n^*$ *atunci* $a^k \equiv a^h \bmod n$ *dacă şi numai dacă* $k \equiv h \bmod t$.

Demonstraţie: $\Rightarrow$ presupunem că $k - h \equiv r \bmod t$, cu $r < t$, atunci $a^{k-h} \equiv a^{it} a^r \bmod n \Rightarrow a^{k-h} \equiv a^r \bmod n$ dar $a^k \equiv a^h \bmod n$ deci $\Rightarrow a^{k-h} \equiv 1 \bmod n$ şi deci $a^r \equiv 1 \bmod n \Rightarrow r = 0 \Rightarrow k - h \equiv 0 \bmod t$. $\Leftarrow$ avem $k \equiv h \bmod t$ deci $\Rightarrow a^{k-h} \equiv 1 \bmod n \Rightarrow a^k \equiv a^h \bmod n$.

Lema 8.1. Fie t ordinul lui $a \in Z_n^*$ atunci $t \mid \phi(n)$ - adică ordinul fiecărui element divide ordinul grupului.

Demonstraţie: demonstraţia decurge direct din proprietatea anterioară pentru că din moment ce $a^t \equiv a^{\phi(n)} \bmod n \Rightarrow t \equiv \phi(n) \bmod t \Rightarrow \phi(n) \equiv 0 \bmod t$ deci $t \mid \phi(n)$.

Definiţia 8.11. *Fie $\alpha \in Z_n^*$ dacă ordinul lui α este $\phi(n)$ atunci α se numeşte generator al lui Z_n^*.*

Definiţia 8.12. *Dacă grupul Z_n^* are generatori atunci Z_n^* se numeşte grup ciclic.*

Teorema 8.9. *Z_n^* are generatori, deci este ciclic, dacă şi numai dacă $n = 2,4, p^k, 2p^k$ unde p este prim iar k un întreg pozitiv.*

O demonstraţie pentru Teorema 8.9 poate fi găsită în capitolul 10 din [2].

8.3.5 CONGRUENŢE POLINOMIALE ÎN ZN

Teorema 8.10. *Fie p un număr prim şi numerele a,b,c întregi relativ primi la p. Ecuaţia de gradul II $ax^2 + bx + c \equiv 0 \bmod p$ este compatibilă (are soluţie) dacă şi numai dacă numărul $\Delta = b^2 - 4ac$ este reziduu cvadratic modulo p sau $\Delta = 0$.*

Lăsăm demonstraţia teoremei 8.10 ca exerciţiu pentru cititor în special datorită similitudinii cu calculul rădăcinii unei ecuaţii de ordinul doi în mulţimea numerelor reale cu care cititorul este familiarizat deja.

Teorema 8.11. *Fie p un număr prim şi f un polinom cu coeficienţi întregi care nu sunt divizibili cu p. Atunci numărul de soluţii ale congruentei $f(x) \equiv 0 \bmod p$ este cel mult gradul polinomului f.*

Nu insistăm pe o demonstraţie riguroasă ci ne oprim asupra unei simple schiţe. În primul rând observăm că un polinom de gradul întâi are o singură rădăcină modulo p, este evident că ecuaţia $x + a \equiv 0 \bmod p$ are o singură soluţie. Se poate apoi observa foarte simplu din Teorema Împărţirii cu Rest că orice polinom de grad k poate fi scris ca $f(x) = (x + a)g(x) + c$ prin împărţire cu rest

la un polinom oarecare $x+a$ iar dacă a este rădăcină a lui f atunci $c=0$ şi deci $f(x)=(x+a)g(x)$ iar numărul de rădăcini ale lui f trebuie să fie 1 plus numărul de rădăcini ale lui g care acum are gradul $k-1$. În mod evident procedura se mai poate repeta şi pentru g şi aşa mai departe până ajungem la un polinom de gradul 1 care va avea o singură rădăcină, de unde rezultă imediat că f are cel mult k soluţii.

8.3.6 REZIDUURI CVADRATICE ÎN ZN

Acest paragraf tratează problema reziduurilor cvadratice în Z_n^* problemă care prezintă un interes major deoarece îşi găseşte aplicaţii atât în numeroase construcţii criptografice, de exemplu algoritmul Rabin [65] de criptare asimetrică şi semnătură digitală, precum şi în tehnicile de analiză criptografică, de exemplu în factorizarea întregilor (fiind demonstrat că posibilitatea de a calcula rădăcini pătrate în Z_n^* duce la factorizarea lui n). De asemenea există şi probleme rămase nerezolvate legate de reziduurile cvadratice în Z_n^*, dintre acestea enunţăm următoarele probleme de importanta majoră în criptografie:

i) Nu se cunoaşte nici un algoritm PTP pentru a distinge reziduurile de pseudo-reziduurile cvadratice în Z_n^* fără a se cunoaşte factorizarea lui n.

ii) Nu se cunoaşte nici un algoritm determinist în timp polinomial în p pentru calcularea reziduurilor cvadratice în Z_p^*, unde p este un număr prim.

Definiţia 8.13. *Fie $x \in Z_n^*$, x se numeşte reziduu cvadratic modulo n, sau pătrat perfect modulo n, dacă există $y \in Z_n^*$ astfel încât $x \equiv y^2 \bmod n$. Dacă nu există $y \in Z_n^*$ astfel încât condiţia anterioară să fie satisfăcuta atunci x se va numi non-reziduu cvadratic modulo n.*

Convenim să notăm mulţimea reziduurilor cvadratice modulo n cu Q_n iar cea a non-reziduurilor cvadratice modulo n cu $\overline{Q_n}$, evident $Q_n \cap \overline{Q_n} = \varnothing$ şi $Q_n \cup \overline{Q_n} = Z_n^*$.

Remarca 8.2. Prin definiţie $0 \notin Z_n^*$ şi deci $0 \notin Q_n$, $0 \notin \overline{Q_n}$ (precizăm că există şi lucrări care consideră că $0 \in Q_n$).

Teorema 8.12. *Dacă p este un număr prim şi $x^2 \equiv y^2 \bmod p$ atunci fie $x \equiv y \bmod p$ fie $x \equiv -y \bmod p$.*

Demonstraţie:
$x^2 \equiv y^2 \bmod p \Rightarrow x^2 - y^2 \equiv 0 \bmod p \Rightarrow (x-y)(x+y) \equiv 0 \bmod p$ deci fie $p \mid (x-y)$ fie $p \mid (x+y)$.

Teorema 8.13. *Fie p un număr prim, următoarele relaţii sunt adevărate: i) $x \in Q_p$, $y \in Q_p \Rightarrow xy \in Q_p$, ii) $x \in Q_p$, $y \in \overline{Q_p} \Rightarrow xy \in \overline{Q_p}$, iii) $x \in \overline{Q_p}$, $y \in \overline{Q_p} \Rightarrow xy \in Q_p$.*

Demonstraţie: i) Este evident că dacă $x \in Q_p$, $y \in Q_p$ atunci $\exists a, b$ astfel încât $x \equiv a^2 \bmod p$ şi $y \equiv b^2 \bmod p \Rightarrow xy \equiv (ab)^2 \bmod p$ deci $xy \in Q_p$. ii) Presupunem că $xy \in Q_p$ atunci avem $xy \equiv a^2 \bmod p$ pentru un număr oarecare $a \in Z_p^*$. Dar cum $x \in Q_p$ înseamnă că $\exists b$ astfel încât $x \equiv b^2 \bmod p$ şi atunci $xy \equiv a^2 \bmod p \Rightarrow b^2 y \equiv a^2 \bmod p$ de unde avem $y \equiv a^2 \left(b^2\right)^{-1} \bmod p$ $\Rightarrow y \equiv \left(ab^{-1}\right)^2 \bmod p$ şi deci $y \in Q_p$ ceea ce este o contradicţie cu ipoteza, deci nu rămâne decât că presupunerea este greşită şi deci $xy \in \overline{Q_p}$. iii) Presupunem că produsul a două non-reziduuri este tot un non-reziduu. Notăm $xy \equiv z \bmod p$ şi presupunem deci că $\Rightarrow z \in \overline{Q_p}$, în mod evident orice non-reziduu cvadratic poate fi scris ca fiind produsul dintre un reziduu şi un non-reziduu cvadratic, convenim astfel pentru un număr oarecare $a \in Z_p^*$ să scriem $x \equiv x' a^2 \bmod p$, $y \equiv y' a^2 \bmod p$, $z \equiv z' a^2 \bmod p$ şi urmează că $x'y'a^4 \equiv z'a^2 \bmod p \Rightarrow x'y'z'^{-1} \equiv \left(a^{-1}\right)^2 \bmod p$. Este uşor de observat că dacă $z' \in \overline{Q_p}$ atunci şi $z^{-1} \in \overline{Q_p}$ pentru că $z^{-1} \equiv z^{p-2} \bmod p$ iar ridicarea la puterea $p-2$ care este un număr impar nu poate schimba apartenenţa la reziduuri sau non-reziduuri cvadratice. Dar noi am presupus că produsul a două non-reziduuri este tot un non-reziduu deci $x'y'$ este non-reziduu şi înmulţit cu z^{-1} care este tot

non-reziduu trebuie să dea tot un non-reziduu, dar $\left(a^{-1}\right)^2$ este reziduu cvadratic deci presupunerea făcută este falsă.

Toate aceste demonstraţii pot fi făcute direct cu ajutorul simbolului Legendre introdus în paragraful următor, lăsăm aceasta ca exerciţiu cititorului. Considerăm de asemenea relevante şi interesante demonstraţiile din [86, p. 18].

8.3.7 NUMĂRUL DE REZIDUURI CVADRATICE ÎN ZN

Lema 8.2. *(pentru Z_p^*): dacă p este un număr prim atunci exact jumătate din elementele din Z_p^* sunt reziduuri cvadratice şi jumătate nu sunt reziduuri cvadratice $\left|Q_p\right| = \left|\overline{Q_p}\right| = \frac{p-1}{2}$. Mai mult fiecare reziduu cvadratic din Z_p^* are exact două rădăcini pătrate.*

Demonstraţia acestei leme se poate face după cum urmează: din moment ce p este un număr prim înseamnă că Z_p^* are generatori. Fie $\alpha \in Z_p^*$ un generator din Z_p^*, atunci toate elementele din Z_p^* pot fi scrise sub forma $a = \alpha^x \bmod p, 1 \le x \le p-1$ iar în cazul în care x este par (în exact jumătate din cazuri) numerele sunt reziduuri cvadratice. De asemenea numerele $a = \alpha^x \bmod p$ pentru x impar nu pot fi reziduuri cvadratice şi rezultatul este uşor de demonstrat pentru că dacă presupunem că a este reziduu cvadratic şi $a = \alpha^x \bmod p$ pentru un x impar, deci $\exists b$ astfel încât $a \equiv b^2 \bmod p$ urmează că $\exists y$ astfel încât $b \equiv \alpha^y \bmod p \Rightarrow a \equiv b^2 \equiv \alpha^{2y} \bmod p$ deci a este o putere pară a generatorului ceea ce este o contradicţie cu ipoteza. Demonstraţia faptului că fiecare reziduu cvadratic are exact două rădăcini este o consecinţă imediată a teoremei 8.12 şi rămâne ca exerciţiu cititorului.

Lema 8.3. *(pentru $Z_{p^e}^*$): Fie un număr prim p şi un întreg $e > 1$ atunci numărul de reziduuri cvadratice din $Z_{p^e}^*$ este $\frac{p^{e-1}(p-1)}{2}$. Mai mult dacă x este rădăcina pătratică a lui a modulo p^e la fel este şi $-x$ şi orice altă rădăcină y a lui*

a *satisface* $y \equiv \pm x \bmod p$. *Mai mult* $\forall a \neq 0 \bmod p$ *avem* $a^{\frac{p^{e-1}(p-1)}{2}} = \pm 1 \bmod p$ *şi* $a \in Q_p \Rightarrow a^{\frac{p^{e-1}(p-1)}{2}} = 1 \bmod p^e$.

Demonstraţie: Demonstraţia se poate face în manieră similară cu cea anterioară deoarece şi $Z_{p^e}^*$ are generatori de ordin $\phi(p^e) = p^e - p^{e-1}$ şi rezultatul teoremei decurge imediat. Din nou, demonstrarea teoremei rămâne ca potenţial exerciţiu pentru cititor.

Lema 8.4. *(pentru* $Z_n^*, n = pq$*): Fie întregul compozit* $n = pq$ *unde* p *şi* q *sunt numere prime (acest caz este important pentru criptografia cu cheie publică fiind utilizat în câteva criptosisteme) atunci:* $|Q_n| = |Q_p||Q_q| = \frac{(p-1)(q-1)}{4}$ *şi* $|\overline{Q_n}| = \frac{3(p-1)(q-1)}{4}$.

Demonstraţia acestei leme poate fi dedusă din următoarea teoremă al cărei caz particular este:

Teorema 8.14. *(pentru* Z_n^**): Fie un întreg compozit impar* n *a cărui factorizare este* $n = \prod_{i=1}^{r} p_i^{e_i}$ *. Atunci numărul de reziduuri cvadratice modulo* n *este* $|Q_n| = \frac{\phi(n)}{2^r}$*. Mai mult dacă* $a \in Q_n$ *atunci* a *are exact* 2^r *rădăcini distincte.*

Demonstraţie: am stabilit prin Lema 8.3 numărul de reziduuri cvadratice din $Z_{p^e}^*$. Din moment ce factorizarea lui n este $n = \prod_{i=1}^{r} p_i^{e_i}$ şi $\forall a \in Q_n \Rightarrow a \in Q_{p_i} \forall p_i \mid n$ cum teorema chineză a resturilor permite calcularea în mod unic a unui reziduu cvadratic modulo n având congruenţele modulo $p_i, \forall p_i \mid n$ rezultă evident că pentru $\forall a \in Q_n$ avem exact 2^r rădăcini distincte. Din aceleaşi considerente se poate deduce uşor şi că numărul de reziduuri cvadratice este $\frac{\phi(n)}{2^r}$.

8.3.8 SIMBOLURILE LEGENDRE ŞI JACOBI

Simbolurile Legendre şi Jacobi sunt de asemenea frecvent utilizate în construcţiile criptografice precum şi în analiza criptografică, aplicaţiile lor vor fi întâlnite în secţiunile următoare, pentru moment introducem câteva aspecte teoretice legate de acestea.

Definiţia 8.14. *Definim simbolul Legendre* $\left(\frac{x}{p}\right)$ *pentru un întreg* x *şi un număr prim* p *după cum urmează:* $\left(\frac{x}{p}\right)=\begin{cases}1 & x\in Q_p\\ -1 & x\in\overline{Q_p}\\ 0 & p\mid x\end{cases}$.

Remarca 8.3. Următoarele proprietăţi pot fi deduse direct din definiţie (nu se insistă asupra demonstraţiei):

i) $\left(\frac{xy}{p}\right)=\left(\frac{x}{p}\right)\left(\frac{y}{p}\right)\ \forall x,y,p$ şi p prim

ii) $\left(\frac{x^2}{p}\right)=1\ \forall x,p$ şi p prim

iii) $x\equiv y \bmod p \Rightarrow \left(\frac{x}{p}\right)=\left(\frac{y}{p}\right)\ \forall x,y,p$ şi p prim

Lema 8.5. *(Euler): Fie* p *un număr prim şi* $x\in Z_p^*$ *atunci* $x\in Q_p$ *dacă şi numai dacă* $x^{\frac{p-1}{2}}\equiv 1 \bmod p$ *respectiv* $x\in\overline{Q_p}$ *dacă şi numai dacă* $x^{\frac{p-1}{2}}\equiv -1 \bmod p$.

Demonstraţie: Conform teoremei introduse în capitolul referitor la congruenţe polinomiale ecuaţia $x^{\frac{p-1}{2}}\equiv 1 \bmod p$ are cel mult $\frac{p-1}{2}$ soluţii, pe de altă parte toate reziduurile cvadratice verifică această proprietate deoarece dacă $x\in Q_n \Rightarrow \exists y$ atunci $x\equiv y^2 \bmod n \Rightarrow x^{\frac{p-1}{2}}\equiv\left(y^2\right)^{\frac{p-1}{2}}\equiv 1 \bmod n$ conform teoremei lui Fermat. Din moment ce numărul de reziduuri cvadratice $\left|Q_n\right|=\frac{p-1}{2}$ urmează

evident că aceasta relaţie este satisfăcută numai de către reziduuri cvadratice. În cazul în care $x \in \overline{Q_p}$ cum $x^{p-1} \equiv 1 \bmod p$, conform Fermat, reiese ca unică alternativă de rădăcină pătrată a lui 1 ca $x^{\frac{p-1}{2}} \equiv -1 \bmod p$ şi partea a doua a lemei este rezolvată.

Corolarul 8.1. $x^{\frac{p-1}{2}} \bmod p = \left(\frac{x}{p}\right)$.

Demonstraţie: relaţia este directă din Lema precedentă, se observă în plus că dacă $p \mid x \Rightarrow x^{\frac{p-1}{2}} \bmod p \equiv 0$.

Corolarul 8.2. $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} \Rightarrow -1 \in Q_p$ *dacă* $p \equiv 1 \bmod 4$ *respectiv* $\Rightarrow -1 \in \overline{Q_p}$ *dacă* $p \equiv 3 \bmod 4$.

Demonstraţie: Această relaţie este din nou directă: se observă că semnul lui $(-1)^{\frac{p-1}{2}}$ depinde de paritatea lui $\frac{p-1}{2}$ iar aceasta este adevărată în cazul $p \equiv 1 \bmod 4$ respectiv falsă în cazul $p \equiv 3 \bmod 4$.

