SECURITATEA CRIPTOGRAFICĂ

Suport curs

Emil SIMION

2011-2012

PREFAŢĂ

In God we trust all other we monitor.
NSA (*No Such Agency*)

Constituită ca ramură interdisciplinară, *Criptologia,* cu cele două componente ale sale, *Criptografia* şi *Criptanaliza* apar tot mai frecvent, ca o necesitate, în curricula de învăţământ ca urmare a dezvoltării exponenţiale a tehnologiilor informaţiei şi comunicaţiilor.

Un sistem criptografic este compus din *algoritm de cifrare* (simetric sau asimetric), *sistem de generare şi management al cheilor de cifrare* (secrete sau publice/private), precum şi *protocoale criptografice* necesare *identificării* (emitentului informaţiei), *autentificării* (informaţiei sau utilizatorului), *transferului* (informaţiei) în condiţii de siguranţă (*protecţie la erori* şi *nerepudierea* informaţiei) şi asigurării *confidenţialităţii.*

Această carte, aflată la ediţia a doua, abordează într-o manieră unitară latura ofensivă a criptologiei şi anume criptanaliza.

Lucrarea prezintă tehnici şi metode utilizate în evaluarea şi spargerea *sistemelor de criptografice.* Astfel, se abordează tehnici de analiză criptografică pentru sistemele de cifrare clasice (substituţii şi transpoziţii), cifrurile cu cheie simetrică (cifrurile flux şi cifrurile bloc), cifrurile cu cheie publică, protocoalele criptografice, precum şi sistemele de cifrare analogică.

Unele paragrafe din lucrare au la bază rezultate obţinute, de autor, din care o parte au fost publicate în reviste de specialitate sau comunicate cu prilejul expunerilor făcute la diferite conferinţe. De asemenea, mare parte din această lucrare a fost subiectul orelor de curs şi seminar ale disciplinelor de criptologie, în cadrul programelor masterale de profil ale Universităţii din Bucureşti, Academiei Tehnice Militare, precum şi Universităţii Politehnica din Bucureşti. Se corectează astfel erorile inerente apărute în ediţia precedentă din anul 2004, fiind actualizate tehnicile criptografice prezentate. Mulţumim pe această cale tuturor celor care în decursul timpului au semnalat astfel de erori.

Fiecare capitol are propuse o serie de aplicaţii, unele dintre acestea constituindu-se ca direcţii ulterioare de studiu.

Cartea se adresează specialiştilor din domeniul analizei criptografice sau a securităţii sistemelor I.T., studenţilor şi masteranzilor de la facultăţile cu profil matematică-informatică, automatică-calculatoare, precum şi tuturor celor interesaţi de domeniul securităţii informaţiilor.

25 noiembrie 2011 Autorul

Cuprins

Capitolul 1

INTRODUCERE

Acestă lucrare abordează o ramură delicată a *Criptologiei* şi anume latura ofensivă a acesteia: *Criptanaliza.* Iniţial, domeniul criptanalizei a fost o preocupare exclusiv guvernamentală dar odată cu dezvoltarea *Tehnologiei Informaţiei* acest domeniu a căpătat şi un aspect comercial. Primele studii criptanalitice moderne au fost realizate în perioada celei de a doua conflagraţii mondiale şi vom menţiona pe următorii cercetători: *William F. Friedman*, *Claude E. Shannon* (SUA), *Hans Rohrbach* (Germania), *Alan Mathison Turing* (Marea Britanie), *Marian Rejewski* (Polonia), *Arne Beurling* (Suedia) şi *Ernst S. Selmer* (Norvegia).

Domeniul criptanalizei este un domeniu interdisciplinar din care se disting: teoria informaţiei, teoria complexiăţii algoritmilor, metodele numerice, statistica matematică, teoria probabilităţilor, teoria codurilor şi nu în ultimul rând, teoria numerelor.

Lucrarea este abordată din perspectiva *evaluatorului* de cifru şi nu a *crackerului* (trebuie făcută distincţia faţă de termenul *hacker,* care realizează acţiuni distructive).

Materialul este structurat în 14 capitole, după cum urmează:

Capitolul 1 realizează o prezentare a întregului material, indicând totodată şi o diagramă de parcurgere a acestuia.

În Capitolul 2 se prezintă *noţiunile generale,* accentul punându-se pe obiectul criptanalizei şi pe operaţiile ce se efectuează în criptanaliză, finalizându-se cu clasificări ale atacurilor criptanalitice. Sunt prezentate, de asemenea, o serie de standarde internaţionale cu implicaţii în evaluarea securităţii sistemelor IT şi a modulelor criptografice.

Capitolul 3 prezintă *teoria complexităţii algoritmilor* mai exact se vor prezenta elemente de teoria maşinilor Turing şi exemple concrete de probleme $NP-$complete.

Analiza statistico-informaţională, prezentată în Capitolul 4, este prima operaţiune ce se execută în criptanaliză (teste statistice de aleatorism şi de complexitate

algoritmică) şi este, de departe, cel mai vast capitol, datorită elementelor practice constitutive. Practic, cu ajutorul tehnicilor prezentate, se pot sparge sistemele criptografice (algoritmi şi protocoale).

Operaţiunea de codificare este realizată după cea de cifrare, pentru a realiza o compatibilitate a ieşirii algoritmului de cifrare cu canalul de comunicaţie. În capitolul 5 sunt prezentate tehnicile de codificare cele mai uzuale, precum şi principalele rezultate referitoare la acestea.

Evident, o carte despre criptanaliză trebuie să înceapă cu *elemente de criptanaliză clasică*: cifruri de permutare şi cifruri de transpoziţie. Capitolul 6 aduce cititorului o nouă viziune a abordării unor astfel de sisteme de cifrare: orice cifru simetric este o combinaţie dintre o transpoziţie şi substituţie polialfabetică.

Capitolul 7 abordează problematica cifrurilor flux, prezentându-se metodele de criptanaliză liniară, de corelaţie, corelaţie rapidă, Siegenthaler, sindrom liniar, corecţia iterativă a erorilor, etc. Capitolul se încheie cu prezentarea unor tehnici de proiectare, precum şi cu exemplu practic de atac criptanalitic.

În mod natural, abordarea continuă cu *criptanaliza cifrurilor bloc,* prezentată în Capitolul 8, în care se face o expunere a principalelor moduri de lucru ale acestora, accentul punându-se pe studiile comparative ale eficienţei acestora. Este realizată o introducere în criptanaliza diferenţială şi criptanaliza liniară, fiind date şi o serie de exemple de implementări şi rezultate experimentale referitoare la standardul de cifrare AES (*Advanced Encryption Standard*) care este succesorul standardului DES (*Data Encryption Standard*).

Capitolul 9 este dedicat introducerii în *criptografia asimetrică* (chei publice) în care securitatea sistemului se bazează pe dificultatea computaţională a unor probleme din teoria numerelor, ca de exemplu: factorizarea numerelor mari, logaritmul discret, extragerea rădăcinii pătrate modulo în număr compozit, etc. Sunt prezentate metodele de cifrare de tip rucsac, RSA, Pohling-Hellman, Rabin, ElGamal, sistemele de cifrare bazate pe curbe eliptice, precum şi tehnicile şi metodele de factorizare a numerelor mari. Se abordează şi problematica PKI (infrastructura cu chei publice).

Semnătura digitală a apărut ca un element de necesitate: realizarea unei funcţii care să aibă aceeaşi valoare ca semnătura olografă. În capitolul 10 sunt prezentate tehnici de realizare a semnăturilor digitale cu ajutorul criptografiei simetrice şi asimetrice.

Un loc aparte în cadrul acestei prezentări îl deţine Capitolul 11, dedicat *criptanalizei protocoalelor criptografice*: protocoale elementare de autentificare şi schimb de chei, protocoale avansate de tip demonstraţii convingătoare fără detalii, cunoaştere zero, dezvăluire minimă, transfer de bit, etc.

În Capitolul 12, se prezintă o introducere în analiza sistemelor de cifrare ana-

logică, abordându-se tematici de calcul operaţional (transformatele Laplace, Fourier, Walsh şi z), conversie semnal analogic în semnal digital şi cifrare analogică (inversarea, rotirea, amestecarea şi multiplexarea spectrului).

Una dintre părţile critice ale unui sistem criptografic este aceea a managementului cheilor criptografice. În Capitolul 13, sunt abordate probleme de management, generare şi protecţie al cheilor criptografice.

Majoritatea capitolelor abordate conţin numeroase exemple practice, precum şi exerciţii rezolvate sau propuse spre rezolvare.

Datorită caracterului interdisciplinar, am considerat utilă prezentarea unor anexe din tehnicile de programare, teoria probabilităţilor, statistică matematică, simulare şi teoria algebrică a numerelor. Problemele puse în cadrul acestor anexe pot constitui cursuri de suport pentru o dezvoltare aprofundată a domeniului criptografic.

Lucrarea se încheie cu o bibliografie în care au fost incluse o serie de referinţe clasice din domeniul criptanalizei. În figura 1.1 se prezintă o abordare optimă a subiectelor tratate în această carte.

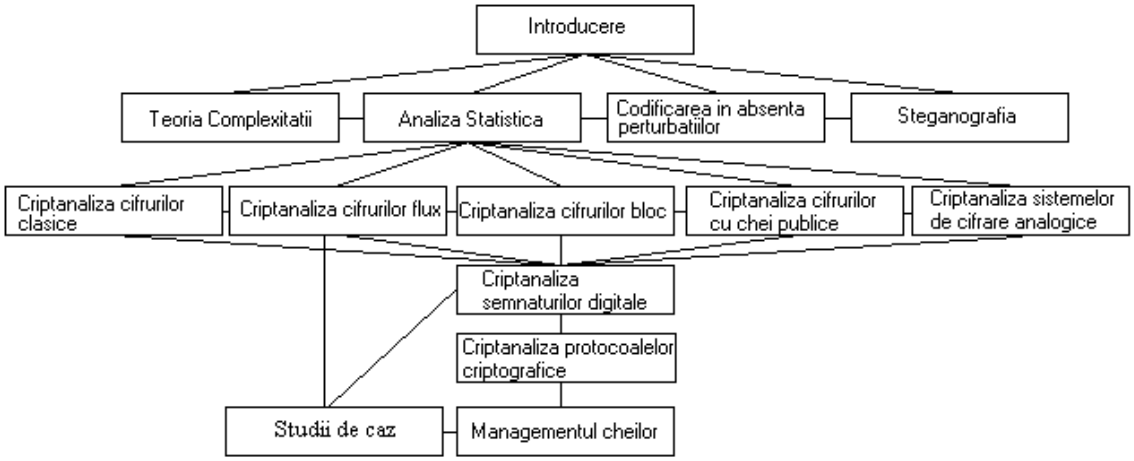

Figura 1.1: Diagrama de parcurgere a cărţii.

S-a avut în vedere elaborarea unei culegerei ce va conţine exemple, exerciţii şi studii de caz care să dubleze materialul de faţă.

Cartea este utilă proiectanţilor de sisteme criptografice, studenţilor de la facultăţile cu profil IT, masteranzilor şi doctoranzilor din domeniul securităţii criptografice, precum şi tuturor celor interesaţi de domeniul securităţii în formă electronică a informaţiilor.

Capitolul 2

NOŢIUNI GENERALE

ars ipsi secreta magistro
Jean Robert du Carlet,1644

2.1. Obiectul criptanalizei

Considerăm util să definim unele dintre principalele noţiuni cu care vom opera în cele ce urmează.

CRIPTOLOGIA este ştiinţa scrierilor secrete, având drept obiect apărarea secretului datelor şi informaţiilor confidenţiale, cu ajutorul sistemelor criptografice.

CRIPTOGRAFIA este latura defensivă a *CRIPTOLOGIEI*, având drept obiect de activitate elaborarea (conceperea) sistemelor criptografice şi a regulilor folosite. Cei care au această specialitate sunt denumiţi criptografi.

CRIPTANALIZA este latura ofensivă a *CRIPTOLOGIEI*, având drept obiect de activitate studierea sistemelor criptografice proprii pentru a le oferi caracteristicile necesare, astfel încât acestea să-şi îndeplinească funcţia pentru care au fost concepute. Totodată criptanaliza poate analiza sistemele criptografice ale terţelor părţi-prin intermediul criptogramelor realizate cu ele- astfel încât spargerea acestora să obţină informaţii utile instituţiei pe care o deserveşte. Specialiştii din acest domeniu se numesc *criptanalişti* sau, folosind un termen mai romantic, *spărgătorii de coduri.*

Prin *algoritm criptografic* înţelegem o mulţime de transformări uniinversabile prin care mulţimea mesajelor (textelor) clare dintr-o limbă se transformă în mulţimea $\mathcal{M}$ a criptogramelor.

Cheia de cifrare constituie o convenţie particulară, materializată, printr-un cuvânt, frază, număr, şir numeric etc. şi care dirijează (reglementează) *operaţia de cifrare.*

Un *protocol criptografic* este un set de reguli, între doi sau mai mulţi parteneri, prin care are loc o operaţie de autentificare şi/sau transfer de cheie sau mesaje.

Un *sistem criptografic* este compus din trei elemente: algoritm de cifrare, sistem de generare al cheilor şi protocol de distribuţie al cheilor de cifrare.

Supracifrarea constă dintr-o mulţime de transformări aplicate asupra criptogramelor şi are rolul să întărească rezistenţa criptogramelor (deci a sistemului criptografic) faţă de atacul criptanaliştilor terţelor părţi.

Descifrarea este operaţia inversă cifrării şi ea constă în aplicarea sistemului de cifrare cunoscut (în prezenţa cheii corecte) asupra criptogramelor pentru aflarea mesajului clar.

Decriptarea este operaţia prin care, numai pe baza analizei criptogramelor realizate cu un sistem de cifru necunoscut, se pune în evidenţă mesajul clar care a fost criptografiat şi se determină caracteristicile sistemului criptografic folosit pentru cifrare.

Sistemele criptografice (cifruri, coduri sau combinaţii ale acestora) se aplică unor mesaje clare redactate într-o limbă oarecare, mesaje care au anumite caracteristici structurale şi statisice, proprii limbii respective. Prin aplicarea sistemelor criptografice aceste caracteristici sunt perturbate, intensitatea şi direcţia acestor perturbaţii regăsindu-se în criptogramă. Un sistem criptografic este cu atât mai bun cu cât intensitatea acestor perturbaţii este mai mare, astfel încât criptograma să nu mai reflecte caracteristicile structurale ale mesajelor clare, deci ale limbii în care au fost ele redactate. Diverse tehnici şi metode permit ca anumite tipuri de sisteme de cifrare să fie invariante la anumiţi parametrii. Aceşti invarianţi constituie elemente de bază în criptanaliză. Cititorul poate consulta pentru o introducere în domeniul criptologiei *Tilborg* [89] şi *Schneier* [69].

2.2. Criterii şi standarde

2.2.1. Beneficii ale standardelor

Standardele, indiferent de categoria lor, sunt importante prin faptul că definesc practici, metode şi măsuri ce urmează a fi urmate de către dezvoltator, integrator şi beneficiar. Din acest motiv standardele cresc fiabilitatea şi eficienţa produselor, iar acest lucru duce la o creştere a gradului de calitate. Standardele oferă soluţii care sunt acceptate de comunitatea căreia i se adresează şi sunt evaluate de către experţi în domeniul respectiv. Prin folosirea standardelor, organizaţiile pot reduce costurile şi protejează investiţiile lor în echipamente tehnologice.

Standardele oferă pentru domeniul IT interoperabilitate, securitate şi integritate.

Interoperabilitatea. Produsele elaborate de un standard specific pot fi folosite cu alte produse elaborate după aceeaşi categorie de standarde. Spre exemplu, prin folosirea unui algoritm de cifrare, datele au fost cifrate prin intermediul unui echipament produs de firma A şi pot fi descifrate cu ajutorul unui echipament produs de către firma B. Prin asigurarea interoperabilităţii dintre diferitele echipamente, standardele permit unei organizaţii sau firme să opteze pentru un anume produs pentru a minimiza cheltuielile.

Securitatea. Standardele pot fi utilizate pentru a atinge un anume nivel comun de securitate. Majoritatea directorilor de firme nu sunt experţi în probleme de securitate, dar prin folosirea unui algoritm standardizat (de exemplu standarde FIPS) au certitudinea că acesta a fost analizat de către organizaţia de standardizare (în acest caz NIST).

Integritatea. Standardele pot fi folosite pentru a asigura integritatea unui anume produs.

Nivel de referinţă. Un standard poate fi un nivel de referinţă în evaluarea produselor elaborate de catre o firmă A (de exemplu standardul *FIPS PUB 140-2* este un standard pentru cerinţele de securitate a modulelor criptografice).

Minimizarea costurilor. Un standard poate duce la minimizarea cheltuielilor prin adoptarea unei soluţii deja existente. Fără aceste standarde, de exemplu, utilizatorii de sisteme informatice vor trebui să devină experţi în orice produs IT.

2.2.2. Organisme de standardizare

Principalele organisme de standardizare din SUA sunt:

- *National Institute of Standards and Technologies* (NIST) este o organizaţie care elaborează standarde FIPS pentru guvernul federal al SUA,

- *American National Standards Institute* (ANSI) este un administartor al standardelor din sectorul privat (orice standard FIPS are un echivalent în standardele ANSI),

- *Institute of Electrical and Electronical Engineering* (IEEE) are ca obiectiv principal elaborarea de teorii şi tehnici avansate din domeniile electric, electronic, ştiinţa calculatoarelor şi informatică,

- *Internet Engineering Task Force* (IETF), este o comunitate internaţională de proiectanţi de reţele, operatori, comercianţi de servicii şi cercetători care se ocupă în principal de evoluţia arhitecturii Internetului. Această comunitate acţionează în grupuri de lucru care sunt organizate după domeniile de activitate specifice ca de exemplu: rutare, transport, securitate, etc,

- *IETF Public-Key Infrastructure (X.509)* (PKIX) *Working Group* se ocupă de standardizarea protocoalelor folosite de reţeaua Internet care au la bază sistemele cu

chei publice. X509 reprezintă un standard acceptat pe scară largă pentru infrastructura ce defineşte formatul datelor şi a procedurilor legate de distribuţia cheilor publice prin certificatele digitale semnate de autorităţile de certificare (CA),

- *International Organization for Standardization* (ISO), este un organism negu-vernamental la care au aderat peste 100 de state şi are ca scop principal promovarea dezvoltării standardizării pentru a facilita schimbul internaţional de bunuri şi servicii, precum şi cooperarea în domeniile intelectual, stiinţific, tehnologic şi economic. ISO lucrează în comitete tehnice de lucru, subcomitete şi grupuri de lucru.

2.2.3. Standardele ISO 15408 şi FIPS 140-2

Criterii Comune [106] (cunoscute şi ca standardul *ISO 15408*), notate în cele ce urmează prin CC, sunt rezultatul încercărilor de evaluare a securităţii IT şi se folosesc în comunitatea internaţională drept sistem de referinţă. Criteriile Comune definesc un set de reguli şi cerinţe din domeniul IT pentru a valida securitatea produselor şi sistemelor IT. Criteriile comune definesc modul de construcţie al profilului protecţiei (PP) care permite eventualilor beneficiari sau producători, creeare de seturi de standarde de cerinţe de securitate care să îndeplinească nevoile acestora. CC este un standard voluntar care este folosit pentru a descrie proprietăţile securităţii (funcţionale şi de asigurare) ale produselor şi sistemelor IT şi sunt organizate pe 7 niveluri de încredere (cunoscute sub acronimul de *EAL*).

Manualele *FIPS PUB* sunt linii directoare care trebuiesc urmate. De exemplu *FIPS PUB 46-3, Data Encryption Standard,* este un set specific de cerinţe tehnice de securitate pentru algoritmul DES. După cum am menţionat mai sus *FIPS 104-2* este un standard de cerinţe pentru modulul criptografic care este organizat pe 4 niveluri.

Când dezvoltăm o specificaţie sau criteriu pentru selectarea unui modul sau produs criptografic trebuie să verificăm atât cerinţele CC cât şi ale FIPS-urilor.

Produsele proiectate cu proprietăţi specificate de CC sau FIPS 140-2 trebuie testate (validate) pentru confirmarea acestor criterii. Validarea se face de un laborator de testare specializat şi acreditat de către organismele abilitate şi care confirmă faptul că produsele testate îndeplinesc specificaţiile de securitate respective.

2.3. Modelul OSI (Open System Interconectation)

2.3.1. Definirea nivelurilor reţelei

Considerăm util prezentarea modului de interconectare a sistemelor actuale de comunicaţii (vezi *Jamsa [34]*) care se poate face conform modelului OSI (*Open Sys-*

tem Interconectation), model ce defineşte cadrul de implementare a protocoalelor de comunicaţii pe şapte niveluri şi anume:

-*Nivelul fizic* în care are loc conversia biţilor în impulsuri electrice, luminoase sau semnale radio;

-*Nivelul legătura de date* realizează împachetarea datelor sub forma de biţi;

-*Nivelul reţea* implementează circuite logice de transmisie a datelor;

-*Nivelul transport* asigură controlul fluxului de date între sisteme de comunicaţii;

-*Nivelul sesiune* realizează conexiuni între aplicaţii;

-*Nivelul prezentare* asigură independenţa faţă de diferenţele de prezentare ale datelor şi sintaxei;

-*Nivelul aplicaţie* în care are loc identificarea partenerilor de comunicaţie;

Cele cinci mari principii, introduse de *Andrew Tanenbaum* în anul 1981, utilizate la realizarea nivelurilor în modelul de referinţă OSI, prezentate mai sus, sunt:

1. Se creează un nou nivel al reţelei ori de câte ori software-ul de reţea necesită un nivel diferit de abstractizare;

2. Fiecare nivel trebuie să îndeplinească o funcţie bine definită;

3. Funcţia fiecărui nivel se alege ţinând cont de definirea protocoalelor standardizate la nivel internaţional;

4. Limitele nivelurilor se aleg astfel încât să minimizeze fluxul de informaţii pe interfeţe;

5. Numărul de niveluri trebuie să fie suficient de mare astfel încât proiectanţii să nu fie obligaţi să implementeze funcţii diferite pe acelaşi nivel. Totuşi, numărul de niveluri nu trebuie să fie atât de mare încât acestea să devină greu de stăpânit.

2.3.2. Nivelul fizic

Nivelul fizic transmite datele prin intermediul canalelor de comunicaţie din reţea. Nivelul fizic include elementele fizice (hardware) necesare pentru îndeplinirea acestei funcţii. Ca atare, liniile de transmisie din reţea-cablurile care conectează toate calculatoarele din reţea -fac parte din nivelul fizic. Metodele de transmisie a datelor, inclusiv semnalele de control şi temporizările, sunt de asemnea componente ale nivelului fizic. Nivelul fizic include de asemenea şi tehnologiile de reţea (Ethernet, ARCNET şi token ring) care definesc parametrii pentru transmisiile de date.

2.3.3. Nivelul legătură date

Nivelul legătură date (sau nivelul legătură) transferă datele brute între nivelul fizic şi nivelul reţea. Placa de reţea reprezintă nivelul legătură de date în calculator. Funcţia principală a nivelului legătură date este de a preveni alterarea datelor în cadrul nivelului fizic.

2.3.4. Nivelul reţea

Nivelul reţea determină ruta sau calea pe care o vor urma datele pentru a ajunge la destinaţie. Ca atare, nivelul reţea trebuie să gestioneze traficul în reţea, congestiile şi ratele de transfer (vitezele) de-a lungul liniilor de transmisie. Nivelul reţea trebuie de asemenea să gestioneze alterările datelor în cadrul canalului de comunicaţie. Nivelul reţea poate fi considerat un sistem de livrare în cadrul reţelei. IP(*Internet Protocol*) este sinonim cu nivelul reţea.

2.3.5. Nivelul transport

Nivelul transport livrează datele în cadrul unui calculator host. Cu alte cuvinte, după ce nivelul reţea livrează datele la adresa host corectă, nivelul transport livrează datele la aplicaţia corespunzătoare din cadrul hostului destinaţie.

2.3.6. Nivelul sesiune

Ca interfaţă a utilizatorului cu reţeaua, nivelul sesiune negociază conexiunea între procese sau aplicaţii de pe calculatoare host diferite. Ca atare, nivelul sesiune gestionează detalii ca nume de cont, parole şi autorizarea utilizatorilor. De exemplu, în multe reţele, trebuie realizată operaţia de login înainte de a folosi serviciile de reţea. Fiecare operaţie de login deschide o sesiune.

2.3.7. Nivelul prezentare

Nivelul prezentare reuneşte funcţii folosite în mod repetat în timpul comunicaţiilor în reţea. Nivelul prezentare gestionează detaliile legate de interfaţa reţeli cu imprimantele, cu monitoarele şi formatele de fişiere. Pe scurt, nivelul prezentare defineşte felul cum se prezintă reţeaua pentru hardwarele şi softwarele instalat. Pentru a înţelege motivele pentru care a fost creat nivelul prezentare, ne putem raporta la cele cinci principii majore utilizate la proiectarea modelului OSI. De exemplu, chiar dacă fiecare aplicaţie de reţea poate executa independent toate funcţiile nivelului prezentare, putem de asemenea defini toate funcţiile acestui nivel fără referinţă la o aplicaţie sau la un alt nivel de reţea (principiul 2). Totuşi, pentru aceasta, trebuie să creeăm un nivel diferit de abstractizare şi, astfel, un nou nivel (principiul 1). Prin localizarea tuturor informaţiilor legate de imprimante, monitoare şi formate de fişiere în acelaşi nivel, minimizăm fluxul de informaţie relativ la acestă interfaţă prin frontierele altor nivele (principiul 4).

2.3.8. Nivelul aplicaţie

Nivelul aplicaţie conţine detalii despre aplicaţiile din întreaga reţea. Ca programator de aplicaţii sau creator de soft, deja suntem familiari cu o mare parte din funcţionarea nivelului aplicaţie. Exemplele de aplicaţii de reţea includ poşta electronică (e-mail) şi bazele de date distribuite. Aplicaţiile dezvoltate pentru utilizarea în Internet vor fi o componentă a nivelului aplicaţie. De fapt, toate programele pentru utilizatorii calculatoarelor din reţea (utilizatorii terminalelor de reţea) fac parte din nivelul aplicaţie.

2.3.9. Protocolul TCP/IP

Cel mai răspândit model de interconectare al sistemelor IT este *TCP/IP* (*Transport Control Protocol/Internet Protocol*) care s-a dezvoltat în mediul universitar şi actualmente este *standardul de comunicaţie pe Internet*. TCP/IP are o segmentare pe patru niveluri. Cele patru niveluri includ serviciile descrie de OSI astfel:

-*Nivelul aplicaţie*: grupează funcţiile OSI din Aplicaţie, Prezentare şi Sesiune (nivelurile cinci, şase şi şapte);

-*Nivelul transport* este similar cu nivelul patru din OSI şi descrie două protocoale: UDP (*User Datagram Protocol*) şi TCP.

-*Nivelul reţea*, denumit şi INTERNET, asigură funcţiile OSI corespunzătoare nivelului trei şi reprezintă baza arhitecturii TCP/IP.

-*Nivelul de acces la reţea* grupează serviciile OSI de pe nivelurile unu şi doi, utilizând standardele existente precum: Ethernet, Token Ring, FDDI, HSSI sau ATM (*Asynchronous Transfer Mode*).

Elementele de criptologie pot interveni pe oricare dintre nivele de interconectare pentru asigurarea serviciilor de confidenţialitate, integritate, autentificare, nerepudiere, disponibilitate a informaţiei precum şi a controlului accesului la resursele sistemului.

2.4. Testarea sistemelor criptografice

2.4.1. Introducere

Serviciile criptografice se implementează prin folosirea modulelor criptografice (criptomodule) care au anumite funcţiuni ca de exemplu: generarea şi verificarea semnăturilor digitale (poate include şi servicii de notariat digital), cifrare şi descifrare, generarea cheilor, distribuţia cheilor, etc. Pentru a atinge un nivel de securitate al serviciilor în sistem trebuie folosite de cele mai multe ori un număr mare de module criptografice, deci o eroare nedetectată în unul din modulele componente

ale sistemului poate afecta funcţiile criptografice ale fiecărui utilizator al sistemului. Figura 2.1 prezintă un model ierarhic de testare a securităţii, model ce include module criptografice. Acest model precum si organismele de testare aplicate sunt descrise în cadrul acestui capitol.

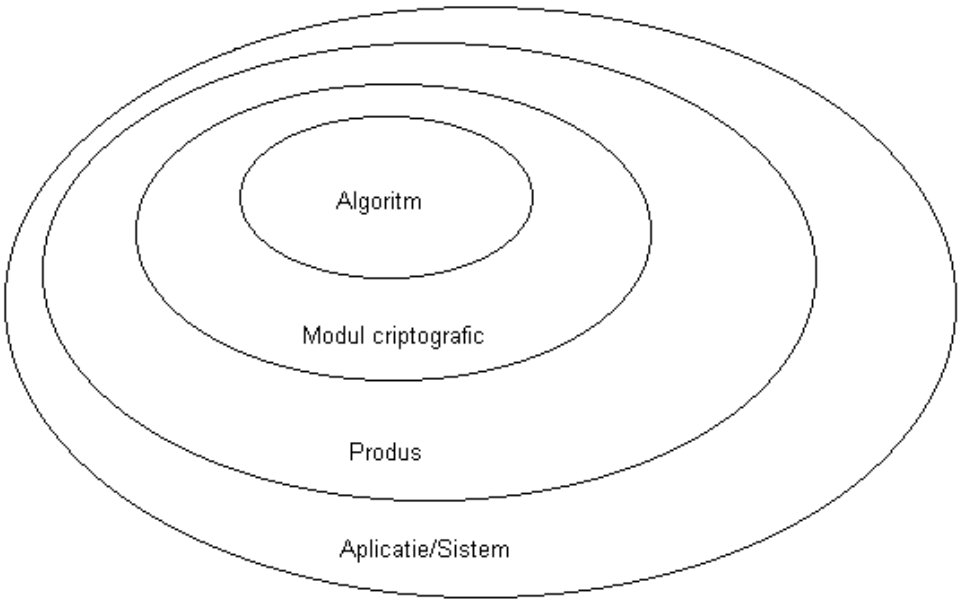

Figura 2.1: Model de testare.

Nivelurile minime sunt algoritmul criptografic şi modulul criptografic. Aceste produse trebuie testate a priori dacă se pot integra într-un sistem existent sau un sistem nou creeat. Modulele criptografice sunt testate de către dezvoltatorul aplicaţiei şi apoi trimise la *Programul de Validare al Modulelor Criptografice* (cunoscut sub acronimul *CMVP*) pentru a fi testate sub incidenţa FIPS 140-2 (Cerinţe de Securitate pentru Modulele Criptografice).

2.4.2. Programul de validare a modulelor criptografice

Prezentare generală

NIST şi *CSE* (Communication Security Establishment) din Canada au stabilit de comun acord programul CMVP. Acesta are ca rol principal furnizarea serviciilor şi agenţiilor federale o metrică a securităţii pentru a fi folosită în procurarea

echipamentelor care conţin module criptografice. Metrica este constituită cu ajutorul testării independente de către laboratoare acreditate. Validarea modului criptografic se face cu ajutorul metodologiei *DRT* (*Derived Test Requirements*) pentru FIPS PUB 140-2. DRT-ul este o listă a tuturor cerinţelor necesare pentru validarea unui modul criptografic şi sunt baza testelor efectuate de laboratoarele (acreditate) de testare a modulului criptografic (*CMT*).

Un *modul criptografic* este o combinaţie a unor procese software şi hardware specializate. Principalele avantajele ale folosirii modulelor validate (acreditate) sunt următoarele:

-asigurarea faptului că modulele au încorporate cerinţele necesare;

-asigură personalul tehnic de faptul că produsul este conform cu un standard agreeat şi de faptul că acesta a fost testat;

-asigură pe utilizator că modulul a fost supus unor cerinţe de securitate bine definite;

-creşte fiabiliatatea cerinţelor de securitate care trebuie îndeplinite pentru realizarea unei aplicaţii specifice.

Printre *responsabilităţile NIST/CSE* menţionăm:

-vizualizarea rapoartelor şi a certificatelor de validare;

-elaborarea politicilor CMVP;

-elaborarea de ghiduri şi clarificări pentru FIPS 140-1 şi alte standarde criptografice.

-asistarea *National Voluntary Laboratory Accreditation Program* (NVLAP).

Cerinţe ale FIPS PUB 140-2

Cerinţele de securitate ale FIPS PUB 140-2 acoperă 11 domenii legate de designul şi implementarea modulului criptografic. În funcţie de cerinţele ce sunt îndeplinite, în majoritatea domeniilor, modulul criptografic primeşte o notă (de la 1 la 4) direct proporţională cu nivelul de securitate asigurat.

Validarea modulelor criptografice

Lista modulelor criptografice validate cuprinde următoarele informaţii pentru fiecare modul criptografic:

-numele producătorului şi punctele de contact;

-numele modulului şi numărul versiunii;

-tipul de modul (software sau hardware);

-data de validare;

-nivelul de validare;

-descrierea modulului (sau a produselor ce sunt incorporate în acest modul).

2.4.3. Proceduri de certificare şi autorizare

Organizaţii industriale şi de standardizare

Nivelul superior de testare, după cel efectuat asupra algoritmilor şi modulelor, se află situat la nivelul producţiei. Produsele sunt testate de către distribuitor, organizaţii de standardizare, precum şi de alte organizaţii independente cu atribuţii de verificare şi de validare, care verifică faptul că produsele analizate funcţionează corect şi într-un mod sigur. Modulele şi componentele criptografice vor fi înglobate în aceste produse, iar pentru aplicaţii guvernamentale aceste module vor trebui să satisfacă prevederile unui standard naţional (echivalent cu FIPS PUB 140-2). La acest nivel, elementul cel mai important este de a obţine garanţii că respectivul produs nu compromite sau elimină caracteristicile criptografice, conducând - prin aceasta - la realizarea unui dispozitiv cu funcţionare nesigură din acest punct de vedere.

2.4.4. Autoritate de certificare

Nivelul cel mai ridicat de testare se realizează în cadrul unui sistem sau al unei aplicaţii. La nivel naţional, această testare va reprezenta chiar testarea de certificare. Prin ea se înţelege o analiză exhaustivă atât a procedurilor de control de securitate tehnice sau de altă natură, precum şi alte măsuri de protecţie ale sistemului. Testarea de certificare precizează în ce măsură un anumit sistem satisface cerinţele de securitate, depinzând de funcţiile sale şi de nevoile operaţionale. Respectiva certificare vine în sprijinul autorizării acordate (de către forurile în drept) privind utilizarea unui sistem. Prin certificare sistemul este examinat în cadrul mediului său operaţional şi împreună cu alte sisteme care sunt legate în reţea. Una din sarcinile importante ale testării de certificare este aceea de a verifica eliminarea oricăror riscuri de compromitere sau de eliminare ale caracteristicilor de securitate, ce ar putea apare datorită unor sisteme externe. Se impune deci nu numai verificarea procedurilor tehnice de control, dar şi a celorlalte măsuri de securitate (fizice, administrative şi de personal). Pentru instituţiile guvernamentale este de dorit ca această testare de certificare să fie efectuată de departamente/organizaţii care nu au activităţi de cercetare/proiectare, asigurându-se astfel o deplină obiectivitate. La toate nivelurile la care se efectuează testarea, o deosebită importanţă are posibiliatea de urmărire a modului cum au fost implementate controalele criptografice şi ale altor parametri de securitate în raport cu cerinţele standardelor în vigoare.

2.5. Procesul de selectare a modulelor criptografice

2.5.1. Faza de planificare

În faza de planificare, scopul activităţii este de a defini obiectivele pe care trebuie să le aibă în vedere tehnicile şi metodele criptografice. Aceste obiective se bazează parţial pe reglementările naţionale şi internaţionale aplicabile. Aceste obiective se stabilesc şi în funcţie de mediul de securitate existent (sau propus), precum şi în legătură cu o analiză preliminara a riscurilor faţă de riscuri şi a vulnerabilităţilor identificate.

Politica de securitate

Politica de securitate (*IT*) este în general definită drept activitatea de stabilire a deciziilor privind securitatea informaţiilor (electronice). Sunt incluse aici politicile manageriale referitoare la :

1. Crearea unui program de securitate a reţelelor de comunicaţii electronice, cu stabilirea sarcinilor şi a responsabilităţilor,

2. Precizarea regulilor de securitate pentru anumite sisteme informatice,

3. Dezvoltarea politicilor necesare în cadrul organizaţiilor.

Proiectarea sistemului include implemntari hardware şi software, proceduri, cerinţe asupra ambientului, considerente privind securitatea fizică, etc. De obicei, politicile de securitate au la bază nevoia de a proteja resursele IT, datele şi informaţiile. Într-o organizaţie politicile de securitate există la toate nivelurile, pornind de la situaţii generale şi până la aplicaţii specifice. Există de asemenea politici de securitate la nivel naţional, care se aplică tuturor organizaţiilor sau agenţiilor şi, din acest punct de vedere, se garantează o direcţie de acţiune precisă pentru protecţia resurselor IT naţionale. După cum vom vedea în continuare, următoarea etapă importantă din faza de planificare este evaluarea riscurilor, după care se vor putea dezvolta politici privind utilizarea sistemelor de operare evaluate şi a modulelor criptografice în anumite condiţii de mediu ambiental. Dăm, mai jos, doar câteva din problemele ce trebuie avute în vedere când se elaborează cerinţele şi politicile criptografice:

1) modul de folosire a algorimilor şi configurarea parametrilor acestora;

2) clasele de utilizatori ce vor folosi metode criptografice şi privilegiile asociate acestor atribuţii;

3) definirea controalelor de confidenţialitate şi integritate, ca şi a tehnicilor de autentificare;

4) procedurile de audit;

5) managementul cheilor, inclusiv distribuţia, generarea, folosirea şi distrugerea;

6) interoperabilitatea între diferite organizaţii şi instituţii, etc.

Evaluarea riscurilor

Managementul riscurilor are două componente: evaluarea, urmată de selecţia şi implementarea contramăsurilor necesare.

- *Evaluarea* riscurilor determină impactul anumitor pierderi şi probabilitatea ca acestea să apară. Pentru metodele criptografice aceste evenimente sunt reprezentate de dezvăluirea neautorizată şi de modificarea datelor.

- *Selecţia şi implementarea* contramăsurilor reduc probabilitatea de apariţie sau minimizează impactul pierderilor.

Evaluarea riscurilor include urmatoarele activităţi:

1) identificarea obiectivelor de protejat;

2) evaluarea mecanismelor curente de securitate şi protecţie;

3) identificarea şi clasificarea ameninţărilor ce afectează integritatea, confidenţialitatea, autentificarea, non-repudierea, disponibilitatea;

4) identificarea şi clasificarea pierderilor potenţiale;

5) dezvoltarea de scenarii de risc;

6) identificarea contramăsurilor de protecţie potenţiale;

7) efectuarea analizei de cost/beneficii pentru contramăsurile propuse.

Obiectivele de securitate

Cea de-a treia sarcină majoră în faza de planificare este stabilirea *obiectivelor de securitate*. Aceste obiective sunt considerate de nivel ridicat şi se referă la securitatea în general şi la criptografie în particular. Iată în continuare doar câteva exemple de obiective de securitate:

1) *Integritatea*: Trebuie garantată corectitudinea cheilor criptografice precum şi a altor CSP (parametri critici de securitate).

2) *Disponibilitatea*: Mecanismele de securitate trebuie să fie permanent disponibile.

3) *Siguranţa*: Trebuie să existe un anumit nivel de siguranţă că mecanismele de securitate funcţionează corect. Nu trebuie să apară vulnerabilităţi sporite la un sistem prin conectarea sa la sisteme/reţele externe.

4) *Autentificarea*: Se va realiza identificarea şi autentificarea utilizatorilor sistemului, precum si urmărirea ulterioară a activităţilor acestora.

5) *Semnături numerice*: Se vor folosi pentru a valida identitatea semnatarului unui mesaj şi integritatea informaţieie recepţionate.

2.5.2. Faza de definire a cerinţelor şi specificaţiilor de securitate

În această fază obiectivul este de a preciza *cerinţele şi specificaţiile* pentru metodele criptografice propuse. După elaborarea acestor prevederi se va trece la definirea criteriilor generale de selecţie, iar în final se vor identifica metodele ce satisfac respectivele cerinţe. Cerinţele vor fi detaliate deoarece ele se constituie într-un ajutor preţios în activitatea de selectarea a produselor, de implementare şi de testare. Cerinţele criptografice pot fi împărţite în trei categorii: funcţionale, de siguranţă şi de mediu ambiental.

Cerinţele funcţionale descriu comportarea de securitate dorită pentru un sistem sau produs şi ele trebuie să satisfacă obiectivele de securitate.

Cerinţele de siguranţă garantează faptul că un produs sau sistem IT răspunde obiectivelor sale de securitate. Certitudinea că obiectivele de securitate sunt satisfacute are la bază următorii factori:

a) încrederea în corectitudinea implementării funcţiilor de securitate, deci evaluarea acestei implementări;

b) încredera în eficienţa funcţiilor de securitate, respectiv evaluarea satisfacerii obiectivelor de securitate stabilite.

Cerinţele ambientale sunt menite să prevină ameninţările şi riscurile din mediul ambiental de operare al produsului sau sistemului şi să acopere orice politici sau ipoteze de securitate din cadrul organizaţiei. Aceste cerinţe se aplică mai curând la sisteme decât la metode criptografice specifice. Ele sunt o componentă a securităţii mediului, care include legi, politici de securitate, expertiză şi alte cunoştinţe relevante pentru acest domeniu.

Etapa finală în faza de definire este identificarea categoriilor de metode şi tehnici criptografice verificate ca răspunzând cerinţelor de securitate prin metode de evaluare a riscurilor.

2.5.3. Faza de achiziţie

În *faza de achiziţie* se va selecta un produs sau modul ce corespunde cerinţelor documentate, pentru ca ulterior acesta să fie configurat, implementat şi testat în sistem. Trebuie subliniat faptul că testarea extensivă a controalelor criptografice este deosebit de importantă datorită rolului acestora în garantarea securităţii sistemului în ansamblul său.

O a doua sarcină majoră în această fază este elaborarea documentaţiei pentru utilizatori şi administratorul de securitate criptografică care să-i instruiască în privinţa responsabilităţilor ce le revin pentru a garanta securitatea sistemului.

Securitatea oferită de un sistem criptografic depinde de valabilitatea matematică a algoritmului folosit, de lungimea cheilor criptografice, de managementul cheilor şi

de modul de operare. În fazele de cercetare şi dezvoltare, responsabilitatea privind realizarea unui modul care să respecte anumite cerinţe de securitate revine producătorului, dar trebuie avut în vedere faptul că prin conformitatea cu un anumit standard nu se garantează automat că un anumit produs este sigur. Responsabilitatea finală revine integratorului de sistem, care trebuie să asigure configurarea şi verificarea unui modul criptografic pentru a garanta funcţionarea sa în condiţii de securitate acceptate.

Următoarele trei reguli guvernează implementarea criptografiei în sistemele IT:

1) Să se determine precis ce informaţii trebuie protejate folosind o funcţie criptografică;

2) Să se protejeze datele înainte de generarea/verificarea semnăturii şi de operaţiile de cifrare sau descifrare;

3) Să se dea utilizatorilor posibilitatea de a monitoriza local toate datele ce sunt semnate sau cifrate.

2.5.4. Faza de operare

În această fază trebuie să se asigure funcţionarea continuă în siguranţă a metodelor criptografice. Cele două domenii critice sunt *pregatirea utilizatorilor* şi *managementul supravegherii componentelor criptografice*.

Este deosebit de important ca toţi utilizatorii să fie conştienţi de responsabilităţile ce le revin, să cunoască procedurile ce trebuie respectate în condiţii normale sau deosebite şi să ştie unde să ceară sprijin în caz de necesitate. Respectivele proceduri trebuie să se constituie în standarde general aplicabile în cadrul sistemului. Dacă nu se respectă un set bine documentat de proceduri, în sistem pot apare slăbiciuni care pot ajunge până la transmiterea datelor în clar. Întreţinerea componentelor criptografice este un factor critic în asigurarea funcţionării în siguranţă, precum şi pentru disponibilitatea modulului/produsului. Ca simplu exemplu, cheile criptografice neschimbate după plecarea unor angajaţi, constituie un element de nesiguranţă.

În acest context, se vor avea în vedere următoarele domenii de întreţinere: hardware/firmware, software, aplicaţii, chei şi personal.

2.6. Operaţii in criptanaliză

2.6.1. Principii criptanalitice

În general în proiectarea unui sistem criptografic trebuie respectate următoarele *principii de proiectare*:

1. Nu trebuie subestimat adversarul;
2. Numai un criptanalist poate evalua securitatea unui sistem criptografic;

3. În evaluarea unui sistem criptografic se ia în considerare faptul că adversarul are cunoştinţe complete despre sistemul evaluat (*Claude E. Shannon* [73]: adversarul cunoaşte în totalitate sistemul criptografic). Tot secretul unui sistem criptografic trebuie să rezide numai în cheie;

4. Complicaţiile superficiale pot fi iluzorii iar acestea pot induce criptografului un sentiment de falsă siguranţă;

5. În evaluarea unui sistem criptografic trebuie luate în calcul toate elementele ca de exemplu reglementările referitoare la distribuţia cheilor, a materialului criptografic, etc.

2.6.2. Criterii de evaluare

Claude Shannon a propus ca următoarele elemente să fie luate în considerare în operaţia de analiză a unui sistem criptografic:

1. Câştigul adversarului rezultat din eventuala decriptare a materialului;
2. Lungimea cheii şi complexitatea modului de manipulare al acestora;
3. Gradul de complexitate al efectuării unui ciclu de cifrare-descifrare;
4. Dimensiunea textului cifrat relativ la dimensiunea textului clar;
5. Modul de propagare al erorilor.

2.6.3. Patru operaţii de bază ale criptanalizei

În general, operaţiile de bază pentru soluţionarea fiecărei criptograme sunt sitentizate în următoarele etape:

1) determinarea limbii folosite în textul respectiv;

2) determinarea tipului de sistem criptografic folosit;

3) reconstrucţia unei chei specifice (parţiale sau complete) a sistemului criptografic echivalent determinat la PASUL 2;

4) reconstrucţia folosirii unui astfel de sistem şi/sau stabilirea textului clar complet.

În unele cazuri, etapa 2 poate precede etapa 1. Aceasta este abordarea clasică în criptanaliză şi poate fi redusă la:

1. Aranjarea sau rearanjarea datelor pentru găsirea caracteristicilor nealeatoare sau a manifestărilor (numărătoare de frecvenţă, repetiţii, forme, fenomene simetrice);

2. Recunoaşterea caracteristicilor nealeatoare sau a manifestărilor când sunt puse în evidenţă (prin statistici sau alte tehnici);

3. Explicarea caracteristicilor nealeatoare când sunt recunoscute (prin noroc, inteligenţă sau perseverenţă). Munca cea mai grea constă în determinarea sistemului general.

În analiza finală, soluţia fiecărei criptograme implică o formă de substituţie ce depinde de reducerea termenilor monoalfabetici dacă textul cifrat nu se exprimă prin literele textului clar.

2.6.4. Evaluare şi spargere

Trebuie să facem de la bun început diferenţa dintre noţiunile de evaluare şi cea de spargere a unui sistem criptografic. *Evaluarea* este un proces prin care se urmăreste punerea în evidenţă a unor neconcordanţe sau hibe ale sistemului criptografic ce pot fi folosite de un eventual cracker. Un model de testare (testare algoritm, testare modul criptografic, testare produs şi testare aplicaţie/sistem) este prezentată în figura 2.4.1.

Evaluarea modulului criptografic se poate face de exemplu cu ajutorul standardului NIST *FIPS 140-2* (pe patru niveluri de ierarhizare) iar evaluarea unui produs cu ajutorul metodologiei *CC Common Criteria,* adoptată de SUA, CANADA şi UE (pe şapte niveluri de ierarhizare). Proiectarea şi evaluarea securităţii unei reţele de calculatoare se poate face conform *Stallings* [88] şi respectiv standardului american *NIST 800-40* [108].

Spargerea este operaţia prin care se poate proiecta o tehnică, metodă sau algoritm prin care să se poată recupera cheia sistemului criptografic sau textul clar cu o complexitate mai mică decât metoda atacului brut:

-evaluatorul doreşte să găsească *cantitatea minimă de informaţie* de tip ieşire din care poate determina, prin folosirea unor instrumente matematice puternice, *o serie de informaţii* despre algoritmul de cifrare, cheia folosită şi/sau textul clar;

-crackerul doreşte să găsească *cantitatea maximă de informaţie* din care poate deduce textul clar.

Aici termenii de maxim sau minim au o semnificaţie generică. De fapt, problema este o problemă de decizie multicriterială (vezi *Preda* [57] pentru o introducere în *teoria deciziilor statistice*): o serie de funcţii obiectiv trebuie maximizate (mărimea cheii de cifrare, gradul de neliniaritate, complexitatea liniar echivalentă, perioada generatorului pseudoaleatoriu, riscul de interceptare a cheii într-o reţea criptografică etc.) iar alte funcţii trebuie minimizate (redundanţa generatorului de chei, gradul de corelare al intrării/ieşirii). Aceste funcţii sunt condiţionate de cunoaşterea sistemului criptografic (adversarul are cunoştinţe complete despre sistemul de cifrare folosit), deci *tăria unui sistem de cifrare trebuie să rezide numai în cheie.*

Să notăm cu $Info_n(\mathbf{m}, \mathbf{c}, \mathbf{k})$ informaţia adiţională referitoare la sistemul criptografic, adică o relaţie vectorială între o mulţime de n mesaje clare, o mulţime de $p(n)$ criptograme şi o mulţime de $q(n)$ chei particulare. Relaţia $Info_n(\mathbf{m}, \mathbf{c}, \mathbf{k})$ este construită cu ajutorul mai multor tipuri de atac, ca de exemplu: atacul pe baza textului clar (text clar cunoscut şi/sau text clar ales), atac diferenţial, cunoaşterea

unei mulţimi particulare de chei ($q(n) > 0$), mesaje identice cifrate cu două sau mai multe chei diferite (unui mesaj clar i se asociază mai multe mesaje cifrate), sau cunoaşterea sistemului criptografic.

Pentru evaluator. Fie $e_n \in [0;1]$ un şir de numere reale şi funcţia obiectiv $n + p(n) + q(n)$. Problema de optimizare pentru evaluator este:

$$\begin{cases} \min(n + p(n) + q(n)) \\ Info_n(\mathbf{m}, \mathbf{c}) = \mathbf{0} \\ H(\mathbf{m}|\mathbf{c}) \geq e_n, \end{cases}$$

unde $H(\mathbf{m}|\mathbf{c})$ este entropia condiţionată (vezi *Guiaşu* [32])de vectorul criptogramelor $\mathbf{c}$ a variabilei aleatoare asignate vectorului mesajelor clare $\mathbf{m}$.

Evaluatorul doreşte ca:

1. $\lim_{n\to\infty} e_n = 1$ (cunoaşterea textului cifrat nu compromite textul clar);
2. să minimizeze pierderea maximă (notată aici prin $L(x, y)$) adică:

$$\alpha = \min_x \max_y L(x, y),$$

unde x este strategia de apărare a evaluatorului (numită şi politică de apărare), iar y este strategia atacatorului.

Pentru cracker. Fie $d_n \in [0;1]$ un şir de numere reale şi funcţia obiectiv $n + p(n) + q(n)$. Problema de optimizare pentru cracker este:

$$\begin{cases} \max(n + p(n) + q(n)) \\ Info_n(\mathbf{m}, \mathbf{c}) = \mathbf{0} \\ H(\mathbf{m}|\mathbf{c}) \leq d_n, \end{cases}$$

unde $H(\mathbf{m}|\mathbf{c})$ este entropia condiţionată de criptograma a variabilei aleatoare asignate mesajului clar.

Crackerul doreşte ca:

1. $\lim_{n\to\infty} d_n = 0$ (cunoaşterea textului cifrat compromite mesajul clar);
2. să maximizeze câştigul minim (notat tot prin $L(x, y)$, câştigul fiind o pierdere negativă) adică:

$$\beta = \max_y \min_x L(x, y),$$

unde x and y au aceleaşi specificaţii.

În general, avem $\beta \leq \alpha$ adică maximul câştigul minim al crackerului nu poate depăşi minimul pierderii maxime a evaluatorului (dacă avem egalitate atunci strategiile corespunzătoare se numesc puncte şa pentru funcţia L).

Avem $\beta < \alpha$ dacă canalul de interceptare este cu perturbaţie.

Cele două probleme de programare (evaluator/cracker) sunt probleme duale.

Avem următoarea relaţie vectorială:

$$\mathbf{c} = \mathbf{f}(\mathbf{m}; \mathbf{k}_t) \tag{2.1}$$

unde $\mathbf{f}$ este operatorul de criptare.

Dacă $\mathbf{k}_t = \mathbf{k}$ pentru orice $t \in T$ (T este perioada de cifrare care este o mulţime discretă) atunci relaţia anterioară se rescrie:

$$\mathbf{c} = \mathbf{f}(\mathbf{m}; \mathbf{k}) \tag{2.2}$$

unde $\mathbf{f}$ este operatorul de criptare. În acest caz, spunem că avem de a face cu o codificare a informaţiei (rolul teoriei codurilor este de a proteja informaţia de erorile ce pot apărea pe canalul de comunicaţie; rolul criptografiei este de a proteja informaţia de interceptarea neautorizată).

În cazul codificării după rezolvarea unui sistem neliniar, putem scrie:

$$\mathbf{m} = \mathbf{h}(\mathbf{c}; \mathbf{k}) \tag{2.3}$$

Deci cunoaşterea lui $\mathbf{f}(.;.)$ ne permite să găsim pe $\mathbf{m}$ din $\mathbf{c}$. Sistemul 2.1, care este un sistem stochastic $(\Omega, \mathcal{K}, \mathcal{P})$, este mult mai dificil de rezolvat decât sistemul 2.2 care este un sistem determinist, deoarece nu apare parametrul t. Soluţia sistemului 2.2, dată de 2.3, este o soluţie particulară a sistemului 2.1 pentru situaţia în care $\mathbf{k}_t = \mathbf{k}$. Cu alte cuvinte, putem spune că operaţia de codificare este o operaţie de criptare cu o cheie particulară.

De foarte multe ori, funcţia de criptare $\mathbf{f}$ este dată în forma scalară:

$$c_i = f(m_i, k_i), \text{ pentru orice } i = 1, ..., n,$$

unde k_i este cheia i obţinută din cheia de bază $\mathbf{k}_t$.

Dacă f poate fi factorizată în modul următor:

$$f(m_i, k_i) = m_i \oplus g(k_i),$$

unde $\oplus$ este operatorul de sumare $\bmod 2$, atunci schema de criptare se numeşte *cifrare flux*, iar funcţia g *generator pseudoaleatoriu*. Datorită simplităţii (din punct de vedere al implementării) şi rapidităţii această schemă de cifrare este foarte folosită în cifrarea comunicaţiilor de tip date şi voce. În acest caz dificultatea spargerii este echivalentă cu dificultatea predicţiei sau chiar a determinării funcţiei g (sau a unei funcţii echivalente care dă aceeaşi ieşire). Tehnica rezolvării problemei este echivalentă cu tehnica ingineriei inverse. Pentru a putea face o comparaţie a complexităţii de implementare sau a complexităţii de spargere a algoritmilor de cifrare se folosesc simbolurile O şi o.

Definiţia 2.6.1. (*Măsura complexităţii*). Fie $f : D \to \mathbf{R}$.
a) Spunem că o funcţie $g : D \to \mathbf{R}$ este $O(f)$ dacă g aparţine mulţimii:

$$O(f) = \{h : D \to \mathbf{R} | \ \exists M > 0 : \\ |h(x)| \leq M|f(x)|, \ \forall x \in D\}.$$

b) Spunem că o funcţie $g : D \to \mathbf{R}$ este $o(f)$ dacă g aparţine mulţimii:

$$o(f) = \{h : D \to \mathbf{R} | \ \lim_{x \to 0} \frac{h(x)}{f(x)} = 0\}.$$

Se poate demonstra (vezi *Knuth* [38]) că $o(f) \subset O(f)$.
Definiţia anterioară se poate reformula în cazul şirurilor.
Definiţia 2.6.2. (*Măsura complexităţii*). Fie şirul x_n.
a) Spunem că şirul z_n este $O(x_n)$ dacă z_n aparţine mulţimii:

$$O(x_n) = \{y_n | \ \exists M > 0 : |y_n| \leq M|x_n|\}.$$

b) Spunem că şirul z_n este $o(x_n)$ dacă z_n aparţine mulţimii:

$$o(x_n) = \{y_n | \ \lim_{n \to \infty} \frac{y_n}{x_n} = 0\}.$$

De asemenea avem $o(x_n) \subset O(x_n)$ (vezi *Knuth* [38]).

Folosind notaţiile anterioare putem evalua complexitatea algoritmilor de spargere. Dacă dorim *să proiectăm un cifru bun* atunci trebuie să garantăm o complexitate minimă de spargere de $O(2^n)$ (acest lucru înseamnă că oponentul nu poate sparge sistemul în timp rezonabil sau cu alte cuvinte cea mai eficientă metodă de spargere este căutarea exhaustivă a cheii sau a parolei), iar dacă dorim *să proiectăm un cracker pentru un algoritm de cifrare* atunci trebuie să garantăm o complexitate de cel puţin $O(n)$ (aceasta înseamnă că spargem sistemul adversarului în timp cel mult polinomial).

Evaluarea complexităţii sistemelor criptografice face parte din clasa *testelor de confirmare* (alte teste de confirmare sunt testele de *complexitate liniară*, *complexitate pătratică* sau *complexitate Lempel-Ziv* (vezi *NIST 800-22* [107])). Aceste teste se execută, de regulă, după procesarea *testelor de referinţă* (*teste statistice* sau alte criterii funcţionale ca: *avalanşă strictă, balans, neliniaritare, simetrie, nedegenerare, necorelare*). O descriere completă a acestor teste statistico-informaţionale este făcută în Capitolul 4. Testele pentru evaluarea sistemelor de cifrare sunt făcute în următoarea ordine (vezi *Simion* [84]):

PAS 1. *Execută teste de REFERINŢĂ*: teste statistice. Dacă sistemul de cifrare cade la aceste teste (decizie multicriterială) atunci *respinge acest sistem de cifrare*,

iar în caz contrar *execută PAS 2*. Aceste teste sunt procesate în timp $O(1)$, cost de memorie $O(1)$ şi au o senzivitate de obicei mai mare ca 95%.

PAS 2. *Execută teste de REFERINŢĂ*: teste funcţionale. Dacă sistemul de cifrare cade la aceste teste (decizie multicriterială) atunci *respinge acest sistem de cifrare*, iar în caz contrar *execută PAS 3*. Aceste teste sunt procesate în timp $O(n)$, cost de memorie $O(1)$ şi au o senzivitate de obicei mai mare ca 98%.

PAS 3. *Execută teste de CONFIRMARE*: teste de complexitate liniară (vezi *Preda* şi *Simion* [59]). Dacă sistemul de cifrare cade la aceste teste (decizie multicriterială) atunci *respinge acest sistem de cifrare*, iar în caz contrar *execută PAS 4*. Aceste teste sunt procesate în timp $O(n)$, cost de memorie $O(n)$ şi au o senzivitate mai mare ca 99%.

PAS 4. *Execută teste de CONFIRMARE*: teste de complexitate Lempel-Ziv şi complexitate pătratică. Dacă sistemul de cifrare cade la aceste teste (decizie multicriterială) atunci *sistemul de cifrare are predictibilitate ascunsă,* iar în caz contrar *sistemul de cifrare trece de bateria de teste*. Aceste teste sunt procesate în timp $O(2^n)$, cost de memorie $O(2^n)$ şi au o senzivitate mai mare ca 99,9%.

Se observă că cele mai rapide teste sunt cele de la paşii 1 şi 2. Cele mai lente teste care cer un cost foarte mare de stocare sunt testele executate la pasul 4.

Observaţie. În ultima perioadă se vehiculează ideea că autoritatea statală trebuie să aibă acces la conţinutul mesajelor cifrate, prin intermediul aşa -numitei proceduri de recuperare a cheii (*key recover*), dar nu trebuie ca acest lucru să afecteze siguranţa sistemului de semnătură digitală. Pentru realizarea acestui deziderat o parte din cheia secretă se transmite pe canalul de comunicaţie (evident în mod criptat şi în urma unor autentificări) către autoritatea desemnată.

2.7. Clasificări ale atacurilor criptanalitice

În acest paragraf sunt prezentate o serie de tipuri de atac asupra sistemelor de cifrare. Astfel sunt exemplificate tipurile de atac asupra algoritmilor de cifrare, asupra cheilor, protocoalelor de autentificare, atacurile asupra sistemului propriu-zis precum şi atacurile neconvenţionale (hardware). Aceste tipuri de atac nu sunt exhaustive, un atac eficient fiind compus, în general, dintr-o submulţime a celor ce urmează.

2.7.1. Tipuri de atac asupra algoritmilor de cifrare

Principalele tipuri de atac care se referă la algoritmul de cifrare sunt următoarele:

Atac cu text clar cunoscut. Criptanalistul deţine un text cifrat şi corespondentul său în clar. Prin acest tip de atac se urmăreşte, de către criptanalist, separarea

informaţiei text de cheia de cifrare, acesta având posibilitatea să obţină, prin metode specifice, algoritmul de cifrare sau o parte a acestuia şi/sau cheia.

Atac cu text clar ales. Criptanalistul poate indica textul clar ce urmează a fi cifrat. Prin acest tip de atac se urmăreşte, de către critpanalist, separarea informaţiei text de cheia de cifrare, acesta având posibilitatea să obţină, prin metode specifice, algoritmul de cifrare şi/sau cheia.

Atac cu text cifrat-cifrat. Criptanalistul deţine un text clar şi corespondentul său cifrat cu două sau mai multe chei diferite. Criptanalistul, prin metode specifice, poate reconstrui algoritmul de cifrare sau o parte a acestuia.

Atac de tip divide et impera. Criptanalistul poate realiza o serie de corelaţii între diversele intrări în algoritm şi ieşirea acestuia, încercând să separe diversele intrări în algoritm, ceea ce îl face pe acesta să spargă problema în două sau mai multe probleme mai uşor de rezolvat.

Atac de tip sindrom liniar. Metoda de criptanaliză constă în elaborarea unui sistem de ecuaţii liniare ale generatorului pseudoaleator şi verificarea acestora de către textul cifrat, obţinându-se astfel textul clar cu o probabilitate destul de mare.

Atacul consistenţei liniare. Metoda de criptanaliză constă în elaborarea unui sistem de ecuaţii liniare ale generatorului pseudoaleator pornind de la o cheie echivalentă de cifrare şi verificarea sistemului, de către generatorul pseudoaleator, cu o probabilitate care tinde la 1, obţinându-se astfel textul clar cu o probabilitate destul de mare.

Atac stochastic asupra ieşirii generatorului, numit şi *atac prin previziune,* este posibil dacă ieşirea generatorului este autocorelată, criptanalistul reuşind să deţină, ca date de intrare, ieşirea generatorului pseudoalator şi textul cifrat obţinând, astfel textul clar corespunzător. Pentru evitarea acestui tip de atac generatorul trebuie să satisfacă condiţiile de:

-balans: toate intrările posibile trebuie să producă toate ieşirile posibile de acelaşi număr de ori;

-nedegenerare: ieşirea trebuie să provină din fiecare element al intrării;

-imunitate la corelaţie: intrări corelate produc ieşiri necorelate;

-avalanşă strictă: schimbarea unui bit să producă schimbări în proporţie de 50%.

Atac informaţional liniar asupra generatorului, numit şi *atac de complexitate liniară,* este posibil dacă se poate echivala generatorul cu un algoritm tip Fibonacci (registru de deplasare liniar) şi dacă complexitatea liniar echivalentă a generatorului este mică. Cu ajutorul acestei tehnici se poate construi un algoritm echivalent şi o cheie echivalentă de cifrare.

Atac cu ajutorul perioadei generatorului pseudoaleator este realizabil dacă perioada generatorului pseudoaleator este mică şi se poate reconstrui textul clar corespunzător.

Atac cu ajutorul viruşilor informatici este posibil dacă algoritmul de cifrarea este implementat şi rulează pe un PC vulnerabil sau neprotejat. Virusul poate substitui sau inhiba algoritmul de cifrare utilizat.

2.7.2. Tipuri de atac asupra cheilor

Atacurile cele mai frecvente care se realizează asupra cheilor de cifrare:

Atacul brut constă în verificarea exhaustivă a cheilor sau parolelor şi este posibil dacă:

-lungimea cheii de cifrare efective (sau a parolei) este mică;

-spaţiul cheii de cifrare (sau a parolei) este mic.

Atacul brut inteligent se realizează dacă gradul de aleatorism al cheii de cifrare (sau parolei) este mic (entropia este mică) şi permite aflarea parolelor care se aseamănă cu cuvinte din limba utilizată.

Atacul de tip backtracking constă în implementarea metodei de căutare de tip backtracking (presupune existenţa unor condiţii de continuare a căutării în direcţia de căutare considerată).

Atacul de tip greedy furnizează cheia optimă locală care poate să nu coincidă cu cheia optimă globală.

Atacul de tip dicţionar (căutarea parolelor sau a cheilor se face după un dicţionar) este posibil dacă parola sau cheia sunt cuvinte cu sens (nume, date etc.).

Atac de tip dicţionar hibrid este posibil prin modificarea cuvintelor din dicţionar, iniţializându-se atacul brut cu ajutorul cuvintelor din dicţionar.

Atac cu ajutorul viruşilor informatici este posibil dacă cheile se stochează pe un PC neprotejat.

Atac asupra hashului parolei sau *cheii de cifrare.* Este posibil dacă hashul parolei este scurt sau necorespunzător elaborat.

Atac prin substituţie. Cheia originală este substituită, de către o terţă persoană, şi înlocuită în toată reţeaua de comunicaţii (sau într-o parte a ei) cu o cheie a acestuia. Este posibil cu ajutorul viruşilor informatici.

Stocarea cheii de cifrare în mod necorespunzător (împreună cu datele cifrate) în clar fără măsuri de protecţie fizice sau criptografice (soft sau hard) poate duce la atac asupra mesajului cifrat.

Stocarea necorespunzătoare a cheilor vechi sau scoase din uz duce la compromiterea documentelor cifrate vechi.

Compromiterea cheii. Dacă se compromite cheia simetrică, se compromit numai acele documente cifrate cu aceasta. Dacă se compromite cheia publică, care se poate afla pe diverse servere, atacatorul se poate substitui utilizatorului legal provocând daune în toată reţeaua de comunicaţii.

Concluzie: *Existenţa cheilor master* sau *a cheilor de salvare* constituie trape în sistemul criptografic.

Timpul de viaţă al cheii este o componentă esenţială care exclude posibilitatea unui atac reuşit, dar nedetectat.

Existenţa unui sistem de generare, gestiune şi management al cheilor de cifrare este cheia întregului succes.

2.7.3. Tipuri de atac asupra protocoalelor de autentificare

Protocoalele de autentificare pot fi supuse următoarelor tipuri de atacuri:

Atac criptografic asupra cheii publice folosite (dacă se foloseşte sistemul cu chei publice) pentru semnătură în cadrul protocolului.

Atac criptografic asupra algoritmului simetric folosit (dacă se foloseşte sistemul simetric) pentru semnătura din cadrul unui protocolului de autentificare.

Pentru *evitarea atacului asupra semnăturilor digitale* trebuie ca:

-semnătura să fie nefalsificabilă: semnătura este o dovadă că semnatarul a semnat în mod deliberat documentul;

-semnătura este autentică: semnătura convinge destinatarul că semnatarul a semnat în mod deliberat documentul;

-semnătura este nereutilizabilă: semnătura face parte din document şi nu poate fi mutată pe alt document;

-documentul semnat este inalterabil: după semnare, documentul nu poate fi modificat;

-semnătura este nerepudiabilă: semnatarul nu poate pretinde mai târziu că nu a semnat documentul respectiv;

Semnătura invizibilă poate fi citită numai de către destinatarul documentului.

Semnătura de tip fail-stop este un protocol criptografic în care expeditorul poate aduce dovezi dacă semnătura sa a fost falsificată.

Atac de concordanţă, numit în literatura de specialitate şi *atacul zilei de naştere*, este posibil dacă probabilitatea ca funcţia semnătură aplicată la două documente diferite să producă aceeaşi semnătură.

Atac pasiv asupra protocolului de autentificare. Interceptorul monitorizează comunicaţia pe canal fără a face nici un fel de intervenţie, scopul acestuia fiind de a trage concluzii despre procesul de autentificare.

Atac prin a treia persoană. Comunicaţia dintre cei doi parteneri ai canalului de comunicaţie este interceptată activ de către o terţă persoană.

2.7.4. Tipuri de atac asupra sistemului

Sistemul de cifrare (algoritm, cheie şi protocol de autentificare) este supus următoarelor tipuri de atacuri:

Atac la nivel de algoritm. Sunt cele prezentate anterior.

Folosirea necorespunzătoare a algoritmului de cifrare:

-nu există algoritm de cheie de mesaj;

-folosirea necorespunzătoare a supracifrării poate duce la un algoritm echivalent mult mai slab decât fiecare algoritm în parte.

Atac la nivel de cheie. Sunt cele prezentate anterior.

Atac la nivel de protocol de autentificare sau *de transmitere a cheii.*

Atacuri datorate erorilor de implementare.

2.7.5. Atacuri hardware asupra modulelor criptografice

Metode de atac ce presupun o serie de măsurători hardware asupra modului criptografic:

Atacuri prin măsurarea timpului de execuţie. Prin măsurarea *timpului necesar efectuării unor operaţii asupra cheii private*, atacatorul poate determina exponenţii utilizaţi în protocolul Diffie-Hellman, factorul RSA (în special asupra algoritmului RSA ce foloseşte pentru semnătură lema chinezească a resturilor CRT), precum şi o serie de alte sisteme criptografice cum ar fi algoritmul de semnătură digitală DSS (vezi *Kocher* [43]).

Atacuri prin măsurarea puterii consumate. Atacul cu ajutorul *analizei simple a puterii* (SPA) constă în măsurarea puterii consumate de dispozitiv în timpul operaţiei criptografice. Acest tip de atac se aplică, de regulă, dispozitivelor cu sursă de tensiune exterioară (ca de exemplu smart-cardurile). Consumul de putere depinde de instrucţiunea executată. Astfel, monitorizând consumul de putere, se poate deduce secvenţa de instrucţiuni (codul sursă). Dacă secvenţa de instrucţiuni depinde de lungimea cheii, atunci consumul de putere poate da informaţii despre cheie. În majoritatea procesoarelor, patternul puterii consumate de o instrucţiune depinde şi de valoarea operanzilor (de exemplu setarea unui bit într-un registru consumă mai multă energie decât ştergerea acestuia). Măsurători efectuate asupra mai multor intrări pot deduce valoarea operandului. Tehnica se numeşte *analiza diferenţială a puterii* (DPA).

Atacuri cu ajutorul defecţiunilor (erorilor) hardware. Echipamentele hardware pot genera erori (tranziente, latente sau induse) în timpul efectuarii unor operaţii aritmetice. Prin exploatarea raţională a acestor erori se pot recupera cheia privată pentru algoritmii de semnătură RSA şi Rabin. O serie de protocoale criptografice

cum ar fi Fiat-Schamir şi Schnorr se pot sparge prin folosirea judicioasă a rezultatelor acestor erori (vezi *Boneh* [11]).

Analiza diferenţială a defecţiunilor. Analiza diferenţială a defecţiunilor (DFA) este o schemă ce se utilizează pentru recuperarea cheilor secrete ale unui sistem criptografic dintr-un dispozitiv HSM (Hardware Security Module) securizat fizic. Modelul de defect este acela al defectelor tranziente (aleatoare) şi al defectelor induse. Metoda foloseşte la identificarea cheilor în cazul utilizării unor cifruri cunoscute (de exemplu DES) şi/sau a unor cifruri cu algoritm necunoscut (de exemplu SkipJack), reconstrucţia algoritmului (cu o structură cunoscută) (vezi *Biham* [8]).

2.8. Aplicaţii

Exerciţiul 2.8.1. Care este diferenţa dintre standardele ISO 15408 şi FIPS 140-2?

Exerciţiul 2.8.2. La ce nivel OSI se încadrează cifrarea software?

Exerciţiul 2.8.3. Care sunt principalele caracteristici ale unui modul criptografic?

Exerciţiul 2.8.4. Descrieţi fazele de dezvoltare a unui modul criptografic.

Exerciţiul 2.8.5. Specificaţi pe scurt principalele beneficii ale standardelor.

Exerciţiul 2.8.6. Care este diferenţa dintre activitatea de evaluare şi activitatea de spargere a unui modul criptografic?

Exerciţiul 2.8.7. Descrieţi principalele clasificări ale atacurilor criptanalitice.

Exerciţiul 2.8.8. Care sunt criteriile de evaluare a sistemelor criptografice?

Exerciţiul 2.8.9. Precizaţi operaţiile de bază efectuate în criptanaliză.