Corolarul 8.3. $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} \bmod p \Rightarrow 2 \in Q_p$ *dacă* $p \equiv 1 \bmod 8$ *sau* $p \equiv 7 \bmod 8$ *respectiv* $\Rightarrow 2 \in \overline{Q_p}$ *dacă* $p \equiv 3 \bmod 8$ *sau* $p \equiv 5 \bmod 8$.

Nu insistăm pe o demonstraţie dar pentru cititorii interesaţi trimitem către [2, p. 181] sau [86, p. 20].

Teorema 8.15. *(a reciprocităţii cvadratice, facilitează calculul simbolului Legendre): Pentru oricare două numere prime* p *şi* q *avem:* $\left(\frac{p}{q}\right) = (-1)^{\frac{(p-1)(q-1)}{4}} \left(\frac{q}{p}\right)$. *Sau echivalent* $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}$. *(Nu vom insista pe demonstraţia acestei teoreme, diverse demonstraţii pot fi găsite în [86, p. 21], [2, p. 185], [49, p. 76]).*

Exemplu de utilizare: pentru clarificarea aplicabilităţii acestui rezultat recurgem la următorul exemplu. Dorim să calculăm $\left(\frac{p}{q}\right)$ pentru $p = 11$ şi

$q=239$, conform legii reciprocităţii cvadratice avem $\left(\frac{p}{q}\right)=(-1)^{\frac{(p-1)(q-1)}{4}}\left(\frac{q}{p}\right)$

$\Rightarrow\left(\frac{11}{239}\right)=(-1)^{\frac{(11-1)(239-1)}{4}}\left(\frac{239\bmod 11}{11}\right)= \quad =-\left(\frac{8}{11}\right)=-\left(\frac{2^2}{11}\right)\left(\frac{2}{11}\right)=-\left(\frac{2}{11}\right)$ şi cum $\left(\frac{2}{11}\right)=(-1)^{\frac{11^2-1}{8}}=-1\Rightarrow\left(\frac{11}{239}\right)=1.$

Simbolurile Jacobi sunt o generalizare a simbolurilor Legendre pentru cazul întregilor compoziţi.

Definiţia 8.15. *(Simbolul Jacobi): Pentru un întreg impar* $n\geq 3$ *a cărui factorizare este* $n=\prod_{i=1}^{r}p_i^{e_i}$ *şi un număr a definim simbolul Jacobi* $\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\ldots\left(\frac{a}{p_r}\right)^{e_r}$ *(se remarcă faptul că simbolul Jacobi este identic cu Simbolul Legendre pentru numere prime).*

Remarca 8.4. Următoarele proprietăţi reies direct din definiţie (nu se insistă asupra demonstraţiei):

i) $c.m.m.d.c.(x,n)\neq 1\Rightarrow\left(\frac{x}{n}\right)=0$

ii) $\left(\frac{x}{n}\right)\neq 0\Rightarrow c.m.m.d.c.(x,n)=1$

iii) Dacă $x\equiv y\bmod n\Rightarrow\left(\frac{x}{n}\right)=\left(\frac{y}{n}\right)$

iv) $\left(\frac{xy}{n}\right)=\left(\frac{x}{n}\right)\left(\frac{y}{n}\right)$ şi $\left(\frac{x}{st}\right)=\left(\frac{x}{s}\right)\left(\frac{x}{t}\right)$

v) $\left(\frac{x^2}{n}\right)=1$ şi $\left(\frac{x}{n^2}\right)=1$

Următoarele două leme stabilesc valoarea simbolului Jacobi pentru numerele -1 şi 2, aceste valori fiind extrem de utile în calcule. Nu prezentăm o demonstraţie pentru acestea pentru aceasta putând fi consultată oricare din lucrările [2], [49], [86]:

Lema 8.6. $\left(\frac{-1}{s}\right)=(-1)^{\frac{s-1}{2}}$ *pentru orice* s *impar.*

Lema 8.7. $\left(\frac{2}{s}\right)=(-1)^{\frac{s^2-1}{8}}$ *pentru orice* s *impar.*

Teorema 8.16. *(Reciprocitatea cvadratică pentru simboluri Jacobi)* $\left(\frac{s}{t}\right)\left(\frac{t}{s}\right)=(-1)^{\frac{(s-1)(t-1)}{4}}$ *şi în consecinţă* $\left(\frac{s}{t}\right)=\left(\frac{t}{s}\right)$ *dacă* $s\equiv 1 \bmod 4$ *şi* $t\equiv 1 \bmod 4$ *respectiv* $\left(\frac{s}{t}\right)=-\left(\frac{t}{s}\right)$ *dacă* $s\equiv 3 \bmod 4$ *şi* $t\equiv 3 \bmod 4$.

Pentru demonstraţia acestei teoreme trimitem către [86, p. 24], [2, p. 189].

Definiţia 8.16. *Pentru orice întreg* $n\geq 3$ *definim* $J_n=\left\{a\in Z_n^* \mid \left(\frac{a}{n}\right)=1\right\}$. *Mulţimea pseudo-reziduurilor cvadratice* Z_n^* *o vom nota cu* Q_n *şi este* $Q_n=J_n-Q_n$.

8.3.9 UTILIZAREA REZIDUURILOR CVADRATICE ÎN CRIPTANALIZĂ

Dorim să prezentăm şi un exemplu în care proprietăţile reziduurilor cvadratice sunt utilizate în criptanaliză. Considerăm în acest sens protocolul prezentat în [73, p. 90][7], menţionăm că sunt multe alte exemple mai relevante decât acesta dar care sunt ceva mai greu de înţeles, exemplul prezent fiind ilustrativ pentru faptul că un bit de informaţie poate fi recuperat dintr-un criptotext RSA (pentru claritate, acest paragraf poate fi citit şi după citirea capitolului 5.2. cu privire la RSA). Alice şi Bob doresc să „dea cu banul” pentru a decide care dintre ei are câştig asupra unei cauze, aceştia nu dispun însă de o

[7] Protocolul este prezentat în [73, p. 90] ca fiind funcţional în cazul general al unei algoritm de criptare comutativ, i.e. pentru care este satisfăcută relaţia $D_{K_1}\left(E_{K_2}\left(E_{K_1}(m)\right)\right)=E_{K_2}(m)$, aici arătăm că pentru cazul în care funcţia de criptare este RSA (funcţie care respectă această condiţie) protocolul poate fi spart.

monedă fizică deoarece comunicarea se face prin telefon (sau orice altă linie de comunicare) şi doresc să realizeze acest lucru prin mijloace criptografice. Următorul protocol, care poate fi fraudat, este propus, se presupune că cei doi utilizează aceeaşi funcţie RSA fără a-şi divulga cheile publică/privată:

i. Alice şi Bob generează câte o pereche de chei publică/privată de RSA cu acelaşi modul, comun cunoscut, dar nu fac cunoscuţi exponenţii public-privat.

ii. Alice generează două mesaje: unul indicând „cap" şi altul „stemă". Aceste mesaje sunt concatenate cu o secvenţă aleatoare şi sunt criptate cu cheia publică a lui Alice iar apoi trimise către Bob într-o ordine aleatoare (pentru ca Bob să nu poată distinge între ele) $E_A(m_1), E_A(m_2)$

iii. Bob alege un mesaj la întâmplare şi îl criptează cu cheia sa publică apoi îl trimite lui Alice $E_B(E_A(m_x))$.

iv. Alice, care nu poate citi acest mesaj, îl decriptează cu cheia sa publică şi îl trimite către Bob $E_B(m_x)$.

v. Bob decriptează mesajul cu cheia s-a privată şi îi trimite rezultatul lui Alice m_x.

vi. Alice citeşte rezultatul şi verifică corectitudinea mesajului (comparând cu mesajul criptat la pasul ii).

vii. Ambii utilizatori îşi dezvăluie cheile publică/privată şi verifică dacă mesajele transmise de cealaltă parte au fost corecte.

Atacul asupra protocolului este simplu. Moneda are două feţe, deci gradul de incertitudine al mesajului, i.e. entropia, este 1 indiferent de valoarea sa. Alice poate „fura" protocolul în pasul patru prin aflarea caracterului cvadratic al celor două mesaje, dacă mesajele au fost intenţionat alese cu caractere cvadratice diferite atunci Alice poate afla încă din pasul 4 care este rezultatul (deci prezumţia că Alice nu poate citi acest mesaj la pasul 4 este falsă). În cazul în care Alice nu este mulţumită de rezultat ea poate refuza să continue protocolul, şi să solicite repornirea lui până când va avea în pasul 4 valorile dorite.

8.3.10 CURBE ELIPTICE

Definiţia 8.17. *(Curbă eliptică) O curbă eliptică peste un câmp* K *de caracteristică diferită de 2 sau 3 este mulţimea puncte care răspund ecuaţiei* $y^2 = ax^3 + bx + c$ alături de punctul infinit notat cu O.

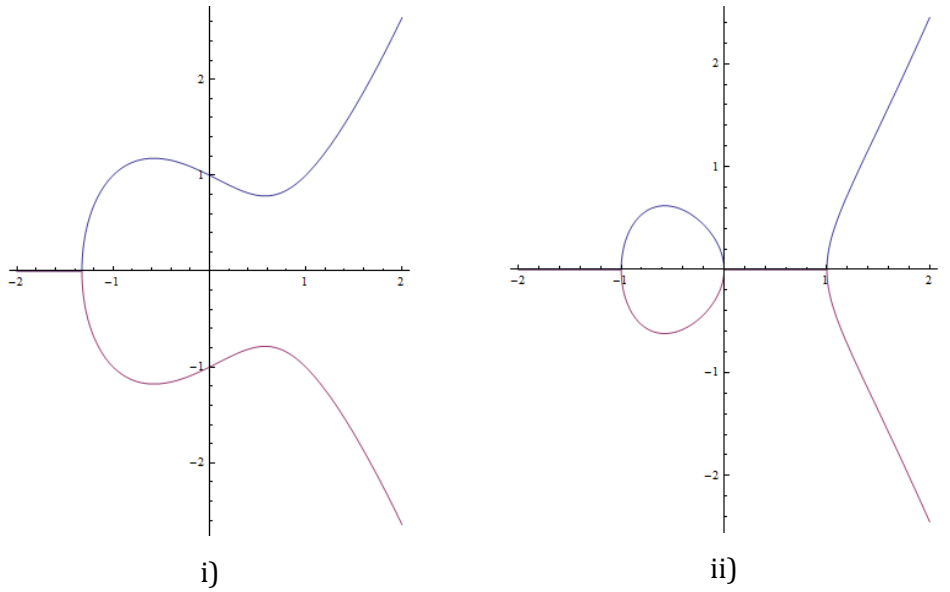


FIGURA 8.1. GRAFICUL A DOUĂ CURBE ELIPTICE: I) $y^2 = x^3 - x + 1$ ŞI II) $y^2 = x^3 - x$

Câmpul K poate fi construit peste orice mulţime, de exemplu mulţimea numerelor reale sau complexe, dar pentru scopuri criptografice se folosesc câmpuri finite (numite şi câmpuri Galois după numele matematicianului care le-a descoperit Evariste de Galois). Exemplu de astfel de câmpuri finite sunt F_p şi F_{p^n} unde p este un număr prim. Figura 8.1 prezintă grafic două curbe în mulţimea numerelor reale.

Deasemenea se impune condiţia $4a^3 + 27b^2 \neq 0$ pentru a evita auto-intersecţii ale curbei (această valoare provine din calculul discriminantului unei ecuaţii cubice).

Rămâne de definit adunarea a două puncte de pe curbă ca o operaţie a grupului descris de curba eliptică.

Definim în primul rând inversul unui punct P ca fiind $-P$ adică punctul care are aceeaşi coordonată x dar coordonată opusă pe y. Adică dacă P are coordonatele (x_P, y_P) atunci $-P$ va avea coordonatele $(x_P, -y_P)$ (adică inversul punctului este imaginea sa în oglindă faţă de axa Ox). Observăm că acest lucru este întotdeauna posibil datorită termenului în y^2 din ecuaţia curbei. Pentru cazul special când P este punctul la infinit O spunem că $-P$ este tot O.

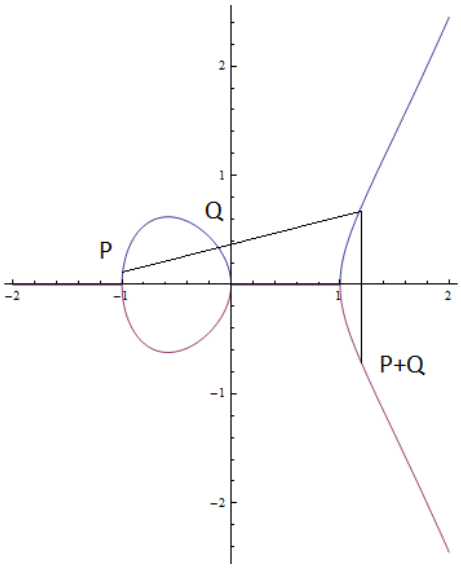


FIGURA 8.2. ADUNAREA A DOUĂ PUNCTE P ŞI Q DE PE CURBA ELIPTICĂ

Definim adunarea a două puncte P şi Q de pe curbă după cum urmează: fie R punctul în care dreapta ce trece prin P şi Q (tangentă la curbă în cazul $P = Q$) intersectează curba. Suma $P + Q$ este $-R$. Figura 8.2 prezintă grafic această operaţie. Se poate observa uşor că dacă $P = -Q$ atunci prin adunare $P + Q$ se obţine punctul la infinit O. La fel, $P + O = P$ deci O este elementul neutru la adunare.

9 FUNDAMENTE COMPUTAŢIONALE

„…modern cryptography is based on a gap between efficient algorithms provided for legitimate users and the computational infeasibility of abusing or breaking these algorithms (via illegitimate adversarial actions).” O. Goldreich[8].

Una dintre cele mai importante probleme înainte de a construi criptosisteme este a lămuri ce se poate calcula eficient şi ce nu se poate calcula eficient în grupuri de întregi. Aşa cum sugerează şi fragmentul de mai sus posibilitatea de a construi criptosisteme se bazează pe existenţa unei instanţe a unui probleme care poate fi eficient rezolvată de către o anume entitate (posesorul legitim al cheii de criptare/decriptare) şi care nu poate fi eficient rezolvată de către altă entitate (adversarul). În figurile 10.1 şi 10.2 este sugerată această discrepanţă pentru criptosistemul Rabin. Se observă că funcţia de criptare presupune efectuarea unei singure multiplicări $O(1)$, iar decriptarea pentru utilizatorul legitim care cunoaşte factorizarea modulului presupune $O\left(\frac{3}{2}\log_2 \phi(n)\right)$ multiplicări (am considerat cazul pentru module Blum, vezi 4.4 şi 4.5). În acelaşi timp pentru un adversar criptarea este fezabilă dar decriptarea presupune găsirea factorizării modulului, operaţie care necesită $O\left(e^{(c+o(1))\sqrt[3]{\ln n}\sqrt[3]{(\ln\ln n)^2}}\right), c \approx 2$ operaţii folosind algoritmul General Number Field Sieve care este unul dintre cei mai buni algoritmi cunoscuţi în prezent.

În acest capitol dorim doar crearea unei imagini de suprafaţă asupra acestei probleme, astfel pentru algoritmi eficienţi de calcul trimitem cititorul în special către cartea lui Shoup [77] dar şi către capitolul 14 din [62]. Pentru a crea o imagine de ansamblu Tabelul 9.1 sintetizează operaţiile care pot fi eficient efectuate şi Tabelul 9.2 pe cele care nu pot fi eficient efectuate, urmând ca procedurile să fie discutate în secţiunile următoare.

---

[8] Din volumul I al cărţii de referinţă “Foundations of Cryptography”.

Utilizator legitim (cunoaste factorizarea lui *n*)

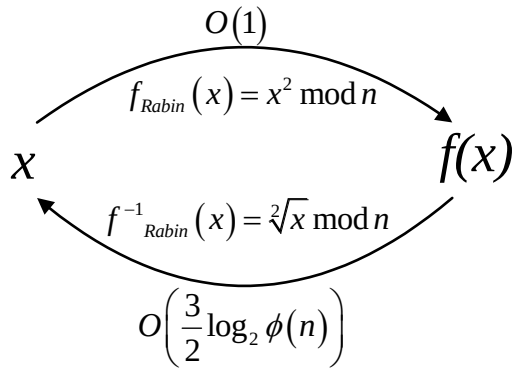


FIGURA 9.1. CALCULUL DIN CRIPTOSISTEMUL RABIN PENTRU UN UTILIZATOR LEGITIM.

Adversar (nu cunoaste factorizarea lui *n*)

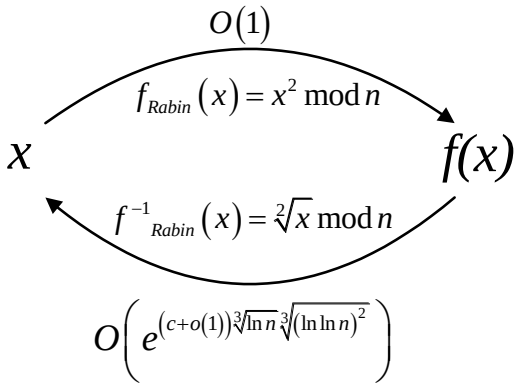


FIGURA 9.2. CALCULUL DIN CRIPTOSISTEMUL RABIN PENTRU UN UTILIZATOR NELEGITIM (ADVERSAR).