Exerciţiul 2.8.10. Care sunt caracteristicile cerinţelor funcţionale şi a specificaţiilor de securitate IT?

Capitolul 3

TEORIA COMPLEXITĂŢII ALGORITMILOR

En cryptographie, aucune règle
n'est absolute.
Étienne Bazeries, 1901

3.1. Introducere

Teoria complexităţii algoritmilor (se pare că acestă denumire provine din *al-Khowârizmî* numele unui matematician persan din secolul al IX-lea) se ocupă în principal cu evaluarea timpului necesar execuţiei unui algoritm dat (*complexitate dinamică*) precum şi a resurselor de stocare sau memorie necesare (*complexitate spaţiu*) rulării acestuia. Având la dispoziţie aceste două noţiuni putem introduce o relaţie de ordine pe mulţimea algoritmilor, care rezolvă o problemă bine precizată, putem face o *comparaţie obiectivă* a doi sau mai mulţi algoritmi şi vom putea da o definiţie a terminologiei de algoritm eficient. Deorece prezenta lucrare abordează domeniile Teoriei deciziilor, Cercetării operaţionale şi Criptografiei din perspectiva elaborării algoritmilor (eficienţi) de rezolvare a problemelor specifice acestor domenii am considerat utilă prezentarea unei scurte introduceri în teoria complexităţii algoritmilor.

3.2. Algoritmi şi maşini Turing

Informal, un *algoritm* este orice procedură bine definită, care pentru o anumită mulţime de valori, numită şi *intrare*, produce o *ieşire*. Putem, de asemenea, privi un algoritm ca un element ajutător pentru rezolvarea *problemelor computaţionale* bine

specificate. Algoritmul descrie o procedură computaţională pentru realizarea unei relaţii între datele de la intrare şi datele de la ieşire.

Complexitatea în timp a unui algoritm este o funcţie de lungimea intrării. Un algoritm are complexitatea în timp $f(n)$, cu $f : \mathbf{N}^* \to \mathbf{N}^*$, dacă şi numai dacă pentru orice n şi orice intrări de lungime n, execuţia algoritmului se face în cel mult $f(n)$ paşi. Dacă n este un întreg, lungimea lui este numărul de cifre sau biţi din n. Desigur, pot exista algoritmi lenţi sau rapizi pentru aceeaşi problemă, în anumite cazuri este posibilă o accelerare nelimitată. Este dificil însă de stabilit limite inferioare pentru complexităţi, de a arăta, de exemplu că orice algoritm pentru o anumită problemă este cel puţin de complexitate pătratică în timp.

Evident, complexitatea depinde de modelul de algoritm adoptat. Numărul de paşi scade pe măsura creşterii sarcinilor atribuite spre rezolvare fiecărui pas. Totuşi, noţiuni fundamentale, ca de pildă complexitatea polinomială în timp sunt independente de model. Modele frecvente pentru algoritmi sunt *maşinile Turing deterministe* (în funcţie de starea maşinii şi de intrare aceasta produce o ieşire unică) sau *nedeterministe* (de exemplu, maşina ghiceşte sau foloseşte un număr arbitar de procesoare paralele). Mai precis maşina Turing deterministă:

-operează în cuante de timp;

-în fiecare moment de timp ea are o stare internă bine precizată;

-numărul stărilor interne posibile este finit;

-dispune de un mecanism citire/scriere de pe o bandă a unui şir de caractere, mişcarea acestuia fiind stânga, dreapta sau pe loc;

-dacă se ajunge cu operaţia de citire/scriere la capătul şirului de caractere atunci i se adaugă acestuia caracterul vid (extinderea nelimitată a memoriei externe).

Deci *complexitatea maşinii Turing deterministe A* este definită ca:

$f_A(n) = \max\{m | A$ se opreşte după m paşi pentru o intrare ω de lungime $n\}$, iar *complexitatea maşinii Turing nedeterministe A* este definită ca:

$f_A(n) = \max\{1, m | s(\omega)$ are m paşi pentru o intrare ω de lungime $n\}$, unde prin $s(\omega)$ se consideră cel mai scurt calcul încheiat cu succes pentru o intrare ω (calcul ce duce la o stare finală).

3.3. Teoria problemelor $NP-$ complete

Teoria problemelor $NP-$ complete se aplică în principal problemelor de decizie care nu pot avea decât două răspunsuri codificate 0 (fals) şi 1 (adevăr): răspunsul da şi răspunsul nu.

Definiţia 3.3.1. O problemă de decizie Π este formată dintr-o mulţime de instanţe D_Π şi dintr-o submulţime $0_\Pi \subseteq D_\Pi$ de 0-instanţe.

Vom prezenta în continuare un exemplu celebru de astfel de problemă de decizie.

Exemplul 3.3.1. (Problema comis-voiajorului, vezi *Cormen* [14])
Instanţe: O mulţime finită $C = \{c_1, ..., c_m\}$ de oraşe, o distanţă $d(c_i, c_j)$ între acestea, o margine superioară M.
Întrebare: Se poate organiza un tur al oraşelor din mulţimea C astfel încât distanţa totală parcursă să nu fie mai mare decât M ?

Definiţia 3.3.2. Un *algoritm* se numeşte *polinomial* dacă complexitatea sa este $O(p(n))$, unde $p(n)$ este un polinom de grad n. Clasa algoritmilor polinomiali se notează cu P. Un algoritm care nu este polinomial, deci care nu face parte din clasa P, face parte din clasa NP, iar în acest caz algoritmul se numeşte *exponenţial*.

Definiţia 3.3.3. O *problemă* se va numi *intractabilă* dacă nu există nici un algoritm polinomial care să o poată rezolva.

Se conjecturează (vezi *Cormen* [14]) că $P \neq NP$ (avem relaţia $P \subset NP$). Având formulată o problemă pe care dorim să o studiem din punctul de vedere al complexităţii cel mai adesea se face reducerea acesteia la o problemă a cărei complexitate este deja cunoscută. Această reducere se face însă după un algoritm care trebuie să fie un algoritm de complexitate polinomială adică din clasa P.

Exemplul 3.3.2. (Problema circuitului hamiltonian, vezi *Cormen* [14])
Instanţe: Un graf $G = (V, E)$;
Întrebare: Graful G conţine un circuit hamiltonian?
Această problemă este echivalentă cu problema comis-voiajorului.

Definiţia 3.3.4. O problemă de decizie L se numeşte $NP-$ completă dacă este din clasa NP şi orice altă problemă $L^* \in NP$ se reduce, printr-o transformare polinomială, la problema L sau, cu alte cuvinte, problema L este cel puţin la fel de grea ca problema L^*. Acest lucru se va nota $L^* \prec L$.

Definiţia 3.3.5. O problemă de decizie L se numeşte co $-NP$ dacă complementara acesteia este din clasa NP.

Avem $P \subseteq NP$ şi $P \subseteq co-NP$. S-a conjecturat (vezi *Cormen* [14]) că incluziunile de mai sus sunt stricte şi mai mult $NP \neq co - NP$ şi $P \neq NP \cap co - NP$. Aceste conjecturi sunt explicitate în figura 3.1.

Din cele prezentate reiese importanţa existenţei problemelor $NP-$ complete. Pentru aceasta fie $U = \{u_1, ..., u_m\}$ o mulţime de *variabile booleene*. O *asignare de adevăr* pentru mulţimea U este o funcţie $t : U \to \{0, 1\}$. Dacă u este o variabilă din U, atunci u şi negaţia sa $\bar{u}$ sunt *afirmaţii* peste U. Afirmaţia u este 1 (u este adevărată) sub t dacă şi numai dacă $t(u) = 1$, afirmaţia $\bar{u}$ este adevărată dacă şi numai dacă $t(u) = 0$ (u este falsă). Vom numi *clauză* peste U o mulţime de afirmaţii peste U, care reprezintă disjuncţia afirmaţiilor din U. O clauză este 1 (adevărată) dacă şi numai dacă cel puţin una din afirmaţiile din clauza respectivă este satisfăcută (cel puţin o afirmaţie este adevărată), deci o clauză este 0 dacă şi numai dacă toate

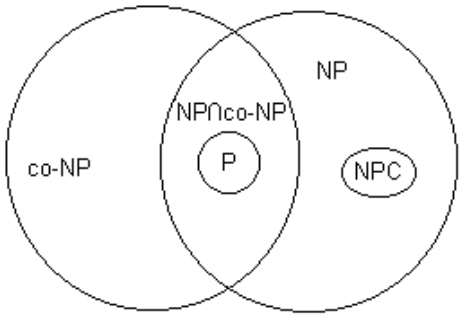

Figura 3.1: Relaţiile conjecturate între mulţimile P, NP $NP-$ complete şi co $-NP$.

afirmaţiile din clauza respectivă sunt false. Vom spune despre o mulţime de clauze $\mathcal{C}$ că este *satisfabilă* dacă şi numai dacă există o asignare de adevăr care satisface simultan toate clauzele din mulţimea $\mathcal{C}$. O asemenea funcţie se va numi funcţie de *satisfabilitate a valorii de adevăr.*

Problema satisfabilităţii este:

Instanţa: O mulţime U de variabile şi o mulţime $\mathcal{C}$ de clauze peste U.

Întrebare: Există o funcţie de satisfabilitate a valorii de adevăr pentru $\mathcal{C}$?

Cook a demonstrat că problema satisfabilităţii este o problemă $NP-$ completă. Pornind de la ea se construiesc toate celelate probleme $NP-$ complete. Se poate demonstra că problema comis-voiajorului este tot o problemă $NP-$ completă, deci şi problema ciclului hamiltonian. Din motivele prezentate anterior ne vom axa numai pe problemele $NP-$ complete. După cum se observă, existenţa unei singure probleme $NP-$ complete duce la construirea altor probleme $NP-$ complete.

Algoritmul de demonstrat $NP-$ completitudinea unei probleme L este următorul:

Pas 1. Demonstrează că $L \in NP$.

Pas 2. Selectează L^* problemă $NP-$ completă.

Pas 3. Construieşte o transformare f de la problema L^* la problema L.

Pas 4. Demonstrează că f este o transformare polinomială.

3.4. Exemple de probleme $NP-$ complete

Există o multitudine de exemple de probleme $NP-$ complete, ne vom rezuma însă la a menţiona pe câteva dintre cele mai sugestive. Unele exemple fac apel la noţiunea de graf.

Exemplul 3.4.1. (Circuit hamiltonian)
Instanţa: Graful G=(V,E);
Întrebare: Graful G conţine un circuit hamiltonian?

Exemplul 3.4.2. (Drum hamiltonian)
Instanţa: Graful G=(V,E);
Întrebare: Graful G conţine un drum hamiltonian?

Exemplul 3.4.3. (Drum hamiltonian orientat)
Instanţa: Graful G=(V,E);
Întrebare: Graful G conţine un drum hamiltonian orientat?

Exemplul 3.4.4. (Problema comis-voiajorului)
Instanţa: O mulţime finită C=$\{c_1, ..., c_m\}$ de oraşe, o distanţă $d(c_i, c_j) \in \mathbf{N}$, o margine superioară M.
Întrebare: Se poate organiza un tur al oraşelor din mulţimea C astfel încât distanţa totală parcursă să nu fie mai mare decât M ?

Exemplul 3.4.5. (Problema rucsacului)
Instanţa: O mulţime U, o mărime $s(u) \in \mathbf{N}$ şi o valoare $v(u) \in \mathbf{N}$ definite pentru orice $u \in U$, o restricţie de mărime $B \in \mathbf{N}$ şi o restricţie de valoare $K \in \mathbf{N}$.
Întrebare: Există o submulţime $U^* \subset U$ astfel ca:

$$\sum_{u \in U^*} s(u) \leq B$$

şi

$$\sum_{u \in U^*} v(u) \geq K.$$

Exemplul 3.4.6. (Calcul paralel)
Instanţa: O mulţime finită A de tascuri, de lungimi $l(a) \in \mathbf{N}$,$a \in A$ un număr $m \in \mathbf{N}$ de procesoare şi un timp limită $D \in \mathbf{N}$.
Întrebare: Există o partiţie a lui $A = A_1 \cup ... \cup A_m$ în m mulţimi astfel încât $max\{\sum_{a \in A_I} l(a) : i = \overline{1, m}\} \leq D$?

Exemplul 3.4.7. (Congruenţe pătratice)
Instanţa: $a, b, c \in \mathbf{N}$;
Întrebare: Există $x < c$ astfel ca $x^2 \equiv a \bmod b$?

Exemplul 3.4.8. (Rezolvarea ecuaţiei diofantice)
Instanţa: $a, b, c \in \mathbf{N}$;
Întrebare: Există $x, y, c \in \mathbf{N}$ astfel ca $ax^2 + by = c$?

Criptografia cu chei publice se bazează pe existenţa problemelor $NP-$ complete dar care pot fi rezolvate cu informaţie suplimentară în timp polinomial ca de exemplu:

Exemplul 3.4.9. (Factorizarea unui număr)
Instanţa: n număr întreg pozitiv;
Întrebare: Este n produs de numere prime?

Problema de mai sus este $NP-$ completă, dar dacă se cunoaşte un factor (netrivial) al lui n atunci acest lucru se poate verifica în timp polinomial. Nu se ştie dacă problema de mai sus este din clasa P.

Să considerăm un sistem de cifrare cu chei publice. Cheia de criptare este publică. Să combinăm acest fapt cu cerinţa pusă pentru orice criptosistem, fie el clasic sau cu cheie publică: criptarea este uşoară de îndată ce cheia de criptare şi textul clar este cunoscut (altfel, criptosistemul va fi greu de folosit). Aceasta implică faptul că, în orice criptosistem cu chei publice rezonabil, *problema criptanalizei este de clasă* NP. Dându-se un text criptat, criptanalistul ghiceşte, mai întâi, textul clar şi apoi textul cifrat, după care îl criptează pentru a stabili dacă acesta conduce la criptotextul dat. Chiar dacă metoda de criptare făcută publică este nedeterministă, întreaga procedură este din clasa NP.

Problema criptanalizei este de asemenea în $co-NP$. Dacă metoda de criptare este deterministă, atunci acest lucru este evident, deoarece se poate proceda exact ca mai sus: stabilim dacă textul clar prezumtiv *nu conduce* la criptotextul dat. În cazul general, independent de faptul că metoda de criptare este deterministă, vom raţiona astfel. Avem tripletul: $(\mathbf{m}, \mathbf{k}, \mathbf{c})$ unde $\mathbf{m}$ este candidatul pentru textul clar, $\mathbf{k}$ este cheia publică de criptare şi $\mathbf{c}$ este criptotextul. Presupunem că acceptăm tripletul exact în cazul în care $\mathbf{w}$ nu este textul clar care va genera pe $\mathbf{c}$. Este clar că există numai un astfel de text clar (în caz contrar decriptarea ar fi ambiguă). Algoritmul nostru ghiceşte mai întâi textul clar $\mathbf{p}$, apoi stabileşte (în timp polinomial nedeterminist) dacă $\mathbf{p}$ generează pe $\mathbf{c}$ conform cu $\mathbf{k}$. Numai în cazul unui răspuns pozitiv algoritmul continuă, comparând pe $\mathbf{p}$ şi $\mathbf{w}$ literă cu literă. Dacă este găsită o diferenţă, algoritmul acceptă textul. Am abordat problema criptanalizei în următorul mod evident: să se stabilească textul clar, când criptotextul şi cheia publică sunt cunoscute. Mergând pe aceeaşi linie, se poate arăta că anumite probleme analoage fac parte din intersecţia $NP \cap co-NP$. Prin urmare, dacă problema $\mathcal{C}$ de criptanaliză ar fi $NP-$completă, atunci am avea $NP = co-NP$. Pentru aceasta să considerăm un element $\mathcal{L}$ din NP. Deoarece C este $NP-$completă, $\mathcal{L}$ este reductibil în timp polinomial la $\mathcal{C}$. În consecinţă la fel şi complementul lui $\mathcal{L}$ este reductibil în timp polinomial la complementul lui $\mathcal{C}$, care este în $co-NP$, conform presupunerii noastre. Aceasta implică faptul că $\mathcal{L}$ este în $co-NP$ şi prin urmare, NP este inclus în $co-NP$. Din această implicaţie, incluziunea inversă este evidentă. Să luăm orice $\mathcal{L}$ din $co-NP$. Complementul lui $\mathcal{L}$ este în NP şi în consecinţă în $co-NP$. Aceasta implică faptul că $\mathcal{L}$ este în NP. Am arătat că dacă problema criptanalizei pentru criptosisteme cu chei publice este $NP-$completă, atunci $NP = co-NP$. Aceasta

implică faptul că este extrem de improbabil ca problema criptanalizei pentru criptosisteme cu chei publice să fie $NP-$completă sau cu un grad mare de complexitate. Pentru situaţia în care dispunem numai de cheie de cifrare problema criptanalizei este o problemă $NP-$completă. Este evident că nu poate fi dată nici o limită superioară pentru complexitatea criptanalitică a criptosistemelor clasice. În esenţă, acestea se datorează faptului că simplitatea criptării şi decriptării pentru utilizatorii legali nu conduce la nici o consecinţă în ce priveşte munca criptanalistului, deoarece toate elementele criptosistemului se păstrează în secret (de fapt în evaluarea unui sistem criptografic se presupun cunoscute toate aceste elemente, toată tăria sistemului de criptare trebuie să rămână în cheia de cifrare).

3.5. Limite actuale ale calculatoarelor

Datorită legii lui Moore care spune că la fiecare doi ani puterea de calcul se dublează, se va ajunge, cu aproximaţie, în anul 2015 la bariera siliciului fiind nevoie de noi suporturi pentru calcul.

Calculatoarele clasice de tip Turing-von Neumann sunt aproape de limita de perfecţionalitate şi au şi limite intrinseci legate de secvenţialitate. Din acest motiv, de mai mulţi ani se caută noi suporturi pentru calcul. Dintre acestea, două sunt direcţiile fundamentale de cercetare:

1. calculatoarele cuantice;

2. calculatoarele moleculare sau biologice: deorece molecula de ADN se comportă sintactic s-au putut realiza experimente de *calcul în eprubetă.*

În 1994 Adelman a realizat primul experiment de calcul în eprubetă de rezolvare a problemei drumului hamiltonian: dacă într-un graf există sau nu un drum hamiltonian între două noduri. Ulterior, calculul genetic şi-a găsit aplicaţii în criptografie: în 1995 algoritmul *DES* (*Data Encryption Standard*) a fost spart în trei luni, iar în 1996 în două luni. Rezultate similare s-au obţinut cu sistemul *RSA* (sistem cu chei publice elaborate de Rivest, Shamir şi Adelman).

3.6. Aplicaţii

Exerciţiul 3.6.1. Evaluaţi valorile $O(x_n)$ şi $o(x_n)$ unde x_n este numărul de operaţii aritmetice necesar calculului Transformatei Fourier Discrete pentru un şir binar de lungime n.

Exerciţiul 3.6.2. Daţi exemple de probleme din clasa P.

Exerciţiul 3.6.3. Daţi exemple de probleme din clasa NP.

Exerciţiul 3.6.4. Daţi exemplu de o problemă din clasa NP care nu este în clasa P.

Exerciţiul 3.6.5. Daţi exemplu de o problemă $NP-$ completă.

Exerciţiul 3.6.6. Problema criptanalizei este din clasa P, NP sau $NP-$ completă?

Exerciţiul 3.6.7. Daţi exemplu de o problemă din clasa $co-NP$ care nu este $NP-$ completă.

Capitolul 4

ANALIZA STATISTICO-INFORMAŢIONALĂ

Even in cryptology, silence is golden.
Laurence D. Smith

4.1. Noţiuni teoretice

Problema testării echipamentelor industriale (procese de fiabilitate), incluzând şi verificarea şi testarea echipamentelor electronice care sunt generatoare de aleatorism (generatorare pseudoaleatoare criptografice) sau a echipamentelor mecanice de aleatorism (jocuri din clasa casino de tip bingo sau ruletă), are ca rol principal luarea unei decizii de genul este sau nu echipamentul testat o sursă cu proprietăţile specificate de către producător. Modelul matematic considerat în studiu este acela al unui şir de variabile aleatoare $X_1, \ldots, X_n$ ce iau valori într-o mulţime de numere întregi bine specificată. Din acestă succesiune de numere întregi vom construi o succesiune de valori binare 0 şi 1. Bineînţeles că având un generator aleator binar de 0 şi 1 putem lesne construi un generator de numere întregi aleatoare între 0 şi $m = 2^\alpha$. Reciproca este evident adevărată m este o putere a lui 2. Prezentul capitol încearcă, printre altele, să acopere acest *spaţiu dintre puterile lui 2,* mai exact se încearcă o modalitate de testare statistică a generatoarelor pseudoaleatoare de numere întregi, cu numere cuprinse între două limite $n_{\min}$ şi $n_{\max}$ ce nu sunt puteri ale lui 2.

Prezentul capitol este organizat după cum urmează: se prezintă conceptele de generatoare pseudoaleatorii şi de test statistic, se prezintă schematic, sub forma de pseudocod, algoritmul de estimare a probabilităţii ideale a sursei testate. Mai mult

decât atât, este prezentată o formulă analitică pentru calculul acestei probabilităţi. Se vor prezenta, tot sub formă de pseudo-cod, testele frecvenţei, serial, autocorelaţiei circulare/temporale precum şi a autocorelaţiilor circulare/temporale. Ca o caracteristică a acestui capitol remarcăm caracterul formal al schemelor de testare prezentate sub formă algoritmică. Astfel, acestea pot fi foarte uşor implementate în diverse limbaje de programare lucru ce este exemplificat în ultimul paragraf al acestui capitol.

4.2. Generatoare şi teste statistice

4.2.1. Generatoare uniforme

Un generator aleatoriu de biţi este un mecanism a cărui ieşire este un şir binar de variabile aleatoare independente şi simetric distribuite, adică poate fi implementat de o sursă binară simetrică (**BSS**). Un generator pseudoaleatoriu este un algoritm care produce un şir binar care simulează o sursă binară simetrică (un exemplu de astfel de generator, cu proprietăţi bune din punct de vedere al aleatorismului, este de exemplu implementat de relaţia $X_n = X_{n-24} + X_{n-55} \bmod m$ unde m este un număr par, iar $X_0, \ldots, X_{54}$ sunt cuvinte pe m biţi nu toate pare). Aleatorismul este o proprietate a unui model matematic abstract caracterizat de un câmp de probabilitate. Un model probabilistic care să poată da o descriere exactă a realităţii este o problemă filozofică legată de întrebarea dacă universul este sau nu determinist. Pe de altă parte, există procese haotice naturale ca zgomotul termic într-un tranzistor ce permite construcţia unui generator aleatoriu de biţi, care pentru aplicaţiile practice este echivalent cu o sursă binară simetrică. În aplicaţiile practice, în procesul de proiectare al unui echipament electronic, care exploatează aceste proprietăţi haotice ale proceselor fizice, se impune să nu existe dependenţe între biţi sau deplasări ale acestora. Aleatorismul reprezintă pentru procesele matematice ceea ce este haosul pentru procesele fizice. Matematic, aleatorismul unei surse revine la:

Definiţia 4.2.1. Generatorul G se numeşte generator aleatoriu dacă eşantioanele $g_1, \ldots, g_n$ sunt variabile aleatoare independente şi:

$$P(G = 0) = P(G = 1) = 0,5.$$

Din motivele prezentate anterior, pentru un asemenea echipament, este esenţială testarea statistică extinsă după proiectare şi de asemenea periodic de-a lungul operaţiilor ce le efectuează.

4.2.2. Conceptul de test statistic

În acest paragraf discutăm din punct de vedere teoretic problema deciziei dacă ieşirea unei surse este statistic independentă şi uniform distribuită. Decizia trebuie

luată pe un şir eşantion de o anumită lungime N. Vom spune la momentul oportun ce volum trebuie să aibă eşantionul. Să notăm cu **B** mulţimea {0,1}. Un algoritm determinist care are ca intrare un şir binar şi ca ieşire o variabilă binară de decizie se numeşte test statistic şi poate fi privit ca o funcţie: $T : \mathbf{B}^N \rightarrow \{accept, resping\}$ care împarte mulţimea $\mathbf{B}^N$ a şirurilor $s^N = s_1, \ldots, s_N$ de lungime N într-o mulţime (de obicei mai mică) $S^T = \{s^N : T(s^N) = resping\} \subset \mathbf{B}^N$ a şirurilor proaste sau nealeatorii, complementara acesteia fiind mulţimea şirurilor bune sau aleatorii. Sunt numeroase definiţii date aleatorismului: Kolmogorov, de exemplu, defineşte acest concept ca lungimea celei mai scurte reguli de generare a şirului. Un şir este aleatoriu dacă regula amintită anterior este chiar şirul însuşi. Mai formal, aleatorismul (sau *complexitatea Kolmogorov*) unui şir este lungimea celui mai mic program Turing care generează şirul pentru o maşină Turing universală fixată. Martin-Lof a arătat că asimptotic un şir este aleatoriu dacă satisface această definiţie pentru toate testele statistice calculabile. O mică problemă apare la definiţia lui Kolmogorov: lungimea celui mai scurt program depinde de maşina Turing particular aleasă. Probleme mai mari apar la faptul că această complexitate Kolmogorov nu este calculabilă, chiar folosind un calculator cu putere de calcul infinită. Această problemă este echivalentă problemei opririi pentru maşini Turing.

Pentru orice model probabilistic cu parametri specificaţi (de exemplu, o sursă binară *fără memorie* care emite 1 cu probabilitatea $0,4$ şi 0 cu probabilitatea $0,6$), problema deciziei dacă şirul de la ieşire este în concordanţă cu modelul se rezolvă cu ajutorul testelor statistice. Pentru modelele parametrizate, testele statistice nu sunt optimale în ceea ce priveşte testarea ipotezelor. În primul rând chiar fixând o distribuţie pe anumite modele nu pot defini un criteriu de optimalitate globală. În al doilea rând, ca şi în cazul testării ipotezelor statistice, strategia optimală chiar pentru o alegere particulară a parametrilor poate fi greu de implementat. Din motivele arătate anterior multe teste statistice sunt euristice. Câteva teste (de exemplu testul frecvenţei şi testul serial) pot fi interpretate în modul următor: parametrii unui model statistic sunt estimaţi din şirul eşantion şi se construieşte o funcţie test bazată pe diferenţa dintre acest parametru şi parametrul unei **BSS**. Bazându-ne pe repartiţia funcţiei test pentru un şir cu adevărat aleatoriu şirul testat este respins sau acceptat. Principalele teste care se efectuează în statistică, cu aplicaţii în problematica testării cheilor de cifrare se referă la:

-compararea mediei unei selecţii cu valoarea mediei repartiţie teoretice din care se presupune că provine aceasta (cu testul t);

-compararea dispersiilor a două selecţii independente (cu testul F);

-compararea frecvenţelor;

-compararea repartiţiilor: se compară dacă o repartiţie empirică nu diferă semnificativ de repartiţia teoretică (se foloseşte testul χ^2 şi/sau testul Kolmogorov-

Smirnov);

-corelaţie care poate fi de mai multe tipuri şi anume: *Pearson* (măsoară legătura lineară între variabile), *Spearman* (este coeficientul de corelaţie al lui Pearson, dar calculat din ranguri), *Kendall-Tau* (măsoară probabilitatea ca datele să fie corelate din punctul de vedere al ordinii de apariţie).

Se mai poate efectua şi analiză de tip ANOVA, adică analiza dispersiei, care analizează diferenţele dintre mediile unor grupuri de date, diferenţiate pe baza unui factor a cărui influenţă asupra valorilor medii ale grupurilor dorim să o investigăm. Aceste teste se pot aplica pe biţi şi/sau bytes, sau alt parametru de codificare (hexa, word etc). În continuare, majoritatea testelor prezentate sunt la nivel logic binar.

4.2.3. Modele statistice pentru generatoare

Cel mai simplu model pentru un generator este sursa binară fără memorie(**BMS**) a cărei ieşire este un şir de variabile aleatoare independente şi uniform distribuite fiind caracterizată de un singur parametru: probabilitatea p de a emite 1. Acest model se va nota cu $\mathbf{BMS}_p$. Remarcăm faptul că $\mathbf{BMS}_{1/2}$ este echivalentă cu **BSS**. Un alt model simplu, notat cu $\mathbf{ST}_p$, emite 0 şi 1 cu aceeaşi probabilitate, dar probabilităţile de tranziţie sunt deplasate: o cifră binară este urmată de complementul ei cu probabilitatea p şi de aceeaşi cifră cu probabilitatea $1-p$. Aceasta este un exemplu de o sursă binară staţionară cu memorie de 1 bit. În general, distribuţia de probabilitate a bitului i de la ieşire poate să depindă de M ieşiri anterioare, unde M este memoria sursei. În multe aplicaţii practice se poate presupune că un generator defect sau slab criptografic este modelat de o asemenea sursă cu memorie relativ mică.

Considerăm o sursă $\mathbf{S}$ care emite şirul $U_1, U_2, U_3, \ldots$de variabile aleatoare binare. Dacă există un întreg pozitiv $M > 0$ astfel încât pentru orice $n > M$ probabilitatea lui U_n condiţionată de $U_1, U_2, \ldots, U_{n-1}$ depinde numai de ultimii M biţi de la ieşire adică dacă:

$$P_{U_n|U_{n-1}\ldots U_1}(u_n|u_{n-1}\ldots u_1) = P_{U_n|U_{n-1}\ldots U_{n-M}}(u_n|u_{n-1}\ldots u_{n-M}),$$

pentru un $n > M$ şi orice şir binar $(u_1, u_2, \ldots, u_n) \in \mathbf{B}^n$, atunci cel mai mic astfel de M se numeşte *memoria sursei* $\mathbf{S}$ şi prin $\mathbf{\Sigma}^n = [U_{n-1}, \ldots, U_{n-M}]$ se notează starea la momentul n. Fie $\mathbf{\Sigma}^1 = [U_0, \ldots, U_{-M+1}]$ *starea iniţială* unde $U_{-M+1}, \ldots, U_0$ sunt variabile aleatoare. Dacă în plus sursa satisface:

$$P_{U_n|\Sigma_n}(u|\sigma) = P_{U_1|\Sigma_1}(u|\sigma),$$

pentru orice $n > M$ şi orice $u \in \mathbf{B}$ şi $\sigma \in \mathbf{B}^M$ sursa se va numi *staţionară*.

Deci o sursă staţionară cu memoria M este complet determinată de distribuţia de probabilitate a stării iniţiale P_{Σ_1} şi distribuţia de probabilitate a tranziţiilor stărilor $P_{\Sigma_2|\Sigma_1}$. Şirul stărilor formează un *lanţ Markov* cu proprietatea suplimentară că fiecare din cele 2^M stări are cel mult două stări succesoare cu probabilitate nenulă. Vom identifica cele 2^M posibile stări ale lanţului Markov cu numerele întregi din intervalul $[0, 2^M - 1]$ ($\Sigma_n = j$ înseamnă că $U_{n-1}, \ldots, U_{n-M}$ este reprezentarea binară a lui j). Pentru clasa *lanţurilor Markov ergodice* care conţine toate clasele de interes practic există o distribuţie de tranziţie a stărilor invariantă $p_0, \ldots, p_{2^M-1}$ astfel încât:

$$\lim_{n\to\infty} P_{\Sigma_n}(j) = p_j \qquad \text{pentru} \quad 0 \le j \le 2^M - 1.$$

Mai mult probabilităţile p_j sunt soluţiile sistemului linear:

$$\begin{cases} \sum_{j=0}^{2^M-1} p_j = 1, \\ p_j = \sum_{k=0}^{2^M-1} P_{\Sigma_2|\Sigma_1}(j|k)p_k \qquad \text{pentru} \quad 0 \le j \le 2^M - 2. \end{cases}$$

4.2.4. Teste elementare de aleatorism statistic

Aşa cum am amintit, un test statistic T pentru şiruri de lungime N este o funcţie $T : \mathbf{B}^N \to \{accept \quad resping\}$ care împarte mulţimea $\mathbf{B}^N$ de şiruri binare de lungime N $s^N = s_1, \ldots, s_N$ într-o mulţime mai mică $\mathbf{S}_T = = \{s^N : T(s^N) = resping\} \subset \mathbf{B}^N$ a mulţimii şirurilor nealeatorii. Mulţimea complementară este mulţimea şirurilor aleatorii. Probabilitatea ca un şir generat de o sursă **BSS** să fie respins este:

$$\rho = \frac{card(S_T)}{2^N}$$

şi se numeşte *rata de respingere*. Aceasta reprezintă probabilitatea de a respinge un şir aleatoriu şi se numeşte *riscul de ordinul* 1 (sau riscul producătorului). În cazul în care testul statistic se construieşte pe baza intervalelor de estimare apare şi a doua probabilitate de eroare statistică şi anume: probabilitatea de a accepta o ipoteză falsă cunoscută sub numele de *riscul de ordinul* 2 (sau riscul beneficiarului). Ideal este ca ambele riscuri să fie minime ceea ce nu se poate. De aceea, pentru situaţiile practice vom stabili ρ şi vom evalua cel de-al doilea risc. Reamintim că testele de tip *Neyman-Pearson* minimizează riscul de ordinul 2 pentru un risc de ordinul 1 fixat(deci în acest caz nu mai trebuie să calculăm funcţia caracteristică operatoare a testului care ne dă riscul beneficiarului).

Un test statistic T, pentru o selecţie rezonabilă de lungime N, se implementează cu ajutorul unei funcţii f_T definită astfel: $f_T : \mathbf{B}^N \to \mathbf{R}$. Se determină distribuţia

de probabilitate a unei variabile aleatoare reale $f(R^N)$ unde R^N este un şir de N variabile binare independente şi simetric distribuite, limitele inferioară şi superioară t_1 şi t_2 astfel încât:

$$Pr[f_T(R^N) \leq t_1] + Pr[f_T(R^N) \geq t_2] = \rho.$$

De obicei, $Pr[f_T(R^N) \leq t_1] = Pr[f_T(R^N) \geq t_2] \approx \rho/2$ (în acest caz t_1 se numeşte *cuantila de ordinul* $\rho/2$, iar t_2 se numeşte *cuantila de ordinul* $1 - \rho/2$) ceea ce corespunde *testului bilateral.* Mulţimea şirurilor proaste S_T, cu cardinalul $\rho 2^N$ este definită de:

$$S_T = \{s^N \in \mathbf{B}^N : f_T(s^N) \leq t_1 \text{ sau } f_T(s^N) \geq t_2\}.$$

De cele mai multe ori, $f_T(R^N)$ are o distribuţie relativ cunoscută ca: normală, $\chi^2(n)$, $t(n)$, $F(n_1, n_2)$, $exp(\lambda)$. Distribuţia normală apare când un număr mare de variabile independente identic distribuite sunt însumate. Distribuţia $\chi^2(n)$ se obţine din însumarea pătratelor a n variabile independente distribuite gaussian de medie 0 şi dispersie 1.

4.2.5. Interpretarea rezultatelor testelor statistice

Operaţia de interpretare a rezultatelor este destul de laborioasă deoarece dacă executăm o baterie de teste, ca în cazul testării generatoarelor pseudoaleatoare criptografice, acestea pot să nu fie independente. Interpretarea unui singur test este mult mai simplă. Astfel, fie două ipoteze statistice şi anume:

$H_0 : s^N$ aleatoriu(nu prezintă componentă predictibilă);

$H_A : s^N$ nealeatoriu(prezintă cel puţin componentă predictibilă).

$1 - \alpha = \Pr(\text{accept } H_0 | H_0 \text{ adevărată})$− probabilitatea apariţiei unui *rezultat negativ* pentru o sursă aleatoare;

$\alpha = \Pr(\text{resping } H_0 | H_0 \text{ adevărată})$− probabilitatea apariţiei unui *rezultat fals pozitiv* pentru o sursă aleatoare;

$1 - \beta = \Pr(\text{resping } H_0 | H_0 \text{ falsă})$− probabilitatea apariţiei unui *rezultat pozitiv* pentru o sursă aleatoare;

$\beta = \Pr(\text{accept } H_0 | H_0 \text{ falsă})$− probabilitatea apariţiei unui rezultat *fals negativ* pentru o sursă aleatoare.

Decizia referitoare la apartenenţa la una din cele două ipoteze se face pe baza unei funcţii de test f calculată dintr-un eşantion $X_1, ..., X_n$ de volum n. Ideal este ca ambele tipuri de erori să fie minime. Din păcate, acest lucru nu este posibil. Testele de tip *Neymann-Pearson* minimizează riscul de ordinul 2 pentru un risc de ordinul 1 fixat, iar testele bazate pe intervale de încredere fixăm α şi calculăm pe β beneficiarului.

Valoarea P este cel mai mic prag de semnificaţie (risc de ordinul 1) pentru care se respinge ipoteza nulă H_0 (pe baza valorilor statisticii $f(X_1, ..., X_n)$) deci:

$$P \leq \alpha \qquad \text{resping } H_0 \text{ la pragul } \alpha,$$

şi

$$\alpha < P \qquad \text{accept } H_0 \text{ la pragul } \alpha.$$

Observaţia 4.2.1. Valoarea P asociată unei statistici $f(X_1, ..., X_n)$ se obţine din ecuaţia $P = Pr(\text{resping } H_0 | \text{ regiunea de respingere pentru } H_0 \text{ dată de valoarea } f)$.

4.3. Entropia variabilelor aleatoare discrete

Fie X o variabilă aleatoare discretă care ia valori într-o mulţime finită sau numărabilă, dar infinită. Variabila aleatoare X este modelată de distribuţia de probabilitate $P_X : \mathcal{X} \to \mathbf{R}$ care satisface:

$$\sum_{x \in \mathcal{X}} P_X(x) = 1.$$

Definiţia 4.3.1. Variabilele aleatoare $X_1, \ldots, X_n$ sunt *statistic independente* dacă:

$$P_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = P_{X_1}(x_1) \ldots P_{X_n}(x_n).$$

Definiţia 4.3.2. *Entropia Shannon* sau *incertitudinea* variabilei aleatoare X este definită de:

$$H(X) = - \sum_{x \in \mathcal{X}: P_X(x) \neq 0} P_X(x) \log_2 P_X(x).$$

Dacă X este finită atunci: $0 \leq H(X) \leq \log_2 |\mathcal{X}|$, unde prin $|\mathcal{X}|$ am notat cardinalul mulţimii $\mathcal{X}$. Avem egalitatea:

$$H(X) = 0 \text{ dacă şi numai dacă } (\exists)\ x \in \mathcal{X} \text{ astfel încât } P_X(x) = 1.$$

ceea ce corespunde cazului *dezordine minimă* (caz determinist) şi egalitatea:

$$H(X) = \log_2 |\mathcal{X}| \text{ dacă şi numai dacă } (\forall)\ x \in \mathcal{X} \text{ avem } P_X(x) = \frac{1}{|\mathcal{X}|}.$$

ceea ce corespunde cazului *dezordine maximă.*

Definiţia 4.3.3. *Entropia Renyi de ordinul* α care este definită ca:

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \sum_{x \in \mathcal{X}} P_X^\alpha(x).$$

În cazul limită $\alpha \to 1$ obţinem $H(x) = \lim_{\alpha \to 1} H_\alpha(X)$ (*entropia Shannon*), pentru $\alpha \to \infty$ obţinem *min-entropia* $H_\infty(X) = -\log_2 \max_{x \in \mathcal{X}} P_X(x)$ iar pentru $\alpha = 2$ obţinem *entropia de coliziune.* Avem de asemenea următoarele inegalităţi:

$$\log_2 |\mathcal{X}| \geq H(X) \geq H_2(X) \geq H_\infty(X)$$

şi

$$0 < \alpha < \beta \text{ atunci } H_\alpha(X) \geq H_\beta(X)$$

cu egalitate dacă şi numai dacă X este uniform distribuită peste $\mathcal{X}$ sau o submulţime a lui $\mathcal{X}$.

Definiţia 4.3.4. *Redundanţa* variabilei aleatoare finite X este definită prin formula: $R(X) = \log_2 |\mathcal{X}| - H(X)$.

Entropia variabilelor $X_1, \ldots, X_n$ este definită ca o extensie a definiţiei anterioare şi anume:

$$H(X_1, \ldots, X_n) = -\sum P_{X_1,\ldots,X_n}(x_1, \ldots, x_n) \log_2 P_{X_1,\ldots,X_n}(x_1, \ldots, x_n).$$

unde sumarea se face după acele probabilităţi nenule.

Definiţia 4.3.5. *Redundanţa* variabilelor aleatoare finite $X_1, \ldots, X_n$ este definită prin formula: $R_n(X_1, \ldots, X_n) = n \cdot \log_2 |\mathcal{X}| - H(X_1, \ldots, X_n)$.

Definiţia 4.3.6. *Distanţa de unicitate* a unui sistem criptografic cu spaţiul cheiii $\mathcal{K}$ şi spaţiul textelor clar $\mathcal{M}$ este definită prin:

$$\min\{n \in N^* | R_n \geq H(K)\}$$

unde $H(K)$ reprezintă entropia cheii iar R_n redundanţa textului clar.

Observaţia 4.3.1. Distanţa de unicitate este un indicator *teoretic şi ideal* pentru a efectua schimbarea cheii (în momentul în care redundanţa textului clar este mai mare ca entropia cheii criptanalistul are suficiente resurse pentru a determina, în cazul în care dispune de *capacitate de stocare* şi *putere de calcul nelimitată*, textul clar din criptogramă).

Observaţia 4.3.2. În cazul în care $\mathcal{X} = \{0, 1\}$ iar cheia este pe k biţi şi toate acestea sunt echiprobabile atunci distanţa de unicitate este k. Sistemul de cifrare care distanţa de unicitate infinit este *sistemul cu cheie de unică folosinţă* (OTP-*one time pad*) care constă în sumarea modulo 2 a textului clar cu o secvenţă cheie generată, prin mecanisme hardware, în mod aleator (nu pseudoaleator).

Definiţia 4.3.7. *Entropia condiţionată* a variabilei aleatoare X dată fiind varabila aleatoare Y este definită ca:

$$H(X|Y) = -\sum_{(x,y)} P_{X|Y}(x, y) \log_2 P_{X|Y}(x, y),$$

unde $P_{X|Y}(x, y)$ este probabilitatea condiţionată ca X să ia valoarea x dacă se observă $Y = y$.

Se poate arăta că:

$$0 \leq H(X|Y) \leq H(X),$$

avem $H(X|Y) = 0$ dacă şi numai dacă Y *determină în mod unic* pe X şi $H(X|Y) = H(X)$ dacă şi numai dacă X şi Y sunt s*tatistic independente*. O regulă importantă de transformare a nedeterminărilor este *regula lanţului*:

$$H(X_1, \ldots, X_n) = H(X_1) + H(X_2|X_1) + \ldots + H(X_n|X_1, \ldots, X_{n-1}),$$

care pentru entropia condiţionată are forma:

$$H(X_1, \ldots, X_n|Y) = H(X_1|Y) + H(X_2|X_1, Y) + \ldots + H(X_n|X_1, \ldots, X_{n-1}, Y).$$

Definiţia 4.3.8. *Informaţia mutuală* între variabilele aleatoare X şi Y este definită ca:

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y; X),$$

şi este o *măsură a reducerii nedeterminării* lui X dacă se observă Y.

Similar, putem defini *informaţia mutuală condiţionată* dintre X şi Y dacă se dă Z ca:

$$I(X; Y|Z) = H(X|Z) - H(X|Y, Z).$$

Să arătăm că pentru $|\mathcal{X}| = n$ entropia $H(X)$ este maximă dacă şi numai dacă $p_i = \dfrac{1}{n}, i = 1, \ldots, n$.

Problema de optimizare este:

$$\begin{cases} \max H(X) \\ \sum\limits_{i=1}^{n} p_i = 1 \\ p_i \geq 0. \end{cases}$$

Lagrangeianul $L = -\sum\limits_{i=1}^{n} p_i \log_2 p_i - \lambda(\sum\limits_{i=1}^{n} p_i - 1)$.

Rezolvarea sistemului:

$$\begin{cases} \dfrac{\partial L}{\partial p_i} = 0, \ i = \overline{1,n} \\ \dfrac{\partial L}{\partial \lambda} = 0 \end{cases}$$

conduce la $p_i = \dfrac{1}{n}, \ i = \overline{1,n}$, adică distribuţia uniformă.

4.4. Surse de aleatorism de numere intregi

4.4.1. Formula analitică a probabilităţii ideale a unei surse de aleatorism de numere întregi

Fie X o variabilă aleatoare ce ia valori întregi în intervalul de numere întregi $\{n_{\min}, \ldots, n_{\max}\}$. Şirul realizărilor $X_1, \ldots, X_n$, care este un şir de numere întregi va fi în prealabil transformat în şirul binar ataşat. Ideal ar fi să găsim o formulă pentru calculul probabilităţii p a simbolului 1 şi pentru $q = 1 - p$ probabilitatea apariţiei simbolului 0. Astfel, dacă notăm cu N_t numărul total de simboluri şi cu N_1 numărul de simboluri de 1 emise de sursă obţinem (făcând formal notaţia $0 := \log_2 0$):

$$p = \frac{N_1}{N_t} = \frac{\sum\limits_{i=n_{\min}}^{n_{\max}} w(bin(i))}{\sum\limits_{i=n_{\min}}^{n_{\max}} ([\log_2 i] + 1)},$$

deci avem:

$$p = \frac{\sum\limits_{i=n_{\min}}^{n_{\max}} w(bin(i))}{\sum\limits_{i=n_{\min}}^{n_{\max}} [\log_2 i] + (n_{\max} - n_{\min})},$$

unde prin $w(.)$ s-a notat *operatorul pondere* $w : B^n \to \mathbf{N}$, iar prin $bin(.)$ funcţia ce transformă un caracter întreg în binar, $bin : \mathbf{N} \to B^n$. Pentru ca formula să

fie operaţională trebuie găsită o expresie pentru compunerea $w(bin(.)) : \mathbf{N} \to \mathbf{N}$. Aceasta este dată de:

$$w(bin(i)) = \sum_{j=0}^{[\log_2 i]} [\frac{i}{2^j}] \bmod 2,$$

(cu aceeaşi convenţie ca mai sus $0 := \log_2 0$). Deci avem următoarea formulă pentru p :

$$p = \frac{\sum\limits_{i=n_{\min}}^{n_{\max}} \sum\limits_{j=0}^{[\log_2 i]} [\frac{i}{2^j}] \bmod 2}{\sum\limits_{i=n_{\min}}^{n_{\max}} [\log_2 i] + (n_{\max} - n_{\min})}.$$

O problemă pe care o avem în vedere este aceea a comportamentului asimptotic al probabilităţii anterioare. Astfel, trebuie să vedem cum evoluează $p = p(n_{\max}, n_{\min})$ când:

i) diferenţa $n_{\max} - n_{\min} \to \infty$;

ii) $n_{\max} \to \infty$.

Metoda de mai jos este o metodă computaţională de calcul a probabilităţii p ce poate fi folosită ca o alternativă la formula de mai sus.

4.4.2. Metoda de calcul efectiv al lui p respectiv q

Următorul algoritm, prezentat în pseudocod, furnizează la ieşire valoarea probabilităţii p a unei surse de numere întregi.

PAS 0. Se citeşte valoarea minimă $n_{\min}$ şi valoarea maximă $n_{\max}$ a realizărilor posibile.

Se iniţializează lungimea reprezentării binare $L = 0$;

Se iniţializează ponderea (numărul de 1) $W = 0$;

PAS 1. Pentru $i = n_{\min}$ până la $n_{\max}$ execută:

se converteşte numărul i în binar.

se află ponderea $w(i)$ corespunzătoare numărului i precum şi $l(i)$ lungimea sa în biţi.

$W = W + w(i); \quad L = L + l(i);$

PAS 2. Se estimează probabilitatea lui 1 prin formula: $p = \dfrac{W}{L}$ şi probabilitatea lui 0 prin relaţia: $q = 1 - \dfrac{W}{L}$;

4.5. Metode de decorelare

Metodele de decorelare au ca scop eliminarea corelaţiilor binare dintr-o secvenţă aleatoare binară care prezintă o descentrare a $Pr(X = 0) \neq 0,5$. Aceste metode se aplică, de regulă, generatoarelor hardware de secvenţe aleatoare. Metodele pot fi deterministe (se poate recupera -parţial sau în totalitate- secvenţa de intrare cunoscând ieşirea) sau nedeterministe (nu se poate recupera informaţia de intrare).

4.5.1. Metode deterministe

Diferenţierea unei secvenţe binare

Fie (X_n) un şir de variabile aleatoare independente cu $Pr(X_n = 0) = \frac{1}{2} + p$ cu $p < 0,5$. Atunci şirul $Z_i = X_i \oplus X_{i-1}$, obţinut prin operaţia de diferenţiere, are $Pr(Z_n = 0) = \frac{1}{2} + 2p^2$, probabilitatea acestuia fiind mai apropiată de $0,5$ decât probabilitatea lui X_n. Evident din secvenţa Z se poate recupera secvenţa X, în concluzie tehnica nu este indicată a se folosi la aplicaţii criptografice.

Sumarea a două sau mai multe secvenţe independente

O variantă a metodei prezentate mai sus se aplică pentru obţinerea unei secvenţe mai bune (din punct de vedere al probabilităţii) din două secvenţe binare aleatoare independente (X_n) şi (Y_n) cu $Pr(X = 0) = p_1$ respectiv $Pr(Y = 0) = p_2$ (presupunem $p_1, p_2 > 0,5$). Atunci:

$$\Pr(X \oplus Y = 0) = 1 + 2p_1p_2 - p_1 - p_2 < \min\{p_1, p_2\}.$$

Metoda se aplică, de exemplu, la construcţia generatorarelor hardware care au în componenţă două sau mai multe generatoare de secvenţe binare independente dar descentrate de la uniformitate $(p_1, p_2 \neq 0,5)$.

4.5.2. Metode nedeterministe

Vom prezenta metoda de decorelare a lui *Von Neumann* care este bazată pe ideea decimării secvenţei generate. Astfel secvenţele de tipul 00 şi 11 sunt ignorate pe când secvenţele de tipul 01 şi 10 sunt codificate în 1 respectiv 0. Din secvenţa de ieşire, care prezintă o corelaţie binară mai mică decât secvenţa de intrare, nu se poate recupera şirul binar iniţial. Procedeul se poate aplica recursiv.

4.6. Teste statistice de aleatorism

O descriere exhaustivă a testelor statistice ce se vor prezenta se poate găsi şi în [107]. Vom prezenta o serie de teste de aleatorism statistic. Acestea se bazează pe construcţia unor funcţii de test, distribuţia de referinţă fiind distribuţia normală sau chi-pătrat. Testele presupun un volum suficient de mare al eşantionului. Pentru volum mic al eşantionului se aplică alte teste în care distribuţia de referinţă este Bernoulli, binomială sau Poisson. Aceste teste (care presupun calcul în discret) sunt mai dificil de implementat decât prima categorie de teste (care presupun manipularea de funcţii continue). *Eroarea de decizie poate fi facută oricât de mică* dacă volumul eşantionului tinde la infinit.

4.6.1. Algoritmul de implementare al testului frecvenţei

Descrierea testului: Acest test este un test fundamental şi se bazează pe distribuţia numărului de 1 din şirul testat. Statistica testului este de tip gaussian (vezi pasul 2 din algoritmul anterior). Testul se poate executa la diverse rate de respingere şi pentru diverse valori ale parametrului p_0, probabilitate care indică valoarea ideală a probabilităţii de apariţie a simbolului 1.

Scopul testului: Testul pune în evidenţă o abatere semnificativă a probabilităţii (reale) de simboluri de 1 faţă de probabilitatea teoretică (ideală).

Pseudocodul este prezentat în continuare (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Probablilitatea p_0 de apariţie a simbolului 1 (dacă $p_0 = 0,5$, atunci avem o sursă binară simetrică). Această probabilitate se poate calcula utilizând algoritmul descris anterior sau formula analitică corespunzătoare.

Notăm $q_0 = 1 - p_0 = \Pr(S = 0)$ probabilitatea de apariţie a simbolului 0.

Ieşire: Decizia de acceptare sau respingere a aleatorismului, adică şirul s^N este realizarea unei surse staţionare cu $\Pr(S = 1) = p_0$, ipoteza alternativă fiind $\Pr(S = 1) = p_1 \neq p_0$.

PAS 0. Citeşte şirul s^N, rata de respingere α.

PAS 1. Calculează funcţia de test:

$$f_{T_F}(s^N) = \frac{1}{\sqrt{N * p_0 * q_0}}(\sum_{i=1}^{N} s_i - N * p_0).$$

PAS 2. Dacă $f_{T_F}(s^N) \in [u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}]$ ($u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$ sunt cuantilele repartiţiei normale de ordinul $\alpha/2$ respectiv $1 - \alpha/2$) se acceptă ipoteza aleatorismului, iar în caz contrar se respinge ipoteza aleatorismului.

PAS 3. Calculează riscul de ordinul 2 (probabilitatea de a accepta o ipoteză falsă):

$$\beta = 1 + \Phi((u_{\frac{\alpha}{2}} - \frac{N(p_1 - p_0)}{\sqrt{Np_0q_0}})\sqrt{\frac{p_0q_0}{p_1q_1}}) - \Phi((u_{1-\frac{\alpha}{2}} - \frac{N(p_1 - p_0)}{\sqrt{Np_0q_0}})\sqrt{\frac{p_0q_0}{p_1q_1}}),$$

unde $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int\limits_{-\infty}^{z} e^{-\frac{x^2}{2}} dx$ este *funcţia lui Laplace.*

4.6.2. Algoritmul de implementare al testului serial

Descrierea testului: Testul serial verifică dacă distribuţia la nivel de element de codificare (bit -test echivalent cu *testul frecvenţei*, dublet -test numit şi *testul frecvenţei dubleţilor*, caracter hexa -*testul frecvenţelor caracterelor hexa*, byte -*testul frecvenţei bytes*, sau *word* etc) a elementelor şirului este uniformă. Statistica testului este de tip χ^2, numărul de grade de libertate al statisticii depinzând de nivelul de codificare. Testul se poate executa la diverse rate de respingere.

Scopul testului: Acest test este de fapt o clasă de teste, fiecare punând în evidenţă abateri de la distribuţia de probabilitate ideală a numărului de simboluri (bit, dublet, hexa, byte, word sau altă dimensiune a cuvântului).

Pseudocodul este prezentat în continuare (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Şirul probablilităţilor $\mathbf{p}^{(0)}$ de apariţie a simbolului i (dacă $p_i^{(0)} = \dfrac{1}{2^L}$, atunci avem o sursă binară simetrică). Acest vector se estimează prin generarea succesiuni binare corespunzătoare valorii minime $n_{\min}$ şi valorii maxime $n_{\max}$ a realizărilor posibile şi aflarea cardinalului tuturor tăieturilor de lungime L din această succesiune.

Ieşire: Decizia de acceptare sau respingere a aleatorismului, adică şirul s^N este realizarea unei surse staţionare cu $\Pr(S = i) = p_i^{(0)}$.

PAS 0. Citeşte şirul s^N, rata de respingere α, dimensiunea blocului de procesat L.

PAS 1. Şirul eşantion s^N este împărţit în $\dfrac{N}{L}$ blocuri consecutive de lungime L (pentru $L = 2$ testul este echivalent cu testul frecvenţei pe dubleţi, pentru $L = 8$ testul este echivalent cu testul frecvenţei pe octeţi etc; alţi parametri uzuali de codificare sunt $L = 4$ (hexa) şi $L = 16$ (word)) şi se determină numărul $n_i(s^N)$ al apariţiilor reprezentării binare a numărului i pentru $0 \leq i \leq 2^L - 1$.

PAS 2. Calculează funcţia de test:

$$f_{T_S}(s^N) = \sum_{i=0}^{2^L-1} \frac{(n_i(s^N) - \frac{N}{L}p_i^{(0)})^2}{\frac{N}{L}p_i^{(0)}}.$$

PAS 3. Dacă $f_{T_S}(s^N) \in [0; \chi^2_{1-\alpha}(2^L - 1)]$ ($\chi^2_{1-\alpha}(2^L - 1)$ este cuantila de ordinul $1 - \alpha$ a distribuţiei $\chi^2(2^L - 1)$) se acceptă ipoteza aleatorismului, iar în caz contrar respinge ipoteza aleatorismului.

Observaţia 4.6.1. O valoare prea mică a lui $f_{T_S}(s^N)$ (apropiată 0) poate fi considerată *suspectă,* în concluzie este indicat de a utiliza un test bilateral în care regiunea de decizie este

$$[\chi^2_{\frac{\alpha}{2}}(2^L - 1); \chi^2_{1-\frac{\alpha}{2}}(2^L - 1)]$$

($\chi^2_{\frac{\alpha}{2}}(2^L-1)$ şi $\chi^2_{1-\frac{\alpha}{2}}(2^L-1)$ fiind cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale distribuţiei $\chi^2(2^L - 1)$).

4.6.3. Algoritmul de implementare al testului succesiunilor

Descrierea testului: Prezentul test se bazează pe distribuţia numărului de succesiuni de 0 şi de 1 de o anumită lungime. Statistica testului este de tip χ^2. Testul se poate executa la diverse rate de respingere.

Scopul testului: Testul succesiunilor [1] (runs test în literatura de specialitate) pune în evidenţă abateri de la uniformitatea succesiunilor de 0 şi de 1.

Pseudocodul este prezentat mai jos (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Ieşire: Decizia de acceptare sau respingere a aleatorismului, adică şirul s^N este realizarea unei surse staţionare binare simetrice cu $\Pr(S = 1) = 0,5$.

PAS 0. Citeşte şirul s^N, rata de respingere α, dimensiunea maximă a succesiunilor L.

PAS 1. În *testul succesiunilor* cu parametrul L se determină numărul $n_i^0(s^N)$ al succesiunilor de 0 de lungime i şi $n_i^1(s^N)$ al succesiunilor de 1 de lungime i pentru $1 \leq i \leq L$(e.g. $L = 15$).

PAS 2. Calculează funcţia de test:

$$f_{T_R}(s^N) = \sum_{b \in \{0,1\}} \sum_{i=1}^{L} \frac{(n_i^b(s^N) - \frac{N}{2^{i+2}})^2}{\frac{N}{2^{i+2}}}.$$

PAS 3. Dacă $f_{T_R}(s^N) \in [0; \chi^2_{1-\alpha}(2L)]$ ($\chi^2_{1-\alpha}(2L)$ este cuantila de ordinul $1 - \alpha$ a distribuţiei $\chi^2(2L)$) se acceptă ipoteza aleatorismului, iar în caz contrar respinge, ipoteza aleatorismului.

[1] O succesiune de 1, de lungime k este un şir de 1 de lungime $k + 2$ ce are ca prefix şi sufix 0. Analog se defineşte succesiunea de 0.

Observaţia 4.6.2. O valoare prea mică a lui $f_{T_R}(s^N)$ (apropiată 0) poate fi considerată *suspectă,* în concluzie este indicat de a utiliza un test bilateral în care regiunea de decizie este

$$[\chi^2_{\frac{\alpha}{2}}(2L); \chi^2_{1-\frac{\alpha}{2}}(2L)]$$

($\chi^2_{\frac{\alpha}{2}}(2L)$ şi $\chi^2_{1-\frac{\alpha}{2}}(2L)$ fiind cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale distribuţiei $\chi^2(2L)$).

4.6.4. Algoritmul de implementare al testului autocorelaţiei temporale

Descrierea testului: Testul autocorelaţiei temporale este echivalent într-o oarecare formă cu testul frecvenţei: astfel, dacă S este o variabilă aleatoare binară cu $p_0 = \Pr(S = 1)$, atunci testul frecvenţei se va aplica cu parametru de probabilitate $2p_0q_0$ unui şir diferenţiat la distanţa bine specificată (pasul 1). De regulă, acest test se aplică pentru orice valoare a parametrului de codificare ($d = 8, 16$, etc). Testul se poate executa la diverse rate de respingere.

Scopul testului: Se verifică ipoteza conform căreia nu sunt corelaţii de tip cuvânt shiftat.

Pseudocodul este prezentat în continuare (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Ieşire: Decizia de acceptare sau respingere a necorelaţiei temporale la o distanţă fixată apriori.

PAS 0. Citeşte şirul s^N, rata de respingere α, distanţa de autocorelare temporală d.

PAS 1. Aplică testul frecvenţei, cu parametru de probabilitate $2p_0q_0$, şirului binar de lungime $N - d$:

$$s_1 \oplus s_{1+d}, s_2 \oplus s_{2+d}, \ldots, s_{N-d} \oplus s_N,$$

unde prin $\oplus$ s-a notat suma modulo 2.

PAS 2. Dacă şirul trece testul frecvenţei, atunci decidem ipoteza necorelării temporale a şirului $s^N = s_1, \ldots, s_N$, la distanţa d şi la riscul de ordinul 1 α (deci a aleatorismului şirului), iar în caz contrar respingem ipoteza necorelării temporale la distanţa d.

4.6.5. Algoritmul de implementare al testului autocorelaţiilor temporale

Descrierea testului: Acest algoritm decide dacă un şir prezintă sau nu autocorelaţii temporale. Este o compunere a testelor de autocorelaţie temporale şi se poate

executa la diverse rate de respingere.

Scopul testului: Se verifică ipoteza conform căreia nu sunt corelaţii de tip cuvinte shiftate.

Pseudocodul este prezentat în continuare (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Ieşire: Decizia de acceptare sau respingere a necorelării temporale.

PAS 0. Citeşte şirul s^N, rata de respingere α, distanţa maximă de autocorelare temporală d.

PAS 1. $nc = 0$;

For $i = 1$ to d execută:

aplică testul autocorelaţiilor la distanţa i;

dacă nu se trece testul $nc = nc + 1$;

EndFor.

PAS 2. Dacă $nc < \alpha d$, atunci şirul trece testul autocorelaţiilor până la distanţa d şi la riscul de ordinul 1 α, iar în caz contrar respingem ipoteza necorelării temporale.

4.6.6. Algoritmul de implementare al testului autocorelaţiei circulare

Descrierea testului: Este un test care răspunde la problema existenţei autocorelaţiei circulare în şirul testat. Deosebirea dintre testul autocorelaţiei temporale şi testul autocorelaţiei circulare este aceea că şirul procesat este rotit şi nu shiftat. Testul se poate executa la diverse rate de respingere.

Scopul testului: Se verifică ipoteza conform căreia nu sunt corelaţii de tip cuvânt rotit.

Pseudocodul este prezentat în continuare (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Ieşire: Decizia de acceptare sau respingere a necorelaţiei circulare la o distanţă fixată apriori.

PAS 0. Citeşte şirul s^N, rata de respingere α, distanţa de autocorelare circulară d.

PAS 1. Aplică testul frecvenţei, cu parametru de probabilitate $2p_0q_0$, şirului binar de lungime N :

$$s_1 \oplus s_{1+d}, s_2 \oplus s_{2+d}, \ldots, s_i \oplus s_{(i+d) \bmod N}, \ldots, s_N \oplus s_d,$$

unde prin $\oplus$ s-a notat suma modulo 2.

PAS 2. Dacă şirul trece testul frecvenţei decidem ipoteza necorelării circulare a şirului $s^N = s_1, \ldots, s_N$ la distanţa d şi la riscul de ordinul 1 α (deci a aleatorismului şirului), iar în caz contrar respingem ipoteza necorelării temporale la distanţa d.

4.6.7. Algoritmul de implementare al testului autocorelaţiilor circulare

Descrierea testului: Este un test care răspunde la problema existenţei autocorelaţiilor circulare în şirul testat.

Scopul testului: Se verifică ipoteza conform căreia nu sunt corelaţii de tip cuvânt rotit.

Pseudocodul este prezentat în continuare (se specifică intrările şi ieşirile algoritmului).

Intrare: Succesiunea binară $s^N = s_1, \ldots, s_N$.

Ieşire: Decizia de acceptare sau respingere a necorelării circulare.

PAS 0. Citeşte şirul s^N, rata de respingere α, distanţa maximă de autocorelare circulară d.

PAS 1. $nc = 0$;

For $i = 1$ to d execută:

aplică testul autocorelaţiilor la distanţa i;

dacă nu se trece testul $nc = nc + 1$;

EndFor.

PAS 2. Dacă $nc < \alpha d$, atunci şirul trece testul autocorelaţiilor, până la distanţa d şi la riscul de ordinul 1 α, iar în caz contrar respingem ipoteza necorelării temporale.

4.6.8. Algoritmul de implementare al testului poker

Descrierea testului: Testul poker clasic consideră n grupe de câte cinci întregi succesivi $\{Y_{5n}, Y_{5n+1}, Y_{5n+2}, Y_{5n+3}, Y_{5n+4}\}$ pentru $0 \leq j \leq n$ şi urmăreşte care dintre următoarele modele se potriveşte fiecăruia dintre cvintupluri sau patternuri:

Toate diferite: *abcde*

O pereche: *aabcd*

Două perechi: *aabbc*

Trei bucăţi: *aaabc*

Full: *aaabb*

Careu: *aaaab*

Cinci la fel: *aaaaa*

Scopul testului: Acest test detectează abateri semnificative de la statistica ideală a cvintuplurilor anterioare. Statistica testului este de tip χ^2.

Schema de implementare eficientă este prezentată în continuare.

Pentru a uşura munca de programare necesară putem să numărăm valorile distincte din mulţimea de cinci valori. Am avea atunci cinci categorii:

5 valori = toate diferite;

4 valori = o pereche;

3 valori = două perechi sau trei bucăţi;

2 valori = full sau careu;

1 valoare = cinci la fel.

Acest compromis conduce la o determinare statistică mai uşoară, iar testul este aproape la fel de bun.

Statistica testului partiţiei este de $\chi^2(k-1)$ şi cu probabilitatea pentru i valori diferite:

$$p_i = \frac{d(d-1)\dots(d-r+1)}{d^k} \left\{ {k \atop i} \right\} \text{ pentru } i = 1, \dots, k,$$

unde formula probabilităţii este dată pentru cazul general în care am considerat n grupe de câte k numere succesive, iar d este numărul maxim de valori din şirul testat ($d = 2$ codificare binară, $d = 8$ codificare la nivel de octet etc.). Numerele $\left\{ {k \atop i} \right\}$ reprezintă numărul de moduri în care se poate partiţiona o submulţime de k elemente în i submulţimi disjuncte nevide (sau clase). Aceste numere se numesc *numerele lui Stirling de speţa a doua* şi se pot calcula recursiv după formula următoare:

$$\begin{cases} \left\{ {0 \atop 0} \right\} = 1, \\ \left\{ {n \atop 1} \right\} = 1 \text{ pentru } n > 0, \\ \left\{ {n \atop 0} \right\} = 0 \text{ pentru } n > 0, \\ \left\{ {0 \atop m} \right\} = 0 \text{ pentru } m > 0, \\ \left\{ {n \atop m} \right\} = m \left\{ {n-1 \atop m} \right\} + \left\{ {n-1 \atop m-1} \right\} \text{ pentru } m > 1 \text{şi } n \geq 1. \end{cases}$$

Intrare: O succesiune $s^N = s_1, \dots, s_N$ cu $s_i \in \{0, 1, \dots, d-1\}$.

Ieşire: Decizia de acceptare sau respingere a aleatorismului.

PAS 0. Citeşte următoarele elemente:

• succesiunea de testat s^N ;

• rata de respingere a testului α;

• parametrul d (sau se extrage din succesiunea s^N) reprezintă valorile pe care le poate lua un element al succesiunii de la intrare (parametrul d se poate seta, de exemplu implicit 8 şi atunci spunem că efectuăm un test la nivel de byte sau cod *ASCII* extins: numere între 0 şi 255);

• lungimea patternului notată prin k (de exemplu, aceasta poate fi setată la 5, iar în acest caz numerele lui Stirling sunt $1, 15, 25, 10$ respectiv 1);

PAS 1. For $i = 1$ to k calculează (după formula anterioară) p_i probabilitatea ideală ca într-o succesiune de lungime k să avem i elemente distincte.

PAS 2. For $i = 0$ to k do $a[i] = 0$;

PAS 3. For $i = 0$ to $\frac{N}{k}$ do if al $i-$lea grup de k elemente consecutive are j elemente distincte atunci $a[j] = a[j] + 1$;

PAS 4. Calculează statistica testului:

$$\chi^2 = \sum_{i=1}^{k} \frac{(a[i] - \frac{N}{k}p_i)^2}{\frac{N}{k}p_i}.$$

PAS 5. Dacă $\chi^2 < \chi^2_{1-\alpha}(k-1)$ (cuantila de ordinul $1-\alpha$ a distribuţiei $\chi^2(k-1)$), atunci se admite ipoteza aleatorismului succesiunii la riscul α.

Observaţia 4.6.3. O valoare prea mică a lui χ^2 (apropiată 0) poate fi considerată *suspectă*, în concluzie este indicat de a utiliza un test bilateral în care regiunea de decizie este

$$[\chi^2_{\frac{\alpha}{2}}(k-1); \chi^2_{1-\frac{\alpha}{2}}(k-1)]$$

($\chi^2_{\frac{\alpha}{2}}(k-1)$ şi $\chi^2_{1-\frac{\alpha}{2}}(k-1)$ fiind cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale distribuţiei $\chi^2(k-1)$).

4.6.9. Algoritmul de implementare al testului $CUSUM$ (sumelor cumulate)

Descrierea testului: Testul *CUSUM* constă în determinarea drumului aleator maxim (începând cu zero) definit de sumele cumulative parţiale ale cifrelor de -1 şi $+1$ din secvenţa binară testată. Testul determină dacă sumele cumulate ale secvenţelor parţiale sunt prea mari sau prea mici relativ la valoarea aşteptată a sumelor cumulate pentru secvenţele aleatoare. Această sumă cumulată poate fi considerată ca un drum aleator. Pentru o secvenţă aleatoare, valoarea unui drum aleator trebuie să fie aproape de zero. Pentru un anume tip de secvenţe nealeatoare drumul până la zero trebuie să fie mare.

Scopul testului: Testul este bazat pe valoarea maximă absolută a sumelor parţiale ale secvenţei reprezentată în forma ± 1. Valorile mari ale acestei statistici indică faptul ca sunt prea multi biţi de unu sau prea mulţi biţi de zero în faza de început a secvenţei. Valorile mici arată că biţii de unu şi de zero sunt mixaţi foarte des.

Fie $s : s_1, \ldots, s_k$ un şir binar ($s_i \in \{0, 1\}$). Construim $x : x_1, \ldots, x_n$, $x_i = 2s_i - 1$.

Distribuţia de probabilitate a variabilei s este dată de $\Pr(S = 1) = p$ de unde rezultă că distribuţia variabilei x este $\Pr(X = 1) = p$.

Fie

$$S_k = \sum_{i=1}^{k} x_i = 2\sum_{i=1}^{k} s_i - k.$$

Valoarea medie a lui S_k este:

$$E(S_k) = k(2p-1),$$

iar dispersia sa este:

$$D^2(S_k) = 4pk(1-p).$$

Variabila aleatoare $\sum_{i=1}^{k} s_i$ are, la limită, o distribuţie normală $N(kp, k(1-p)p)$.

Deci

$$\frac{\sum_{i=1}^{k} s_i - kp}{\sqrt{kp(1-p)}} \sim N(0,1),$$

adică

$$\frac{\frac{S_k+k}{2} - kp}{\sqrt{kp(1-p)}} \sim N(0,1),$$

de unde rezultă:

$$\frac{S_k + k - 2kp}{2\sqrt{kp(1-p)}} \sim N(0,1).$$

Avem, pentru valori mici ale lui z:

$$\begin{aligned}\lim_{n\to\infty} \Pr(\frac{\max_{1\le k\le n}|S_k|}{\sqrt{n}} &\le z) = \frac{1}{\sqrt{2\pi}} \int_{-z}^{z} \sum_{k=-\infty}^{\infty} (-1)^k \exp\left\{-\frac{(u-2kz)^2}{2}\right\} du \\ &= \frac{4}{\pi} \sum_{j=0}^{\infty} \frac{(-1)^j}{2j+1} \exp\left\{-\frac{(2j+1)^2\pi^2}{8z^2}\right\}.\end{aligned}$$

Pentru valori mari ale lui $\frac{\max_{1\le k\le n}|S_k|}{\sqrt{n}}$ vom folosi formula:

$$\begin{aligned}\lim_{n\to\infty} \Pr\ (\frac{\max_{1\le k\le n}|S_k|}{\sqrt{n}} \le z) &= \sum_{k=-\infty}^{\infty} (-1)^k [\Phi((2k+1)z) - \Phi((2k-1)z)] \\ &= \Phi(z) - \Phi(-z) + 2\sum_{k=1}^{\infty} (-1)^k [\Phi((2k+1)z) - \Phi((2k-1)z)] \\ &= \Phi(z) - \Phi(-z) + 2\sum_{k=1}^{\infty} [2\Phi((4k-1)z) - \Phi((4k+1)z) - \Phi((4k-3)z)] \\ &\approx \Phi(z) - \Phi(-z) - 2[2\Phi(3z) - \Phi(5z) - \Phi(z)] \\ &\approx 1 - \frac{4}{\sqrt{2\pi}z} \exp\left\{-\frac{z^2}{2}\right\},\end{aligned}$$

unde $\Phi(X)$ este distribuţia normală standardizată.