Se pot calcula eficient	**Condiţie**
Operaţii elementare: <, >, -, +, *, /, a^x	-
c.m.m.d.c., invers multiplicativ în Z_n, x^{-1}, *c.m.m.m.c.*	-
Simbolurile Legendre şi Jacobi	-
Testarea dacă un număr este prim sau compozit	-
Rădăcina pătrată în Z_n, i.e. $\sqrt[2]{x} \bmod n$	Dacă **şi numai dacă** se cunoaşte factorizarea lui n
Rădăcina de ordin ε în Z_n când $c.m.m.d.c.\left(\varepsilon, \phi(n)\right) = 1$, i.e. $\sqrt[\varepsilon]{x} \bmod n$	Dacă se cunoaşte factorizarea lui n
Apartenenţa la Q_n	Dacă se cunoaşte factorizarea lui n

TABELUL 9.1. OPERAŢII CARE POT FI EFICIENT REZOLVATE.

Nu se pot calcula eficient	**Condiţie**
Logaritmul discret	Pentru orice element (de preferinţă generator al grupului) care are un ordin suficient de mare, sute, mii de biţi etc.
Factorizarea unui întreg	Întregi mari de sute, mii de biţi etc., excepţie cazurile banale (de exemplu putere a unui număr prim) şi numerele având factori care pot fi eficient găsiţi datorită unor proprietăţi algebrice ale acestora
Rădăcina pătrată în Z_n, i.e. $\sqrt[2]{x} \bmod n$	Dacă nu se cunoaşte factorizarea lui n
Rădăcina de ordin ε în Z_n când $c.m.m.d.c.\left(\varepsilon, \phi(n)\right) = 1$, i.e. $\sqrt[\varepsilon]{x} \bmod n$	Dacă nu se cunoaşte factorizarea lui n
Apartenenţa la Q_n	Dacă nu se cunoaşte factorizarea lui n

TABELUL 9.2. OPERAŢII CARE NU POT FI EFECTUATE EFICIENT.

9.1 ELEMENTE DE TEORIA COMPLEXITĂŢII

O **problemă** este o mulţime nevidă de întrebări între care există o relaţie, pot fi una sau mai multe întrebări şi este obligatoriu ca ele să aibă o dimensiune finită. De exemplu factorizarea unui întreg este o problemă cu următorul enunţ: având un întreg *n* să se găsească numerele prime al căror produs este. Un **algoritm** este un set bine definit de paşi pentru rezolvarea unei probleme. Altfel spus, un algoritm este ansamblul de paşi prin care un set de date de intrare este transformat într-un set de date de ieşire în scopul rezolvării unei probleme. De exemplu pentru rezolvarea problemei factorizării întregului de mai sus se poate folosi următorul algoritm: pentru toţi întregii *i* de la 2 la radical din *n* verifică dacă *n* este divizibil cu *i*.

Este evident că eficienţa unui algoritm este o funcţie care depinde de dimensiunea datelor de intrare şi totodată eficienţa unui algoritm trebuie să fie o caracteristică intrinsecă a algoritmului care să nu depindă de un anume model al maşinii de calcul. În acest context este impropriu să numim un algoritm ca fiind eficient în baza faptului că pe un anumit procesor a avut un timp de rulare oarecare, şi mai mult, faptul că pe un procesor a avut un anume timp de rulare nu va spune nimic cu privire la timpul de rulare în momentul în care dimensiunea datelor de intrare se dublează. Să presupunem ca exemplu naiv doi algoritmi care caută un element într-un şir ordonat crescător, algoritmul A1 implementează o căutare naivă în care şirul este parcurs de la un capăt la altul element cu element în scopul identificării elementului căutat iar A2 implementează o căutare binară, prin înjumătăţirea succesivă a şirului în care se face căutarea. Să presupunem că timpul de calcul pentru un tablou cu 1.000.000 de elemente este pentru primul algoritm 5 milisecunde şi pentru al doilea 5 microsecunde. La dublarea dimensiunii datelor de intrare timpul de calcul pentru primul algoritm se va dubla în timp ce pentru al doilea va creşte nesemnificativ. Aceasta deoarece, pentru găsirea unui element într-un şir de *n* elemente, primul algoritm efectuează cel mult *n* paşi iar cel de-al doilea cel mult $log_2 n$ paşi. În consecinţă performanţa unui algoritm nu trebuie descrisă în funcţie de timpul de rulare al acestuia ci ca funcţie de numărul de paşi pe care algoritmul îi necesită. Aici intră în joc teoria complexităţii care este domeniul care ne oferă un răspuns cu privire la numărul de paşi necesari ca un algoritm să ofere un rezultat. În cazul nostru, al criptologiei, ne putem referi la numărul de paşi pentru a cripta, decripta informaţie sau chiar pentru a sparge un cod. Informal, din punct de vedere al criptanalizei teoria complexităţii ne spune cât timp şi cât spaţiu este necesar pentru a sparge un criptosistem.

În general cu privire la un algoritm, sub raportul complexităţii, ne interesează atât **timpul** (despre care s-a spus că se măsoară în număr de paşi) cât

şi **spaţiul** (care înseamnă cantitate de memorie necesară) – de aici şi noţiunile de complexitate de timp şi complexitate de spaţiu. Pentru măsurarea complexităţii folosim următorii indicatori de performanţă:

i) **limita asimptotică superioară**:

$$f(n)=O(g(n))\Leftrightarrow \exists c,n_o\ a.i. 0\le f(n)\le c\cdot g(n)\forall n\ge n_0$$

ii) **limita asimptotică inferioară**:

$$f(n)=\Omega(g(n))\Leftrightarrow \exists c,n_o\ a.i. 0\le c\cdot g(n)\le f(n),\forall n\ge n_0$$

iii) **limita asimptotică restrânsă**:

$$f(n)=\Theta(g(n))\Leftrightarrow \exists c_1,c_2,n_o\ a.i. c_1\cdot g(n)\le f(n)\le c_2\cdot g(n)\forall n\ge n_0$$

iv) **limitele asimptotice relaxate**:

$$f(n)=o(g(n))\Leftrightarrow \exists n_o\ a.i.\forall c\ 0\le f(n)<c\cdot g(n)\forall n\ge n_0$$

$$f(n)=\omega(g(n))\Leftrightarrow \exists n_o\ a.i.\forall c\ 0\le c\cdot g(n)< f(n)\forall n\ge n_0$$

Este important de spus că $f=F\left(g\left(n\right)\right)$, unde F este oricare din indicatorii Ω, Θ, O, o, nu se citeşte: “f egal F de g(n)”, ci corect este “f este de ordinul F al lui g(n)” sau mai simplu “f este F(g(n))”. Intuitiv semnul “=“ nu are semnificaţia semnului egal, în acest caz este echivalent cu $\in$.

Astfel în funcţie de timpul de calcul algoritmii se împart în clase după cum urmează: constant $O(1)$, logaritmic $O(\lg(n))$ (se observă că $\log_c n=\Theta(\lg n),\forall c>0$), poli-logaritmic $O\left(\lg^c\left(n\right)\right)$, fracţionar $O\left(n^c\right), 0<c<1$, liniar $O(n)$, liniar logaritmic $O\left(n\log_2 n\right)$, pătratic (sau cvadratic) $O\left(n^2\right)$, cubic $O\left(n^3\right)$, polinomial $O\left(n^m\right)$ (cu observaţia că: liniar, pătratic, cubic sunt timpi polinomiali), super-polinomial $O\left(c^{f(n)}\right)$ (unde c este o constantă iar $f(n)$ nu este constant dar este mai mic decât $O(n)$), exponenţial $O\left(c^{f(n)}\right)$ (unde c este o constantă iar $f(n)$ este un polinom de gradul 1), factorial (sau combinatorial) $O\left(n!\right)$, dublu exponenţial $O\left(2^{c^n}\right)$.

Următoarele proprietăţi intuitive decurg direct din definiţia acestor indicatori:

i) $f(n)=O(g(n))\Leftrightarrow g(n)=\Omega(f(n))$

ii) $f(n)=\Theta(g(n)) \Leftrightarrow f(n)=O(g(n)) \wedge f(n)=\Omega(g(n))$

iii) $f(n)=O(h(n)) \wedge g(n)=O(h(n)) \Rightarrow (f+g)(n)=O(h(n))$

iv) $f(n)=O(h(n)) \wedge g(n)=O(i(n)) \Rightarrow (f \cdot g)(n)=O(h(n) \cdot i(n))$

v) $f(n)=O(f(n)) - reflexivitate$

vi) $f(n)=O(g(n)) \wedge g(n)=O(h(n)) \Rightarrow f(n)=O(h(n)) - tranzitivitate$

Reţinem de asemenea următoarele aproximări utile:

i) $f(n)=a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0 \Rightarrow f(n)=\Theta(n^k)$

ii) $n! = o(n^n) \wedge n! = \Omega(2^n)$

iii) $1 < \ln\ln n < \ln n < e^{\sqrt{(\ln n)(\ln\ln n)}} < n^{\varepsilon} < n^c < n^{\ln n} < c^n < n^n < c^{c^n}$

Pentru a ilustra importanţa cunoaşterii complexităţii prezentăm următorul tabel al unor magnitudini uzuale şi de asemenea vom considera 4 algoritmi A_1, A_2, A_3, A_4 având complexităţile $O(n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$ precum şi un sistem capabil să execute 10^{10} operaţii/secundă, în scopul unei comparaţii vom evalua timpul necesar rezolvării algoritmilor pentru $n=10^6$. Tabelul 9.1 sintetizează aceste rezultate.

Este necesar de asemenea să introducem noţiunea de reducţie în timp polinomial. Se observă că dacă avem doi algoritmi A_1 şi A_2 care au complexitate polinomială, chiar dacă A_2 devine subrutină a lui A_1 algoritmul rezultat va funcţiona tot în timp polinomial. Următoarea definiţie introduce noţiunea de reducţie în timp polinomial (polynomial reduction):

Definiţia 9.1. (Reducţie în timp polinomial) *Spunem că problema Π_1 se reduce în timp polinomial la Π_2 şi notăm $\Pi_1 <_P \Pi_2$ dacă există un algoritm A_1 care rezolvă Π_1 şi primeşte ca subrutină un algoritm A_2 care rezolvă Π_2 iar complexitatea lui A_1 este polinomială dacă complexitatea lui A_2 este şi ea polinomială.*

Cărţi de referinţă în domeniul algoritmilor şi a complexităţii sunt cartea lui Knuth [55] şi a lui Cormen et al. [25]. De asemenea este util glosarul de termeni din [28] iar în ceea ce priveşte teoria cu privire la NP-completness, care aici nu a fost prezentată dar are implicaţii majore în criptografie deoarece faimoasa

problemă $P \neq NP$ condiţionează existenţa funcţiilor criptografice one-way, este de referinţă lucrarea lui Garey şi Johnson [41].

Secunde într-un an	3×10^{7}
Vârsta sistemului solar în secunde	2×10^{17}
Electroni în univers	8.37×10^{77}
Chei pe 256 biţi	1.2×10^{77}
Chei pe 1024 biţi	1.8×10^{308}
Timpul pentru a rezolva A_1	1×10^{-3}
Timpul pentru a rezolva A_2	1×10^{2}
Timpul pentru a rezolva A_3	1×10^{8}
Timpul pentru a rezolva A_4	1×10^{303020}

TABELUL 9.3. CÂTEVA MAGNITUDINI UZUALE.

9.2 ELEMENTE DE TEORIA INFORMAŢIEI

În limbaj informal, din punct de vedere criptanalitic, Teoria Informaţiei ne spune de câtă informaţie criptată avem nevoie pentru a sparge un mecanism de criptare, sau din considerente criptografice care este dimensiunea unei chei pentru ca criptosistemul să nu poată fi spart. În ceea ce priveşte ultimul aspect, este demonstrat că pentru a construi un algoritm de criptare care să nu poată fi spart este necesară o cheie aleatoare de dimensiune egală cu a mesajului criptat, bineînţeles cheia nu poate fi refolosită. Astfel teoria informaţiei poate fi utilizată pentru a construi criptosisteme care au **securitate necondiţionată** (numită şi confidenţialitate perfectă (perfect secrecy)) – care nu pot fi sparte chiar dacă puterea de calcul este infinită.

Într-adevăr există şi criptosisteme care sunt imposibil de spart. Sigur la o primă vedere existenţa unor astfel de criptosisteme ar face să pară că problemele

criptografiei sunt în totalitate rezolvate. Nu este nici pe departe aşa, problema de bază a criptografiei fiind dimensiunea cheii de criptare, despre care s-a afirmat anterior că pentru ca criptosistemul să nu poată fi spart trebuie să fie egală cu dimensiunea mesajului, ori în general în practică dorim cu ajutorul unei chei cât mai mici să criptăm cantităţi cât mai mari de informaţie, fiind ne-fezabil să avem chei de dimensiunea unui mesaj.

Un exemplu de sistem care nu poate fi spart este binecunoscutul **one-time pad**. Acesta este derivat din codul Vernam care este un cod stream ce presupune utilizarea unei chei aleatoare k de aceeaşi dimensiune cu a mesajului m şi criptotextul c se calculează ca $c_i = m_i \oplus k_i$ (se face XOR bit cu bit între mesaj şi cheie) cheia nu poate fi reutilizată pentru transmiterea altui mesaj. Se cunoaşte că sistemul a fost utilizat în practică pentru a cripta o linie de comunicaţie între Washington şi Moscova, dar mai relevant decât atât este faptul că sistemul mai apare ca bloc constructiv şi în protocoale de securitate contemporane.

Noţiunea de bază cu care lucrează teoria informaţiei este **entropia** informaţiei. Entropia măsoară gradul de incertitudine al informaţiei. Ea este utilizată, şi poate fi mai uşor văzută dintr-o perspectivă inginerească în acest fel, pentru a calcula numărul de biţi necesari pentru a codifica o anume informaţie. Spre exemplu dacă considerăm un zar cu n feţe, şi considerăm că zarul este perfect echilibrat (probabilitatea ca fiecare faţă să cadă fiind egală) avem nevoie de exact $\log_2 n$ biţi pentru a codifica fiecare eveniment de aruncare a zarului (eveniment finalizat prin marcarea numărului indicat de zar). Pentru cazul general definim entropia ca fiind:

Definiţia 9.2. *(Entropia) Pentru o variabilă* X *ce poate lua aleator valorile* $x_1, x_2, ..., x_n$ *cu probabilităţile* $p_1, p_2, ..., p_n$*, i.e.* $\Pr(X = x_i) = p_i$*, definim entropia lui* X *ca fiind* $H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$ *(se consideră* $p_i \log_2 p_i = 0$ *dacă* $p_i = 0$*).*

Definiţia poate părea greoaie, dar putem să o clarificăm uşor printr-un exemplu. Fie X o variabilă aleatoare de k biţi, conform definiţiei entropia trebuie să fie k. Avem $H(X) = -\sum_{i=1}^{2^k} \frac{1}{2^k} \log_2 2^{-k} = -\sum_{i=1}^{2^k} \frac{1}{2^k}(-k) = k$ şi astfel valoarea este verificată. Pe baza definiţiei putem înţelege şi de ce o parolă de k biţi, în cazul în care anumite caractere sunt mai probabile decât altele, nu are de fapt k biţi de entropie ci mai puţin.

9.3 CALCULUL OPERAŢIILOR ELEMENTARE

În continuare convenim să notăm cu $(x_k x_{k-1}...x_1)_b$ reprezentarea întregului x în baza b. Se poate demonstra uşor, fiind o consecinţă imediată a teoremei împărţirii cu rest, că fiecare întreg are o reprezentare unică în orice bază $b \geq 2$. Reprezentarea într-o baza b se face prin împărţire succesivă a câtului şi păstrarea restului, lucru care trebuie să fie deja cunoscut de cititor şi care nu îl vom mai detalia.

Prin operaţii elementare desemnăm operaţiile pe care le efectuăm în mod uzual cu orice întreg, aceste operaţii sunt: adunare, scădere, multiplicare, împărţire şi ridicare la putere. Subliniem că toate aceste operaţii pot fi efectuate în timp polinomial asupra întregilor de orice dimensiune. Enumerăm în continuare schema de principiu şi complexitatea acestor operaţii:

i) **Compararea** a doi întregi de k digiţi (dacă au număr diferit de digiţi este mai mare întregul cu mai mulţi digiţi) se face prin parcurgerea de la stânga la dreapta a digiţilor şi compararea la nivel de digit presupune $O(k)$ comparaţii la nivel de digit.

INTRARE: doi întregi $x = (x_k x_{k-1}...x_1)_b$, $y = (y_k y_{k-1}...y_1)_b$.

1. Setează $i = k$.
2. Atâta timp cât $x_i = y_i$ şi $i > 1$ calculează $i = i - 1$.
3. Dacă $x_i \geq y_i$ returnează 1 altfel returnează 0.

IEŞIRE: 1 dacă $x \geq y$ şi 0 în caz contrar.

ALGORITMUL 9-1. COMPARAREA A DOI ÎNTREGI

ii) **Adunarea** a doi întregi de k digiţi în baza b presupune $O(k)$ adunări la nivel de digit în baza b.

INTRARE: doi întregi $x=\left(x_k x_{k-1}...x_1\right)_b$, $y=\left(y_k y_{k-1}...y_1\right)_b$.

1. Setează $c=0$.
2. Pentru $i=\overline{1,k}$:

   $z_i=\left(x_i+y_i+c\right)\bmod b$,

   $c=\left(x_i+y_i+c\right)\text{div}b$,
3. $z_{k+1}=c$.
4. Returnează $z=\left(z_{k+1}z_k...z_1\right)_b$.

IEŞIRE: $z=x+y$.

ALGORITMUL 9-2. ADUNARA A DOI ÎNTREGI

iii) **Scăderea** a doi întregi de k digiţi în baza b presupune $O\left(k\right)$ scăderi la nivel de digit în baza b.

INTRARE: doi întregi $x=\left(x_k x_{k-1}...x_1\right)_b$, $y=\left(y_k y_{k-1}...y_1\right)_b$ cu $x\geq y$.