Utilizând teorema 2.6 din *Revesz* [66] obţinem:

$$\Pr(\max_{1\le k\le n}|S_k| \ \ge \ z) = 1 - \sum_{k=-\infty}^{\infty}\Pr((4k-1)z < S_n < (4k+1)z))$$

$$+\sum_{k=-\infty}^{\infty}\Pr((4k+1)z \ < \ S_n < (4k+3)z)).$$

Observaţii. i) Dacă $p = 0,5$ (distribuţie uniformă) atunci $E(S_k) = 0$ şi $D^2(S_k) = k$.

ii) Dacă secvenţa are o distribuţie uniformă ($p = 0,5$) atunci funcţia de test devine:

$$\frac{S_k}{\sqrt{k}} \sim N(0,1).$$

iii) Testul se poate aplica şi în format regresiv în acest caz calculul sumelor cumulate S_k făcându-se de la ultimul către primul element:

$$S_k = \sum_{i=1}^{k} x_{n-i+1}, \ k = 1,\ldots,n.$$

iv) Statistica $CUSUM$ este definită ca fiind egală cu $\max\limits_k |S_k|$, iar distribuţia acesteia se construieşte cu ajutorul statisticii maximum (vezi paragraful 4.6.15.).

În cele ce urmează se prezintă o variantă, în pseudocod, a algoritmului de testare $CUSUM$.

Intrare: Succesiunea binară $s_1,\ldots,s_n$.

Ieşire: Decizia referitoare la acceptarea sau respingerea aleatorismului.

PAS 0. Citeşte probabilitatea $p = \Pr(S = 1)$, rata de respingere a testului α, lungimea minimă a drumului $k_{\min}$ (minim 1000).

PAS 1. Se transformă şirul binar $s_1,\ldots,s_n$ în şirul $\mathbf{X}$ de ± 1 după formula: $x_i = 2s_i - 1$.

PAS 2. For $j = k_{\min}$ to n do calculează statistica:

$$CUSUM[j] = \frac{\sum\limits_{i=1}^{j} s_i - jp}{\sqrt{jp(1-p)}}.$$

PAS 3. Dacă pentru orice $j \in \{k_{\min},\ldots,n\}$ statistica $CUSUM[j] \in [u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}]$ ($u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$ sunt cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale repartiţiei $N(0;1)$) atunci secvenţa testată este declarată aleatoare. Decizia se poate lua şi pe baza statisticii $\max_{1\le k\le n}|S_k|$ definite mai sus. În acest caz este dificil de calculat cuantila de

ordinul α a statisticii $\max_{1\leq k\leq n}|S_k|$ şi deci vom lua decizia cu ajutorul valorii P a probabilităţii:

$$\begin{aligned} P &= 1-\sum_{k=(-\frac{n}{z}+1)/4}^{(n/z-1)/4}\left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)-\Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] \\ &+\sum_{k=(-\frac{n}{z}-3)/4}^{(n/z-1)/4}\left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right)-\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right] \end{aligned}$$

prin compararea acesteia cu riscul α : dacă $P<\alpha$ atunci vom respinge ipoteza aleatorismului, în caz contrar vom accepta ipoteza aleatorismului.

4.6.10. Algoritmul de implementare al testului de aproximare a entropiei

Descrierea testului: Testul de aproximare a entropiei se bazează pe evaluarea diferenţei aproximării empirice a entropiei Shannon, realizată cu ajutorul blocurilor intercalate, dintre două lungimi consecutive a blocurilor. Statistica testului de aproximare a entropiei este de tip χ^2, descrierea acesuia fiind dată în *Rukhin* [67].

Scopul testului: Valori mari ale diferenţei dintre aproximarea empirică a entropiei Shannon, calculată intercalat pe două lungimi de bloc consecutive m şi $m+1$, pune în evidenţă fluctuaţii mari sau iregularităţi ale şirului iar o valoare mică a acestei diferenţe apare ca urmare a unei regularităţi substanţiale a şirului testat.

În cele ce urmează se prezintă o variantă, în pseudocod, a algoritmului de testare.

Intrare: Şirul (codificat pe s biţi) $x_1,\ldots,x_n$.

Ieşire: Decizia referitoare la acceptarea sau respingerea aleatorismului la riscul α.

PAS 0. Citeşte probabilitatea $p=\Pr(X=1)$, rata de respingere a testului α, parametrul s de codificare a datelor, gradul de aproximare al datelor (lungimea blocului pe care se face evaluarea entropiei) m.

PAS 1. Se calculează:

π_l^m frecvenţa relativă a patternului $l=(i_1,...,i_m)$ din şirul $x_1,\ldots,x_n$, aceasta calculându-se pe blocuri intercalate de lungime m;

π_l^{m+1} frecvenţa relativă a patternului $l=(i_1,...,i_{m+1})$ din şirul $x_1,\ldots,x_n$, aceasta calculându-se pe blocuri intercalate de lungime $m+1$.

PAS 2. Se calculează diferenţa de aproximare a entropiilor de ordinul m şi $m+1$:

$$ApEn(m)=\Phi^m-\Phi^{m+1},$$

unde

$$\begin{aligned}\Phi^m &= \sum_{i=1}^{s^m} \pi_l^m \ln \pi_l^m, \\ \Phi^{m+1} &= \sum_{i=1}^{s^{m+1}} \pi_l^{m+1} \ln \pi_l^{m+1}.\end{aligned}$$

PAS 3. Se calculează statistica testului:

$$f_{ApEn} = 2 \cdot n \cdot [\ln s - ApEn(m)] \sim \chi^2(s^{m+1} - s^m)$$

PAS 4. Dacă $f_{ApEn} < \chi^2_{1-\alpha}(s^{m+1} - s^m)$ (cuantila de ordinul $1-\alpha$ a distribuţiei $\chi^2(s^{m+1} - s^m)$), atunci se admite ipoteza aleatorismului secvenţei testate la riscul α.

Observaţia 4.6.4. O valoare prea mică a lui f_{ApEn} (apropiată 0) poate fi considerată *suspectă,* în concluzie este indicat de a utiliza un test bilateral în care regiunea de decizie este

$$[\chi^2_{\frac{\alpha}{2}}(s^{m+1} - s^m); \chi^2_{1-\frac{\alpha}{2}}(s^{m+1} - s^m)]$$

($\chi^2_{\frac{\alpha}{2}}(s^{m+1} - s^m)$ şi $\chi^2_{1-\frac{\alpha}{2}}(s^{m+1} - s^m)$ fiind cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale distribuţiei $\chi^2(s^{m+1} - s^m)$).

Observaţia 4.6.5. Valoarea uzuală a parametrului s este 2, a lui $m \geq 14$ şi condiţia de consistenţă $m < [\log_2 n] - 2$.

Observaţia 4.6.6. Testul se poate generaliza, în cadrul pasului 2, utilizând aproximarea empirică a entropiei lui Rényi de ordinul γ.

4.6.11. Algoritmul de implementare al testului lui Maurer (1992) şi testul entropiei

Descrierea testului: Testul lui Maurer, proiectat în 1992, este un test care pe lângă decizia referitoare la aleatorism calculează şi o estimare a redundanţei (gradul de predictibilitate) şi a entropiei (gradul de dezordine) sursei testate. Acest test poate fi utilizat şi ca un test de compresie. De fapt, ideea principală care a stat la proiectarea acestui test este testul de compresie Lempel-Ziv: cu o probabilitate foarte mare o sursă staţionară emite toate patternurile posibile (de o anumită lungime) într-o perioadă de timp bine specificată. Statistica testului este de tip normal, el putându-se realiza la diverse rate de respingere.

Scopul testului: Decizia asupra aleatorismului sursei testate şi o estimare a entropiei acesteia.

Algoritmul de implementare al testului lui Maurer este descris în continuare, specificându-se intrările şi ieşirile din algoritm.

PAS 0. Citeşte şirul binar $s^N = s_1, \ldots, s_n, \ldots s_N$

PAS 1. Citeşte parametrii testului N, L, Q. Aceştia sunt legaţi prin relaţia: $N = (Q + K)L$. N este lungimea şirului, K este numărul total de paşi, iar Q este numărul paşilor iniţiali.

Vom nota $b_n(s^N) = [s_{L(n-1)+1}, \ldots, s_{Ln}]$ pentru $n = 1, \ldots Q + K$. Deci şirul este împărţit în $Q + K$ blocuri adiacente de lungime L.

PAS 2. Citeşte rata de respingere ρ. Calculează cuantila repartiţiei normale $y = -u_{\frac{\rho}{2}}$.

PAS 3. Determină $A_{Q+1}(s^N), \ldots, A_{Q+K}(s^N)$, unde:

$$A_n(s^N) = \begin{cases} \min\{i : b_n(s^N) = b_{n-i}(s^N)\} \text{ dacă } \{i : b_n(s^N) = b_{n-i}(s^N)\} \neq \emptyset \\ n \text{ dacă } \{i : b_n(s^N) = b_{n-i}(s^N)\} = \emptyset. \end{cases}$$

PAS 4. Calculează :
statistica testului:

$$f_{T_U}(s^N) = \frac{1}{K} \sum_{n=Q+1}^{Q+K} \log_2 A_n(s^N),$$

constanta de proporţionalitate:

$$c(L, K) \approx 0.7 - \frac{0.8}{L} + (1.6 + \frac{12.8}{L}) K^{-\frac{4}{L}},$$

valoarea medie:

$$E[f_{T_U}(R^N)] = E[log_2 A_n(R^N)] = 2^{-L} \sum_{i=1}^{\infty} (1 - 2^{-L})^{i-1} \log_2 i,$$

dispersia variabilei aleatoare ce reprezintă un bloc:

$$Var[log_2 A_n(R^N)] = 2^{-L} \sum_{i=1}^{\infty} (1 - 2^{-L})^{i-1} (\log_2 i)^2 - (E[f_{T_U}(R^N)])^2,$$

dispersia:

$$\sigma = c(k, L) \sqrt{\frac{Var[log_2 A_n(R^N)]}{K}}.$$

PAS 5. Calculează limitele de respingere ale testului:

$$t_1 = E[f_{T_U}(R^N)] - y\sigma,$$

şi

$$t_2 = E[f_{T_U}(R^N)] + y\sigma.$$

PAS 6. Se respinge aleatorismul dacă $f_{T_U} < t_1$ sau $f_{T_U} > t_2$, în caz contrar se acceptă ipoteza aleatorismului.

PAS 7. Calculează estimarea entropiei: $H_S = \frac{E[f_{T_U}(U_S^N)]}{L}$.

PAS 8. Calculează estimarea redundanţei: $R_S = 1 - H_S$.

PAS 9. Estimarea mărimii efective a cheii (vezi *Maurer* [46] şi *Simion* [81]) este valoarea f_{T_U} (calculată la Pasul 4).

Observaţii privind implementarea testului. i) Pentru implementarea testului se recomandă alegerea parametrului L între 6 şi 16, $Q \geq 10 \times 2^L$, K cât de mare se poate(de exemplu $K = 1000 \times 2^L$). Această alegere a lui Q garantează că cu o probabilitate foarte mare fiecare bloc de lungime L apare cel puţin o dată în primele Q blocuri ale şirului aleatoriu.

ii) Recomandăm alegerea lui ρ între $0,05$ şi $0,001$.

iii) Funcţia test construită f_{T_U} are o repartiţie normală de medie m şi dispersie σ^2.

iv) Pentru o sursă staţionară ergodică S, cu ieşirea U_S^N, avem:

$$\lim_{L\to\infty} \frac{E[f_{T_U}(U_S^N)]}{L} = H_S.$$

4.6.12. Algoritmul de implementare al testului χ^2

Descrierea testului: Acest test foloseşte ca bază pentru o mulţime de alte teste particulare. Algoritmul de implementare este descris mai jos.

Intrare: O succesiune $s^N = s_1, \ldots, s_N$ cu $s_i \in \mathcal{S}$ unde $\mathcal{S} = \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_k$ este o partiţie a valorilor lui s, şirul frecvenţelor ideale $p_i = \Pr(S \in \mathcal{E}_i)$.

Ieşire: Decizia de acceptare sau respingere a aleatorismului.

PAS 0. Citeşte succesiunea s^N şi rata de respingere α.

PAS 1. For $i = 1$ to k calculează f_i frecvenţa celui de al i-lea simbol în şirul s^N.

PAS 2. Calculează statistica testului:

$$\chi^2 = \sum_{i=1}^{k} \frac{(f_i - Np_i)^2}{Np_i} = \sum_{i=1}^{k} \frac{f_i^2}{Np_i} - N.$$

PAS 3. Dacă $\chi^2 < \chi^2_{1-\alpha}(k-1)$ (cuantila de ordinul $1-\alpha$ a distribuţiei $\chi^2(k-1)$), atunci se admite ipoteza aleatorismului succesiunii la riscul α.

Observaţii. i) Pe baza testului χ^2 se construiesc şi testele de frecvenţă digrame, poker, al colectării cupoanelor, al golurilor, iteraţiilor ş.a. Un exemplu complet de aplicare al acestui tip de test este testul statistico-informaţional al lui Berlekamp-Massey (vezi cap. 6).

ii) Calculul cuantilei de ordinul α a unei distribuţii de tip χ^2 se reduce la rezolvarea ecuaţiilor de forma:

$$\int_{-\infty}^{x} t^{a(\alpha)} e^{-t} dt = b(\alpha),$$

unde $a(\alpha)$ şi $b(\alpha)$ sunt polinoame în funcţie de α. Aceste cuantile se pot aproxima cu ajutorul cuantilelor repartiţiei normale şi sunt de obicei tabelate.

iii) Valoarea ideală a statisticii χ^2 pentru verificarea uniformităţii ($p_i = \dfrac{1}{k}$) este 0 şi cu cât această valoare este mai mare cu atât şirul este mai predictibil. Statistica testului este în acest caz:

$$\chi^2 = \sum_{i=1}^{k} \frac{(f_i - \frac{N}{k})^2}{\frac{N}{k}}.$$

O valoare prea mică a lui χ^2 (apropiată 0) poate fi considerată *suspectă,* în concluzie este indicat de a utiliza un test bilateral în care regiunea de decizie este

$$[\chi^2_{\frac{\alpha}{2}}(k-1); \chi^2_{1-\frac{\alpha}{2}}(k-1)]$$

($\chi^2_{\frac{\alpha}{2}}(k-1)$ şi $\chi^2_{1-\frac{\alpha}{2}}(k-1)$ fiind cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale distribuţiei $\chi^2(k-1)$).

Documentaţia *NIST 800-22* [107] sugerează utilizarea testelor de tip χ^2 în varianta unilaterală şi la riscul $\alpha = 0,01$ (*sunt acceptate şi secvenţele perfecte*). Standardul American *FIPS PUB 140-2* [105] şi *Welschenbach* [96] impun teste bilaterale de tip χ^2 şi la riscul $\alpha = 0,0001$ (*secvenţele perfecte sunt considerate suspecte deci respinse*).

Pentru $k = 2$ (test la nivel de bit) vom obţine:

$$\chi^2 = \frac{4(f_1 - \frac{N}{2})^2}{N},$$

unde f_1 este numărul de biţi de 1 din sucesiunea testată. Vom observa mai departe că acestă variabilă $\chi^2(1)$ este de fapt o variabilă normală (de la testul frecvenţei) ridicată la pătrat, deci testul frecvenţei este un caz particular al testului χ^2.

iv) Dacă $p_i = \frac{1}{k}$ (distribuţia uniformă), pentru orice i, atunci calcule elementare conduc la $E(\chi^2) = \frac{N}{k}$ şi $D^2(\chi^2) = k - 1$.

4.6.13. Algoritmul de implementare al testului Kolmogorov-Smirnov

Descrierea testului: Testele precedente decideau între două ipoteze şi anume dacă o succesiune avea o abatere de la uniformitate. Testul se poate aplica la diverse riscuri de ordinul 1, statistica testului fiind determinată printr-o formulă asimptotică.

Scopul testului: Testul Kolmogorov-Smirnov este un test care se aplică pentru a decide asupra repartiţiei variabilei aleatoare (sursă de zgomot etc.) testate.

Testul care urmează, introdus de *Kolmogorov* şi *Smirnov*, decide între două ipoteze asupra distribuţiei $F(x)$ a unei variabile aleatoare (continue) X şi anume:

$$H_0 : X \text{ are funcţia de repartiţie } F(x),$$

respectiv:

$$H_1 : X \text{ nu are repartiţie } F(x).$$

Reamintim că funcţia de repartiţie $F(x)$ este egală cu $\Pr(X \leq x)$. Statistica testului este determinată prin formulele asimptotice de la pasul 3.

Algoritmul de implementare este prezentat în continuare:

PAS 0. Citeşte eşantioanele $X_1, \ldots, X_n$ şi rata de respingere a testului α.

PAS 1. Calculează funcţia de repartiţie empirică:

$$F_n(x) = \frac{card\{X_i \leq x | i = 1, \ldots, n\}}{n}.$$

PAS 2. Calculează abaterile K_n^+ şi K_n^- :

$$K_n^+ = \sqrt{n} \max_x (F_n(x) - F(x)) \text{ când } F_n(x) \geq F(x),$$

şi

$$K_n^- = \sqrt{n} \max_x (F(x) - F_n(x)) \text{ când } F_n(x) < F(x).$$

PAS 3. Calculează distribuţiile lui K_n^+ şi K_n^- după una din formulele (exacte):

$$\Pr(K_n^{+(-)} \leq \frac{t}{\sqrt{n}}) = \frac{t}{n^n} \sum_{k=0}^{t} C_n^k (k-t)^t (t+n-k)^{n-k-1},$$

$$\Pr(K_n^{+(-)} \leq \frac{t}{\sqrt{n}}) = 1 - \frac{t}{n^n} \sum_{t<k\leq n} C_n^k (k-t)^t (t+n-k)^{n-k-1}$$

sau după formula asimptotică (Smirnov 1951):

$$\Pr(K_n^{+(-)} \leq s) = 1 - e^{-2s^2}(1 - \frac{2}{3}\frac{s}{\sqrt{n}} + O(\frac{1}{n})).$$

PAS 4. Calculează cuantila $u_{1-\alpha}$ de ordinul $1-\alpha$ după formula:

$$\Pr(K_n^{+(-)} \leq u_{1-\alpha}) = 1-\alpha.$$

PAS 5. Dacă K_n^+ şi $K_n^- \in [0, u_{1-\alpha}]$, atunci se decide că repartiţia variabilei X este $F(x)$. În caz contrar se respinge ipoteza asupra repartiţiei lui X.

Observaţii: i) Dacă $F(x) = \dfrac{x-a}{b-a}, x \in [a,b]$, atunci testul respectiv face o verificare a uniformităţii variabilei $X \in [a,b]$.

ii) Calculul valorilor K_n^+ şi K_n^- poate fi mult simplificat prin următoarea procedură:

1. Se reţin observaţiile independente: $X_1, \ldots, X_n$.

2. Sortează ascendent aceste observaţii: $X_1 \leq \ldots \leq X_n$ (acest lucru se poate face cu ajutorul algoritmului de sortare rapidă, sortare prin interclasare etc.).

3. K_n^+ şi K_n^- se calculează după formulele:

$$K_n^+ = \sqrt{n} \max_{1 \leq j \leq n} (\frac{j}{n} - F(X_j)),$$

respectiv:

$$K_n^- = \sqrt{n} \max_{1 \leq j \leq n} (F(X_j) - \frac{j-1}{n}).$$

4.6.14. Testul spectral (transformarea Fourier discretă)

Descrierea testului: Acest test urmăreşte să distingă elementele periodice (patternuri care se repetă şi care sunt foarte apropiate, puse în evidenţă prin intermediul transformatei Fourier) care pot indica o anume abatere de la aleatorismul şirului (autocorelare). Statistica testului este de tip normal, iar acesta se poate aplica la diverse rate de respingere.

Scopul testului: Se testează distribuţia spectrului în amplitudine al şirului testat.

Algoritmul de implementare al testului spectral descris în continuare, specificându-se intrările şi ieşirile din algoritm.

Intrare: Succesiunea binară $s_1, \ldots, s_n$, pragul de referinţă pentru modulul transformatei $\tau \in [0,1]$ (în *NIST 800-22* [107] acesta este setat la 0,95) şi pragul de semnificaţie α.

Ieşire: decizia asupra aleatorismului la pragul de semnificaţie α.

PAS 1. Se transformă şirul binar $s_1, \ldots, s_n$ în şirul $\mathbf{X}$ de 1 şi -1 după formula $x_i = 2s_i - 1$.

PAS 2. Calculează (vezi observaţiile) *transformata Fourier discretă* $F = DTF(X)$ a lui X după formula:

$$f_j = \sum_{k=1}^{n} x_k \exp(-\frac{2\pi(k-1)ij}{n}), \text{ pentru } j = 0, \ldots, n-1$$

unde $\exp(\frac{2\pi(k-1)j}{n}) = \cos(\frac{2\pi(k-1)j}{n}) + i\sin(\frac{2\pi(k-1)j}{n})$, şi $i = \sqrt{-1}$.

Datorită simetriei transformării numerelor reale în numere complexe vom considera numai valorile de la 0 la $\left[\frac{n}{2}\right]$.

PAS 3. Calculează $M = modul(F^{'})$, unde $F^{'}$ este prima jumătate a lui F :

$$F^{'} = (f_0, \ldots, f_{[\frac{n}{2}]}).$$

Se produce astfel şirul (numit *spectrul în amplitudine*, măsurat în *decibeli*) $M = (m_0, \ldots, m_{[\frac{n}{2}]})$.

Evident $0 \leq m_j$ şi o valoare m_j poate lua numai 2^n valori. Mai mult $m_j = m_{n-j}$ pentru $j = 1, ..., [\frac{n}{2}]$.

PAS 4. Calculează limita $T = \sqrt{-n\ln(1-\tau)}$ pentru care, cu probabilitatea de τ, amplitudinea m este mai mică ca T.

PAS 5. Se calculează numărul mediu (ipoteza aleatorismului) de amplitudini din M care sunt mai mici ca T :

$$N_0 = \frac{\tau \cdot n}{2}.$$

PAS 6. N_1 numărul de amplitudini din M care sunt mai mici ca T.

PAS 7. Diferenţa normalizată (funcţia de decizie):

$$d = \frac{N_1 - N_0}{\sqrt{\frac{n}{4} \cdot \tau \cdot (1-\tau)}}.$$

PAS 8. Dacă $d \in [u_{\frac{\alpha}{2}}, u_{1-\frac{\alpha}{2}}]$ ($u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$ sunt cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1 - \frac{\alpha}{2}$ ale repartiţiei normale standardizate), atunci decide ipoteza aleatorismului la nivelul de risc α, în caz contrar respinge ipoteza aleatorismului.

Observaţii. i) *O valoare prea mică* a lui d indică faptul că mai puţin de $100 \cdot \tau\%$ din frecvenţe sunt peste limita T şi prea multe (mai mult ca $100 \cdot (1-\tau)\%$) peste aceeiaşi limită. O valoare prea mare a lui d indică faptul că numărul de frecvenţe care sunt peste limita T este peste pragul de $100 \cdot \tau\%$ şi numărul de frecvenţe sub limita T este mai mic ca $100 \cdot (1-\tau)\%$.

ii) Pentru calculul transformatei Fourier discrete se poate utiliza algoritmul transformatei Fourier rapide care reduce numărul operaţiilor aritmetice de la $O(n^2)$ la $O(n\log_2 n)$.

iii) Transformata Fourier se poate defini şi pentru funcţii din $L^1(\mathbf{R})$ prin formula:

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt,$$

acesta realizează o transformare a semnalului din domenul timp în domeniul frecvenţă.

iv) Există şi un alt test statistic mult mai util în activitatea de criptanaliză şi anume testul bazat pe *transformata Walsh-Hadamard* care este definită, pentru funcţii booleene $f : \mathbf{Z}_2^n \to \mathbf{Z_2}$, prin formula:

$$\hat{f}(\boldsymbol{\omega}) = \sum_{\mathbf{x}} f(\mathbf{x})(-1)^{\boldsymbol{\omega}.\mathbf{x}}.$$

unde $\boldsymbol{\omega}.\mathbf{x}$ este produsul scalar dintre vectorii $\boldsymbol{\omega}$ şi $\mathbf{x}$, iar sumarea se face ca în $\mathbf{R}$. Acest test se foloseşte pentru detectarea algoritmului folosit în recurenţele de tip neliniar. Testele de acest tip se mai numesc şi *teste spectrale.*

v) Ţinând cont de faptul că $x_i = \pm 1$, relaţia lui Parserval devine în acest caz:

$$\sum_{j=0}^{n-1} |f_j|^2 = n \sum_{i=1}^{n} |x_i|^2 = n^2.$$

4.6.15. Teste de corelaţie

Fie şirurile x_n şi y_n care sunt realizările unor surse binare staţionare cu $\Pr(X = 1) = p_0$ respectiv $\Pr(Y = 1) = p_1$. Un *test de corelaţie* la distanţa d este un test de aleatorism statistic aplicat şirului binar $z_n = x_n \oplus y_{n+d}$ cu $\Pr(Z = 1) = p_0q_0 + p_1q_1$. Se pot construi astfel teste de corelaţie temporală, circulară, serială, corelaţie a succesiunilor, corelaţie Poker, corelaţie Fourier etc. Paragaful următor prezintă testul corelaţiilor temporale şi circulare.

4.6.16. Algoritmul de implementare al testului corelaţiilor temporale şi circulare

Descrierea testului: Testul corelaţiei la distanţa d pentru şirurile x_n şi y_n care sunt realizările unor surse staţionare cu $\Pr(X = 1) = p_0$ respectiv $\Pr(Y = 1) = p_1$ este de fapt testul frecvenţei, cu parametru de probabilitate $p_0q_0 + p_1q_1$, aplicat şirului $z_n = x_n \oplus y_{n+d}$.

Scopul testului: Acest test este deosebit de util, de exemplu, în aplicaţiile criptografice pentru a stabili gradul de corelare a două texte criptate sau a textului clar cu textul cifrat.

4.6.17. Creşterea senzitivităţii algoritmilor de testare statistică

După cum am văzut unele dintre testele prezentate sunt un conglomerat de teste statistice (de exemplu, testele de autocorelaţii), iar decizia de acceptare sau respingere a ipotezei nule (de exemplu, ipoteza necorelării) se bazează pe statistica

numărului de decizii de neacceptare dată de teste(de exemplu, testele de frecvenţă): dacă acest număr este mai mic ca $n\alpha$, unde α este pragul de semnificaţie al testelor, atunci acceptăm ipoteza nulă. Această regulă induce pentru schema de testare un număr foarte mare de rezultate fals pozitive, deci rata de respingere este extrem de mare. Vom încerca să menţinem această rată în limite rezonabile.

Agregarea rezultatelor. Pentru a fixa ideile să presupunem că dorim să testăm, prin intermediul unei proceduri de test T, parametrii unei variabile aleatoare. Pentru aceasta efectuăm, la rata de respingere α, un număr de n teste T asupra a n rezultate independente ale variabilei aleatoare ai cărei parametri fac obiectul testării. Să notăm cu $t(n)$ numărul de respingeri al ipotezei nule date de testul T din cele n teste efectuate la riscul de ordinul 1 α. Deci numărul de teste care decide ipoteza nulă este $n-t(n)$. Dacă ipoteza nulă este adevărată, atunci variabila ce dă rezultatul unui test T este o variabilă de tip Bernoulli cu $\Pr(T=1)=\alpha$ şi deci variabila $t(n)$ are o repartiţie binomială de parametrii n şi α :

$$\Pr(t(n)=k)=C_n^k\alpha^k(1-\alpha)^{n-k}.$$

La limită $t(n)$ urmează o repartiţie $N(\alpha n, n\alpha(1-\alpha))$ (vezi Anexa E) deci:

$$\frac{t(n)-\alpha n}{\sqrt{n\alpha(1-\alpha)}}\sim N(0,1).$$

Folosind relaţia anterioară dacă: $\frac{t(n)-\alpha n}{\sqrt{n\alpha(1-\alpha)}}\in[u_{\frac{\alpha}{2}};u_{1-\frac{\alpha}{2}}]$ ($u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$ sunt cuantilele repartiţiei normale de ordinul $\frac{\alpha}{2}$, respectiv $1-\frac{\alpha}{2}$,) acceptăm ipoteza nulă, iar în caz contrar, respingem ipoteza nulă. Această regulă de decizie, dacă este folosită în pasul de decizie, creşte senzitivitatea testelor de autocorelaţii temporale şi circulare.

Agregarea celor n rezultate ale statisticii notate prin f_i cu $i=1,...,n$, se poate face şi cu ajutorul statisticii:

$$\chi^2(m)=\sum_{j=1}^{m}\frac{\left(F_j-\frac{n}{m}\right)^2}{\frac{n}{m}}$$

unde F_j este numărul acelor p_i (cel mai mic prag de semnificaţie la care resping ipoteza nulă: în cazul *testului bilateral* aceasta fiind $\frac{\Pr(X\le f_i)}{2}$ iar în cazul *testului unilateral* fiind $\Pr(X\le f_i)$) care sunt în intervalul $[\frac{j-1}{m},\frac{j}{m})$ pentru $j=1,...,m$. Această statistică *verifică uniformitatea* valorilor p.

Testul maximum din t. Un alt procedeu este acela de a considera statistica $t=\max_{1\le i\le n}(t_i)$, unde t_i este valoarea statisticii corespunzătoare eşantionului i

(presupunem că eşantioanele sunt independente), care în ipoteza H au aceeaşi funcţie de repartiţie $H(x)$. Funcţia de repartiţie a lui t este:

$$F(x) = \Pr(t \leq x) = \Pr(\max_{1 \leq i \leq n}(t_i) \leq x) = (H(x))^n.$$

La riscul de ordinul 1 egal cu α cuantila $t_{1-\alpha}$ satisface relaţia:
$(H(t_{1-\alpha}))^n = 1 - \alpha$. Deci $H(t_{1-\alpha}) = (1-\alpha)^{\frac{1}{n}}$ unde $t_{1-\alpha}$ este cuantila de ordinul $(1-\alpha)^{\frac{1}{n}}$ a repartiţiei $H(x)$ (notată prin $h_{(1-\alpha)^{\frac{1}{n}}}$). Testul poate fi *lateral* (domeniul critic sau de respingere a ipotezei nule fiind de exemplu $[h_{(1-\alpha)^{\frac{1}{n}}}, \infty)$) sau *bilateral* (în acest caz, domeniul de acceptare este $[h_{(\frac{\alpha}{2})^{\frac{1}{n}}}, h_{(1-\frac{\alpha}{2})^{\frac{1}{n}}}]$).

Testul bazat pe statistica maximum-t dă informaţii şi în ceea ce priveşte *stabilitatea repartiţiei.*

Analiza datelor semnificative. O altă problemă ce trebuie avută în vedere este aceea a *datelor semnificative* adică a acelor date care se îndepărtează de media variabilei aleatoare cu ajutorul căreia se face agregarea testului. În acest caz se foloseşte *inegalitatea lui Cebâşev*:

$$\Pr(|X - m| \geq \varepsilon\sigma) \leq \frac{1}{\varepsilon^2}.$$

4.7. Teste de aleatorism algoritmic

Aceste teste, spre deosebire de testele statistice care scot în evidenţă defecte de tip stochastic, pun în evidenţă o comportare algoritmică a sursei testate (cel puţin în probabilitate). Dintre aceste teste reţinem testele de compresie (de exemplu, testul Lempel-Ziv, testul Maurer) precum şi teste de tip algoritm recursiv (complexitate liniară sau pătratică). Reţinem că o descriere a acestora poate fi găsită de exemplu la adresa de internet a **NIST***(National Institute of Standards and Technologies).* În cadrul acestui paragraf vom prezenta o abordare a aleatorismului în funcţie de complexitatea şirurilor finite. Vom simula diferite şiruri binare sub diverse modele stochastice pentru a estima puterea testului.

4.7.1. Scurt istoric

Complexitatea linear echivalentă a unei succesiuni de numere (nu neapărat binare) este lungimea celei mai scurte recurenţe de tip Fibonacci care generează succesiunea dată, fiind introdusă iniţial de *Massey* în 1969. *Jansen* şi *Boekee* au definit *ordinul maxim al complexităţii* ca lungimea celui mai mic registru de deplasare, nu neapărat linear, care generează şirul. Media ordinului maxim al complexităţii a unui şir binar

de lungime n este aproximativ de $n \log n$. O altă măsură a complexităţii este *complexitatea Lempel-Ziv*. Această măsură cuantifică rata de apariţie a unor structuri noi în şir. *Kolmogorov* şi *Chaitin* au introdus noţiunea de complexitate Turing-Kolmogorov-Chaitin care măsoară lungimea minimă a intrării într-o maşină Turing universală care generează şirul. Nu există un algoritm eficient pentru calculul complexităţii Kolmogorov (de fapt, această complexitate nu este calculabilă). *Beth* şi *Dai* au demonstrat că complexitatea Turing-Kolmogorov-Chaitin este aproximativ de două ori complexitatea liniară pentru majoritatea şirurilor suficient de lungi. În continuare ne axăm pe studiul complexităţii Kolmogorov.

4.7.2. Măsurarea complexităţii

Aşa cum am menţionat anterior, complexitatea computaţională este un concept fundamental în definirea sau recunoaşterea unui şir aleatoriu. Pentru a fundamenta ideile vom face o prezentare informală a complexităţii *Kolmogorov*. Presupunem că cititorul este familiarizat cu definiţiile şi proprietăţile fundamentale ale maşinilor *Turing*.

În esenţă, scopul principal al lui Kolmogorov a fost să dea o definiţie a *cantităţii de informaţie* conţinută într-un *singur* obiect, care poate fi spre exemplu un şir binar, noţiune ce este complementară altor concepte (exemplu entropia) care sunt aplicabile surselor stochastice (sau modelelor), dar nu sunt aplicabile elementelor individuale generate de sursă (model). Dacă avem un şir binar de lungime fixă vom considera că acesta este mai mult sau mai puţin aleatoriu dacă demonstrăm că este dificil să-l descriem în câteva cuvinte. De fapt, şirul este foarte complex (aleatoriu) dacă cea mai bună descriere a sa este el însuşi. În spiritul acestei idei, Kolmogorov (şi alţi ca de exemplu Löfgren) au introdus noţiunea de *complexitate descriptivă* a unui singur obiect (un şir binar, un sistem etc.). A fost demonstrat ulterior (Martin-Löf, Löfgren, Schnorr) că această noţiune corespunde noţiunii de aleatorism.

Aceasta înseamnă că şirurile cu complexitate mare vor fi declarate aleatorii de către orice test statistic. Cu alte cuvinte, dacă se construieşte un test de aleatorism cu ajutorul complexităţii descriptive, atunci acele şiruri clasificate ca aleatorii de către acest test vor fi acceptate ca aleatorii de către orice test statistic.

Prin *complexitate Kolmogorov* a unui şir înţelegem dificultatea descrierii şirului sau a stocării şirului prin intermediul unei reguli. O descriere a unui şir poate fi gândită ca un set de instrucţiuni pentru un anume computer care permite reconstrucţia şirului.

Pentru simplificare, ne restrângem la cazul unui şir binar de lungime n, notat cu S_n. Procedura pentru calculul lui S_n poate fi un computer, o maşină sau algoritm α care acceptă ca intrare un program (descriere) p şi generează la ieşire şirul S_n, de exemplu:

$$S_n = \alpha(p).$$

O măsură a complexităţii lui S_n este lungimea $l(p)$ a lui p. În orice caz, p este el însuşi un şir binar şi probabil există o descriere mai simplă $p^{'}$ a lui p astfel încât $l(p^{'}) \leq l(p)$ şi $S_n = \alpha$. Bazându-se pe aceste consideraţii, Kolmogorov a definit complexitatea relativă $K_\alpha(S)$ a unui şir finit S relativ la α ca:

$$K_\alpha(S) = \min\{l : \alpha(p) = S,\ l = l(p)\}.$$

Dacă nu există nici un p astfel încât $\alpha(p) = S$, atunci $K_\alpha(S) = \infty$. Cu alte cuvinte : complexitatea unui şir finit S în raport cu o maşină α este lungimea celui mai scurt program (pe maşina α) care calculează S, dacă un asemenea program există.

Pentru a face definiţia independentă de maşina α este luată o maşină *Turing* universală (**MTU**). (O **MTU** este o maşină Turing cu stările finite care poate simula orice maşină Turing). Se poate arăta că nu există o procedură pentru calculul lui K_α. Aceasta revine la a afirma că nu există nici o maşină Turing β astfel încât:

$$\forall S_n, \beta(S_n) = K_\psi(S_n),$$

unde prin ψ am notat o maşină Turing universală fixată.

S-a demonstrat de către *Löfgren*(1972, 1977) că pentru n arbitrar de mare se poate evita dificultatea de a determina dacă un şir dat este de complexitate mare. Pentru a evita aceste limitări ne propunem să găsim măsuri eficiente de măsurare a complexităţii.

Lempel şi *Ziv* (1976) au propus o măsură alternativă a complexităţii unui şir finit care satisface o serie de axiome. Vom prezenta în continuare aceste axiome.

Fie X o mulţime finită de simboluri numită alfabet (în cazul nostru $X = \{0, 1\}$) şi X^l mulţimea şirurilor de lungime l care poate fi formată cu elemente din X. Vom nota cu λ şirul vid (adică şirul de lungime zero) şi vom defini mulţimea X^* prin:

$$X^* = \bigcup_{l=0}^{\infty} X^l.$$

Fie $S \in X^*$ şi K două funcţii reale definite pe X^*. Atunci K se numeşte măsură de complexitate pe X^* dacă satisface următoarele axiome:

Axioma 1. $\forall S \in X^* : K(S) \geq 0$ $(K(\lambda) = 0)$.

Complexitatea este o funcţie nenegativă şi este zero pentru şirul vid.

Axioma 2. Fie SQ concatenarea a două şiruri arbitrare $S, Q \in X^*$, atunci:

$$K(SQ) \geq K(S).$$

Creşterea şirului nu trebuie să descrească complexitatea.

Axioma 3. Fie $S = \{x_1, \ldots, x_n\}$ atunci $\forall x_i, x_j \in X$:

$$K(x_i) = K(x_j).$$

Complexitatea şirului format dintr-un singur simbol este aceeaşi pentru orice simbol.

Axioma 4. Fie X_i^m şirul de lungime $m \geq 1$ cu toate elementele egale cu $x_i \in X$. Atunci:

$$K(X_i^m) = \min\{K(S) : l(S) = m\} \text{ pentru orice } i = 1, \ldots, n.$$

Şirul format din *repetarea* unui singur simbol trebuie să aibă complexitatea minimală dintre toate şirurile de aceeaşi lungime.

Axioma 5. Fie $(x_{i_1}, \ldots, x_{i_m})$ o permutare de m elemente distincte din S şi fie $S_m = x_{i_1} \ldots x_{i_m}$. Atunci:

$$K(S_m) = \max\{K(S) : l(S) = m\}.$$

Şirul al cărui elemente sunt *toate* distincte trebuie să aibă complexitatea maximă printre toate şirurile de aceeaşi lungime.

Axioma 6. Fie A o aplicaţie a lui X în X, prin $A(S)$ vom nota transformarea şirului $S \in X^*$ obţinut prin aplicarea lui A fiecărui element al lui S. Atunci $\forall S \in X^*$:

$$K(A(S)) \leq K(S).$$

O transformare a şirului într-un şir de aceeaşi lungime nu trebuie să crească complexitatea şirului.

Sunt o mulţime de funcţii care satisfac aceste axiome, de exemplu măsura trivială $K(S) = c_1 l(S) + c_2$ unde c_1, c_2 sunt constante, iar $l(S)$ este lungimea lui S. Pentru a restrânge această mulţime impunem următoarea axiomă:

Axioma 7: $\forall S \in X^*$: $K(S) \leq l(S) + c$ unde c este o constantă independentă de S. În particular c poate fi 0, de unde rezultă:

$$0 \leq K(S) \leq l(S), \forall S \in X^*.$$

Aceasta înseamnă că valoarea K trebuie să fie mărginită de lungimea şirului. Deci, dacă ne raportăm la şiruri de lungime n, trebuie să avem:

$$0 \leq K \leq n, \forall S \in X^*, \text{ cu } l(S) = n.$$

Mai sunt şi alte măsuri posibile care satisfac axiomele anterioare. Una dintre acestea, numită *complexitatea modificată a segmentului*, o vom discuta în subcapitolul următor.

4.7.3. Complexitatea segmentului

Lempel şi Ziv au definit vocabularul unui şir finit S_n notat prin $v(S_n)$ ca submulţimea lui X^* formată din toate subşirurile *diferite* S_j $(j = 1, \ldots, n)$, de lungime j, conţinut în S_n. De exemplu, vocabularul şirului $S = \{0010\}$ este:

$$v(0010) = \{0, 1, 00, 01, 001, 010, 0010\}.$$

Complexitatea segmentului S_n este definită ca fiind cardinalul lui $v(S_n)$, de exemplu $K_S(S_n) = |v(S_n)|$. În cazul exemplului nostru, complexitatea lui S este $K_S(S) = 8$.

Observăm că este rezonabil să luăm ca măsură a aleatorismului complexitatea segmentului, deoarece şiruri foarte regulate (de exemplu nealeatorii) vor avea un vocabular mic (considerăm cazul extrem în care $S = 111111$) în timp ce şirurile foarte neregulate (aleatorii) tind să aibă un vocabular foarte mare.

Să notăm prin q cardinalul lui X. Se poate arăta că pentru $n \geq 1$, şi $q \geq n$:

$$n \leq K_S(S_n) \leq \frac{n(n+1)}{2}.$$

De asemenea, dacă $q < n$ avem:

$$n \leq K_S(S_n) \leq \frac{q^m - q}{q-1} + n(n-m+1) - \frac{n(n-1) - (m-1)(m-2)}{2},$$

unde m este cel mai mic număr întreg $(1 \leq m \leq n)$ pentru care avem inegalitatea $q^m > n - m + 1$. Complexitatea $K_S(S_n)$ poate fi transformată în complexitatea modificată a segmentului notată prin $K_T(S_n)$ după regula:

$$K_T(S_n) = \frac{2K_S(S_n)}{n} - 1,$$

pentru $n \geq 1$, iar pentru şirul vid λ avem $K_T(\lambda) = 0$. Pentru versiunea modificată K_T a complexităţii avem nişte inegalităţi similare şi anume dacă $q \geq n$ atunci:

$$1 \leq K_T(S_n) \leq n.$$

Pentru $q < n$ avem:

$$1 \leq K_T(S_n) \leq k,$$

unde

$$k = \frac{2(q^m - q)}{n(q-1)} + n - 2(m-1) + \frac{(m-1)(m-2)}{n}.$$

Este trivială demonstraţia în cazul în care complexitatea modificată a segmentului K_T satisface cele şapte axiome prezentate în secţiunea anterioară. Vom alege complexitatea modificată a segmentului ca un potenţial test statistic în testele de aleatorism pentru un studiu prin simulare.

4.7.4. Complexitatea segmentului ca măsură a aleatorismului

Aşa cum am menţionat anterior, testele statistice de aleatorism sunt, de regulă, instrumente puternice pentru studiul altor variante ale aleatorismului.

Ideea de bază este de a genera un şir stochastic binar cu o dependenţă internă ce nu poate fi detectată decât folosind un model specific.

Printr-un şir binar *aleatoriu* de lungime $l = n$ înţelegem un şir de variabile aleatoare independente şi identic distribuite (i.i.d) *Bernoulli* $X_1, \ldots, X_n$ astfel încât $X_i = 0$ sau 1 $(i = 1, \ldots, n)$ cu:

$$P(X_k = x_k | X_{k-1} = x_{k-1}, \ldots, X_1 = x_1) = P(X_k = x_k) = \theta = 0,5$$

pentru orice $k = 1, \ldots, n$.

Ipoteza nulă care trebuie testată este:

$H_0 : X_1, \ldots, X_n$ sunt variabile aleatoare independente Bernoulli.

iar ipoteza alternativă va fi,

$H_1 : X_1, \ldots, X_n$ nu sunt variabile aleatoare independente Bernoulli.

În cazul complexităţii Kolmogorov, un şir S_n de lungime n se numeşte aleatoriu (sau $t-$aleatoriu) dacă pentru o maşină Turing universală fixată ψ şi pentru t întreg nenegativ avem:

$$K_\psi(S_n) > n - t.$$

Se poate demonstra că folosind această definiţie se poate construi un test de aleatorism astfel încât şirurile declarate aleatorii de către acest test să fie declarate aleatorii de către orice alt test statistic.

Dar complexitatea Kolmogorov nu este calculabilă şi din acest motiv trebuie să folosim altă măsură, ca de exemplu complexitatea modificată a segmentului care este efectiv calculabilă şi îndeplineşte axiomele $1 - 7$. Deci, dacă notăm prin K_T complexitatea modificată a segmentului vom declara un şir finit binar S_n ca fiind aleatoriu (în concordanţă cu ipoteza H_0) dacă:

$$K_T(S_n) > k_\alpha,$$

pentru k_α astfel încât sub H_0 :

$$P(K_T(S_n) \leq k_\alpha) \leq \alpha.$$

Aceasta înseamnă că k_α este valoarea critică corespunzătoare unui test de mărime α cu regiunea critică:

$$C_\alpha = \{S_n : K(S_n) \leq k_\alpha\}.$$

În această prezentare folosim ca alternative particulare H_1 un tip special de şiruri Markov pentru care $P(X_m = x_m | X_{m-1} = x_{m-1}, \ldots, X_1 = x_1)$ depinde numai de

$a-(m-k)$ variabilă X_{m-k} pentru un întreg pozitiv k. Cazul $k=1$ corespunde *procesului Markov* uzual, iar pentru un k general procesul se va numi *proces Markov cu lagul* k. Dacă valoarea k este cunoscută este uşor să adaptăm testele existente pentru a testa H_0 contra ipotezei alternative:

$$H_1 : P(X_m = x_m | X_{m-1} = x_{m-1}, \ldots, X_1 = x_1)$$
$$= P(X_m = x_m | X_{m-k} = x_{m-k}) = \lambda_m \neq 0,5,$$

pentru un $0 < \lambda_m < 1$. Probabilitatea λ_m ia două valori deoarece fiecare x_i poate lua două valori. Dacă k nu este cunoscut, atunci nu este uşor să testăm aleatorismul şi acesta este cazul în care aplicarea lui K_T ca test statistic să poată fi mult mai folositoare.

În continuare vom descrie modelele şi procedurile pe care le vom aplica. Deoarece avem un şir de variabile aleatoare Bernoulli probabilităţile sub H_1 pentru $i = k+1, k+2, \ldots,$ sunt:

$$P(X_i = 1 | X_{i-k} = 1) = \lambda > \theta = 0,5,$$

$$P(X_i = 0 | X_{i-k} = 1) = 1 - \lambda < \theta = 0,5,$$

$$P(X_i = 0 | X_{i-k} = 0) = \lambda > \theta = 0,5,$$

şi

$$P(X_i = 1 | X_{i-k} = 0) = 1 - \lambda < \theta = 0,5.$$

Pentru orice $l = 10, 20, 30, 40, 50$ generăm N şiruri după modelul anterior. Simulările au fost efectuate pentru $\lambda = 0,6;\ 0,7;\ 0,8;\ 0,9$ care corespund diferitelor grade de dependenţă.

Să remarcăm că primele k elemente ale şirului sunt generate aleatoriu şi din ele orice alt bloc de k elemente va avea tendinţa de a fi o copie a blocului anterior de k elemente. Mai exact un bloc de k simboluri va avea tendinţa de a se reproduce (cât de puternică este această tendinţă depinde de valoarea λ), dar odată produsă o schimbare a unui simbol în noul bloc acesta va avea tendinţa de a se reproduce el însuşi.

Exemplu. Prezentăm o serie de şiruri binare ce au fost obţinute prin simulare pentru diferite valori ale lui l, k şi λ.

a) lungimea $l = 10, k = 1, \lambda = 0,6$.

1001100011

1000001110

1110001111
b) lungimea $l = 20, k = 3, \lambda = 0,8$.
11001000110110010011
10010010110011001011
01000001000100100100
c) lungimea $l = 30, k = 3, \lambda = 0,7$.
101001101101101110100100101101
000000001011010000000010010010
001011111001011111100101101101
d) lungimea $l = 40, k = 3, \lambda = 0,8$.
1001001011011010000100101111011011011011
0001001001001001001100100110110110011011
1101101101111001001001001001001001001001
e) lungimea $l = 50, k = 5, \lambda = 0,9$.
10000101001110011100111001110011000100001000010100
00100001000010000100001000011100111001110111101111
00000000000010000100001000010001101011000100001000

4.8. Teste de necorelare algoritmică

4.8.1. Formularea problemei

Una dintre practicile cele mai frecvente de a elabora algoritmi criptografici este aceea de a modifica anumite elemente din algoritmi deja proiectaţi şi evaluaţi de către comunitatea ştiinţifică. *Scopul* realizării acestor modificări sunt:

-realizarea unui algoritm cel puţin la fel de sigur (din punct de vedere criptografic) ca algoritmul original;

-gradul de necorelare al celor doi algoritmi să fie suficient de mare;

-complexitatea de implementare a noului algoritm să fie comparabilă cu cea a algoritmului original;

-viteza de calcul a noului algoritm să fie comparabilă cu cea a algoritmului original.

4.8.2. Principii de test

Vom prezenta în cele ce urmează o serie de principii de testare a necorelării a doi algoritmi criptografici. Pentru fixarea ideiilor fie A_1 şi A_2 doi algoritmi criptografici cu aceeaşi spaţiu de intrare al cheilor $\mathcal{K}$ şi al mesajelor $\mathcal{M}$. Vom defini

distanţa, relativ la ponderea Hamming a sumei ieşirilor, dintre algoritmii A_1 şi A_2 prin formula:

$$d(A_1; A_2) = \max_{M \in \mathcal{M}, k \in \mathcal{K}} cor\ (A_1(M; k), A_1(M; k)),$$

corelaţia fiind calculată ca

$$1 - 2\frac{weight(A_1(M; k) \oplus A_1(M; k))}{n},$$

unde n este lungimea blocului de ieşire (exprimată în biţi).

Observaţia 4.8.1. Evident $d \in [-1, 1]$ ($d = -1$ înseamnă că cei doi algoritmi produc ieşiri negate unul faţă de altul, $d = 0$ înseamnă că cei doi algoritmi sunt necorelaţi, $d = 1$ înseamnă că cei doi algoritmi produc aceleaşi ieşiri).

Observaţia 4.8.2. Se poate opta şi pentru altă formulă de calcul a distanţei: acesta se construieşte şi pe baza structurii algortimului original şi a modificărilor efectuate.

Observaţia 4.8.3. În practică calculul distanţei d este nefezabil dearece atât $card(\mathcal{K})$ cât şi $card(\mathcal{M})$ sunt numere foarte mari. Din acest motiv se iau deregulă M şi k cuvinte de pondere Hamming maxim p respectiv q (fezabil $p = 1, q = 1$), restricţionând calculul lui d la aceste cuvinte se obţinâne astfel distanţa $d_{p,q}(A_1; A_2)$. Complexitatea de calcul a distanţei $d_{p,q}(A_1; A_2)$ creşte exponenţial cu p şi q :

$$O(\frac{card(\mathcal{M})^p}{p!} \cdot \frac{card(\mathcal{K})^q}{q!}).$$

În acest caz trebuie estimată *eroarea de aproximaţie*:

$$d(A_1; A_2) - d_{p,q}(A_1; A_2).$$

4.9. Teste de verificare a jocurilor de tip Casino

Testele prezentate în cadrul acestui paragraf sunt teste dedicate jocurilor de tip Casino, care sunt de două tipuri şi anume: jocuri de tip bilă întoarsă (ruletă) şi bilă neîntoarsă (loto, 6/49, 5/40 sau jackpot). Acestea sunt prezentate în pseudocod, iar sintaxa este similară limbajului C.

4.9.1. Metoda $3-\sigma$ pentru ruletă

Iniţializări: for $j = 0$ to 36 do $k[j] := 0$;
Intrare. Şirul realizărilor $x_0, x_1, x_2, \ldots, x_{n-1} \in \{0, 1, 2, .., 36\}$;
PAS 1. for $j = 0$ to $n - 1$ do $k[x[j]] := k[x[j]] + 1$;
PAS 2. Calculăm:

$$\mu := \frac{n}{37},$$

$$\sigma := \sqrt{\frac{n}{37}(1 - \frac{1}{37})}.$$

PAS 3. Dacă pentru orice $j = 0, 1, 2.., 36$ avem $k[j] \in [\mu - 3\sigma; \mu + 3\sigma]$, atunci se decide **ruleta funcţionează corect**.

Dacă există $j = 0, 1, 2.., 36$ astfel încât $k[j] \notin [\mu - 3\sigma; \mu + 3\sigma]$, atunci se decide **ruleta este defectă**.

4.9.2. Metoda $3-\sigma$ pentru diferenţe la ruletă

Intrare. Şirul $x_0, x_1, x_2, \ldots, x_{n-1} \in \{0, 1, 2, .., 36\}$, n minim 3700.
PAS 1. Calculăm diferenţele: for $i = 0$ to $n - 2$ $y[i] := x[i + 1] - x[i]$;
PAS 2. Transformăm diferenţele: for $i = 1$ to $n - 2$ do

$$t[i] = \begin{cases} y[i] \text{ dacă } y[i] \geq 0 \\ 37 + y[i] \text{ dacă } y[i] < 0. \end{cases}$$

PAS 3. Aplicăm algoritmul 1 (metoda $3 - \sigma$ pentru ruletă) şirului diferenţelor.

4.9.3. Metoda X^2 pentru ruletă

Iniţializare. for $j = 0$ to 36 do $k[j] := 0$;
Intrare. Şirul $x_0, x_1, x_2, \ldots, x_{n-1} \in \{0, 1, 2, .., 36\}$, n minim 3700.
PAS 1. for $j = 0$ to $n - 1$ do $k[x[j]] := k[x[j]] + 1$;
PAS 2. Calculăm:

$$X^2 = \sum_{j=0}^{36} \frac{(k[j] - \frac{n}{37})^2}{\frac{n}{37}}$$

PAS 3. Dacă $X^2 < u_{0,9973}(36)$ (cuantila de ordinul $0,9973$ a distribuţiei $\chi^2(36)$) atunci **ruleta este corectă**.
$u_{0,9973}(36) = 64,1$;

4.9.4. Metoda X^2 aplicată diferenţelor pentru ruletă

Intrare. Şirul $x_0, x_1, x_2, \ldots, x_{n-1} \in \{0, 1, 2, .., 36\}$, n minim 3700.
PAS 1. Calculăm diferenţele: for $i = 0$ to $n-2$ do $y[i] := x[i+1] - x[i]$;
PAS 2. Transformăm diferenţele: for $i = 1$ to $n-2$ do

$$t[i] = \begin{cases} y[i] \text{ dacă } y[i] \geq 0 \\ 37 + y[i] \text{ dacă } y[i] < 0. \end{cases}$$

PAS 3. Aplicăm algoritmul 3 şirului diferenţelor $t[i]$.

4.9.5. Metoda X^2 pentru jocurile de tip loto

Iniţializări:
la 5 din 40 : $n = 5$ şi $v = 40$;
la 6 din 49 : $n = 6$ şi $v = 49$.
Iniţializări. for $j = 1$ to v do $k[j] := 0$;
Intrare. Numărul de extrageri complete N, şirul realizărilor $x_0, x_1, x_2, \ldots x_{Nn-1}$ din $\{1, 2, \ldots, v\}$
PAS 1. for $j = 0$ to $Nn-1$ do $k[x[j]] := k[x[j]] + 1$;
PAS 2. Calculăm statistica testului :

$$X_L^2 = \sum_{j=1}^{v} \frac{(k[j] - \frac{Nn}{v})^2}{\frac{Nn}{v}}$$

PAS 3. Dacă $X_L^2 < u_{0,9973}(v-n)$ (cuantila de ordinul $0,9973$ a distribuţiei $\chi^2(v-n)$) atunci **jocul este corect**, unde:
$u_{0,9973}(35) = 66,77$ (5 din 40);
$u_{0,9973}(43) = 66,1$(6 din 49).

4.10. Aplicaţii

Exerciţiul 4.10.1. Să se calculeze entropia $H(X)$ a variabilelor repartizate $Exp(\lambda)$, $N(\mu; \sigma^2)$ şi $U(a, b)$.

Exerciţiul 4.10.2. Să se determine distribuţiile de probabilitate de forma:

$$\begin{pmatrix} x_1 & \ldots & x_n \\ p_1 & \ldots & p_n \end{pmatrix}$$

care maximizează entropia dar au media egală cu m.

Răspuns. Problema de optimizare este:

$$\begin{cases} \max H(X) \\ \sum\limits_{i=1}^{n} p_i = 1 \\ \sum\limits_{i=1}^{n} x_i p_i = m. \end{cases}$$

Lagrangeianul va fi:

$$L = -\sum_{i=1}^{n} p_i \log p_i - \alpha(1 - \sum_{i=1}^{n} p_i) - \beta(m - \sum_{i=1}^{n} x_i p_i).$$

Sistemul:

$$\frac{\partial L}{\partial p_i} = 0 \quad \forall i = 1, \ldots, n$$

ne dă:

$$p_i = e^{-1+\alpha+\beta x_i},$$

unde α şi β se determină din restricţiile problemei.

Exerciţiul 4.10.3. Pentru testarea conformităţii implementării testelor statistice (mai în general pentru orice algoritm), se calculează teoretic ce valoare au funcţiile de test pentru un set de intrări particulare. Folosiţi ca intrări particulare secvenţele binare 11101110 ... 1110 (secvenţă de lungime 1024) şi 1010 ... 1010 (secvenţă de lungime 1024) pentru calculul funcţiilor de test din acest capitol. Acelaşi lucru pentru secvenţa de lungime 256 (exprimată în caractere hexa) 00,01,02, ..., FF.

Exerciţiul 4.10.4. Calculaţi valorile minime şi maxime ale funcţiilor de testare statistică din acest capitol. Cum interpretaţi rezultatele?

Răspuns. Vom exemplifica pe statistica testului frecvenţei. Avem

$$-\frac{Np_0}{\sqrt{Np_0q_0}} \leq f_{T_F}(s^N) \leq \frac{Nq_0}{\sqrt{Np_0q_0}},$$

valorile extreme se ating pentru şirul $0, \ldots, 0$ respectiv $1, \ldots, 1$.

Exerciţiul 4.10.5. Cum se poate detecta rata de respingere a filtrării la care a fost supusă o secvenţă binară (prin filtrare înţelegem operaţia de triere a secvenţelor cu ajutorul unui test statistic efectuat la rata de respingere α).

Exerciţiul 4.10.6. Propuneţi o metodă de integrare a rezultatelor obţinute în urma aplicării de n ori ale aceluiaşi test statistic asupra a n eşantioane binare de volum m.

Exerciţiul 4.10.7. Care este distanţa de unicitate pentru un cifru bloc cu m biţi de cheie şi n biţi lungimea blocului de date?

Capitolul 5

CODIFICAREA IN ABSENŢA PERTURBAŢIEI

The logic of secrecy was the mirror image of the logic of information.
Colin Burke, 1994

5.1. Introducere

Pentru a optimiza transmisia printr-un canal de comunicaţie, informaţia care circulă prin acel canal de comunicaţie trebuie codificată (vezi *Guiaşu* [32]). Codificarea presupune însă şi construcţia unor tehnici şi metode matematice ce permit detectarea erorilor şi corectarea acestora (aceasta este diferenţa dintre teoria codurilor şi criptologie: protecţia contra erorilor ce apar pe canalul de comunicaţie, respectiv protecţia contra accesului neautorizat). Aceste tipuri de coduri se numesc coduri corectoare şi detectoare de erori şi nu vor face subiectul prezentei expuneri. În cele ce urmează vom presupune că dispunem de un canal de comunicaţie fără perturbaţie, ocupându-ne de problema optimizării transmisiei. După prezentarea unor definiţii generale din teoria codurilor, în care rolul central este ocupat de inegalitatea lui Kraft, vom prezenta algoritmii de codificare ai lui Huffman şi Fano. În final se prezintă teoremele de codificare optimală ale lui Shannon-Fano.

5.2. Codificarea în absenţa perturbaţiei

În acest subcapitol vom prezenta principalele definiţii din teoria codificării în absenţa pertubaţiei. Se prezintă inegalitatea lui Kraft referitoare la lungimile cuvintelor de cod ale unui cod instantaneu.

Să considerăm $\{A_0, \mu\}$ o *sursă de informaţie staţionară* (nu îşi modifică distribuţia de probabilitate în timp) astfel încât card$A_0 = a_0$. Să notăm *canalul fără perturbaţie* cu $\{A, e_{\{\omega\}}, A\}$ cu $e_{\{\omega\}}(S) = 1,\ \omega \in S$ şi $e_{\{\omega\}}(S) = = 0, w \notin S$, pentru orice $S \in \mathcal{F}_A = \mathcal{B}(\mathcal{C})$, unde $\mathcal{C} = \{[\alpha_1, \ldots, \alpha_n; t_1, \ldots, t_n], n \text{ finit}\}$ este familia cilindrilor n-dimensionali: $[\alpha_1, \ldots, \alpha_n; t_1, \ldots, t_n] = \{\omega \in A^I | x_{t_i} = = \alpha_i, i = 1, \ldots, n\}$.

Prin *procedeu de codificare* înţelegem o regulă prin care fiecare literă din A_0 este codificată într-un şir finit de semnale de cod din A. Fie $S(A)$ mulţimea tuturor şirurilor finite de semnale din A. Prin *cod* înţelegem o aplicaţie:

$$\varphi : A_0 \to S(A), \varphi(v_i) = x_{j_1} x_{j_2} \ldots x_{j_{n_i}},\ i = 1, \ldots, a_0$$

Observaţie. Uneori se numeşte cod mulţimea $\varphi(A_0)$.

Definiţia 5.2.1. (*Caracterizări ale codurilor*).

i) Codul φ se numeşte *cod singular* dacă există $i, j \in \{1, \ldots, a_0\},\ i \neq j$ astfel încât $\varphi(v_i) \neq \varphi(v_j)$.

ii) Codul φ se numeşte *cod nesingular* dacă pentru orice $i \neq j$ avem $\varphi(v_i) \neq \varphi(v_j)$.

iii) Codul φ se numeşte *cod uniform* dacă $n_i = n$ pentru $i = 1, \ldots, a_0$.

iv) Codul φ se numeşte cu *decodificare unică* dacă pentru orice succesiune de cuvinte cod există o singură succesiune de litere din A_0 ce poate fi codificată în respectiva succesiune de cuvinte cod.

v) Codul φ se numeşte *cod instantaneu* dacă nici un cuvânt de cod nu este prefixul altui cuvânt de cod.

Observăm că, codurile instantanee sunt cu decodificare unică.

Definiţia 5.2.2. (*Lungimea medie a cuvintelor de cod*). Fie $\{A_0, \mu\}$ o sursă staţionară şi $\mu[v_i] > 0,\ i = 1, \ldots, a_0$ cu $\sum\limits_{i=1}^{a_0} \mu[v_i] = 1$. Fie aplicaţia $\varphi : A_0 \to S(A), \varphi(v_i) = x_{j_1} x_{j_2} \ldots x_{j_{n_i}},\ i = 1, \ldots, a_0$ un cod instantaneu. *Lungimea medie* a cuvintelor de cod este definită prin formula:

$$L = \sum_{i=1}^{a_0} n_i \mu[v_i].$$

Următoarea teoremă ne furnizează o condiţie necesară şi suficientă ca o mulţime de numere naturale să fie lungimile unui cod instantaneu.

Teorema 5.2.1. *(Inegalitatea lui Kraft). Dacă $a_0 =$ cardA_0 şi $a =$ card$A$ atunci numerele pozitive $n_1, \ldots, n_{a_0}$ pot fi lungimile cuvintelor unui cod instantaneu*

dacă şi numai dacă:

$$\sum_{i=1}^{a_0} a^{-n_i} \leq 1.$$

Demonstraţie. *Necesitatea.* Fie φ un cod instantaneu şi $n_1, \ldots, n_{a_0}$ lungimile cuvintelor de cod.

Fie $n = \max_{1 \leq i \leq a_0} n_i$ şi $r_i = \{j | n_j = i\}, 1 \leq \ i \leq \ n$ numărul cuvintelor de cod de lungime i. Atunci:

$$\sum_{i=1}^{a_0} a^{-n_i} = \sum_{i=1}^{n} r_i a^{-i}$$

Dar

$$\begin{aligned} r_1 &\leq a, \\ r_2 &\leq (a - r_1)a = a^2 - r_1 a, \\ r_3 &\leq [(a - r_1)a - r_2]a = a^3 - r_1 a^2 - r_2 a, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ r_n &\leq a^n - r_1 a^{n-1} - r_2 a^{n-2} - \ldots - r_{n-1} a, \end{aligned}$$

din ultima relaţie rezultă $\sum\limits_{i=1}^{n} r_i a^{-i} \leq 1$ adică $\sum\limits_{i=1}^{a_0} a^{-n_i} \leq 1$.

Suficienţa. Fie $n_1, \ldots, n_{a_0}$ întregi pozitivi cu proprietatea $\sum\limits_{i=1}^{a_0} a^{-n_i} \leq 1$.

Fie $n = \max_{1 \leq i \leq a_0} n_i$ şi $r_i = \{j | n_j = i\}, 1 \leq \ i \leq \ n$. Atunci ipoteza devine $\sum\limits_{i=1}^{n} r_i a^{-i} \leq 1$, adică:

$$\begin{cases} \frac{r_1}{a} \leq 1 \Rightarrow r_1 \leq a, \\ \frac{r_1}{a} + \frac{r_2}{a^2} \leq 1 \Rightarrow r_2 \leq (a - r_1)a, \\ \frac{r_1}{a} + \frac{r_2}{a^2} + \frac{r_3}{a^3} \leq 1 \Rightarrow r_3 \leq [(a - r_1)a - r_2]a, \\ \ldots\ldots\ldots\ldots\ldots\ldots \end{cases}$$

deci $n_i, 1 \leq i \leq a_0$ pot fi lungimile cuvintelor unui cod instantaneu. ∎

Următoarea teoremă dă o margine inferioară a lungimii medii a unui cod instantaneu.

Teorema 5.2.2. *(Marginea inferioară a lungimii medii).* *Fie* $\{A_0, \mu\}$ *o sursă staţionară şi* $\mu[v_i] > 0, \ i = 1, \ldots, a_0$ *cu* $\sum\limits_{i=1}^{a_0} \mu[v_i] = 1$ *şi fie:*

$$H_1 = -\sum_{i=1}^{a} \mu[v_i] \log_2 \mu[v_i].$$

Fie $\varphi : A_0 \to S(A)$ cod instantaneu. Atunci:

$$L \geq \frac{H_1}{\log_2 a}.$$

Demonstraţie.

$$\begin{aligned} H_1 &= -\sum_{i=1}^{a} \mu[v_i] \log_2 \mu[v_i] \leq -\sum_{i=1}^{a_0} \mu[v_i] \log_2 \left(\frac{a^{-n_i}}{\sum_{j=1}^{a_0} a^{-n_j}} \right) \\ &= \log_2 \left(\sum_{j=1}^{a_0} a^{-n_j} \right) + \log_2 a \sum_{i=1}^{a} n_i \log_2 \mu[v_i] \leq L \log_2 a. \end{aligned}$$

∎

5.3. Codurile Fano şi Huffman

5.3.1. Algoritmul de implemenare a codului Fano

Etapele implementării codului Fano sunt următoarele:

PAS 1. Ordonăm literele v_i în ordinea descrescătoare a probabilităţii lor de apariţie.

PAS 2. Împărţim A_0 în a clase cât mai echiprobabile.

PAS 3. Asociem literelor dintr-o clasă un anumit simbol din A.

PAS 4. Repetăm procedeul pentru fiecare clasă trecând la subclase. Se face aceasta până când fiecare subclasă constă dintr-o singură literă.

PAS 5. Codul corespunzător unei litere constă în concatenarea simbolurilor asociate la pasul 3.

5.3.2. Algoritmul de implementare a codului Huffman

Exemplificăm codul Huffman pentru cazul în care card $A_0 = a_0$ şi card $A = a(= 2)$, presupunem că $a_0 = k(a-1) + a$ (dacă $a_0 \neq k(a-1) + a$ atunci se adaugă fictiv litere de probabilitate zero pentru a atinge această condiţie.)

PAS 1. Ordonăm $v_i, i = 1, \ldots, a_0$ în ordine descrescătoare a probabilităţilor v_i.

PAS 2. Înlocuim ultimele a litere cu o nouă literă compusă de probabilitate egală cu suma probabilităţilor literelor din care provine.

PAS 3. Celor a litere de la pasul 2 le atribuim câte un simbol din A.

PAS 4. Reordonăm mulţimea formată din $a_0 - a$ litere iniţiale şi litera compusă.
PAS 5. Reluăm procedeul de la pasul 2.
PAS 6. Codul corespunzător unei litere este format din concatenarea literelor asociate la pasul 3.

5.4. Coduri optime

Deoarece comunicaţia presupune o serie de costuri de comunicaţie care uneori pot fi foarte mari, se pune în mod natural următoarea problemă: cum să îmbunătăţesc lungimea medie a cuvintelor de cod (cât mai aproape de marginea inferioară)? Procedeul de codificare este următorul: *secvenţe finite de litere din alfabetul iniţial se codifică în secvenţe finite de semnale cod.* Pentru sursa staţionară $\{A_0, \mu\}$, notăm cu $V_m^{(i)}$ blocurile m-dimensionale $1 \leq i \leq a_0^m$. Codificarea se face prin:

$$\varphi(V_m^{(i)}) = x_{j_1} x_{j_2} \dots x_{j_{n_i}},$$

şi lungimea medie a cuvintelor de cod va fi:

$$L_m = \sum_{i=1}^{a_0} n_i \mu(V_m^{(i)}).$$

Următoarele două teoreme ne asigură de existenţa codurilor asimptotic optime pentru surse Bernoulli respectiv pentru sursele staţionare.

Teorema 5.4.1. *(Shannon-Fano) Presupunem că $\{A_0, \mu\}$ este sursă Bernoulli $\{A_0, p_v\}$ (probabilitatea de apariţie a unui cuvânt este egală cu produsul probabilităţilor de apariţie a literelor individuale) şi fie H_0 entropia sa (avem evident $H_0 = H_1 = -\sum_{i=1}^{a_0} p_i \log_2 p_i$). Atunci pentru orice număr întreg pozitiv m există un cod instantaneu a cărui lungime medie L_m verifică:*

$$\lim_{m\to\infty} \frac{L_m}{m} = \frac{H_0}{\log_2 a}.$$

Teorema 5.4.2. *(Shannon-Fano) Presupunem că $\{A_0, \mu\}$ este sursă staţionară şi fie H_0 entropia sa (în general $H_0 \neq H_1$). Atunci pentru orice număr întreg pozitiv m există un cod instantaneu a cărui lungime medie L_m verifică:*

$$\lim_{m\to\infty} \frac{L_m}{m} = \frac{H_0}{\log_2 a}.$$

5.5. Aplicaţii

Exerciţiul 5.5.1. Să se construiască *codul Fano* pentru alfabetul format din literele $v_i, \quad i = \overline{1,8}$ cu probabilităţile $0,35;\ 0,15;\ 0,15;\ 0,13;\ 0,12;\ 0,06;\ 0,03;\ 0,01$.

Răspuns. Se aplică algoritmul corespunzător. Obţinem clasele:
Iteraţia 1. $\{v_1, v_2\}, \{v_3, v_4, v_5, v_6, v_7, v_8\}$.
Iteraţia 2. $\{v_1\}, \{v_2\}, \{v_3, v_4\}, \{v_5, v_6, v_7, v_8\}$.
Iteraţia 3. $\{v_3\}, \{v_4\}, \{v_5\}, \{v_6, v_7, v_8\}$.
Iteraţia 4. $\{v_6\}, \{v_7, v_8\}$.
Iteraţia 5. $\{v_7\}, \{v_8\}$.
Obţinem codul:
$\varphi(v_1) = 11$;
$\varphi(v_2) = 10$;
$\varphi(v_3) = 011$;
$\varphi(v_4) = 010$;
$\varphi(v_5) = 001$;
$\varphi(v_6) = 0001$;
$\varphi(v_7) = 00001$;
$\varphi(v_8) = 00000$;
Lungimea medie a cuvintelor codului este: $L = \sum\limits_{i=1}^{a_0} n_i \mu[v_i] = 2.64$.
Entropia sursei de informaţie este:

$$H_1 = -\sum_{i=1}^{a_0} \mu[v_i] \log_2 \mu[v_i] = 2.5625.$$

Exerciţiul 5.5.2. Să se construiască *codul Huffman* pentru alfabetul format din literele v_i, $i = \overline{1,8}$ cu probabilităţile $0,35;\ 0,15;\ 0,15;\ 0,13;\ 0,12;\ 0,06;\ 0,03;\ 0,01$.