1. Setează $c=0$.
2. Pentru $i=\overline{1,k}$:

   $z_i=\left(x_i-y_i+c\right)\bmod b$,

   Dacă $x_i-y_i+c<0$ atunci $c=-1$ altfel $c=0$
3. Returnează $z=\left(z_k z_{k-1}...z_1\right)_b$

IEŞIRE: $z=x+y$.

ALGORITMUL 9-3. SCĂDEREA A DOI ÎNTREGI

iv) **Multiplicarea** a doi întregi de k digiţi în baza b presupune $O\left(k^2\right)$ multiplicări şi $O\left(k^2\right)$ adunări la nivel de digit în baza b. Operaţia de înmulţire având pondere mai mare faţă de adunare convenim să evaluăm complexitatea ca fiind $O\left(k^2\right)$ multiplicări la nivel de digit în baza b. Ridicarea la pătrat a unui întreg este un caz particular de multiplicare, fiind vorba de multiplicarea unui întreg cu el însuşi, şi există algoritmi optimizaţi pentru acest caz, fiind în principiu de 2 ori mai simplă decât multiplicarea a doi întregi diferiţi. Dacă întregii au număr diferiţi de digiţi, să spunem k respectiv l, atunci complexitatea este $O(kl)$, următorul algoritm poate fi utilizat:

INTRARE: doi întregi $x=\left(x_k x_{k-1}...x_1\right)_b$, $y=\left(y_l y_{l-1}...y_1\right)_b$.

1. Setează $c=0$.
2. Setează $z=\left(z_{k+l}z_{k+l-1}...z_1\right)_b=0$.
3. Pentru $i=\overline{1,k}$:

   Pentru $j=\overline{1,l}$:

   $z_{i+j}=\left(z_{i+j}+x_i y_j+c\right)\bmod b$,

   $c=\left(z_{i+j}+x_i y_j+c\right)\mathrm{div} b$,

   $z_{i+l}=\left(z_{i+j}+x_i y_j+c\right)\mathrm{div} b$,
4. Returnează z.

IEŞIRE: $z=xy$.

ALGORITMUL 9-4. ÎNMULŢIREA A DOI ÎNTREGI

v) **Împărţirea** (ne referim la împărţire cu rest) este cea mai costisitoare operaţie, are complexitate tot $O(kl)$ operaţii la nivel de digit dar indicatorul asimptotic pierde din acurateţe şi în practică împărţirea este mult mai costisitoare decât multiplicarea. Algoritmul nu este atât de direct, pentru care evităm prezentarea detaliată a lui (trimitem cititorul către algoritmul din [58] care este simplu de implementat) şi ne rezumăm doar la prezentarea unei strategii fără detalii (paşii 2.1. şi 2.2. nefiind detaliaţi)

INTRARE: doi întregi $x=(x_k x_{k-1}...x_1)_b$, $y=(y_l y_{l-1}...y_1)_b$.

1. Setează $r=x$.
2. Pentru $i=k-l+1$ până la 0 :

   $q_i=\lfloor r/b^i y \rfloor$,

   $r \leftarrow r-b^i q_i y$.
3. Returnează q,r

IEŞIRE: q,r astfel încât $x=yq+r$.

ALGORITMUL 9-5. ÎMPĂRŢIREA A DOI ÎNTREGI

vi) **Ridicarea la putere** presupune $O\left(\frac{3}{2}\log_2 e\right)$ multiplicări (acest timp a fost calculat având în vedere că multiplicarea unui număr cu el însuşi, i.e. ridicare la pătrat, costă aproximativ jumătate dintr-o multiplicare). Algoritmul poartă numele de „repeated square and multiply" şi presupune că exponentul este codificat în binar. Atenţie,

chiar dacă această metodă este suficient de rapidă pentru a face practică exponenţierea, în implementări se foloseşte în general un mecanism ceva mai avansat numit **sliding-window exponentiation**.

INTRARE: doi întregi $x=\left(x_k x_{k-1}...x_1\right)_b$, $e=\left(e_l e_{l-1}...e_1\right)_2$.

4. Setează $aux=x$, $y=1$
5. Pentru $i=\overline{1,l}$:

   Dacă $e_i=1$ atunci $y=y\cdot aux$,

   $aux=aux^2$,
6. Returnează y

IEŞIRE: $y=x^e$.

ALGORITMUL 9-6. RIDICAREA LA PUTERE MODULO N FOLOSIND REPEATED SQUARE AND MULTIPLY (RSM)

În criptografie, toate aceste operaţii le efectuăm în grupul de întregi Z_n. În acest caz avem nevoie de efectuarea operaţiilor modulo n dar tehnica rămâne în mare parte aceeaşi ca la cazul general cu următoarele observaţii:

i) Comparaţia este identică cu cazul general.

ii) La adunare modulo n dacă rezultatul este mai mare decât n (poate fi cel mult $2(n-1)$) atunci scădem n din rezultat.

iii) Scăderea modulo n este identică dacă descăzutul este mai mare decât scăzătorul. Dacă nu, atunci se face adunare cu opusul, deci scăderea din modul a scăzătorului şi adunarea la descăzut, i.e. $x-y \bmod n = x+(n-y) \bmod n$.

iv) Multiplicarea modulo n se face identic iar la final se recurge la împărţire cu rest la n.

v) Împărţirea modulo n este înmulţire cu inversul multiplicativ ceea ce înseamnă că mai întâi se calculează un invers multiplicativ (vezi secţiunea 4.2) şi apoi se trece la multiplicare modulo n ca la iii), i.e. $x/y \bmod n = xy^{-1} \bmod n$.

vi) Ridicarea la putere modulo n se face cu specificaţia că fiecare multiplicare (din paşii 5.1. şi 5.2.) se face modulo n.

Operaţiile de multiplicare şi ridicare la putere modulo n sunt operaţiile cele mai comun utilizate în criptografia cu cheie publică şi pentru acestea în practică se impune utilizarea unor tehnici optimizate. Algoritmii prezentaţi aici au doar scop didactic şi chiar dacă nu diferă mult la nivel de complexitate în aplicaţii unde timpul de calcul contează nu se recomandă folosirea lor. Pentru detalii asupra unor tehnici optimizate poate fi consultată orice carte generală de criptografie [62], [73], subiectul este prea amplu pentru a fi discutat în această lucrare. De asemenea este recomandată binecunoscuta carte a lui Knuth [55].

9.4 CALCULUL C.M.M.D.C. ŞI INVERSELOR MULTIPLICATIVE

Calculul celui mare divizor comun pe baza factorizării întregilor utilizând relaţia din Definiţia 8.8. nu este un procedeu convenabil din punct de vedere computaţional deoarece evident nu poate fi aplicat atunci când nu este cunoscută factorizarea întregilor (situaţia cea mai uzuală în practică) iar a factoriza un întreg nu este fezabil din punct de vedere computaţional. Astfel, pentru cazul când factorii celor două numere sunt necunoscuţi poate fi utilizat algoritmul Euclidian. Acesta se bazează pe faptul că dacă avem doi întregi $a > b$ atunci $c.m.m.d.c.(a,b) = c.m.m.d.c.(b, a \bmod b)$ şi această relaţie duce la posibilitatea calculului *c.m.m.d.c.* în timp logaritmic (relaţia nu o demonstrăm deoarece este o noţiune bine-cunoscută de teoria numerelor) . Relaţia poate fi aplicată recursiv pentru a obţine cel mai mare divizor al celor doi întregi a, b şi următorul algoritm sintetizează acest rezultat:

INTRARE: doi întregi a, b cu $a > b$

1. Atâta timp cat $b \neq 0$ execută:
2. $r = a \bmod b, a = b, b = r$
3. Returnează a

IEŞIRE: $c.m.m.d.c.(a,b)$

ALGORITMUL 9-7. CEL MAI MARE DIVIZOR COMUN

Se poate demonstra că pentru oricare doi întregi a, b dacă $d = c.m.m.d.c.(a,b)$ putem găsi o pereche de întregi x, y astfel încât $ax + by = d$. Acest calcul poate fi uşor efectuat folosind Algoritmul Euclidian Extins (AEE) pentru care trimitem către [62, p. 67]. Acum dorim să facem o expunere foarte simplă asupra principiului pe care funcţionează. Presupunem că $a = 97, b = 17$ şi dorim să calculăm $d = c.m.m.d.c.(97,17)$ conform algoritmului anterior, prin împărţirile succesive din pasul 2 obţinem următoarele rezultate intermediare:

1. $97 = 5 \cdot 17 + 12$
2. $17 = 1 \cdot 12 + 5$
3. $12 = 2 \cdot 5 + 2$
4. $5 = 2 \cdot 2 + 1$

Dacă acum introducem în relaţia de la pasul 4 pe cea de la pasul 3 obţinem $5 - 2 \cdot 2 = 1 \Rightarrow 5 - 2 \cdot (12 - 2 \cdot 5) = 1$ şi continuând tot aşa până la relaţia de la pasul 1 cu schimbarea restului cu o combinaţie liniară de termeni obţinem succesiv relaţiile:

$$5-2\cdot 2=1\Rightarrow 5-2\cdot(12-2\cdot 5)=1\Leftrightarrow 5\cdot 5-2\cdot 12=1\Rightarrow 5\cdot(17-1\cdot 12)-2\cdot 12=1\Leftrightarrow$$
$$\Leftrightarrow 5\cdot 17-7\cdot 12=1\Rightarrow 5\cdot 17-7(97-5\cdot 17)=1\Leftrightarrow 40\cdot 17-7\cdot 97=1$$

Într-adevăr la final am ajuns să scriem cel mai mare divizor a două numere ca şi combinaţie liniară între acestea. Este evident că pentru oricare doi întregi acest raţionament poate fi aplicat şi astfel prin parcurgerea în ordine inversă a rezultatelor intermediare ale Algoritmului Euclidian putem calcula **inverse multiplicative** pentru că este evident că dacă $c.m.m.d.c.(a,b)=1$ avem $a\equiv x^{-1}\bmod b$ respectiv $b\equiv y^{-1}\bmod a$. Cel mai comun exemplu de utilizare în criptografie este la calcularea unei perechi de chei publică-privată pentru RSA deoarece exponenţii din aceste chei sunt inverse multiplicative.

9.5 CALCULUL RĂDĂCINII DE ORDIN E CU RELATIV PRIM LA ORDINUL GRUPULUI

Am dedicat o secţiune separată acestei probleme doar datorită importanţei ei în criptografia cu cheie publică, fiind problema pe care se construieşte criptosistemul RSA. Calculul este însă direct, dacă se cunoaşte ordinul grupului Z_n şi exponentul este relativ prim la acesta se poate calcula inversul multiplicativ al exponentului modulo ordinul grupului, i.e. $\delta\equiv\varepsilon^{-1}\bmod\phi(n)$, şi rădăcina se extrage direct ca $y\equiv x^{\delta}\bmod n$, este evident că acum vom avea $y^{\varepsilon}\equiv x\bmod n$ [9]. Următorul algoritm prezintă acest rezultat:

[9] Ceea ce înseamnă de fapt $x\equiv\sqrt[\delta]{y}\bmod n$ şi $y\equiv\sqrt[\varepsilon]{x}\bmod n$.

INTRARE: întregul x, exponentul ε cu $c.m.m.d.c.\left(\varepsilon,\phi(n)\right)=1$ şi modulul n a cărui factorizare este cunoscută.

1. Pe baza factorizării lui n calculează $\phi(n)$ după relaţia din Teorema 8.13.
2. Calculează $\delta\equiv\varepsilon^{-1}\bmod\phi(n)$ folosind algoritmul Euclidian Extins din 4.2.
3. Calculează $y\equiv x^{\delta}\bmod n$.
4. Returnează y.

IEŞIRE: y care este rădăcina de ordin ε a lui x.

ALGORITMUL 9-8. RĂDĂCINA DE ORDIN E MODULO N

9.6 CALCULUL RĂDĂCINII PĂTRATE

Pentru calculul rădăcinilor pătrate vom distinge trei cazuri după cum calculul se desfăşoară în Z_p, Z_{p^l} sau Z_n unde p este un număr prim iar n un compozit.

Pentru cazul Z_p distingem primul caz ca fiind $p\equiv 3\bmod 4$, caz în care se observă că dacă $x\in Q_p$ şi $x\equiv y^2\bmod p$ atunci $y=x^{\frac{p+1}{4}}\bmod p$. Deci următorul algoritm poate fi utilizat dacă $p\equiv 3\bmod 4$:

INTRARE: un număr prim $p \equiv 3 \bmod 4$ şi un întreg oarecare $a \in Q_p$.

1. Calculează $r = a^{\frac{p+1}{4}} \bmod p$.
2. Returnează $\pm r \bmod p$.

IEŞIRE: cele doua rădăcini pătrate din Z_p^* ale lui a

ALGORITMUL 9-9. RĂDĂCINA PĂTRATĂ MODULO P PENTRU ÎNTRGI BLUM

Pe de altă parte, cel de-al doilea caz este când $p \equiv 1 \bmod 4$ şi algoritmul anterior nu mai poate fi aplicat (intuitiv se poate observa că $\frac{p+1}{4} = \frac{1+4k+1}{4} = \frac{2(2k+1)}{4}$ nu este un număr întreg). Pentru acest caz nu se cunosc algoritmi determinişti pentru calculul rădăcinilor[10] şi problema găsirii unui algoritm determinist a cărui complexitate depinde doar de n rămâne o problemă deschisă în criptografie. Există algoritmi non-determinişti eficienţi şi simplu de înţeles care folosesc ca intrare adiţională un non-reziduu cvadratic din Z_p^* pentru acest caz, i.e. $p \equiv 1 \bmod 4$, recomandăm pentru expunerea concisă [4, p. 32] şi evităm a prezenta un algoritm în ideea că în criptografie în general se folosesc preponderent întregi de tipul $p \equiv 3 \bmod 4$ pentru care există relaţia directă de extragere a rădăcinii pătrate.

Pentru cazul Z_{p^e} se foloseşte procedeul numit Hensell Lifting, se poate consulta pentru detalii [77, p. 288].

Pentru cazul Z_n unde n este un întreg compozit calculul reziduurilor cvadratice este posibil doar dacă se cunoaşte factorizarea întregului n, altfel spus este adevărat că problema factorizării unui întreg n se reduce în timp polinomial la problema calculului rădăcinilor pătrate în Z_n şi invers. Deci problemele sunt echivalente din punct de vedere computaţional. Dacă factorizarea lui n este

[10] În lucrarea [5, p. 32] se face referire către un algoritm determinist care are o complexitate ce depinde de valoarea lui n şi a lui a.

cunoscută calculul rădăcinilor pătrate în Z_n se poate efectua relativ uşor prin calculul rădăcinilor modulo factorii compozitului şi rezolvarea în continuare cu Teorema Chineză a Resturilor. Vom considera în continuare cazul $n = pq$ cu p,q numere prime, acest caz fiind important în criptografie iar restul cazurilor fiind uşor deductibile din acesta. Pentru acest caz prezentăm următorul algoritm:

INTRARE: un întreg compozit $n = pq$ cu p,q numere prime şi un întreg oarecare $a \in Q_n$.

1. Calculează rădăcinile pătrate $(r,-r)$ ale lui a în Z_p.
2. Calculează rădăcinile pătrate $(s,-s)$ ale lui a în Z_q.
3. Foloseşte algoritmul Euclidian extins pentru a calcula două numere c şi d astfel încât $cp + dq = 1$.
4. Calculează $x = (rdq + scp) \bmod n$ şi $y = (rdq - scp) \bmod n$.
5. Returnează $\pm x \bmod n$ şi $\pm y \bmod n$.

IEŞIRE: Cele patru rădăcini pătrate din Z_n^* ale lui a.

ALGORITMUL 9-10. RĂDĂCINA PĂTRATĂ MODUL N

9.7 CALCUL RĂDĂCINII PĂTRATE PENTRU MODULE BLUM

Am considerat utilă dedicarea unui paragraf separat pentru cazul modulelor tip Blum, deoarece acestea sunt utilizate în multe criptosisteme cu cheie publică, cum ar fi de exemplu semnătura digitală din 5.3.2. Astfel, pentru $p \equiv 3 \bmod 4$ şi $q \equiv 3 \bmod 4$ numărul $n = pq$ se numeşte întreg Blum. Următoarele remarci sunt adevărate în cazul unui întreg Blum:

i) Dacă $x \in Q_n$ atunci $x^{\frac{n-p-q+5}{8}} \bmod n$ este rădăcina pătrata a lui x.

ii) Dacă simbolul Jacobi $\left(\frac{x}{n}\right)=1$ atunci

$$x^{\frac{n-p-q+5}{4}} \bmod n = \begin{cases} x, & x \in Q_n \\ n-x, & x \in \overline{Q_n} \end{cases}.$$

Proprietăţile i) şi ii) le demonstrăm simultan după cum urmează: dacă $x^{\frac{n-p-q+5}{8}} \bmod n$ este rădăcina pătrată a lui x atunci $x \in Q_n$ şi $\left(x^{\frac{n-p-q+5}{8}}\right)^2 \equiv x \bmod n$. Datorită izomorfismului descris de teorema chineză a resturilor este suficient să arătam că $x^{\frac{n-p-q+5}{8}}$ este o rădăcină pătrata a lui x în Z_p respectiv Z_q, mai mult este suficient să arătam pentru Z_p deoarece relaţia este simetrică în p şi q. Se observă uşor că $\frac{n-p-q+5}{4} = \frac{(p-1)(q-1)}{4}+1$ deci $\left(x^{\frac{n-p-q+5}{8}}\right)^2 \equiv x^{\frac{n-p-q+5}{4}} \equiv x^{\frac{(p-1)(q-1)}{4}} \cdot x \bmod p$ dar $x^{\frac{(p-1)(q-1)}{4}} \equiv \left(x^{\frac{(q-1)}{2}}\right)^{\frac{(p-1)}{2}} \bmod p$ şi cum $\frac{(q-1)}{2}$ este o cantitate impară urmează că $x^{\frac{(p-1)(q-1)}{4}} \bmod p$ este chiar simbolul Legendre pentru x în Z_p deci $\left(x^{\frac{n-p-q+5}{8}}\right)^2 \equiv \pm x \bmod p$ după cum x este sau nu reziduu cvadratic – ceea ce trebuia demonstrat.