Răspuns. Aplicând algoritmul corespunzător obţinem:
Iteraţia 1, nu este necesară reordonare :

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_2) = 0.15; \\ \mu(v_3) = 0.15; \\ \mu(v_4) = 0.13; \\ \mu(v_5) = 0.12; \\ \mu(v_6) = 0.06; \\ \mu(v_7) = 0.03 \to 1 \\ \mu(v_8) = 0.01 \to 0 \end{cases}$$

Iteraţia 2, nu este necesară reordonare:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_2) = 0.15; \\ \mu(v_3) = 0.15; \\ \mu(v_4) = 0.13; \\ \mu(v_5) = 0.12; \\ \mu(v_6) = 0.06 \to 1 \\ \mu(v_{78}) = 0.04 \to 0 \end{cases}$$

Iteraţia 3, este necesară reordonare:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_2) = 0.15; \\ \mu(v_3) = 0.15; \\ \mu(v_4) = 0.13; \\ \mu(v_5) = 0.12 \to 1 \\ \mu(v_{678}) = 0.1 \to 0 \end{cases}$$

Se vor reordona simbolurile:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_2) = 0.15; \\ \mu(v_3) = 0.15; \\ \mu(v_4) = 0.13; \\ \mu(v_{5678}) = 0.22. \end{cases}$$

Iteraţia 4, este necesară reordonare:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_{5678}) = 0.22; \\ \mu(v_2) = 0.15; \\ \mu(v_3) = 0.15 \to 1 \\ \mu(v_4) = 0.13 \to 0 \end{cases}$$

Se vor reordona simbolurile:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_{5678}) = 0.22; \\ \mu(v_2) = 0.15; \\ \mu(v_{34}) = 0.28. \end{cases}$$

Iteraţia 5, este necesară reordonare:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_{34}) = 0.28; \\ \mu(v_{5678}) = 0.22 \to 1 \\ \mu(v_2) = 0.15 \to 0 \end{cases}$$

Se vor reordona simbolurile:

$$\begin{cases} \mu(v_1) = 0.35; \\ \mu(v_{34}) = 0.28; \\ \mu(v_{25678}) = 0.37. \end{cases}$$

Iteraţia 6, este necesară reordonare:

$\begin{cases} \mu(v_{25678}) = 0.37; \\ \mu(v_1) = 0.35 \to 1 \\ \mu(v_{34}) = 0.28 \to 0 \end{cases}$

Se vor reordona simbolurile:

$\begin{cases} \mu(v_{25678}) = 0.37 \\ \mu(v_{134}) = 0.63 \end{cases}$

Iteraţia 7, după care se opreşte algoritmul:

$\begin{cases} \mu(v_{134}) = 0.63 \to 1 \\ \mu(v_{25678}) = 0.37 \to 0 \end{cases}$

Obţinem codul:

$\varphi(v_1) = 11;$
$\varphi(v_2) = 00;$
$\varphi(v_3) = 101;$
$\varphi(v_4) = 100;$
$\varphi(v_5) = 011;$
$\varphi(v_6) = 0101;$
$\varphi(v_7) = 01001;$
$\varphi(v_8) = 01000;$

Lungimea medie a cuvintelor codului este: $L = \sum_{i=1}^{a_0} n_i \mu[v_i] = 2.64.$

Entropia sursei de informaţie este:

$$H_1 = -\sum_{i=1}^{a_0} \mu[v_i] \log_2 \mu[v_i] = 2.5625.$$

Exerciţiul 5.5.3. Să se decodifice secvenţa cod 100011010101001 ştiind că aceasta corespunde codului de la exerciţiul 5.5.2.

Exerciţiul 5.5.4. Să se construiască *codul Huffman* pentru alfabetul format din literele v_i, $i = \overline{1,8}$ cu probabilităţile $1/8$. Care este lungimea medie a cuvintelor de cod? Cum este această lungime de cod pentru *codul Fano*?

Exerciţiul 5.5.5. Să se construiască *codurile Huffman* şi *Fano* pentru alfabetele limbii române şi engleză.

Răspuns. Pe baza unor fişisere text (eşantioane) se vor estima probabilităţile literelor, cifrelor şi a semnelor de punctuaţie din limba studiată. Pe baza acestor probabilităţi se vor construi codurile respective.

Exerciţiul 5.5.6. Studiaţi din punct de vedere al complexităţii de implementare codurile Fano şi Huffman.

Exerciţiul 5.5.7. Construiţi un cod cu:

-decodificare unică, dar care să nu fie instantaneu, de la $\mathbf{Z}_m$ la $\mathbf{Z}_n (m > n)$;

-instantaneu, deci cu decodificare unică, de la $\mathbf{Z}_m$ la $\mathbf{Z}_n (m > n)$;

Exerciţiul 5.5.8. Deduceţi formulele de conversie din baza de numeraţie m în baza de numeraţie n.

Exerciţiul 5.5.9. Codul *MORSE* este un cod cu decodificare unică? Care este lungimea medie a cuvintelor de cod pentru limba română, franceză şi engleză?

Capitolul 6

CRIPTANALIZA CIFRURILOR CLASICE

Deciphering is, in my opinion, one of the most fascinanting of arts, and I fear I have wasted upon it more time than it deserves.
Charles Babbage, 1864

6.1. Substituţia simplă şi multiplă

6.1.1. Substituţia simplă

Operaţia de cifrare se bazează pe o *corespondenţă* biunivocă între *alfabetul clar* notat prin $\mathcal{A}$ şi *alfabetul cifrat* notat prin $\mathcal{C}$. Pentru exemplificarea ideilor vom prezupune că alfabetul clar este format din cele 26 de litere ale *limbii române* (fără diacritice) plus *delimitatorul de cuvânt* spaţiul. Alfabetul cifrat poate fi format din aceeleaşi caractere sau doar din cele 26 de litere ale limbii române caz în care spaţiul se va înlocui cu cea mai puţin frecventă literă (Q) sau se va ignora pur şi simplu.

Corespondenţa dintre cele două alfabete (presupunem că delimitatorul de cuvânt este înlocuit cu litera Q) poate fi:

-aleatoare;

-pseudoaleatoare: plecând de la o parolă se construieşte alfabetul cifrat.

Dacă în cazul corespondenţei aleatoare lucrurile sunt cât se poate de clare, vom prezenta pe scurt o metodă de construcţie a corespondenţei în cel de-al doilea caz. Pornind de la o parolă, alfabetul cifrat este construit după următorul algoritm:

-se scriu, o singură dată, în ordinea apariţiei, literele din parolă;

-se scriu literele alfabetului ce nu apar în parolă.

Corespondenţa între cele două alfabete se realizează după regula alfabet în alfabet după o permutare fixă σ (aceasta poate fi chiar permutarea identică iar la decriptare se aplică aceeaşi procedeu dar cu inversa permutării σ).

În funcţie de forma permutării substituţia se numeşte:

-*directă* (alfabetul cifrat are acelaşi sens lexicografic cu alfabetul clar, sunt în total 26 astfel de substituţii). Exemplu de substituţie directă:

A	B	C	D	E	F	G	H	I	J	K	L	M
G	H	I	J	K	L	M	N	O	P	Q	R	S

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
T	U	V	W	X	Y	Z	A	B	C	D	E	F

-*inversă* (alfabetul cifrat are sens invers lexicografic cu alfabetul clar, sunt în total 26 de astfel de substituţii). Exemplu de substituţie inversă:

A	B	C	D	E	F	G	H	I	J	K	L	M
U	T	S	R	Q	P	O	N	M	L	K	J	I

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	G	F	E	D	C	B	A	Z	Y	X	W	V

Reamintim aici trei exemple celebre (vechile coduri ebraice) de substituţii reciproce (dacă litera $\mathcal{X}$ se substituie cu litera $\mathcal{Y}$ atunci $\mathcal{Y}$ se va substitui cu $\mathcal{X}$) şi anume:

-*atbash* (prima jumătate a literelor alfabetului se mapează în cea de-a două jumătate în ordine invers lexicografică):

A	B	C	D	E	F	G	H	I	J	K	L	M
Z	Y	X	W	V	U	T	S	R	Q	P	O	N

-*albam* (prima jumătate a literelor alfabetului se mapează în cea de-a două jumătate în ordine lexicografică):

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

-*atbah*:

A	B	C	D	J	K	L	M	E	S	T	U	V
I	H	G	F	R	Q	P	O	N	Z	Y	X	W

Definiţia 6.1.1. Un *cifru de substituţie liniar* de la $\mathbf{Z}_m$ la $\mathbf{Z}_m$ (m fiind numărul de caractere al alfabetului sursă) poate fi descris prin funcţia $f : \mathbf{Z}_m \to \mathbf{Z}_m$ definită prin $f(x) = \alpha x + \beta$ cu $\gcd(\alpha, m) = 1$, funcţia de descifrare fiind $f^{-1}(x) = \alpha^{-1}(x - \beta)$. Cheia de cifrare sunt numerele α şi β.

Observaţia 6.1.1. Primul manual de analiză criptografică a cifrurilor de substituţie a fost scris de *Al-Kindi* [2] în anul 850 AD.

Observaţia 6.1.2. Cifrul de substituţie are *proprietatea de confuzie* (ascunderea legăturii dintre textul clar şi textul cifrat).

6.1.2. Substituţia multiplă

În cazul substituţiei multiple (*substituţie poligrafică*) M caractere din textul clar sunt substituite în N caractere de text cifrat. Evident pentru ca operaţia de cifrare să fie reversibilă trebuie ca $M \geq N$. Dacă $M = N = 2$ atunci cifrul se numeşte cifru de *substituţie digrafică*.

Definiţia 6.1.2. Un cifru de *substituţie poligrafică liniar* de la $\mathbf{Z}_m^k$ la $\mathbf{Z}_m^k$ (m fiind numărul de caractere al alfabetului sursă şi k dimensiunea $k-$gramei) poate fi descris prin funcţia $f : \mathbf{Z}_m^k \to \mathbf{Z}_m^k$ definită prin $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \boldsymbol{\beta}$ cu matricea $\mathbf{A}$ inversabilă, funcţia de descifrare fiind $f^{-1}(\mathbf{y}) = \mathbf{A}^{-1}(\mathbf{y} - \boldsymbol{\beta})$. Cheia de cifrare este matricea $\mathbf{A}$ şi vectorul $\boldsymbol{\beta}$.

Identificarea cifrului de substituţie poligrafică

Un cifru de substituţie poligrafică se poate detecta foarte uşor după frecvenţa $N-$gramelor. Distribuţia acestora este departe de a fi uniformă. Evident, o condiţie necesară de identificare a acestui tip de cifru este aceea ca dimensiunea mesajului cifrat să fie suficient de mare. O procedură de detectare a lui N se bazează, de exemplu, pe minimul entropei H calculate din $R-$grame:

$$H(m; N) = \min_R H(m; R).$$

Alte proceduri de identificare corectă a parametrului n sunt prezentate în secţiunea *Proceduri de identificare a sistemului.* Pentru a se asigura biunivocitatea parametrul M trebuie să fie egal cu N. Construcţia tabelei de substituţie poligrafică se realizează cu ajutorul frecvenţei $N-$gramelor din textul cifrat.

Cifrul celor 4(2) tabele rectangulare

În continuare vom prezenta două sisteme de cifrare de tip substituţie digrafică. Literele alfabetelor (clar şi cifrat) sunt trecute într-un careu de 5×5 (litera I fiind asimilată literei J). Modul de aranjare al literelor albabetelor clar este fixat apriori (de exemplu ordine lexicografică) iar al alfabetelor cifrate în funcţie de parolă. Textul clar este preprocesat astfel încât acesta să fie compatibil cu matricea de cifrare (delimitatorul de cuvânt este ignorat sau este înlocuit cu cea mai puţin frecventă literă, litera I este asimilată cu litera J, şi în fine dacă este cazul mai adjuncţionăm o literă la text pentru a avea un număr par de digrame). Pentru cifrul celor 4 tabele regula de cifrare este dată de regula dreptunghiului: o digramă din textul clar se cifrează în digrama corespunzătoare diagonalei secundare a dreptunghiului determinat de cele două caractere ale digramei clare. Modul de aranjare al celor patru alfabete ($P1$ şi $P2$ sunt alfabetele clare iar $C1$ şi $C2$ sunt alfabetele cifrate în funcţie de parolă) este prezentat mai jos:

P1	C1
C2	P2

.

Algoritmul prezentat mai sus degenerează în algoritmul celor două tabele (verticale sau orizontale) după cum urmează: $P1 \equiv C1$ şi $P2 \equiv C2$ respectiv $P1 \equiv C2$ şi $P2 \equiv C1$. Dacă literele ce formează digrama se află pe aceeaşi coloană, respectiv linie, atunci reprezentarea cifrată a acestora sunt ele însele, respectiv acestea scrise în ordine inversă. Cele două tabele de cifrare sunt:

P1[C1]
C2[P2]

,

respectiv

P1[C2]	P2[C1]

.

Cifrul Playfair

Cifrul Playfair (numele acestui sistem de cifrare provine de la Lordul englez Playfair) este unul dintre cele mai cunoscute sisteme de cifrare digrafice (transformă un grup de 2 litere într-un grup de alte două litere). Acest sistem de cifrare este foarte simplu de folosit dar mult mai sigur decât sistemele de substituţie monoalfabetice. Descriem în continuare modul de utilizare al acestui sistem de cifrare. Literele alfabetului sunt trecute într-un careu de 5×5 (litera I fiind asimilată literei J). Textul clar este preprocesat astfel încât acesta să fie compatibil cu matricea de cifrare (delimitatorul de cuvânt este ignorat sau este înlocuit cu cea mai puţin frecventă literă,

litera I este asimilată cu litera J, şi în fine dacă este cazul mai adjuncţionăm o literă la text pentru a avea un număr par de digrame). Regula de cifrare este următoarea:

i) Dacă digrama ce se doreşte cifrată nu are literele pe aceeiaşi linie sau coloană, atunci regula de cifrare este *regula dreptunghiului*, traseul fiind pe verticală de la cea de-a doua literă a digramei către prima literă.

ii) Dacă digrama ce se doreşte cifrată are literele pe aceeiaşi linie, atunci se aplică regula: *cifrează la dreapta, descifrează la stânga.*

iii) Dacă digrama ce se doreşte cifrată are literele pe aceeiaşi coloană, atunci se aplică regula: *cifrează în jos, descifrează în sus.*

Observaţia 6.1.3. Dacă o digramă apare în textul clar în ordine inversă atunci acelaşi lucru se va întâmpla şi în textul cifrat.

Observaţia 6.1.4. Cifrul Playfair nu are regulă pentru cifrarea literelor duble: digramele ce conţin două litere identice sunt sparte prin introducerea artificială a unei alte litere.

Observaţia 6.1.5. Cifrul Playfair apare ca o extindere, în sensul reducerii numărului de tabele rectangulare folosite (de la două la unul), al cifrului cu 2 tabele.

Metoda cea mai freventă de atac a acestui tip de cifru constă în analiza frecvenţei digramelor de text clar combinată cu metoda comparaţiei patternurilor din textul cifrat cu patternuri din dicţionar. Spectaculos este faptul că în *Manualul Teroristului* [104], în capitolul 13 intitulat *Scrieri Secrete, Coduri şi Cifruri*, se prezintă tehnici steganografice, tehnici de codificare şi o serie de metode de cifrare clasice cum ar fi substituţia simplă şi substituţia multiplă.

6.2. Substituţia polialfabetică

6.2.1. Caracteristicile şi identificarea sistemelor de substituţie polialfabetică

Un sistem de cifrare de tip substituţie polialfabetică este generalizarea sistemului de cifrare de substituţie monoalfabetică. Fie un sistem de cifrare polialfabetic compus dintr-un număr N de alfabete. Fiecare alfabet reprezintă o permutare (stabilită în funcţie de parolă) a alfabetului de intrare. Algoritmul de cifrare constă în substituirea celei de a $i-$a litere m din textul clar cu litera corespunzătoare din cel de al $i \bmod N$ alfabet. Sistemele polialfabetice sunt uşor de identificat prin aplicarea tehnicii frecvenţelor secvenţelor decimate din textul cifrat sau a valorii maxime a funcţiei $Kappa$ (v. paragraful Proceduri de identificare a sistemului).

6.2.2. Atacul sistemelor polialfabetice

Atacul sistemelor polialfabetice este similar cu atacul a N sisteme de substituţie monoalfabetică. Deci, o procedură de *tip divide et impera* are o compexitate de $O(N)$. Procedura este descrisă în continuare:

Intrare: Textul cifrat de lungime M suficient de mare.

Ieşire: Textul clar corespunzător sistemului de cifrare polialfabetic.

Pas 1. Determină numărul de alfabete N.

Pas 2. Pentru $j = 0$ to 4 execută:

pentru $i = 1$ to $N - j$ execută:

aplică procedura de reconstrucţie parţială (pe baza frecvenţelor $(j+1)-$gramelor) a alfabetelor $i, \ldots, i+j$.

Pas 3. Conform celor N alfabete reconstruieşte textul clar.

Observaţia 6.2.1. Procedura descrisă mai sus are ca parametru implicit de analiză numărul maxim de legături 4 : astfel, $1-$gramele sunt caracterele, $2-$gramele sunt dubleţii, etc.

6.3. Soluţia unui cifru de substituţie

În criptanaliză, soluţia unui cifru de substituţie parcurge următoarele etape:

a) *analiza criptogramelor:*

a1) pregătirea unui tabel de frecvenţe;

a2) căutarea repetiţiilor;

a3) determinarea tipului de sistem utilizat;

a4) pregătirea unei foi de lucru;

a5) pregătirea unui alfabet individual;

a6) tabelarea repetiţiilor lungi.

b) *Clasificarea vocalelor şi consoanelor prin studierea:*

b1) frecvenţelor;

b2) spaţiilor;

b3) combinaţiilor de litere;

b4) repetiţiilor.

c) *Identificarea literelor:*

c1) partiţionarea literelor în clase de probabilitate;

c2) verificarea presupunerilor;

c3) înlocuirea valorilor corecte în criptogramă;

c4) descoperirea altor valori pentru a avea soluţia completă.

d) *Reconstrucţia sistemului:*

d1) reconstrucţia tabelei de cifrare;

d2) reconstrucţia cheilor folosite în operaţia de cifrare;

d3) reconstrucţia cheilor sau a cuvintelor cheie ce au fost folosite pentru construcţia şirurilor de alfabete.

6.4. Transpoziţia

Dacă substituţia schimbă valoarea unui caracter din textul cifrat, permutarea schimbă poziţia pe care acest caracter apare în textul cifrat şi nu valoarea sa. Operaţia de cifrare prin permutarea $\sigma \in S_N$ se realizează prin permutarea caracterelor din blocurile adiacente, de lungime N, ale textului clar. Un caz particular al permutării este aceela al transpoziţiei de lungime N. Textul este scris într-o tabelă (completă sau nu) cu N coloane, literele fiind scrise linie cu linie. Pornind de la o parolă literală se construieşte o parolă numerică (spre exemplu se asociează fiecărei litere din parolă poziţia sa în scrierea lexicografică a acesteia). Textul este apoi citit coloană cu coloană într-o ordine stabilită apriori (crescător, descrescător, etc.). Pentru descifrare se aplică acelaşi algoritm dar cu parola numerică σ^{-1}.

Observaţia 6.4.1. Cifrul de transpoziţie (mai general cifrul de permutare) are *proprietatea de difuzie* (disiparea redundanţei textului clar de-a lungul textului cifrat).

6.5. Sisteme mixte

Sistemele mixte de cifrare au la bază o cifrare succesivă a mesajului prin metoda substituţiei iar apoi prin metoda transpoziţiei sau viceversa. Tot ceea ce trebuie să facem acum este să atacăm sistemul de cifrare de la ultima sa componentă către prima. Remarcăm faptul că în cazul substituţiei simple aceasta este comutativă cu operaţia de transpoziţie deci se poate aborda mai întâi substituţia iar apoi transpoziţia. În cazul utilizării unui sistem polialfabetic, cu număr necunoscut de alfabete, recomandarea este ca după stabilirea, prin metode statistice, a numărului de alfabete, să se abordeze concomitent identificarea efectivă a alfabetelor şi al transpoziţiei utilizate. În cazul utilizării unui sistem poligrafic (tabele de cifrare) şi o transpoziţie este recomandabilă o tehnică de tip backtracking.

6.6. Proceduri de identificare a sistemului

Procedurile de criptanaliză prezentate în cadrul acestui paragraf sunt bazate pe calculul unor estimatori pentru o serie de funcţii de test (vezi *Friedman* [20], *Bauer* [6], *Preda* şi *Simion* [58]). Textul clar trebuie să fie omogen din punct de vedere

statistic. Dacă textul nu este omogen, atunci, cu ajutorul unei proceduri, acesta se poate diviza în părţi omogene. Procedurile ce urmează ne permit să identificăm modelul de cifrare, structura statistică a textului clar şi în cele din urmă soluţia problemei.

Să notăm prin $T = (t_1, \ldots, t_M)$ şi $T^{'} = (t_1^{'}, \ldots, t_M^{'})$ două şiruri de lungime M din acelaşi vocabular Z_N (de lungime N). Notăm prin m_i şi $m_i^{'}$ frecvenţele de apariţie ale celei de a $i-a$ litere din alfabetul sursă din şirul T respectiv $T^{'}$. Avem deci:

$$\sum_{i=1}^{N} m_i = M$$

şi

$$\sum_{i=1}^{N} m_i^{'} = M.$$

În cele ce urmează vom descrie funcţiile de decizie *Kappa, Chi, Psi* şi *Phi*.

6.6.1. Funcţia Kappa

Funcţia de test Kappa este definită prin:

$$Kappa(T, T^{'}) = \frac{\sum\limits_{i=1}^{M} \delta(t_i, t_i^{'})}{M},$$

unde δ este simbolul lui *Kronecker.*

Avem inegalităţile:

$$0 \leq Kappa(T, T^{'}) \leq 1,$$

cu egalitate în partea stângă pentru $t_i \neq t_i^{'}$ (pentru orice i) respectiv $T \equiv T^{'}$. Testul Kappa este similar cu testul de corelaţie. Câteodată $Kappa$ se va nota mai simplu prin K.

Avem următoarele teoreme de invarianţă (vezi *Bauer* [6]):

Teorema 6.6.1. *Pentru toate sistemele de cifrare de tip substituţie polialfabetică valoarea lui Kappa a două texte de lungimi egale, cifrate cu aceeaşi cheie, este invariantă.*

Teorema 6.6.2. *Pentru toate sistemele de cifrare de tip transpoziţie valoarea lui Kappa a două texte de lungimi egale, cifrate cu aceeaşi cheie, este invariantă.*

Valoarea medie a lui $Kappa(T, T^{'})$, pentru T şi $T^{'}$ definite de două surse Q respectiv $Q^{'}$ (cu probabilitatea simbolurilor p_i respectiv $p_i^{'}$ pentru $i = = 1, \ldots, N$) este:

$$E[Kappa(T, T^{'})] = \sum_{i=1}^{N} p_i^{'} p_i.$$

Dacă sursele Q şi $Q^{'}$ sunt identice (acest lucru se notează prin $Q \equiv Q^{'}$), adică $p_i = p_i^{'}$, atunci:

$$E[Kappa(T, T^{'})] = \sum_{i=1}^{N} p_i^2.$$

Teorema 6.6.3. *Pentru două surse identice Q şi $Q^{'}$ are loc inegalitatea:*

$$\frac{1}{N} \leq E[Kappa(T, T^{'})] \leq 1,$$

cu egalitate în partea stângă pentru distribuţia uniformă, iar în partea dreaptă pentru sursa deterministă (adică există un indice i pentru care $p_i = 1$).

Observaţia 6.6.1. Dispersia funcţiei Kappa se calculează după formula:

$$D^2(X) = E((E(X) - X)^2).$$

Observaţia 6.6.2. Funcţia Kappa poate fi folosită la atacul sistemelor polialfabetice: diferenţa dintre valorile maxime ale valorilor $Kappa(T^{(i)}, T)$ (am notat prin $T^{(i)}$ textul T deplasat ciclic cu i poziţii la dreapta) este divizibil cu numărul de alfabete utilizate. Pentru a găsi alfabetele utilizate se poate aplica o tehnică de tip divide et impera şi o procedură bazată pe maximul frecvenţei sau atac de tip stereotip.

6.6.2. Funcţia Chi

Funcţia Chi este definită prin formula:

$$Chi(T, T^{'}) = \frac{\sum_{i=1}^{N} m_i m_i^{'}}{M^2}.$$

Vom nota această funcţie şi prin litera grecească χ. Au loc următoarele teoreme de invarianţă (vezi *Bauer* [6]):

Teorema 6.6.4. *Pentru toate sistemele de cifrare de tip substituţie monoalfabetică valoarea lui Chi a două texte de lungimi egale, cifrate cu aceeaşi cheie, este invariantă.*

Teorema 6.6.5. *Pentru toate sistemele de cifrare de tip transpoziţie valoarea lui Chi a două texte de lungimi egale, cifrate cu aceeaşi cheie, este invariantă.*

Avem inegalitatea:

$$Chi(T, T^{'}) \leq 1,$$

cu egalitate pentru $T \equiv T^{'}$. Pentru un text cu distribuţie uniformă T ($m_i = \frac{M}{N}$) şi un text arbitrar $T^{'}$ avem:

$$Chi(T, T^{'}) = \frac{1}{N}.$$

Valoarea medie a lui Chi a două texte T şi $T^{'}$ de lungime egală M peste aceeaşi vocabular Z_N se calculează din probabilităţile p_i şi $p_i^{'}$ ale frecveţei de apariţie ale celui de al i-lea caracter în sursele stochastice Q şi $Q^{'}$ ale textelor:

$$E[Chi(T, T^{'})] = \sum_{i=1}^{N} p_i p_i^{'}.$$

Dacă sursele Q şi $Q^{'}$ sunt identice atunci $p_i = p_i^{'}$ pentru orice i şi

$$E[Chi(T, T^{'})] = \sum_{i=1}^{N} p_i^2.$$

Avem inegalitatea:

$$\frac{1}{N} \leq E[Chi(T, T^{'})] \leq 1,$$

cu egalitate în partea stângă pentru distribuţia uniformă iar în partea dreaptă pentru sursa deterministă.

6.6.3. Funcţia Psi

Funcţia Psi este legată de funcţia Chi prin formula:

$$Psi(T) = Chi(T, T).$$

Vom nota această funcţie prin litera grecească ψ. În mod similar ca pentru funcţia χ avem următoarele teoreme de invarianţă (vezi *Bauer* [6]).

Teorema 6.6.6. *Pentru toate sistemele de cifrare de tip substituţie monoalfabetică valoarea lui Psi este invariantă.*

Teorema 6.6.7. *Pentru toate sistemele de cifrare de tip transpoziţie valoarea lui Psi este invariantă.*

Avem inegalitatea:

$$\frac{1}{N} \leq Psi(T) \leq 1,$$

cu egalitate în partea stângă pentru un text cu distribuţie uniformă ($m_i = \frac{M}{N}$), şi în partea dreaptă dacă T este construit dintr-un singur caracter.

Valoarea medie a lui Psi al unui text de lungime M din alfabetul Z_N se calculează din probabilităţile p_i ale frecvenţei de apariţie a celui de al i-lea caracter din sursa stochastică Q ce produce textul:

$$E[Psi(T)] = \sum_{i=1}^{N} p_i^2.$$

Are loc inegalitatea:

$$\frac{1}{N} \leq E[Psi(T)] \leq 1,$$

cu egalitate în partea stângă pentru un text cu distribuţie uniformă şi în partea dreaptă pentru o sursă deterministă.

Avem teorema $Kappa - Chi$ (vezi *Bauer* [6]):

Teorema 6.6.8. *Pentru două texte T şi T' peste aceeaşi vocabular şi de aceeaşi lungime, valoarea $Chi(T, T')$ este media aritmetică a valorilor $Kappa(T^{(i)}, T')$ ($T^{(i)}$ este textul T rotit ciclic cu i poziţii la dreapta obţinut prin formula $t_j^* = t_{(j-i-1) \bmod M+1}$ pentru $j = 1, \ldots, M$):*

$$Chi(T, T') = \frac{1}{M} \sum_{i=0}^{M-1} Kappa(T^{(i)}, T').$$

Corolarul următor este cunoscut sub numele de teorema $Kappa - Psi$:

Corolar 6.6.1. Pentru un text T de lungime M peste vocabularul Z_N avem:

$$Psi(T) = \frac{1}{M} \sum_{i=0}^{M-1} Kappa(T^{(i)}, T).$$

6.6.4. Funcţia Phi

Vom defini acum o nouă funcţie denumită *Phi* care este dată prin formula:

$$Phi(T) = \frac{\sum_{i=1}^{N} m_i(m_i - 1)}{M(M-1)}.$$

Funcţia *Phi* va fi notată prin litera grecească φ. În cazul în care $T \equiv T'$ avem $Kappa(T^{(0)}, T) = 1$, în timp ce pentru $i \neq 0$ valorile lui $Kappa(T^{(i)}, T)$ sunt relativ mici. Deci cazul $i = 0$ este atipic pentru medie şi deci este mult mai natural să extindem media numai peste cele $M - 1$ cazuri:

$$\frac{1}{M-1} \sum_{i=1}^{M-1} Kappa(T^{(i)}, T).$$

Avem teorema $Kappa - Phi$:

Teorema 6.6.9. (*Bauer* [6]) *Pentru un text T de lungime M peste vocabularul Z_N valoarea lui $Phi(T)$ este media valorilor $Kappa(T^{(i)}, T)$ ($T^{(i)}$ este textul T rotit ciclic cu i poziţii la dreapta):*

$$Phi(T) = \frac{1}{M-1} \sum_{i=1}^{M-1} Kappa(T^{(i)}, T).$$

Avem de asemenea următoarele teoreme de invarianţă (vezi *Bauer* [6]) pentru funcţia *Phi*.

Teorema 6.6.10. *Pentru toate sistemele de cifrare de tip substituţie monoalfabetică valoarea lui Phi este invariantă.*

Teorema 6.6.11. *Pentru toate sistemele de cifrare de tip transpoziţie, valoarea lui Phi este invariantă.*

Teorema 6.6.12. *Dacă T este o secvenţă distribuită uniform atunci*

$$Phi(T) = \frac{M - N}{(M-1)N}$$

unde M este lungimea secvenţei iar N numărul de simboluri (caractere).

Teorema 6.6.13. *Dacă notăm prin $Phi_c(T)$ valoarea funcţie Phi calculată pe $c-$grame atunci:*

$$\lim_{c\to\infty} Phi_c(T) = 0.$$

Valoarea c pentru care $Phi_c(T) = \max_r Phi_r(T)$ este dimensiunea blocului pe care se realizează codificarea datelor după procesul de cifrare.

Observaţia 6.6.3. i) Valoarea medie a lui $Phi(T)$ este egală cu valoarea medie a lui $Psi(T)$ deci este egală cu $\sum_{i=1}^{n} p_i^2$.

ii) Funcţia Phi este utilă deoarece literele rare ($m_i = 1$) sunt eliminate.

iii) $Phi(T) = 0$ dacă şi numai dacă $0 \leq m_i \leq 1$.

iv) Funcţia Phi este utilă în identificarea limbii folosite în textul clar, cu condiţia ca sistemul de cifrare folosit să fie o tranpoziţie şi/sau o substituţie monoalfabetică.

v) Funcţia Phi a fost propusă iniţial de *Solomon Kullback* din considerente stochastice.

6.7. Funcţii simetrice de frecvenţă a caracterelor

Teoremele de invarianţă pentru Psi au loc pentru toate funcţiile de frecvenţă m_i a caracterelor care sunt simetrice. Cea mai simplă funcţie polinomială netrivială este $\sum_{i=1}^{N} m_i^2$. Aceasta aparţine familiei de funcţii:

$$Psi_a(T) = \begin{cases} (\sum_{i=1}^{N} (\frac{m_i}{M})^a)^{\frac{1}{a-1}} & \text{dacă } 1 < a < \infty \\ \exp(\sum_{i=1}^{N} (\frac{m_i}{M}) \ln(\frac{m_i}{M})) & \text{dacă } a = 1 \\ \max_{1,\ldots,N} \frac{m_i}{M} & \text{dacă } a = \infty \end{cases}$$

cu $\sum_{i=1}^{N} \frac{m_i}{M} = 1$. Observăm că Psi_2 este Psi. Cantitatea logaritmică $-\ln Psi_a(T)$ se numeşte entropia *Rényi de ordinul a* a lui T (Alfred Rényi 1921-1970, matematician ungur) . Familia are următoarea reprezentare:

$$-\ln Psi_a(T) = \begin{cases} -\frac{1}{a-1} \ln(\sum_{i=1}^{N} (\frac{m_i}{M})^a) & \text{if } 1 < a < \infty \\ -\sum_{i=1}^{N} (\frac{m_i}{M}) \ln(\frac{m_i}{M}) & \text{if } a = 1 \\ -\max_{1,\ldots,N} \ln(\frac{m_i}{M}) & \text{if } a = \infty. \end{cases}$$

Entropia Rényi de ordinul 1 se numeşte *entropie Shannon* (Claude Shannon 1916-2000, inginer American, părintele teoriei informaţiei) iar entropia Rényi de ordinul 2 este *entropia Kullback.* Toate proprietăţile lui Psi se generalizează la proprietăţi ale lui Psi_a.

6.8. Atac stereotip asupra cifrurilor de substituţie

Fie $t_1, \ldots, t_M$ mesajul cifrat de lungime M (suficient de mare). Notăm prin $c_1, \ldots, c_N$ literele care apar în mesajul cifrat şi prin $p_1, \ldots, p_N$ frecvenţa de apariţie a acestora. Presupunem că sistemul de cifrare folosit este o substituţie monoalfabetică. Notăm prin $c_1^*, \ldots, c_N^*$ alfabetul textului clar şi prin $p_1^*, \ldots, p_N^*$ probabilitatea apariţiei literelor alfabetului. O metodă de rezolvare a acestui sistem de cifrare este atacul brut care constă în generarea tuturor permutărilor alfabetului. Numărul total de permutări ce trebuiesc generate este de $N!$ care este imens şi deci trebuie găsită o altă cale de atac. Tehnica este de a diviza mulţimea literelor din textul cifrat şi din alfabetul clar în K clase astfel încât dispersia din fiecare clasă este mică. Notăm prin m_i numărul de litere din cea de a $i-a$ clasă corespunzătoare textului cifrat şi prin n_i numărul de litere din cea de a $i-a$ clasă corespunzătoare alfabetului clar. Notăm cu $k_i = \max(m_i; n_i)$ şi $l_i = \min(m_i; n_i)$. Astfel trebuie generate numai $\prod_{i=1}^{K} A_{k_i}^{l_i}$ (avem $\sum_{i=1}^{K} n_i = \sum_{i=1}^{K} m_i = N$) număr relativ mic în raport cu $N!$. Pentru a realiza această clasificare vom proceda de exemplu pentru literele $c_1, \ldots, c_N$ după cum urmează:

1) Sortăm literele $c_1, \ldots, c_N$ în ordinea crescătoare a probabilităţilor $p_1, \ldots, p_N$. Notăm prin $\bar{c}_1, \ldots, \bar{c}_N$ şi prin $\bar{p}_1, \ldots, \bar{p}_N$ literele sortate, respectiv probabilităţile sortate.

2) Calculăm $\delta_i = \bar{p}_i - \bar{p}_{i-1}$ $(i = 2, \ldots, n)$. Cele mai mari $K-1$ valori ale lui δ_i sunt puncte delimitatoare ale claselor.

Observaţia 6.8.1. Pentru texte de lungime mică, în loc de probabilitatea p_i se foloseşte f_i frecvenţa absolută a literei i.

6.9. Atac de tip frecvenţă maximă a cifrurilor de substituţie

Fie $t_1, \ldots, t_M$ text clar de lungime M cu $t_i \in \{0, \ldots, 2^L - 1\}$ (textul este codificat pe L-biţi). Notăm prin $\sigma \in S_{2^L}$ cheia (permutarea) şi prin $c_1, \ldots, c_M$ textul permutat (cifrat). Energia informaţională a textului este:

$$\sum_{i=0}^{2^L-1} \left(\frac{f_i}{M}\right)^2$$

unde f_i este frecvenţa de apariţie a caracterului i ($i = 0, \ldots, 2^L - 1$). Utilizând inegalitatea Cauchy-Buniakovsky-Schwartz obţinem:

$$\sum_{i=0}^{2^L-1} \frac{f_i}{M} p_{\sigma_i} \leq \sqrt{\sum_{i=0}^{2^L-1} \left(\frac{f_i}{M}\right)^2 \sum_{i=0}^{2^L-1} p_{\sigma_i}^2},$$

unde p_{σ_i} este probabilitatea de apariţie a caracterului i în textul T. Deoarece

$$\sum_{i=0}^{2^L-1} \left(\frac{f_i}{M}\right)^2 = \sum_{i=0}^{2^L-1} p_{\sigma_i}^2,$$

vom obţine

$$\sum_{i=0}^{2^L-1} \frac{f_i}{M} p_{\sigma_i} \leq \sum_{i=0}^{2^L-1} \left(\frac{f_i}{M}\right)^2.$$

Avem egalitate dacă şi numai dacă:

$$p_{\sigma_i} = \frac{f_i}{M} \text{ pentru orice } i.$$

Deci pentru a obţine permutarea optimă σ (cheia sistemului de cifrare) trebuie rezolvată problema următoare:

$$\begin{cases} \max\limits_{\sigma} \sum\limits_{i=0}^{2^L-1} \frac{f_i}{M} \hat{p}_{\sigma_i} \\ \sum\limits_{i=0}^{2^L-1} \frac{f_i}{M} = 1, \end{cases}$$

unde $\hat{p}_{\sigma_i}$ este un estimator al lui p_{σ_i}.