De asemenea este relevant de remarcat că dacă $p \neq q \bmod 8$ atunci $2 \in \overline{Q}_n$ şi deci înmulţirea cu 2 schimbă simbolul Jacobi din 1 în -1 şi invers (acest lucru se demonstrează direct din definiţia simbolului Jacobi).

Aceste observaţii sunt utile în diverse calcule precum cele de la semnătura digitală Rabin-Williams.

9.8 VERIFICAREA APARTENENŢEI LA MULŢIMEA REZIDUURILOR CVADRATICE

Pentru verificarea apartenenţei unui număr la Q_n sau $\overline{Q}_n$ este mai întâi necesar să discutăm despre calculul simbolurilor Legendre şi Jacobi. Simbolul

Legendre poate fi calculat fie pe baza relaţiei din Lema 8.5 a lui Euler fie cu acelaşi algoritm pentru simbolul Jacobi deoarece cele două sunt egale pentru numere prime. Bazat pe proprietăţile anterior amintite cu privire la reziduuri cvadratice, mai exact legea reciprocităţii cvadratice 8.15 şi caracterul cvadratic al lui 2 din 8.42, următorul algoritm facilitează calculul simbolului Jacobi:

INTRARE: un întreg impar $n \geq 3$ şi un întreg oarecare $0 \leq a < n$.

1. Dacă $a = 0$ returnează 0.
2. Dacă $a = 1$ returnează 1.
3. Scrie $a = 2^e a_1$ unde a_1 este un număr impar.
4. Calculează $s = \left(\frac{2^e}{n}\right)$ (se poate utiliza relaţia 9.30 dar deoarece aceasta presupune o exponenţiere se preferă următorul calcul care este echivalent: dacă e este par atunci $s = 1$, dacă $n \equiv \pm 1 \bmod 8$ atunci $s = 1$, dacă e este impar şi $n \equiv \pm 3 \bmod 8$ atunci $s = -1$.
5. Dacă $n \equiv 3 \bmod 4$ şi $a_1 \equiv 3 \bmod 4$ atunci $s = -s$ (conform legii reciprocităţii cvadratice).
6. Calculează $n_1 \equiv n \bmod a_1$.
7. Dacă $a_1 = 1$ returnează s altfel returnează $s \cdot Jacobi(n_1, a_1)$.

OUTPUT: simbolul Jacobi $\left(\frac{a}{n}\right)$.

ALGORITMUL 9-11. SIMBOLUL JACOBI

Acum, dacă pentru un număr a dorim să verificăm că este reziduu cvadratic modulo p, unde p este număr prim, atunci acest lucru este posibil prin calcularea simbolului Legendre după următoarea relaţia $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \bmod p$ sau aşa cum am mai spus chiar cu algoritmul anterior.

Pentru cazul în care pentru un număr a dorim să verificam că este reziduu cvadratic modulo p^l, unde p este număr prim, acest lucru se poate face prin verificarea rezultatului modulo p deoarece este evident că orice reziduu cvadratic modulo p^l este şi reziduu cvadratic modulo p.

Altfel dacă pentru un număr a dorim să verificăm că este reziduu cvadratic modulo n, unde n este un întreg compozit acest lucru nu mai este posibil deoarece prin calcularea simbolului Jacobi nu se poate trage o concluzie finală în cazul în care numărul este un pseudo-reziduu modulo n, deci $a \in Q_n$. Putem însă calcula simbolul Jacobi $\left(\frac{a}{n}\right)$ şi în cazul în care $\left(\frac{a}{n}\right) = -1$ putem spune cu certitudine că a nu este reziduu cvadratic iar dacă $\left(\frac{a}{n}\right) = 1$ nu se poate trage nici o concluzie. Pe de altă parte dacă factorizarea lui n este cunoscută se poate verifica dacă un număr este sau nu reziduu cvadratic, prin verificare modulo fiecare factor al lui n - este evident că un număr a este reziduu cvadratic modulo n dacă şi numai dacă este reziduu cvadratic modulo oricare factor al lui n. Pentru a verifica dacă un întreg este reziduu cvadratic modulo un număr prim se poate aplica acelaşi algoritm pentru calculul simbolului Jacobi, sau criteriul lui Euler din Lema 8.5, această abordare ducând la un timp de calcul ceva mai scăzut.

9.9 PROBLEMA FACTORIZĂRII ÎNTREGI (IFP)

Problema factorizării întregilor (IFP – Integer Factorization Problem) este o problemă de importanţă majoră deoarece probabil în jur de 50% din criptografia cu cheie publică are securitatea bazată pe imposibilitatea de a factoriza întregi mari. Un exemplu concret pentru a ilustra importanţa acestei probleme îl reprezintă concursul oferit de RSA Security, detalii se găsesc la [71] şi sumele oferite au fost date în Tabelul 1.2. Totodată, problema factorizării întregilor oferă un excelent studiu de caz unde elemente de teoria numerelor se împletesc cu elemente de teoria complexităţii. Acest paragraf urmăreşte să facă o trecere în revistă a câtorva algoritmi utilizaţi în factorizare. Pentru prezentări generale în problematica factorizării se consideră utile lucrările [66], [87] şi articolul [17].

În principiu pentru factorizarea unui întreg este suficientă găsirea unui algoritm care să poată descompune întregul în produs de doi întregi

supraunitari[11] şi interesează o astfel de descompunere doar pentru întregi impari[12]. Pentru factorizarea întregilor mari, vorbim de sute şi mii de cifre, nu se cunosc soluţii eficiente. Excepţie fac următoarele cazuri:

i) $n = p$, întregul n este număr prim, în acest caz se pot aplica teste de primalitate care dovedesc că întregul este prim, demonstrarea primalităţii fiind mult mai uşoară decât factorizarea (vezi secţiunea 4.9).

ii) $n = p^e$, întregul n este o putere a unui număr prim, în acest caz se poate calcula succesiv radicalul de ordin i pentru $2 \le i \le \lfloor \log_2 n \rfloor$ şi se verifică dacă rădăcina de ordin i este număr întreg. Deoarece ridicarea la putere şi radicalul sunt funcţii monoton crescătoare pe mulţimea numerelor reale o astfel de căutare poate fi efectuată în timp logaritmic (căutare binară).

iii) Întregii de forme speciale, având factori ale căror proprietăţi permit găsirea eficientă a acestora.

În continuare problema factorizării se va aborda pentru situaţii diferite de cele de la punctele i), ii), iii) şi subliniem că pentru criptografie este de mare importanţă cazul $n = p \cdot q$ unde p şi q sunt numere prime. Algoritmii pentru factorizare se împart în două mari categorii:

i) **Algoritmi dedicaţi** (pentru forme speciale) – astfel de algoritmi sunt utilizaţi în general pentru a calcula factori mai mici decât o margine superioară şi realizează acest lucru relativ repede (în practică marginea superioară este aleasă în funcţie de resursele de calcul).

ii) **Algoritmi generali** – aceşti algoritmi sunt utilizaţi pentru a factoriza întregi cu factori relativ mari deoarece consumă aceleaşi cantităţi de timp indiferent dacă factorii sunt mici sau mari.

---

[11] Această problemă este cunoscută sub numele de splitting, iar dacă un algoritm pentru această problemă ar fi cunoscut atunci el s-ar putea aplica recursiv până la aflarea factorizării complete a oricărui întreg.

[12] Este important de remarcat că în fapt interesează doar factorizarea numerelor impare deoarece a factoriza un număr par conduce la împărţire cu 2 până când se ajunge tot la un număr impar, deci problema factorizării întregilor poate fi numită problema factorizării întregilor impari.

Astfel o strategie generală de factorizare, din motive evidente, arată ca în Figura 9.3. Metodele prezentate în acest capitol pot fi clasificate conform cu Figura 9.4. Această taxonomie este doar orientativă întrucât pe de o parte prezentarea va sintetiza numai o parte din metodele de factorizare existente în literatură, iar pe de altă parte problema factorizării poate fi considerată în continuare o problemă deschisă.

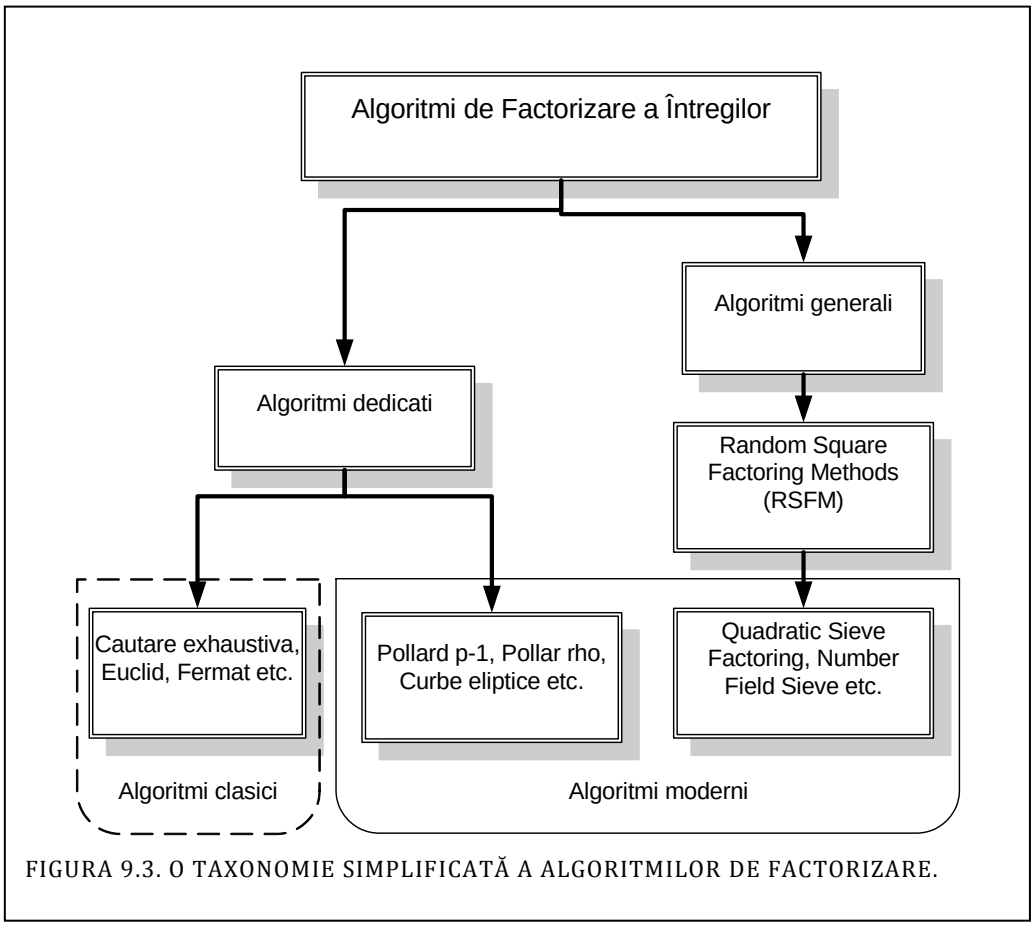


FIGURA 9.3. O TAXONOMIE SIMPLIFICATĂ A ALGORITMILOR DE FACTORIZARE.

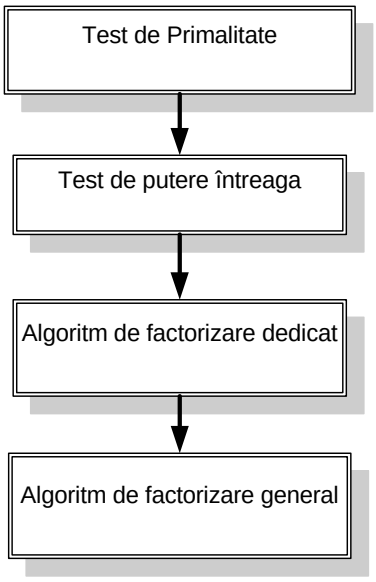


FIGURA 9.4. SCHEMA GENERALĂ A UNEI STRATEGII DE FACTORIZARE.

9.9.1 ALGORITMI DE FACTORIZARE CLASICI

Cel mai bine-cunoscut algoritm de factorizare, cu toate că este şi cel mai ineficient, este **căutarea exhaustivă**. Aceasta presupune încercarea de a-l împărţi pe n la valori succesive în intervalul $[2,\lfloor\sqrt{n}\rfloor]$ în scopul găsirii unui factor (în mod cert orice întreg compozit are cel puţin un factor în acest interval). Complexitatea algoritmului este $O(\sqrt{n})$, această soluţie nefiind practică pentru întregii mari, de sute sau mii de biţi.

Se poate relativ uşor observa că Algoritmul Euclidian poate fi utilizat în scopul factorizării. Factorizarea bazată pe algoritmul lui **Euclid** presupune construcţia de produsului $P_k = \prod_{i=1}^{k} p_i$ a primelor k numere prime (sau, mai util în practică construcţia, mai multor produse a numerelor prime din anumite intervale) şi apoi calcularea $c.m.m.d.c.(n, P_k)$ unde n este întregul care trebuie factorizat. Evident soluţia nu este mai eficientă decât căutarea exhaustivă, mai

mult ea necesită şi spaţii de memorie pentru stocarea produsului, spaţii care pentru întregi mari nu sunt disponibile. Cu toate acestea soluţia prezintă un avantaj pentru care uneori a fost utilizată în practică, acesta este faptul că se pot calcula apriori, înainte de a cunoaşte întregul n care trebuie factorizat, tabele cu aceste produse pentru valori rezonabile şi apoi pot fi utilizate pentru scoaterea factorilor mici din orice întreg care se doreşte factorizat. Problema algoritmului nu constă doar în timpul de calcul deoarece calculul celui mai mare divizor comun se rezolvă în timp polinomial, ci în spaţiul necesar deoarece calcularea produselor va necesita un spaţiu de stocare care creşte exponenţial – deci dacă aplicăm această metodă vom rămâne rapid fără spaţiu de stocare.

Factorizarea **Fermat** nu este nici ea mai eficientă decât o căutare exhaustivă, însă are mare interes atât din punct de vedere didactic cât şi pentru faptul că algoritmi avansaţi de factorizare se bazează pe idei înrudite cu aceasta. Fermat a observat că orice întreg impar poate fi scris ca diferenţă de exact două pătrate perfecte, iar scrierea unui număr ca diferenţă de pătrate conduce spre factorizarea lui pentru că $n=a^2-b^2 \Rightarrow n=(a-b)(a+b)$ şi $c.m.m.d.c.(a-b,n)$ sau $c.m.m.d.c.(a+b,n)$ pot da câte un factor al lui n. În mod cert soluţia banală a problemei, care este $n=(k+1)^2-k^2$ unde $k=\frac{n-1}{2}$, nu conduce decât la factorii banali ai lui n care sunt 1 şi n - deci interesează găsirea unei soluţii non-banale. Pentru întregii de forma $n=pq$ comun utilizaţi în criptografie se observă că $n=\left(\frac{p+q}{2}\right)^2-\left(\frac{p-q}{2}\right)^2$ de unde prin calcularea celui mai mare divizor comun, aşa cum s-a arătat anterior, rezultă imediat cei doi factori primi. În acest scop se alege $i=\lfloor\sqrt{n}\rfloor$ şi se calculează succesiv $j=(i+1)^2-n$ verificându-se dacă j este pătrat perfect. În esenţă aceasta înseamnă o căutare exhaustivă plecând de la $\lfloor\sqrt{n}\rfloor$ în sus. Verificarea faptului că un număr este pătrat perfect poate fi optimizată deoarece pătratele perfecte se termină doar în 0, 4, 5, 6, 9, mai mult modulo 16, extrem de util în inginerie deoarece pot fi extraşi doar ultimii patru biţi, sunt congruente doar cu 0, 1, 4, 9 – prin această metodă putând fi eliminate rapid numere care sigur nu sunt pătrate perfecte. Complexitatea algoritmului este însă tot $O(\sqrt{n})$ iar în practică algoritmul este de multe ori mai slab decât căutarea exhaustivă (de exemplu în cazul când numărul are un factor foarte mic şi unul foarte mare).

Pentru prezenta lucrare este suficientă prezentarea acestor algoritmi clasici de factorizare, principii utilizate de aceştia fiind utilizate şi în algoritmii moderni din următoarele două secţiuni. Menţionăm că mai există şi alte metode clasice de factorizare pentru care cititorul este îndrumat către [66], [87]. Datorită faptului că aceşti algoritmi clasici pot fi utilizaţi destul de eficient pentru găsirea factorilor relativ mici ei intră în categoria algoritmilor de factorizare dedicaţi.

9.9.2 ALGORITMI DE FACTORIZARE DEDICAŢI

Tehnicile introduse de Pollard şi utilizarea curbelor eliptice propuse de Lenstra sunt cele mai relevante metode dedicate de factorizare.

Algoritmul **Pollard p-1** se bazează tot pe construcţia unui produs asemenea celui din cazul metodei Euclid dar acest produs va fi utilizat în conjuncţie cu mica teoremă a lui Fermat. Astfel pentru un întreg compozit n construim produsul $Q=\prod_{q_i<B} q^{\lfloor \log_{q_i} n \rfloor}$ unde numerele q_i sunt toate numerele prime mai mici decât B. Dacă un număr p este un factor al lui n şi toţi divizori lui $p-1$ sunt mai mici decât B atunci $c.m.m.d.c.\left(x^Q-1,n\right)=p$ (această proprietate se poate demonstra imediat din mica teoremă a lui Fermat). Un număr care are toţi factorii mai mici decât un număr B se numeşte B-neted. Considerăm util următorul exemplu:

Exemplu:

$n=13\cdot 17$

$(17-1)\ este\, 2-neted$

$\Rightarrow Q=\prod_{q\le 2} q^{\lfloor \log_q(n) \rfloor}=2^{\lfloor \log_2 221 \rfloor}=128$

$\Rightarrow cmmdc\left(x^{128}-1,n\right)=17\ \forall x\, relativ\ prim\, la\, 17$

Ceva mai eficiente şi inovatoare sunt metodele bazate pe găsirea unor coliziuni de pătrate perfecte, apropiate ca idee de metoda de factorizarea a lui Fermat. Algoritmul **Pollard-rho** se bazează pe posibilitatea găsirii unei coliziuni de numere în Z_n folosind o funcţie $f(x): Z_n \to Z_n$ şi calculând şirul recurent $x_i = f\left(x_{i-1}\right)$. Deoarece funcţia este definită pe o mulţime finită elemente şirului

recurent vor începe la un moment să descrie un ciclu iar dacă se găsesc două elemente $x_i = x_j, i \neq j$, deci o coliziune, putem obţine factorizarea lui n dacă, similar cu cazul metodei Fermat, putem scrie o relaţie de forma $a^2 \equiv b^2 \bmod n \Rightarrow (a-b)(a+b) = kn$ şi deci aşa cum s-a observat şi anterior $c.m.m.d.c.(a-b,n)$ sau $c.m.m.d.c.(a+b,n)$ pot da câte un factor al lui n. Complexitatea de timp a acestui algoritm este $O(\sqrt{n})$ iar cea de spaţiu este $O(\sqrt{n})$ deoarece lungimea unor astfel de cicluri pentru funcţii cu un comportament stohastic este de aproximativ $\sqrt{\frac{\pi n}{8}}$. În mod cert complexitatea de spaţiu face imposibilă utilizarea unui astfel de algoritm în practică. Acest dezavantaj este însă înlăturat prin utilizarea funcţiei $f(x): Z_n \to Z_n, f(x) = x^2 + 1 \bmod p$ unde $p \mid n$. Bineînţeles că factorul p al lui n este necunoscut dar calculul poate fi efectuat şi modulo n deoarece dacă $f(x) = a \bmod n$ atunci $f(x) \equiv a \bmod p$. Astfel vom considera şirul recurent $x_i = f(x_{i-1}), x_0 = 2$ şi calculăm succesiv perechi (x_i, x_{2i}) până la găsirea unei coliziuni $x_i = x_{2i}$. Dacă se găseşte o astfel de coliziune în mod evident $c.m.m.d.c.(x_i - x_{2i}, n) = p$ ceea ce conduce la factorizare (se pot utiliza şi alte valori pentru x_0 sau pentru funcţia $f(x)$, de exemplu orice $f(x) = x^2 + c$). Conform aceloraşi raţionamente asupra ciclului descris de şirul recurent complexitatea de timp acestui algoritm devine $O(\sqrt[2]{p}) \approx O(\sqrt[4]{n})$, spaţiul de stocare solicitat este nesemnificativ. Considerăm util următorul exemplu:

Exemplu:

$n = 13 \cdot 17 = 221$, $f(x) = x^2 + 1$, $x_i = f(x_{i-1})$

$x_0 = 2$	$x_0 = 2$	$cmmdc(x_i - x_{2i}, 221)$
$x_1 = 2^2 + 1 \equiv 5 \bmod 221$	$x_2 = (2^2 + 1)^2 + 1 \equiv 26 \bmod 221$	1
$x_2 = 5^2 + 1 \equiv 26 \bmod 221$	$x_4 = (26^2 + 1)^2 + 1 \equiv 197 \bmod 221$	1
$x_3 = 26^2 + 1 \equiv 14 \bmod 221$	$x_6 = (197^2 + 1)^2 + 1 \equiv 104 \bmod 221$	1
$x_4 = 26^2 + 1 \equiv 197 \bmod 221$	$x_8 = (104^2 + 1)^2 + 1 \equiv 145 \bmod 221$	13

9.9.3 ALGORITMI DE FACTORIZARE GENERALI

Dezvoltăm în continuare ideea de a găsi doi întregi x, y astfel încât $x^2 \equiv y^2 \bmod n$ şi $x \neq \pm y \bmod n$ ca apoi să extragem factorii prin calcularea celui mai mare divizor comun. Strategia pe care se bazează familia de algoritmi **Random Square Factoring Methods** (RSFM) este următoarea:

i) Se alege o mulţime $S = \{p_1, p_2, p_3, ..., p_t\}$ numită bază de factorizare formată din t numere prime.

ii) Se aleg perechi de întregi (a_i, b_i) astfel încât:

- $b_i \equiv a_i^{\,2} \bmod n$
- $b_i = \prod_{j=1}^{t} p_j^{e_{ij}}$ - ceea ce înseamnă conform definiţiilor introduse anterior că b_i este neted raportat la baza de factorizare, acest lucru fiind uşor de verificat prin împărţiri succesive la elementele bazei de factorizare.

iii) Se încearcă găsirea unei mulţimi de T de elemente b_i astfel încât în produsul tuturor elementelor b_i din mulţimea T exponenţii numerelor prime p_i sa fie numere pare. Pentru simplificarea problemei fiecărui element b_i i se asociază un vector $\bar{e}_i = (e_{i1}, e_{i2}, e_{i3}, ..., e_{it})$ care conţine exponenţii tuturor numerelor prime din factorizarea sa:

$$\left.\begin{array}{l} b_1 = p_1^{e_{11}} p_2^{e_{12}} \ldots p_t^{e_{1t}} \\ b_2 = p_1^{e_{21}} p_2^{e_{22}} \ldots p_t^{e_{2t}} \\ \ldots \\ b_t = p_1^{e_{t1}} p_2^{e_{t2}} \ldots p_t^{e_{tt}} \end{array}\right\} \Rightarrow \begin{array}{c} \bar{e}_1 = (e_{11}, e_{12}, ..., e_{1t}) \\ \bar{e}_2 = (e_{21}, e_{22}, ..., e_{2t}) \\ \ldots \\ \bar{e}_t = (e_{t1}, e_{t2}, ..., e_{tt}) \end{array}$$

Fiecare vector este pus în corespondenţă cu un vector binar $\overline{e_i}'$ care reţine doar paritatea elementelor din $\overline{e_i}$:

$$\left.\begin{array}{l} \overline{e_1} = (e_{11}, e_{12}, ..., e_{1t}) \\ \overline{e_2} = (e_{21}, e_{22}, ..., e_{2t}) \\ \quad ... \\ \overline{e_t} = (e_{t1}, e_{t2}, ..., e_{tt}) \end{array}\right\} \Rightarrow \begin{array}{l} \overline{e_1}' = (e_{11} \bmod 2, e_{12} \bmod 2, ..., e_{1t} \bmod 2) \\ \overline{e_2}' = (e_{21} \bmod 2, e_{22} \bmod 2, ..., e_{2t} \bmod 2) \\ \quad ... \\ \overline{e_t}' = (e_{t1} \bmod 2, e_{t2} \bmod 2, ..., e_{tt} \bmod 2) \end{array}$$

Ceea ce trebuie aflat pentru rezolvarea problemei este dacă sistemul format de vectori $\overline{e_i}'$ este sau nu liniar dependent. Dacă este liniar dependent înseamnă că există o combinaţie liniară de vectori a cărei sumă este vectorul nul. În acest caz factorizarea este găsită deoarece în mod evident $\prod_{b_i \in T} b_i = x^2 = \prod_{i=1}^{t} a_i^2$ şi astfel s-a găsit o congruenţă de tipul $x^2 \equiv y^2 \bmod n$.

Exemplu:

$n = 13 \cdot 17 = 221$

$S = \{2,3,5,7\}$

$58^2 \bmod 221 = 49 = 7^2 \Rightarrow e_1 = (0,0,0,0)$

$56^2 \bmod 221 = 42 = 2 \cdot 3 \cdot 7 \Rightarrow e_2 = (1,1,0,1)$

$19^2 \bmod 221 = 140 = 2^2 \cdot 5 \cdot 7 \Rightarrow e_3 = (0,0,1,1)$

$35^2 \bmod 221 = 120 = 2^3 \cdot 3 \cdot 5 \Rightarrow e_4 = (1,1,1,0)$

$45^2 \bmod 221 = 36 = 2^2 \cdot 3^2 \Rightarrow e_5 = (0,0,0,0)$

$Se observa ca e_2 + e_3 + e_4 \equiv 0 \bmod 2$

$\Rightarrow (56 \cdot 19 \cdot 35)^2 \equiv 2^6 \cdot 3^2 \cdot 5^2 \cdot 7^2 \bmod 221$

$\Leftrightarrow (56 \cdot 19 \cdot 35 - 2^3 \cdot 3 \cdot 5 \cdot 7) \cdot (56 \cdot 19 \cdot 35 + 2^3 \cdot 3 \cdot 5 \cdot 7) = k \cdot n$

$cmmdc(56 \cdot 19 \cdot 35 - 2^3 \cdot 3 \cdot 5 \cdot 7, n) = 13, cmmdc(56 \cdot 19 \cdot 35 + 2^3 \cdot 3 \cdot 5 \cdot 7, n) = 17$

Cel mai puternic şi promiţător algoritm de factorizare general cunoscut este **Number Field Sieve** care face parte tot din familia RSFM. Detaliile acestui algoritm sunt destul de complexe, pentru o prezentare mai detaliată trimitem către [66], [87], [58].

9.10 PROBLEMA LOGARITMULUI DISCRET (DLP)

Definiţia 5.3. Problema logaritmului discret (DLP – Discrete Logarithm Problem): Având un număr prim p un generator α al grupului Z_p^* şi valoarea lui $\alpha^x \bmod p$ găseşte-l pe x.

Desigur problema poate fi generalizată pe orice alt grup decât Z_p^* şi mai mult pentru elemente care nu sunt neapărat generatori ai grupului – desigur problema având relevanţă pentru criptografie atunci când elementul are ordin suficient de mare. În cele ce urmează vom nota ordinul lui α cu k, evident dacă α este generator al lui Z_p^* şi p este prim atunci $k = p-1$.

Soluţia banală este **căutarea exhaustivă** dar această presupune $O(k)$ multiplicări pentru care nu poate fi utilizată în practică. O optimizare directă a acestei căutări este algoritmul **baby-step giant-step**. Acesta se bazează pe observaţia simplă că putem scrie $x = i\left\lfloor \sqrt{k} \right\rfloor + j$ şi se poate calcula off-line o tabelă cu puterile lui α până la j, în mod evident ca şi consecinţă a teoremei împărţirii cu rest avem $j < \left\lfloor \sqrt{k} \right\rfloor$. Tabela se păstrează sortată după valorile lui $\alpha^j, j = \overline{1,\left\lfloor \sqrt{k} \right\rfloor}$ pentru găsirea rapidă a valorilor din aceasta (prin căutare binară care are timp logaritmic). Odată realizată această tabelă se poate calcula succesiv pentru $i = \overline{1,\left\lfloor \sqrt{k} \right\rfloor}$ valoarea lui $\left(\alpha^{-\sqrt{k}}\right)^i \cdot \alpha^x \bmod p$ şi se verifică dacă această valoare se află în tabel. În caz afirmativ valorile lui i şi j au fost determinate şi deci x poate fi calculat. În mod evident algoritmul are nevoie de cel mult $O\left(\sqrt{k}\right)$ paşi, fiecare pas constând într-o multiplicare şi o căutare iar căutarea are timp logaritmic deci poate fi neglijată. Problema este faptul că algoritmul utilizează $O\left(\sqrt{k}\right)$ memorie – aceasta fiind limitarea majoră a sa. Pentru a elimina necesităţile de memorie se poate utiliza algoritmul **Pollar-rho**.

Cel mai performant algoritm pentru rezolvarea problemei logaritmului discret este algoritmul de **calcul indexa**t (index calculus) avantajul major al acestuia fiind că poate fi presetat off-line pe un anume grup şi utilizat apoi asupra oricărui generator din grup. Pentru valori mari ale ordinului grupului nici acest algoritm nu este suficient de eficient, problema logaritmului discret rămânând nerezolvabilă în practică. Trebuie spus că algoritmul de calcul indexat nu funcţionează asupra problemei logaritmului discret pe curbe eliptice, iar acest fapt a dus şi la succesul criptografiei pe curbe eliptice în practică, pentru că metodele de a sparge aceste criptosisteme sunt destul de slabe.

9.11 GENERAREA NUMERELOR PRIME

Problema generării numerelor prime reprezintă pasul de plecare în implementarea oricărui criptosistem în practică, aşa cum se va vedea atât criptosistemele bazate pe factorizarea întregilor, precum RSA, cât şi cele bazate pe logaritmi discreţi, precum Diffie-Hellman-Merkle sau ElGamal, au nevoie de numere prime la generarea cheii.

Numerele prime pe care le utilizăm în criptografie sunt numere mari de sute şi mii de biţi. Generarea acestora urmează întotdeauna această paradigmă: se generează o secvenţă aleatoare (de fapt adesea pseudo-aleatoare în practică) având numărul de biţi dorit şi se supune unui test de primalitate. Probabilitatea de a alege aleator un număr şi aceasta să fie prim este destul de mare deoarece numărul de numere prime până la o valoare x este aproximativ $x/\ln x$. Următorul algoritm sintetizează această procedură:

INTRARE: 1^k

1. Generează k biţi aleatori $b_k b_{k-1} \ldots b_1$.
2. Verifică dacă numărul $x = (x_k x_{k-1} \ldots x_1)_2$ este prim şi dacă da returnează x dacă nu întoarce-te la pasul 1.

IEŞIRE: x prim.

ALGORITMUL 9-12. GENERARA UNUI NUMĂR PRIM

Dacă această schemă de principiu a fost lămurită tot ce trebuie să stabilim este cum arată un test de primalitate. Există două tipuri de teste de primalitate: teste probabilistice care validează numărul ca fiind prim cu o anumită probabilitate (deci eroarea testului este ne-nulă) şi teste reale de primalitate care demonstrează că un număr este prim (deci eroarea testului este nulă). În practică este recomandat testul Miller-Rabin, care este un test probabilist suficient de rapid şi cu probabilitate foarte mică de eroare, pentru acesta pot fi consultate lucrările [35], [62] şi orice altă carte de referinţă.

Continuăm cu prezentarea a două teste probabilistice de primalitate. Unul dintre testele cele mai simple de primalitate este **testul Fermat** care se bazează pe mica teoremă a lui Fermat 8.6. Folosind acest rezultat este foarte clar că numărul p este prim doar dacă $a^{p-1} \equiv 1 \bmod p$. Algoritmul de principiu asociat testului Fermat este următorul:

INTRARE: un număr aleator p

1. Pentru $i = \overline{1,t}$

   alege un număr aleator $1 < a < p$,

   calculează $x = a^{p-1} \bmod p$,

   dacă $x \neq 1$ returnează 0.

2. Returnează 1.

IEŞIRE: dacă x este prim 1 altfel 0.

ALGORITMUL 9-13. TESTUL FERMAT DE PRIMALITATE

Dacă algoritmul returnează 0 este foarte clar că numărul nu este prim, dar dacă returnează 1 atunci nu se poate ştii sigur ci doar cu o probabilitate care depinde de numărul t. Există însă suficient de multe numere a pentru care $a^{p-1} \equiv 1 \bmod p$ chiar dacă p nu este prim – aceste numere poartă numele de „Fermat liar”. Pentru a creşte probabilitatea de succes a testului de primalitate poate fi utilizat şi criteriul lui Euler pentru calculul simbolului Legendre din Lema

8.5 folosind relaţia $\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}}$. Testul construit pe acest principiu este cunoscut sub numele de Soloway-Strassen şi are următoarea descriere:

INTRARE: un număr aleator p

3. Pentru $i = \overline{1,t}$:

   Alege un număr aleator $1 < a < p$.

   Calculează $x = a^{\frac{p-1}{2}} \bmod p$.

   Dacă $x \neq \pm 1$ returnează 0.

   Calculează $y = \left(\frac{x}{p}\right)$ folosind algoritmul pentru simboluri Jacobi din 10.6.

   Dacă $x \neq y$ returnează 0.

4. Returnează 1.

IEŞIRE: dacă x este prim 1 altfel 0.

ALGORITMUL 9-14. TESTUL SOLOWAY-STRASSEN DE PRIMALITATE

Probabilitatea ca un întreg să fie declarat prim în mod fals de testul Soloway-Strassen este $\left(\frac{1}{2}\right)^t$ iar testul Miller-Rabin, care nu îl vom descrie, are o probabilitate de eroare de $\left(\frac{1}{4}\right)^t$. Numerele a care fraudează testul Solovay-Strassen poartă numele de „Euler liar” iar cele care fraudează testul Miller-Rabin numele de „strong liar”. Se poate demonstra că sunt cel mult $\phi(n)/2$ astfel de numere pentru orice compozit, de aici şi valoarea probabilităţii de a frauda acest test.

9.12 MULTIPLICAREA UNUI PUNCT DE PE O CURBĂ ELIPTICĂ

Dacă exponenţierea era operaţia de bază în cadrul grupurilor construite pe clase de resturi modulo n, operaţia analoagă în cazul grupurilor construite peste puncte ale curbelor eliptice este adunarea punctelor de pe curbă. Adică operaţia frecvent efectuată g^k pentru un generator g şi un număr aleator k devine kP unde P este un punct de pe curbă.

Multiplicarea unui punct cu o constant, care înseamnă adunare succesivă, se poate calcula eficient în manieră similară cu exponenţierea (care folosea algoritmul Repeated-Square-and-Multiply) prin adunări succesive. De exemplu pentru a calcula $99P$ nu este necesar să adunăm punctul succesiv de 99 de ori cu el însuşi pentru că putem scrie simplu $99P = 2(P + 2*2*2*2*2(P + 2*P)) + P$ ceea ce conduce la 3 adunări şi 7 multiplicări cu 2 (dublări) ale punctului.

INTRARE: punctul P şi un întreg $k = (k_l k_{l-1} \ldots k_1)$.

1. Setează $aux = P$, $y = 1$
2. Pentru $i = \overline{l,1}$:

   $aux = aux + aux$,

   Dacă $k_i = 1$ atunci $aux = aux + P$,
3. Returnează aux

IEŞIRE: kP.

ALGORITMUL 9-15. MULTIPLICAREA UNUI PUNCT DE PE O CURBĂ ELIPTICĂ

10 BIBLIOGRAFIE

[1] R. J. Anderson, (2001), *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, 640 pagini, ISBN 0471389226.

[2] T.M. Apostol, (1976), *Introduction to Analytic Number Theory*, Springer-Verlag, 329 pagini, ISBN 0387901639.

[3] J. Bayne, (2002), *An Overview of Threath and Risk Assesment*, SANS Institute.

[4] M. Bellare, S. Goldwasser, (2001), *Lecture Notes on Cryptography*, Curs: MIT, disponibil la http://www.cs.ucsd.edu/users/mihir/papers/gb.html.

[5] M. Bellare, P. Rogaway, (1993), *Random oracles are practical: A paradigm for designing efficient protocols*, ACM Conference on Computer and Communications Security, pages 62–73.

[6] M. Bellare, P. Rogaway, (1995), *Optimal asymetric encryption – How to encrypt with RSA*, Advances in Cryptology – EuroCrypt 94, LNCS vol. 950, Springer-Verlag.

[7] M. Bellare, P. Rogaway, (1996), *The exact security of digital signatures-how to sign with RSA and Rabin*. In Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques (EUROCRYPT'96), Ueli Maurer (Ed.). Springer-Verlag, Berlin, Heidelberg, 399-416.

[8] M. Bellare, R. Canetti, H. Krawczyk, (1996), *Keying Hash Functions for Message Authentication*, Advances in Cryptology – CRYPTO 96, LNCS vol. 1109, Springer-Verlag.

[9] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, (1998), *Relations among notions of security for public-key encryption schemes*, Advances in Cryptology – CRYPTO 98, LNCS, vol. 1462, pp.26 – 45, Springer-Verlag.

[10] I. Blake, G. Seroussi, N. Smart, (1999), *Elliptic Curves in Cryptography*, Cambridge University Press, ISBN 0521653746.

[11] I. Blake, G. Seroussi, N. Smart, (2005), *Advances in Elliptic Curve Cryptography*, Cambridge University Press, ISBN 052160415X.

[12] D. Bleichenbacher, U. Maurer, (1994), *Directed Acyclic Graphs, One-way Functions and Digital Signatures*, Advances in Cryptology CRYPTO94, 75-82, LNCS 839, Springer-Verlag.

[13] D. Bleichenbacher, U. Maurer, (1996), *On the Efficiency of One-time Digital Signatures*, Advances in Cryptography ASIACRYPT 96, pp. 145—58, LNCS 1163, Springer-Verlag.

[14] L. Blum, M. Blum, M. Shub, (1982), *Comparison of Two Pseudo-Random Number Generators*, Advances in Cryptology – CRYPTO 82, pp. 61-78, Springer-Verlag.

[15] L. Blum, M. Blum, M. Shub, (1986), *A Simple Unpredictable Pseudo-Random Number Generator*, SIAM Journal on Computing, Volume 15, Issue 2 , pp. 364 – 383.

[16] D. Boneh, R. Venkatesan, (1998), *Breaking rsa may not be equivalent to factoring*, Proceedings of Eurocrypt 98, LectureNotes in Computer Science, vol. 1233, pp. 59–71, Springer-Verlag.

[17] R.P. Brent, (1990), *Primality Testing and Integer Factorization*, The Australian National University TR-CS-90-03.

[18] D. Brumley, D. Boneh, (2003), *Remote Timing Attacks are Practical*, Proceedings of the 12th Usenix Security Symposium.

[19] E. Byres, J. Lowe, (2004), The Myths and Facts behind Cyber Security Risks for Industrial Control Systems. VDE Congress'04.

[20] R. Canetti, O. Goldreich, S. Halevi, (2004), *The random oracle methodology, revisited*, Journal of the ACM (JACM), Volume 51 , Issue 4, pp. 557 – 594.

[21] D. Chaum, (1982), *Blind sigantures for Untraceable payments*, Advances in Cryptology – CRYPTO 82, Springer-Verlag.

[22] D. Chaum, (1985), *Security without identification – Card Computers to make Big Brother Obsolete*, Communication of the ACM.

[23] H. Cohen, (1993), *A Course in Computational Algebraic Number Theory*, Springer, ISBN 0387942939, 534 pagini.

[24] H. Cohen, G. Frey et al., (2006), *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Chapman & Hall/CRC, ISBN 1584885181, 848 pagini.

[25] T.H. Cormen, C.E. Leiserson, R.L. Rivest, (1990), *Introduction to Algorithms*, MIT Press, Cambridge, Massachusets, 1048 pagini, ISBN 0262530910.

[26] CNSS, (2006), NATIONAL INFORMATION ASSURANCE (IA) GLOSSARY http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf.

[27] R. Cramer, V. Shoup, (2004), *Design and analysis of practical public-key encryption schemes secure against adaptive chosenciphertext attack*, SIAM Journal on Computing, vol. 33, Issue1, pp. 167 – 226.

[28] DADS, (2007), Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology, availlable at www.nist.gov/dads/.

[29] W. Diffie, M.E. Hellman, **(**1976), *New directions in cryptography*, IEEE Transactions on Information Theory.

[30] D. Dolev, C.Dwork, M. Naor, (1991), *Non-malleable cryptography*. Proceedings of the 23rd Symposium on Theory of Computing, ACM STOC, pp. 542–552.

[31] D. Dzung, M. Naedele, T.P. Hoff, M. Crevatin, (2005), *Security for Industrial Communication Systems*, Proceedings of the IEEE, vol. 93, no. 6.

[32] S. Even, O. Goldreich, S. Micali, (1995), *On-line/offline Digital Signatures*, Journal of Cryptology, pp. 35-67, Springer-Verlag.

[33] T. ElGamal, (1985), *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory.

[34] W. Feller, (1968), *An Introduction to Probability Theory and Its Applications*, Volume 1, Wiley; 3 edition, 528 pagini, ISBN 0471257087.

[35] N. Ferguson, B. Schneier, (2003), *Practical Cryptography*, Wiley, 432 pagini, ISBN 0471223573.

[36] FIPS 46, (1976), re-înnoit în 1988, 1993, 1999 ca FIPS 46-1, 46-2, 46-3, *Data Encryption Standard (DES)*, National Institute of Standards and Technology (NIST)., U.S. Department of Commerce.

[37] FIPS 180-1, (1995), 180-2, (2002), *Announcing the Secure Hash Standard.*, National Institute of Standards and Technology (NIST)., U.S. Department of Commerce.

[38] FIPS 197, (2001), *Announcing the Advanced Encryption Standard*. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[39] E. Fujisaki, T. Okamoto, (1999), *How to enhance the security of public-key encryption at minimum cost*, Workshop on Practice and Theory in Public Key Cryptography, PKC'99, LNCS, vol. 1560, pp. 53–68, Springer-Verlag.

[40] E. Fujisaki, T.Okamoto, D. Pointcheval, J. Stern, (2001), *RSA– OAEP is secure under the RSA assumption*, Advances in Cryptology — CRYPTO'01, LNCS, vol. 2139, Springer-Verlag.

[41] M.R. Garey, D.S. Johnson, (1979), *Computers and Intractability – A Guide to the Theory of NP-Completness*, Bell Telephone Laboratories.

[42] O. Goldreich, (2001), *Foundations of Cryptography*, Cambridge University Press, 392 pagini, ISBN 0521791723.

[43] O. Goldreich, (2004), *Foundations of Cryptography Volume II Basic Applications*, Cambridge University Press, 448 pagini, ISBN 0521830842.

[44] S. Goldwasser, S. Micali, (1984), *Probabilistic encryption*, Journal of Computer and System Sciences, vol. 28, pp. 270-299.

[45] S. Goldwasser, S. Micali, R. Rivest, (1988), *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2), pp. 281-308.

[46] B. Groza, (2007), *Broadcast authentication protocol with time synchronization and quadratic residues chains*, Second International Conference on Availability, Reliability and Security (ARES'07), International Symposium on Frontiers in Availability, Reliability and Security (FARES'07), Vienna, Austria, pp. 550-557, IEEE Comp. Soc..

[47] B. Groza, (2007), *An extension of the RSA trapdoor in a KEM/DEM Framework*, Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'07, Main Track, pp. 168-173, IEEE Comp. Soc..

[48] D. Hankerson, A. Menezes, S. Vanstone, (2003), *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 311 pagini, ISBN 038795273X.

[49] G.H. Hardy, E.M. Wright, (1975), *An Introduction to the Theory of Numbers*, Oxford University Press, 456 pagini, ISBN 0198531710.

[50] J. Hastad, (1988), *Solving simultaneous modular equations of low degree,* SIAM Jorunal of Computing, vol. 17, pp. 336-341.

[51] J. Herranz, D. Hofheinz, E. Kiltz, (2006), *The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure*, http://eprint.iacr.org/2006/207.pdf .

[52] IACR, (2007), *International Association for Cryptologic Research (IACR)*, http://www.iacr.org.

[53] David Kahn, (1996), The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet, Scribner, ISBN-10: 0684831309, ISBN-13: 978-0684831305.

[54] Jonathan Katz, Yehuda Lindell, (2007), Introduction to Modern Cryptography: Principles and Protocols, Chapman & Hall/CRC Cryptography and Network Security Series, ISBN-10: 1584885513, ISBN-13: 978-1584885511.

[55] D.E. Knuth, (1969), *The Art of Computer Programming, vol. 2 Seminumerical Algorithms*, Addison Wesley, 784 pagini, ISBN 0201896842.

[56] N. Koblitz, (1994), *A Course in Number Theory and Cryptography*, (Graduate Texts in Mathematics), Springer, 235 pagini, ISBN 0387942939.

[57] K. Kurosawa, Y. Desmedt, (2004), *A New Paradigm of Hybrid Encryption Scheme*, Advances in Cryptology - CRYPTO 2004, LNCS vol. 3152, pp. 426-442, Springer-Verlag.

[58] E. Landquist, (2002), *The Number Field Sieve Algorithm*, MATH 420: Computer Algebra Systems, http://www.math.uiuc.edu/~landquis/nfsieve.pdf .

[59] A.K. Lenstra, (2005), *Further progress in hashing cryptanalysis*, http://cm.bell-labs.com/who/akl/hash.pdf.

[60] W. Mao, (2003), *Modern Cryptography: Theory and Practice*, Prentice Hall PTR, 740 pagini, ISBN 0130669431.

[61] R. C. Merkle, (1987), *A digital signature based on a conventional encryption function*, Advances in Cryptology - CRYPTO 87, pp. 369-378, LNCS 293, Springer-Verlag.

[62] A.J. Menezes, P.C. Oorschot, S.A. Vanstone, (1996), *Handbook of Applied Cryptography*, CRC Press, 816 pagini, ISBN 0849385237.

[63] V. S. Miller, (1986), *Use of elliptic curves in cryptography*, Advances in Cryptology - CRYPTO'85, LNCS 218, Springer-Verlag.

[64] NIST, (2007), *Recommendation for Key Management — Part 1: general,* NIST Special Publication 800-57. March, 2007 http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf.

[65] M. Rabin, (1979), *Digitalized signatures and public key functions as intractable as factorization,* MIT/LCS/TR-212, MIT Laboratory for Computer Science.

[66] H. Riesel, (1994), *Prime Numbers and Computer Methods for Factorization*, Birkhäuser Boston; 2nd ed. edition, 494 pagini, ISBN 0817637435.

[67] R. Rivest, A. Shamir, L. Adleman, (1978), *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM.

[68] R. Rivest, (1992), *The MD5 Message-Digest Algorithm*, MIT Laboratory for Computer Science and RSA Data Security, RFC 1321.

[69] R. Rivest, A. Shamir, D.A. Wagner, (1996), *Time-lock puzzles and timed-release Crypto*, available at http://theory.lcs.mit.edu/~rivest/publications.html.

[70] RSA Laboratories, (2003), *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, RFC 3447, http://tools.ietf.org/html/rfc3447.

[71] RSA Laboratories, (2005), *RSA Factoring Challenge* http://www.rsasecurity.com/rsalabs/challenges/ factoring/numbers. http://www.rsa.com/rsalabs/node.asp?id=2093.

[72] W. F. Rush, J. A. Kinast, **(**2003), *Here's what you need to know to protect SCADA systems from cyber-attack*, Pipeline & Gas Journal.

[73] B. Schneier, (1996), APPLIED CRYPTOGRAPHY, John Wiley & Sons, 784 pagini, ISBN 0471117099.

[74] B. Schneier, (2004), *Cryptanalysis of MD5 and SHA: Time for a New Standard*, http://schneier.com/essay-074.html.

[75] B. Schneier, (2005), *Cryptanalysis of SHA-1*, http://www.schneier.com /blog /archives/2005/02/cryptanalysis_o.html

[76] K. Schmidt-Samoa, (2006), *A new rabin-type trapdoor permutation equivalent to factoring,* Electronic Notes in Theoretical Computer Science, 157(3).

[77] V. Shoup, (2004), *Computational Introduction to Number Theory and Algebra* availlable at www.shoup.net/ntb - 2004

[78] V. Shoup, (2001), *OAEP reconsidered*, Lecture Notes in Computer Science, 2139, Springer-Verlag.

[79] V. Shoup, (2001), *A proposal for an ISO standard for public key encryption*, Input for Committee.

[80] W. Stallings, (2005), *Cryptography and Network Security* (4th Edition), Prentice Hall, 592 pagini, ISBN 0131873164.

[81] D. R. Stinson, (2005), *Cryptography: Theory and Practice*, Third Edition Chapman & Hall/CRC, 616 pagini, ISBN 1584885084.

[82] Y. Tsiounis, M. Yung, (1998), *On the Security of ElGamal based Encryption*, Workshop on Practice and Theory in Public Key Cryptography, PKC'98, LNCS, vol. 1431, Springer-Verlag.

[83] Samuel S. Wagstaff, (2002), *Cryptanalysis of Number Theoretic Ciphers*, Chapman & Hall/CRC, 336 pagini, ISBN 1584881534.

[84] L. C. Washington, (2003), *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, 440 pagini, ISBN 1584883650.

[85] X. Wang, Y.L. Yin, H. Yu, (2005), *Collision search on SHA1*, http://theory.csail.mit.edu/~yiqun/shanote.pdf.

[86] D.R. Wilkins, (2001), *Topics in Number Theory*, Course 311: Michaelmas Term 2001, http://www.maths.tcd.ie/~dwilkins/Courses/311/311NumTh.pdf .

[87] Song Y. Yan, (2003), *Primality Testing and Integer Factorization in Public-Key Cryptography*, Springer, 256 pagini, ISBN 1402076495.

[88] U.S. Department of Energy, 21 Steps to Improve Cyber Security of SCADA Networks, 2002.

11 ANEXE

A - NOTAŢII COMUN UTILIZATE

*	- denotă o valoare oarecare
$ct.$	- constantă
$\lvert m \rvert$	- lungimea în biţi a unui mesaj (sau a altui obiect după caz)
$\phi(n)$	- Funcţia Euler Phi, *n* este un întreg oarecare.
$O(\cdot)$	- limita asimptotică superioară (complexitate).
Q_n	- mulţimea reziduurilor cvadratice (pătratelor perfecte) modulo *n*.
$\overline{Q}_n$	- mulţimea non-reziduurilor cvadratice modulo *n*.
Q_n	- mulţimea pseudo-reziduurilor cvadratice modulo *n*.
Z_n	- mulţimea resturilor modulo *n*.
Z_n^*	- mulţimea resturilor modulo *n* relativ prime la *n*.

$\left(\frac{x}{p}\right)$	- simbolul Legendre, *x* este un întreg oarecare şi *p* este un număr prim.
$\left(\frac{x}{n}\right)$	- simbolul Jacobi, *x* şi *n* sunt întregi oarecare.
$v(k)$	- funcţie (cantitate) neglijabilă.
$\bar{v}(k)$	- funcţie (cantitate) ne-neglijabilă.
c.m.m.d.c.	- cel mai mare divizor comun.
c.m.m.m.c.	- cel mai mic multiplu comun.
$A <_P B$	- algoritmul *A* se reduce în timp polinomial la *B*.
$A \Leftrightarrow B$	- $A <_P B$ şi $B <_P A$, deci algoritmul *A* este echivalent cu *B*.
CPA	- atac de tip mesaj ales.
CCA	- atac de tip criptotext ales.
CCA2	- atac de tip criptotext ales adaptiv.
DLP	- problema logaritmului discret.
PFI	- problema factorizării întregi.
DDH	- problema decizională Diffie-Hellman.
CDH	- problema computaţională Diffie-Hellman.
IND	- imposibilitatea de a distinge (Indistinguishability).
NM	- non-maleabilitate (Non-malleability).
PTP	- Probabilist în Timp Polinomial, Algoritm.
QRP	- Problema reziduurilor cvadratice.

B – CÂTEVA DEFINIŢII MAI RIGUROASE

> "Theorists view a one-way function as a basic object and build pseudo-random functions from them. But in practice, as indicated by Luby and Rackoff, the DES provides a pseudorandom function of 64 bits to 64 bits. Ironically, if one needs a practical protocol for a one-way function, likely one would construct it from DES thereby reducing the "simple" primitive to the "complex" one (...) Let us look at a second efficiently-computable primitive: the map defined by the MD5 algorithm (...) What really is this object? To date, there has been no satisfactory answer. That is, there is no formal definition which captures a large fraction of the nice properties this function seems to possess and it is not clear that one can be found."- M. Bellare & P. Rogway[13].

Relaţia între teorie şi practică nu este una uşoară. Am evitat încărcarea capitolelor anterioare cu definiţii riguroase ale funcţiilor criptografice deoarece cred că acestea şi-ar găsi mai bine locul în volumul 3. Totuşi cred că este util a defini în această anexă mod riguros, din punct de vedere matematic, câteva dintre noţiunile frecvent întâlnite în construcţia criptosistemelor.

Definiţiile cu privire la criptosisteme, şi în particular cu privire la securitatea acestora, fac adeseori apel la noţiunea de cantitate neglijabilă sau funcţie neglijabilă. De exemplu, putem spune că o funcţie criptografică hash este rezistentă la coliziune dacă probabilitatea de a găsi o coliziune este neglijabilă.

[13] Din lucrarea "Random oracles are practical: A paradigm to design efficient protocols".

Subliniem însă că în general se lucrează cu noţiunea de funcţie neglijabilă, valoarea neglijabilă depinzând astfel de un parametru. Iată definiţia unei funcţii neglijabile:

Definiţia A.1. (Funcţie neglijabilă) *O funcţie* $v: D_v \to C_v$ *se numeşte neglijabilă dacă pentru orice constantă* $c \geq 0$ *există un întreg* k_c *astfel încât* $v(k) < k^{-c}, \forall k \geq k_c$.

Într-o manieră similară definim noţiunea de funcţie ne-neglijabilă:

Definiţia A.2. (Funcţie ne-neglijabilă) *O funcţie* $\bar{v}: D_{\bar{v}} \to C_v$ *se numeşte ne-neglijabilă dacă există un polinom* $P(k)$ *astfel încât pentru valori suficient de mari ale lui* k *avem* $\bar{v}(k) > \frac{1}{P(k)}$.

Toate funcţiile criptografice sunt **funcţii one-way**, adică funcţii greu de inversat. O funcţie greu de inversat nu trebuie confundată cu o funcţie ireversibila. Din punct de vedere matematic o funcţie ireversibila este o funcţie care nu este bijectivă, în timp ce o funcţie greu de inversat este o funcţie a cărei inversă nu poate fi calculată în mod eficient. Această noţiune poate fi uşor formalizată folosind noţiunea de funcţie neglijabilă, presupunând că probabilitatea cu care un algoritm ar putea inversa această funcţie este neglijabilă. În general se merge însă mult mai departe de atât şi se impune imposibilitatea găsirii unei coliziuni în imaginea acestei funcţii, condiţie care este chiar mai puternică decât aceea de a inversa efectiv funcţia. Acest aspect este sintetizat în următoarea definiţie.

Definiţia A.3. (Funcţie one-way (puternică one-way)) *O funcţie* $f: D_f \to C_f$ *se numeşte one-way dacă:*

1) Există un algoritm PTP (Probabilist în Timp Polinomial) care calculează $y = f(x)$ *pentru aproape*[14] *orice valoare* $x \in D_f$.

2) Orice algoritm PTP care primeşte ca intrare pe $y = f(x)$ *returnează cu o probabilitate neglijabilă o valoare* z *astfel încât* $f(z) = y$, *adică pentru o*

[14] Noţiunea de „aproape orice valoare" este şi ea lipsită de rigurozitate, acum însă pentru rigurozitate cititorul poate uşor să pună în corespondenţă acest termen cu noţiunea de funcţie neglijabilă.

valoare k numită parametru de securitate avem $\Pr\left[x \xleftarrow{\Theta} D_f, y \leftarrow f(x), z \leftarrow \mathrm{A}(1^k, y) : f(z) = y\right] \leq v_{\mathrm{A}}(k)$ *(simbolul* $\xleftarrow{\Theta}$ *denotă că valoarea a fost aleasă aleator).*

Este natural să considerăm valoarea lui k ca fiind $k = \left\lfloor \log_2 \left| D_f \right| \right\rfloor$ deoarece în mod evident dimensiunea domeniului de definiţie este cea care face să scadă probabilitatea ca algoritmul PTP să returneze o valoare corectă.

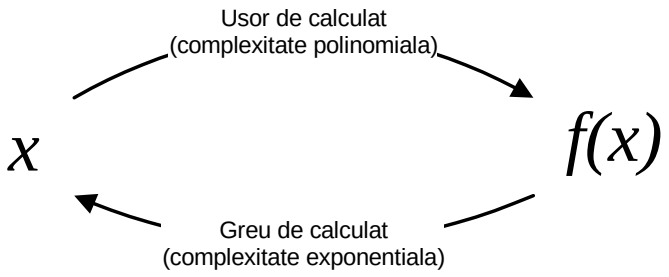


FIGURA 11.1. IMAGINE INTUITIVĂ ASUPRA UNEI FUNCŢII ONE-WAY.

Definiţia A.3 se referă la ceea ce se numeşte de fapt funcţie one-way puternică. Existenţa funcţiilor one-way puternice este condiţionată însă de existenţa funcţiilor one-way slabe şi invers. Pentru aceasta se impune să definim şi ce este o funcţie one-way slabă, definiţia este în cele din urmă evidentă.

Definiţia A.4. (Funcţie slabă one-way) *O funcţie* $f : D_f \to C_f$ *se numeşte one-way slaba dacă:*

1) Există un algoritm PTP care calculează $y = f(x)$ *pentru aproape orice valoare* $x \in D_f$ *.*

2) Orice algoritm PTP care primeşte ca intrare pe $y = f(x)$ *returnează cu o probabilitate ne-neglijabila o valoare* z *astfel încât* $f(z) \neq y$*, adică pentru o valoare k numită parametru de securitate avem* $\Pr\left[x \xleftarrow{\Theta} D_f, y \leftarrow f(x), z \leftarrow \mathrm{A}(1^k, y) : f(z) \neq y\right] \geq \bar{v}_A(k)$.

Se poate demonstra că existenţa funcţiilor one-way slabe şi cea a funcţiilor one-way puternice este echivalentă, pentru demonstraţii pot fi consultate lucrările [4, p. 20] şi [42, p. 20]. Altfel se poate observa imediat că o funcţie puternică one-way poate fi oricând transformată în funcţie slabă one-way (şi se observă de altfel că orice funcţie care respectă Definiţia A.4 respectă şi Definiţia A.3). În ceea ce priveşte reciproca, faptul că o funcţie slabă one-way poate fi transformată în funcţie puternică one-way, acest lucru de asemenea poate fi intuitiv înţeles deoarece dacă considerăm o funcţie slabă one-way f putem construi uşor funcţia $f'(\bar{x}) = f(x_0) \| f(x_1) \| \ldots \| f(x_{d-1})$, unde $\bar{x}$ denotă un vector de dimensiune d şi || denotă concatenare, care devine mai greu de inversat decât f deoarece probabilitatea de a inversa f' devine produsul probabilităţilor de a inversa f în toate punctele $x_i, i = \overline{0, d-1}$ (evident probabilitatea fiind o valoare subunitară, produsul probabilităţilor tinde exponenţial la 0 odată cu creşterea lui d).

Blocul principal de construcţie al sistemelor criptografice îl reprezintă funcţiile one-way cu trapă. Informal o funcţie one-way cu trapă este o funcţie one-way care poate fi inversată dacă se foloseşte o informaţie suplimentară numită trapă. Astfel definim o funcţie one-way cu trapă după cum urmează:

Definiţia A.5. (Funcţie trapdoor one-way) *O funcţie one-way cu trapă este o funcţie one-way pentru care există o informaţie numită trapă t şi un algoritm PTP pe care îl notăm cu Inv astfel încât $z \leftarrow Inv(t, y = f(x))$ şi $f(z) = f(x)$ pentru aproape orice valoare din domeniul de definiţie al funcţiei.*

În criptografie însă nu lucrăm cu o anume funcţie one-way, ci cu familii (colecţii) de funcţii one-way. Acestea se definesc după cum urmează, atât în cazul funcţiilor simple one-way cât şi al celor cu trapă:

Definiţia A.6. (Colecţii de funcţii one-way) *Fie $F = \{ f_i \mid f_i : D_{f_i} \to C_{f_i} \}$ unde $i \in I$ iar I este o mulţime de indici. Spunem despre F că este o colecţie de funcţii cu trapă one-way dacă:*

1) Există un algoritm PTP care primind ca intrare 1^k (unde k este parametru de securitate) returnează $i \in \{0,1\}^k \cap I$.

2) Există un algoritm PTP care la intrarea $i \in I$ returnează o valoare aleatoare $x \in D_{f_i}$ aleasă uniform din D_{f_i} (prin aleasă uniform înţelegem faptul că toate valorile din D_{f_i} au şanse egale de a fi selectate de algoritm).

3) $\forall i \in I$, $\forall x \in D_{f_i}$ *există un algoritm PTP care calculează* $f_i(x)$.

4) Pentru orice algoritm PTP probabilitatea de a găsi o valoare pentru care imaginea funcţiei este egală cu o valoare dată a imaginii este neglijabilă: $\Pr\left[i \xleftarrow{\Theta} I, x \xleftarrow{\Theta} D_{f_i}, y \leftarrow f_i(x), z \leftarrow \mathrm{A}\left(1^k, y\right) : f_i(z) = y \right] \leq v_{\mathrm{A}}(k)$.

La 1) am folosit notaţia 1^k, această notaţie se utilizează frecvent în criptografie şi însemnă prin convenţie reprezentarea numărului k în baza 1, care reprezintă k valori succesive de 1.

Definiţia A.7. (Colecţii de funcţii one-way cu trapă) *Fie* $F = \left\{ f_i \middle| \, f_i : D_{f_i} \to C_{f_i} \right\}$ *unde* $i \in I$ *iar* I *este o mulţime de indici. Spunem despre* F *că este o colecţie de funcţii one-way cu trapă dacă:*

1) Există un algoritm PTP care primind ca intrare 1^k *(unde* k *este parametru de securitate) returnează* $(i, t_i), i \in \{0,1\}^k \cap I, |t_i|_2 < p(k)$ *(ultima condiţie semnifică faptul că dimensiunea trapei are complexitate polinomială) unde* t_i *este trapa funcţiei* f_i.

2) Există un algoritm PTP care la intrarea $i \in I$ *returnează o valoare aleatoare* $x \in D_{f_i}$ *aleasă uniform din* D_{f_i}.

3) $\forall i \in I$, $\forall x \in D_{f_i}$ *există un algoritm PTP care calculează* $f_i(x)$.

4) $\forall i \in I$, $\forall x \in D_{f_i}$ *există un algoritm PTP, care îl notăm cu* Inv, *şi care având* t_i *şi* $y = f_i(x)$ *calculează* $z \leftarrow Inv(t_i, y)$ *şi avem* $f(z) = y$ *(condiţie necesară pentru corectitudinea inversării) şi suplimentar în cazul în care funcţia cu trapă este o permutare avem* $z = x$.

5) Pentru orice algoritm PTP, care nu primeşte trapa ca intrare, probabilitatea de a găsi o valoare pentru care imaginea funcţiei este egală cu o imagine dată este neglijabilă: $\Pr\left[i \xleftarrow{\Theta} I, x \xleftarrow{\Theta} D_{f_i}, y \leftarrow f_i(x), z \leftarrow \mathrm{A}\left(1^k, y\right) : f_i(z) = y \right] \leq v_{\mathrm{A}}(k)$.

Considerăm utilă, pe de o parte pentru clarificarea definiţiilor introduse anterior, dar şi pentru simplificarea expunerii ce va urma prezentarea unor funcţii cu trapă one-way definite peste grupuri de întregi. Importanţa acestor funcţii constă în faptul că toate aceste funcţii constituie baza unor criptosisteme cu cheie

publică. În cele ce urmează vom nota cu Z_n^* mulţimea numerelor din Z_n care sunt prime raportate la n adică $Z_n^* = \{x \in Z_n \mid cmmdc(x,n) = 1\}$.

Funcţia RSA stă la baza criptosistemului propus de Rivest, Shamir şi Adlemann, care este poate cel mai răspândit criptosistem cu cheie publică este şi totodată oferă prima soluţie completă pentru criptare asimetrică şi semnătură digitală [67]. Funcţia RSA este definită ca:

$$f_{RSA} : Z_n^* \to Z_n^*,\ f_{RSA}(x) = x^{\varepsilon} \bmod n$$

Aici, $n = pq$ este un întreg produs a exact două numere prime, ε reprezintă un întreg oarecare pentru care este adevărat că $c.m.m.d.c.(\varepsilon, \phi(n)) = 1$, unde $\phi(n) = (p-1)(q-1)$. Inversa acestei funcţii este:

$$f_{RSA}^{-1} : Z_n^* \to Z_n^*,\ f_{RSA}^{-1}(x) = x^{\delta} \bmod n, \delta \equiv \varepsilon^{-1} \bmod \phi(n)$$

De subliniat că funcţia RSA este o permutare one-way cu trapă, deci o funcţie bijectivă. Funcţia RSA se poate inversa dacă se cunoaşte factorizarea lui n. Nu se cunoaşte însă dacă inversarea acestei funcţii este echivalentă cu factorizarea lui n, deci dacă securitatea RSA este echivalentă cu problema factorizării întregilor. În principiu RSA Security promovează această idee prin concursurile de factorizare iniţiate, dar recent există lucrări care aduc destul de mult scepticism în această direcţie. Un lucru este însă foarte clar, în cei aproape 30 de ani de la introducerea acestei funcţii în criptografie nu s-a făcut nici un fel de progres în a demonstra echivalenţa securităţii RSA-ului cu problema factorizării dar în acelaşi timp singura cale de a sparge RSA-ul (exceptând cazuri particulare cu exponenţi de criptare, decriptare prost aleşi, utilizare necorespunzătoare etc.) este factorizarea întregului folosit ca modul. O generalizare a funcţiei RSA pentru construcţia criptosistemelor poate fi găsită în [47].

Funcţia Rabin este din punct de vedere cronologic al doilea candidat de funcţie one-way cu trapă folosită în criptosisteme asimetrice [65]. Funcţia este definită ca:

$$f_{Rabin} : Z_n^* \to Q_n,\ f_{Rabin}(x) = x^2 \bmod n$$

Din nou $n = pq$ este un întreg produs a exact două numere prime. Prin Q_n se notează mulţimea pătratelor perfecte din Z_n^*, i.e. $Q_n = \left\{ x \in Z_n^* \middle| \exists y \in Z_n^*, x \equiv y^2 \bmod n \right\}$. De subliniat faptul că funcţia Rabin nu este un caz particular al funcţiei RSA deoarece funcţia RSA impune ca $c.m.m.d.c.(\varepsilon, \phi(n)) = 1$ în timp ce în cazul funcţiei Rabin exponentul 2 nu poate îndeplini aceasta condiţie, funcţia lui Euler, i.e. $\phi(n)$, fiind întotdeauna pară. Inversa funcţiei Rabin nu poate fi sintetizată într-o formulă, dar există algoritmi pentru calculul acestei inverse dacă şi numai dacă factorizarea lui n este cunoscută (în capitolul 4 se prezintă câteva metode pentru aceasta). Se subliniază de asemenea faptul că funcţia Rabin nu este bijectivă şi descrie în Z_n^* o transformare de tip 4 la 1. Funcţia Rabin poate fi inversată dacă şi numai dacă se cunoaşte factorizarea lui n - deci are securitatea echivalentă cu problema factorizării întregilor.

Impunând o restricţie asupra modulului compozit n, funcţia Rabin poate fi transformată într-o permutare cunoscută sub numele de **funcţia Rabin-Williams.**. Astfel în cazul în care n este un întreg Blum, adică produsul a două numere prime p, q cu $p \equiv 3 \bmod 4, q \equiv 3 \bmod 4$ funcţia Rabin definită pe Q_n devine o permutare one-way cu trapă.

$$f_{Rabin-Williams} : Q_n \to Q_n,\ f_{Rabin-Williams}(x) = x^2 \bmod n$$

În acest caz funcţia are şi o inversa care poate fi definită după cum urmează:

$$f^{-1}{}_{Rabin-Williams} : Q_n \to Q_n,\ f^{-1}{}_{Rabin-Williams}(x) = x^{\frac{n-p-q+5}{8}} \bmod n$$

Funcţia Rabin-Williams are securitatea echivalentă cu dificultatea problemei factorizării întregilor.

Funcţia Schmidt-Samoa este una dintre cele mai recente propuneri de funcţie one-way trapdoor a cărei securitate este echivalentă cu problema factorizării [76]. Funcţia lucrează cu un modul compozit de forma $n = p^2 q$ şi este definită după cum urmează:

$$f_{Samoa} : Z^*_{pq} \to N - R(n),\ N - R(n) = \left\{ x \in Z^*_n \middle| \exists y \in Z^*_n, x = y^n \bmod n \right\}$$

$$f_{Samoa}(x) = x^n \bmod n$$

Funcţia are ca inversă:

$$f^{-1}{}_{Samoa}(x) = x^d \bmod pq,\ d = n^{-1} \bmod \phi(pq)$$

Funcţia este bijectivă, adică este o permutare one-way cu trapă. Alte detalii pot fi găsite în lucrarea [76].