Observaţia 6.9.1. Această procedură se poate folosi şi ca un test de confirmare a rezultatului obţinut prin atacul de tip stereotip.

6.10. Concluzii

Funcţiile de test definite în cadrul acestui capitol sunt invariante în cazul folosirii anumitor sisteme de cifrare cum ar fi sistemele de cifrare de tip transpoziţie şi de substituţie mono sau polialfabetică. Aceste funcţii de test pot fi folosite în identificarea limbii folosite în textul clar, a sistemului de cifrare folosit, a cheii şi în cele din urmă a textului clar. Funcţia *Kappa* este folosită în identificarea sistemului de cifrare şi în identificarea limbii folosite în textul clar. Procedura de identificare a

limbii se bazează pe compararea valorilor lui Psi şi Phi ale textului cifrat cu valorile lui Psi şi Phi ale fiecărei limbi (aceste funcţii de test devin în acest moment teste de confirmare). Funcţiile Chi sunt folosite în atacuri de *tip isolog* (texte diferite cifrate cu aceeiaşi cheie). Testele $Kappa, Chi, Psi$ şi Phi se pot efctua pe digrame sau trigrame, etc.

Alte aplicaţii criptografice sunt cele de tip căutare inteligentă a parolelor sau de tip dicţionar modificat (conform unor reguli lexicale).

Cele mai bune rezultate sunt obţinute dacă se analizează textul format numai din litere mari sau mici. Textele pot fi cu delimitator de cuvânt (spaţiu) sau fără delimitator de cuvânt.

Descrierea completă a acestor tipurilor de cifruri prezentate în cadrul acestui capitol precum şi principalele moduri de atac se poate găsi de exemplu în *Bauer* [6].

6.11. Aplicaţii

Exerciţiul 6.11.1. Care este diferenţa dintre proprietatea de confuzie şi proprietatea de difuzie a unui algoritm de cifrare?

Exerciţiul 6.11.2. Să se construiască alfabetul cifrat cu ajutorul parolei de cifrare $TESTARESISTEM$ iar apoi să se cifreze mesajul "IN CRIPTOGRAFIE NICI O REGULA NU ESTE ABSOLUTA". Permutarea ce realizează corespondenţa este:

0	1	2	3	4	5	6	7	8	9	10	11	12
25	24	23	22	21	20	19	18	17	16	15	14	13

13	14	15	16	17	18	19	20	21	22	23	24	25
12	11	10	9	8	7	6	5	4	3	2	1	0

Răspuns. Corepondenţa dintre alfabetul clar şi alfabetul cifrat (înainte de realizarea permutării) este:

A	B	C	D	E	F	G	H	I	J	K	L	M
T	E	S	A	R	I	M	B	C	D	F	G	H

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
J	K	L	N	O	P	Q	U	V	W	X	Y	Z

Corepondenţa dintre alfabetul clar şi alfabetul cifrat după realizarea permutării este:

A	B	C	D	E	F	G	H	I	J	K	L	M
Z	Y	X	W	V	U	Q	P	O	N	L	K	J

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	G	F	D	C	B	M	I	R	A	S	E	T

Mesajul clar se procesează astfel încât spaţiul este înlocuit cu cea mai puţin frecventă literă:

$$INQCRIPTOGRAFIEQNICIQREGULAQNUQESTEQABSOLUTA.$$

Mesajul cifrat va fi:

$$OHDXCOFMGQCZUOVDHOXODCVQIKZDHIDVBMVDZYBGKIMZ.$$

Exerciţiul 6.11.3. Să se cifreze mesajul "SI IN CRIPTOGRAFIE TACEREA ESTE AUR" cu ajutorul metodei celor 4 tabele iniţializate cu parolele de cifrare $CRIPTOGRAFIE$ şi $TEST$.

Exerciţiul 6.11.4. Să se construiască matricea de cifrare Playfair cu ajutorul parolei $CRIPTOGRAFIE$ iar apoi să se cifreze mesajul "SI IN CRIPTOGRAFIE TACEREA ESTE AUR".

Răspuns. Matricea Playfair se obţine trecând literele din parolă o singură dată în careul de 5×5, iar apoi celelalte litere ale alfabetului în ordinea lexicografică:

C	R	I/J	P	T
O	G	A	F	E
B	D	H	K	L
M	N	Q	S	U
V	W	X	Y	Z

Mesajul este preprocesat prin introducerea literei Q ca delimitator de cuvânt, adjuncţionându-se la finalul mesajului (pentru ca acesta să aibă lungime pară) litera Q:

$$SIQINQCRIPTOGRAFIEQTACEREAQESTEQAURQ.$$

Respectând regulile de cifrare Playfair mesajul cifrat devine:

$$QPXAQSRIPTCEDGFETAUIOIGTOFUAUPAUEQIN.$$

Exerciţiul 6.11.5. Să se cifreze prin metoda transpoziţiei ($N = 12$), pornind de la parola $CRIPTOGRAFIE$ mesajul "SI IN CRIPTOGRAFIE TACEREA ESTE AUR".

Răspuns. Vom construi secvenţa numerică de cifrare asociind fiecărei litere din parolă indicele din ordinea lexicografică: astfel literele din parolă, scrise în ordine lexicografică sunt:

1	2	3	4	5	6	7	8	9	10	11	12
A	C	E	F	G	I	I	O	P	R	R	T

deci parola $CRIPTOGRAFIE$ produce permutarea: 2 10 6 9 12 8 5 11 1 4 7 3.

Textul clar este scris într-o tabelă cu 12 coloane:

2	10	6	9	12	8	5	11	1	4	7	3
S	I	Q	I	N	Q	C	R	I	P	T	O
G	R	A	F	I	E	Q	T	A	C	E	R
E	A	Q	E	S	T	E	Q	A	U	R	Q

Deoarece lungimea textului nu este divizibilă cu 12 vom completa ultimul rând cu o secvenţă cunoscută (în acest caz caracterul Q). Textul cifrat se obţine citind coloanele tabelei de cifrare în ordinea indicată de parola numerică:

$$IAASGEORRQPCUCQEQAQTERQETIFEIRARTQNIS.$$

Descifrarea se va realiza în mod similar folosind permutarea inversă σ^{-1}.

Dacă dimensiunea transpoziţiei N este mai mică decât lungimea parolei atunci se vor reţine N caractere din parolă.

Exerciţiul 6.11.6. Studiaţi comutativitatea operatorilor de cifrare substituţie mono/polialfabetică şi a transpoziţiei.

Exerciţiul 6.11.7. Implementaţi algoritmul de decriptare a unei transpoziţii.

Exerciţiul 6.11.8. Implementaţi algoritmul de decriptare a unei subtituţii simple.

Exerciţiul 6.11.9. Implementaţi algoritmul de decriptare a unui cifru obţinut prin compunerea unei transpoziţii şi a unei substituţii simple.

Exerciţiul 6.11.10. Implementaţi algoritmul de decriptare a unui cifru polialfabetic.

Capitolul 7

CRIPTANALIZA CIFRURILOR FLUX

The quality of a machine
depends largely on its use.
Boris Hagelin

7.1. Atacul generatoarelor pseudoaleatoare

Multe din generatoarele pseudoaleatoare făcute publice au fost sparte. Reamintim că *a sparge* un generator de chei-flux revine la acelaşi lucru cu a conduce un atac împotriva unui cifru în flux folosind textul clar. O comunicaţie sigură trebuie să reziste la astfel de atacuri. A sparge înseamnă fie a obţine parţile componente, fie a descoperi o slăbiciune (pentru a elabora o metodă de atac mai eficientă decât calculul brut). Pentru generatoarele de chei-flux, aceasta înseamnă fie a găsi cheia secretă, fie a arăta că o parte a ei este redundantă, dacă este privită din unghiul diverselor metode de atac. Se foloseşte cuvântul *a sparge* în ambele cazuri, deoarece a descoperi şi îndepărta redundanţa unei chei ajută la îmbunatăţirea calităţii generatorului. Vom trece în revistă o serie de tehnici de criptanaliză cum ar fi: criptanaliza liniară, criptanaliza diferenţială, metoda corelaţiei, metoda corelaţiei rapide (bazate pe transformata Walsh-Hadamard), metoda sindromului liniar, atacul consistenţi liniare precum şi metode bazate pe corecţia iterativă a erorilor. Sunt realizate şi o serie de studii comparative ale eficienţi (din punct de vedere al recuperaării datelor şi din punct de vedere al timpului de rulare) algoritmilor discutaţi. Se vor prezenta deasemenea o serie de tehnici de proiectare a generatoarelor de tip flux precum şi exemple practice de atac asupra unor clase de generatoare pseudoaleatoare. Aplicaţiile din finalul capitolului ar trebui să încheie un ciclu primar de pregătire în domeniu.

7.2. Criptanaliza liniară

7.2.1. Complexitatea liniară

Prezentăm pe scurt o metodă de criptanaliză şi anume metoda criptanalizei liniare, care îşi are originea în teoria codurilor, mai exact în problema decodificării codurilor de tip **BCH**. Odată cu dezvoltarea criptografiei această tehnică şi-a găsit aplicaţii în evaluarea algoritmilor de cifrare în estimarea complexităţii liniar echivalente a unui generator pseudoaleatoriu. Odată estimată această complexitate liniar echivalentă, se poate trece la reconstrucţia textului clar ce a fost însumat modulo 2 cu ieşirea acestui generator. Metoda de criptanaliză se numeşte *metoda sindromului liniar* şi poate fi găsită în literatura de specialitate. Această metodă poate fi generalizată pentru algoritmii de tip cascadă Gollmann care generalizează algoritmul de cifrare folosit de sistemul de telefonie mobilă **GSM**. Vom prezenta o serie de rezultate referitoare la complexitatea liniar echivalentă precum şi metoda de calcul efectiv al acesteia. Cu ajutorul complexităţii liniar echivalente se pot construi testele statistico-algoritmice (vezi *Simion* [82] şi [84]).

Intuitiv, aleator înseamnă imprevizibil. Pentru ca o secvenţă să fie aleatoare, trebuie ca perioada secvenţei să fie suficient de mare, iar diversele forme ale ei să fie uniform distribuite în secvenţe. În continuare, vom descrie câteva dintre cele mai simple metode care generează secvenţe pseudoaleatoare. În particular, registrele de deplasare cu feedback liniar detaliaţi în această secţiune servesc ca blocuri componente pentru construirea generatoarelor pe care le vom discuta ulterior. Cea mai discutată metodă de a genera numere pseudoaleatoare se bazează pe recurenţe de forma: $X_i = aX_{i-1} + b \bmod m$ unde a, b, m sunt parametrii care descriu generatorul şi pot fi utilizaţi drept chei secrete, iar X_0 este dat. Dacă parametrii sunt aleşi corect, numerele X_i nu se vor repeta decât după m termeni. Ca un exemplu, secvenţa generată de $X_i = 5X_{i-1} + 3 \bmod 16$, cu $X_0 = 1$ este $\{1, 8, 11, 10, 5, 12, 15, 14, 9, 0, 3, 2, 13, 4, 7, 6, 1, 8, \ldots\}$.

S-a arătat că secvenţele produse de astfel de generatoare bazate pe congruenţe liniare, nu sunt sigure din punct de vedere criptografic. Fiind dată o porţiune suficient de mare din secvenţă, se pot reconstrui parametrii m, a, b. Generatoarele bazate pe congruenţe liniare care folosesc trunchieri (doar un prefix în biţi al fiecarui număr X_i) s-au dovedit a fi nesigure. Secvenţe puternice din punct de vedere criptografic pot fi generate folosind registrele de deplasare chiar dacă feedback-ul este liniar. Un registru de deplasare cu feedback constă în n locaţii de memorie de câte un bit care se ,,deplasează" spre dreapta şi o funcţie de feedback care exprimă orice element nou $a(t)$, cu $t \leq n$, al secvenţei în funcţie de elementele generate anterior $a(t-n), a(t-n+1), \ldots, a(t-1)$. Funcţia de feedback trebuie să fie nesingulară,

adică de forma: $a(t) = g(a(t-1), \ldots, a(t-n+1)) \oplus a(t-n)$, unde $\oplus$ desemnează operaţia SAU exclusiv. Dacă funcţia de feedback este liniară (se poate implementa doar folosind operaţia SAU exclusiv) spunem că generatorul este un registru de deplasare cu feedback liniar(**LFSR**). Altfel, spunem că generatorul este un registru de deplasare cu feedback neliniar (**NLFSR**). O locaţie de memorie a registrului se numeşte nivel, iar semnalele binare $a(0), a(1), \ldots, a(n-1)$ sunt încărcate ca date iniţiale. Perioada secvenţei produse depinde atât de numărul de niveluri, cât şi de detaliile conexiunilor de feedback. Mai exact, perioada maximă a secvenţei care poate fi generată de un registru de deplasare cu feedback, având n niveluri şi o funcţie de feedback nesingulară este $2^n - 1$, adică numărul maxim de stări în care se poate afla un registru cu n niveluri (se exclude starea nulă). Nu vom insista asupra **NLFSR**-urilor, care sunt în general greu de implementat şi în multe cazuri uşor predictibile. **LFSR**-urile sunt folosite de mult timp pentru teste **VSLI**, comunicaţii cu spectru larg etc. Aceste circuite sunt însă şi printre cele mai importante generatoare pseudoaleatoare de biţi. Funcţia lor de feedback are forma:

$$a(t) = c_1 a(t-1) \oplus c_2 a(t-2) \oplus \ldots \oplus c_{n-1} a(t-n+1) \oplus a(t-n), \tag{7.1}$$

unde $c_i \in \{0, 1\}$. Conexiunea de feedback a unui **LFSR** poate fi exprimată printr-un polinom de feedback:

$$f(X) = 1 + c_1 X + c_2 X^2 + \ldots + c_{n-1} X^{n-1} + X^n,$$

cu nedeterminata X. Acest polinom decide perioada şi comportarea statistică a secvenţei de ieşire. Pentru a preveni o secvenţă de ieşire trivială, trebuie ca starea ,,zero peste tot" să nu fie stare iniţială. De exemplu, dacă un **LFSR** cu patru niveluri are polinomul de feedback:

$$f(X) = 1 + X + X^2 + X^3 + X^4,$$

dependent de starea iniţială, atunci el va genera una din secvenţele de perioadă 5.

a) $1111011110\ldots$,

b) $1000110001\ldots$,

c) $0100101001\ldots$,

Sau, alt exemplu, dacă **LFSR** are polinomul de feedback dat de $f(X) = 1 + X + X^4$, atunci el generează o singură secvenţă netrivială de perioadă 15, cu cea mai bună statistică pe care o astfel de secvenţă o poate avea:

$101100100011110\ldots$

Dacă **LFSR** nu are perioada maximă, în cazul în care cunoaştem structura statistică a sursei de informaţie, se poate aplica pentru decriptare *metoda verosimilităţii maxime*.

În general, pentru a garanta cea mai mare perioadă posibilă $2^n - 1$, polinomul de feedback $f(X)$ al **LFSR**-ului trebuie să fie *primitiv*. Aceasta însemnă că $f(X)$ trebuie ales astfel încât cel mai mic număr întreg pozitiv T pentru care $X^T - 1$ este divizibil cu $f(X)$ să fie $T = 2^n - 1$. O definiţie echivalentă este următoarea.

Definiţia 7.2.1. Un polinom $g(X)$ din inelul de polinoame $Z_p[X]$, p număr prim, se numeşte *polinom primitiv*, dacă $ord(\alpha) = p^n - 1$ unde $\alpha = \{X\} \bmod g(X)$ (clasa de resturi a polinomului X modulo $g(X)$) şi $n = \deg(g(X))$. Polinomul $g(X)$ se numeşte *polinom generator* pentru corpul Galois $GF(p^n)$.

Observaţia 7.2.1. Polinomul de feedback $f(X)$ este polinomul reciproc al polinomului generator $g(X)$.

Sunt două tipuri de implementare a unei secvenţe binare obţinute cu ajutorul polinomului generator: schema de tip *Fibonacci* (ce implementează recurenţa 7.1) şi schema *Galois* (ce implementează clasele de resturi), acestea fiind exemplificate în figurile 7.1 şi 7.2.

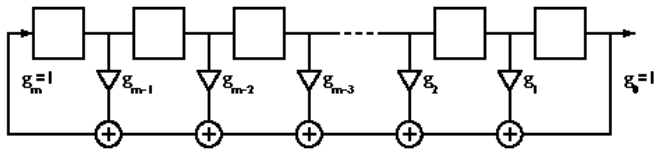

Figura 7.1: Implementarea Fibonacci a secvenţei generate de polinomul generator $g(X)$.

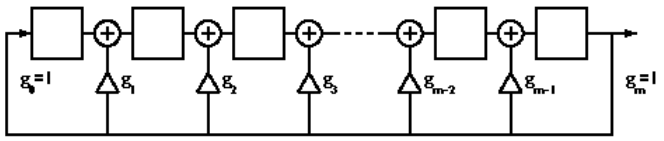

Figura 7.2: Implementarea Galois a secvenţei generate de polinomul generator $g(X)$.

Reţinem faptul că având stări iniţiale convenabil alese cele două scheme de implementare produc aceeiaşi ieşire.

Există algoritmi care testează primitivismul unui polinom. Numărul de polinoame primitive de grad n este:

$$N_p(n) = \frac{\Phi(2^n - 1)}{n},$$

unde $\Phi(x)$, cunoscută ca *funcţia lui Euler*, desemnează cardinalul de numere naturale mai mici ca x şi relativ prime cu x. Observăm că dacă un polinom $f(X)$ este primitiv atunci şi polinomul *reciproc* lui adică $X^n f(\frac{1}{X})$ este primitiv. Se ştie că orice polinom primitiv este ireductibil. Reciproca nu este adevărată. Numărul de polinoame ireductibile de grad n în algebra mod p ($p = 2$) este dat de formula următoare:

$$N_I(n) = \frac{1}{n} \sum_{d|n} p^d \mu(\frac{n}{d}),$$

unde μ este *funcţia lui Möebius* definită în felul următor pentru $n = \prod_1^k p_i^{\alpha_i}$: $\mu(n) = 0$ dacă $\prod_i^k \alpha_i > 1$, $\mu(n) = (-1)^k$ dacă n este produsul a k numere prime distincte şi $\mu(1) = 1$. Legătura între funcţia lui Moebius şi funcţia lui Euler este dată de:

$$\frac{\phi(n)}{n} = \sum_{d|n} \frac{\mu(d)}{d}.$$

Dacă k este un *număr prim Mersenne*, adică k este număr prim de forma $2^n - 1$ unde n este număr prim, atunci orice polinom ireductibil de grad k (în algebra mod 2) este primitiv:

$$\begin{aligned} N_I(k) &= \frac{1}{2^n - 1} \sum_{d|2^n-1} 2^d \mu(\frac{2^n - 1}{d}) = \frac{1}{2^n - 1}[-2 + 2^{2^n-1}] \\ &= \frac{\Phi(2^{2^n-1} - 1)}{2^n - 1} = N_P(k). \end{aligned}$$

Noţiunea de **LFSR** se poate generaliza pentru algebra $(\mathbf{Z}_p, +, .)$, cu p număr prim, prin următoarea:

Definiţia 7.2.2. Vom numi **LFSR** (registru de deplasare liniar) de lungime L o recurenţă liniară $a_n = f(a_{n-1}, \ldots, a_{n-L})$ în algebra $(\mathbf{Z}_p, +, \cdot)$.

Deci să notăm cu s un şir infinit ai cărui termeni sunt s_i sunt numere din mulţimea $\{0, 1, \ldots, p-1\}$ cu p număr prim (dacă $p = 2$ atunci avem de a face cu un şir binar) şi prin s^n şirul finit de lungime n $s_0, \ldots, s_{n-1}$. Unele din teoreme sau observaţii sunt valabile numai pentru $p = 2$.

Definiţia 7.2.3. Vom spune că un registru de deplasare **LFSR** *generează un şir infinit* s dacă există o stare iniţială pentru care ieşirea acestuia este s. Similar, spunem că **LFSR** *generează şirul finit* s^n dacă există o stare iniţială pentru care ieşirea **LFSR** are primii n termeni şirul finit s^n.

Definiţia 7.2.4. *Complexitatea liniară a şirului infinit* s, notată prin $L(s)$ este definită în modul următor:

i) dacă s este şirul nul $s = 0, 0, 0, \ldots$ atunci $L(s) = 0$;

ii) dacă nu există un **LFSR** care să genereze s atunci $L(s) = \infty$;

iii) în caz contrar $L(s)$ este lungimea celui mai mic **LFSR** care generează pe s.

Definiţia 7.2.5. *Complexitatea liniară a şirului finit* s^n, notată prin $L(s^n)$ este lungimea celui mai scurt registru de deplasare **LFSR** care pentru o anumită stare iniţială generează o ieşire ce are ca primi n termeni şirul finit s^n.

Pentru un polinom $C(D) \in \mathbf{Z}_2[D]$ şi pentru $L \in \mathbf{N}$ vom nota prin $< L, C(D) >$ registrul de deplasare **LFSR** de lungime L şi cu reacţiile date de polinomul $C(D)$. Principalele proprietăţi ale complexităţii liniare sunt sistematizate în următoarele teoreme.

Teorema 7.2.1. *(Proprietăţi fundamentale). Fie* s *şi* t *două şiruri.*

i) complexitatea şirului finit s^n *este cel mult* n*: pentru orice număr* $n \geq 1$ *avem* $0 \leq L(s^n) \leq n$*;*

ii) complexitatea şirului finit s^n *este minimă adică* $L(s^n) = 0$ *dacă şi numai dacă* s^n *este un şirul* $0, 0, \ldots, 0$ *de lungime* n*;*

iii) complexitatea şirului finit s^n *este maximă adică* $L(s^n) = n$ *dacă şi numai dacă* s^n *este un şirul* $0, 0, \ldots, \alpha$ *(cu* $\alpha \neq 0$*) de lungime* n*;*

iv) dacă s *este periodic cu perioada* N *atunci* $L(s) \leq N$*;*

v) $L(s \oplus t) \leq L(s) + L(t)$*.*

Teorema 7.2.2. *(Legătura cu polinoamele ireductibile). Pentru orice polinom* $C(D) \in \mathbf{Z}_p[D]$ *de grad* L *şi ireductibil peste* $\mathbf{Z}_p$ *orice stare nenulă, din cele* $p^L - 1$ *stări posibile, ale unui registru de deplasare liniar* **LFSR** $< L, C(D) >$ *generează un şir cu complexitatea* L*.*

Teorema 7.2.3. *(Media şi dispersia complexităţii liniare a unui şir aleatoriu). Fie* s^n *un şir binar finit de lungime* n *ales aleatoriu din mulţimea tuturor şirurilor binare finite de lungime* n*. Fie* $L(s^n)$ *complexitatea sa. Vom nota cu* $B(n)$ *funcţia de paritate adică* $B(n) = 0$ *pentru* n *par şi* $B(n) = 1$ *pentru* n *impar.*

i) Media complexităţii liniare a lui s^n *este:*

$$E(L(s^n)) = \frac{n}{2} + \frac{4 + B(n)}{18} - \frac{1}{2^n}(\frac{n}{3} + \frac{2}{9}),$$

deci pentru n suficient de mare $E(L(s^n)) \approx \frac{n}{2} + \frac{2}{9}$ *dacă n este par şi* $E(L(s^n)) \approx \frac{n}{2} + \frac{5}{18}$ *dacă n este impar.*

ii) Dispersia complexităţii liniare a lui s^n *este:*

$$Var(L(s^n)) = \frac{86}{81} - \frac{1}{2^n}(\frac{14 - B(n)}{27}n + \frac{82 - 2B(n)}{81}) - \frac{1}{2^{2n}}(\frac{1}{9}n^2 + \frac{4}{27}n + \frac{4}{81}),$$

deci pentru n mare $Var(L(s^n)) \approx \frac{86}{81}$.

Observaţie. O problemă care este în studiu la ora actuală este aceea a comportamentului asimptotic al profilului complexităţii liniare. Ideea de bază este aceea de a lega testul statistic de algoritmul de generare adică să răspundem, cu o anumită probabilitate, la întrebări de genul şirul este generat sau nu de un registru de deplasare cu anumite caracteristici (vezi *Simion* [79]).

Teorema 7.2.4. *(Media complexităţii liniare a unui şir aleatoriu periodic). Fie* s^n *un şir binar finit de lungime n ales aleatoriu din mulţimea tuturor şirurilor binare finite de lungime* $n = 2^t$. *Fie s şirul infinit obţinut prin concatenarea infinită a şirului finit* s^n *cu el însuşi. Media complexităţii liniare a şirului s este:*

$$E(L(s)) = n - 1 + 2^{-n}.$$

Teorema 7.2.5. *(Profilul complexităţii). Fie* $s = s_0, s_1, \ldots$ *un şir din algebra* $(\mathbf{Z}_p, +, \cdot)$. *Vom nota cu* L_N *complexitatea liniară a subşirului* $s^N = s_0, s_1, \ldots, s_{N-1}$, $N \geq 0$. *Şirul* $L_1, L_2, \ldots$ *se numeşte profilul complexităţii liniare a şirului infinit s. Similar dacă* $s = s_0, s_1, \ldots, s_{n-1}$ *este un şir binar şirul* $L_1, L_2, \ldots, L_n$ *se numeşte profilul complexităţii liniare a şirului finit* s^n.

Teorema 7.2.6. *(Proprietăţi ale profilului complexităţii). Fie* $L_1, L_2, \ldots$ *profilul complexităţii liniare a şirului infinit* $s = s_0, s_1, \ldots$. *Atunci:*

i) dacă $j > i$ *avem* $L_j \geq L_i$;

ii) dacă $L_N > \frac{N}{2}$ *atunci* $L_{N+1} = L_N$;

iii) dacă $L_{N+1} > L_N$ *atunci* $L_{N+1} + L_N = N + 1$.

Teorema 7.2.7. *(Media profilului complexităţii liniare a unui şir aleatoriu). Fie* $s = s_0, s_1, \ldots$ *un şir binar aleatoriu şi fie* L_N *complexitatea liniară a subşirului* $s^N = s_0, s_1, \ldots, s_{N-1}, N \geq 1$. *Pentru orice index fixat* $N \geq 1$ *media celui mai mic j pentru care* $L_{N+j} > L_N$ *este 2 dacă* $L_N \leq \frac{N}{2}$ *şi* $2 + 2L_N - N$ *dacă* $L_N > \frac{N}{2}$. *Mai mult, creşterea medie a complexităţii liniare este 2 dacă* $L_N \geq \frac{N}{2}$ *sau* $N - 2L_N + 2$ *dacă* $L_N < \frac{N}{2}$.

7.2.2. Algoritmul Berlekamp-Massey. Rezultate teoretice

Algoritmul Berlekamp-Massey este un algoritm eficient de determinare a complexităţii liniare a unui şir finit s^n de lungime n. Algoritmul are n iteraţii, iar la iteraţia N se calculează complexitatea liniară a subşirului s^N format din primii N termeni ai lui s^n.

Definiţia 7.2.6. Să considerăm şirul finit $s^{N+1} = s_0, s_1, \ldots, s_{N-1}, s_N$. Pentru $C(D) = 1 + c_1 D + \ldots + c_L D^L$ fie $< L, C(D) >$ un **LFSR** care generează subşirul $s^N = s_0, s_1, \ldots, s_{N-1}$. *Discrepanţa următoare* d_N este diferenţa dintre s_N şi al $(N+1)$ termen generat de **LFSR** :

$$d_N = (s_N + \sum_{i=1}^{L} c_i s_{N-i}) \bmod p.$$

Teorema 7.2.8. *Fie şirul finit* $s^N = s_0, s_1, \ldots, s_{N-1}$ *de complexitate* $L = L(s^N)$ *şi fie* $< L, C(D) >$ *un* **LFSR** *care generează* s^N.

i) registrul **LFSR** $< L, C(D) >$ *generează* $s^{N+1} = s_0, s_1, \ldots, s_{N-1}, s_N$ *dacă şi numai dacă discrepanţa următoare* d_N *este* 0;

ii) dacă $d_N = 0$ *atunci* $L(s^{N+1}) = L$;

iii) să presupunem că $d_N \neq 0$. *Fie* m *cel mai mare număr întreg mai mic ca* N *astfel încât* $L(s^m) < L(s^N)$ *şi fie* $< L(s^m), B(D) >$ *un registru de deplasare* **LFSR** *de lungime* $L(s^m)$ *care generează* s^m. *Atunci* $< L', C'(D) >$ *este un registru* **LFSR** *de lungime minimă care generează* s^{N+1} *unde:*

$$L' = \begin{cases} L & \text{dacă } L > N/2 \\ N + 1 - L & \text{dacă } L \leq N/2, \end{cases}$$

şi $C'(D) = C(D) + db^{-1} B(D) D^{N-m}$, *unde* b^{-1} *este inversul, în algebra* $(\mathbf{Z}_p, +, \cdot)$, *a discrepanţei de la pasul* m.

7.2.3. Implementarea algoritmului Berlekamp-Massey

Rezultatele precedente ne permit implementarea algoritmului de determinare a complexităţii liniar echivalente a unui şir binar. Această procedură de test poate fi asimilată procedurilor de analiză criptografică liniară.

Prezentăm în continuare schema acestuia în pseudocod. Pentru portabilitatea algoritmului, el va fi descris în algebra $\mathbf{GF}(p)$, cu p număr prim.

Intrare: şirul $s^n = s_0, s_1, s_2, \ldots, s_{n-1}$ cu elemente din algebra $\mathbf{GF}(p)$ de lungime n.

Ieşire: complexitatea liniară $L(s^n)$ a lui şirului s^n, $0 \leq L(s^n) \leq n$.

1. *Iniţializare.*
$C(D) = 1,\ B(D) = 1,$
$L = 0,\ m = -1,\ N = 0, b = 1,$
$p = 2$ (caracteristica corpului);
2. **While** $(N < n)$ **execută**:
2.1. Calculează discrepanţa următoare d:
$d = (s_N + \sum_{i=1}^{L} c_i s_{N-i}) \bmod p.$
2.2. **Dacă** $d \neq 0$ **execută**:
$T(D) = C(D), \quad C(D) = C(D) + db^{-1}B(D)D^{N-m}.$
Dacă $L \leq \frac{N}{2}$ atunci $L = N + 1 - L,\ m = N,\ B(D) = T(D),\ d = b.$
2.3. $N = N + 1.$
3. **Return**(L).

Observaţii. i) La fiecare iteraţie la sfârşitul pasului $2 < L, C(D) >$ este cel mai scurt registru **LFSR** care generează s^N. Deci algoritmul anterior poate fi folosit la calculul profilului complexităţii liniare.

ii) Timpul de execuţie a algoritmului *Berlekamp-Massey* pentru determinarea complexităţii liniare a unui şir binar de lungime n este $O(n^2)$.

iii) Fie s^n un şir finit de lungime n şi fie L complexitatea sa liniară. Atunci există un unic registru **LFSR** de lungime L care generează s^n dacă şi numai dacă $L \leq \frac{N}{2}$.

iv) Fie s un şir binar infinit de complexitate liniară L şi fie t un subşir finit al lui s de lungime cel puţin $2L$. Atunci algoritmul *Berlekamp-Massey* cu intrarea t determină un **LFSR** de lungime L care generează s.

v) Dacă *polinomul de feedback* care generează şirul este *primitiv* atunci algoritmul *Berlekamp-Massey* va găsi un registru de aceeaşi lungime cu acesta.

vi) O altă problemă este aceea a reconstrucţiei celui mai mic registru de deplasare, dar neliniar, care generează şirul respectiv. Aceasta lucrează cu noţiunea de şir *De Bruijn* care corespunde noţiunii de primitivism în cazul liniar.

7.2.4. Testul Berlekamp ca test statistico-informaţional

Descrierea testului: Algoritmul Berlekamp-Massey (detecţia celei mai scurte recurenţe liniare care produce o succesiune dată) ne conduce la testul Berlekamp-Massey ca test statistico-informaţional. Acest test se mai numeşte şi test de liniaritate. Statistica testului este o variabilă aleatoare de tip χ^2.

Scopul testului: Testul decide dacă secvenţa testată prezintă valenţe ale unei secvenţe generate liniar.

Algoritmul de test este prezentat în continuare specificându-se intrările şi ieşirile din program.

Intrare: Succesiunea binară, de lungime N, s şi riscul de ordinul 1 al testului notat prin α.

Ieşire: Decizia referitoare la inexistenţa componentei liniare (aleatorism).

PASUL 1. Şirul de biţi este partiţionat în N blocuri de lungime M.

PASUL 2. Utilizând algoritmul Berlekamp-Massey calculez complexitatea liniară L_i a fiecăruia din cele N blocuri.

PASUL 3. Calculăm valoarea medie a complexităţii după formula:

$$\mu = \frac{M}{2} + \frac{4 + B(M)}{18} - \frac{1}{2^M}(\frac{M}{3} + \frac{2}{9}).$$

PASUL 4. Pentru $i = 1, \ldots, N$ calculăm valorile:

$$T_i = (-1)^M(L_i - \mu) + \frac{2}{9}.$$

PASUL 5. Valorile T_i determină creşterea valorilor v_i după cum urmează:

$$\begin{cases} T_i \leq -2,5 \text{ creşte } v_0, \\ -2,5 \leq T_i \leq -1,5 \text{ creşte } v_1, \\ -1,5 \leq T_i \leq -0,5 \text{ creşte } v_2, \\ -0,5 \leq T_i \leq 0,5 \text{ creşte } v_3, \\ 0,5 \leq T_i \leq 1,5 \text{ creşte } v_4, \\ 1,5 \leq T_i \leq 2,5 \text{ creşte } v_5, \\ 2,5 \leq T_i \text{ creşte } v_6. \end{cases}$$

(s-a partiţionat spaţiul realizărilor variabilei T_i în $K = 7$ clase)

PASUL 6. Calculăm statistica testului care este de tip $\chi^2(K-1)$:

$$\chi^2_{calc} = \sum_{i=0}^{K-1} \frac{(v_i - N\pi_i)^2}{N\pi_i},$$

unde:

$$\begin{cases} \pi_0 = 0,01047 \\ \pi_1 = 0,03125 \\ \pi_2 = 0,125 \\ \pi_3 = 0,5 \\ \pi_4 = 0,25 \\ \pi_5 = 0,0625 \\ \pi_6 = 0,02078. \end{cases}$$

(aceste valori reprezintă probabilităţile a priori conform celor K clase). Evident $\sum_{i=0}^{6} \pi_i = 1$.

PASUL 7. Dacă $\chi^2_{calc} \leq \chi^2(K-1,\alpha)$ (cuantila de ordinul α a distribuţiei $\chi^2(K-1)$) atunci se decide acceptarea aleatorismului sucsesiunii testate, în caz contrar se respinge ipoteza aleatorismului.

Observaţia 7.2.2. O valoare prea mică a lui χ^2_{calc} (apropiată 0) poate fi considerată *suspectă,* în concluzie este indicat de a utiliza un test bilateral în care regiunea de decizie este $[u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}]$ ($u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$ fiind cuantilele de ordinul $\frac{\alpha}{2}$ respectiv $1-\frac{\alpha}{2}$ ale distribuţiei $\chi^2(K-1)$).

7.3. Metoda corelaţiei

Metoda corelaţiei este o tehnică care se aplică în principiu la analiza algoritmilor de cifrare de tip flux ce sunt compuşi din mai multe registre de deplasare (recurenţe liniare) combinate cu ajutorul unei funcţii booleene. Metoda constă în corelarea ieşirii algoritmului, deci a funcţiei booleene, cu fiecare registru (intrarea în funcţie). Odată realizată această corelaţie se poate previziona fiecare registru (intrare în funcţie) cu ajutorul algoritmului Berlekamp-Massey. În principiu metoda se poate aplica nu numai pentru combinaţii de registre de deplasare ci şi pentru orice structură algoritmică cu condiţia adaptării consistente a algoritmului Berlekamp-Massey.

7.4. Metoda corelaţiei rapide

Metoda corelaţiei rapide este o generalizare a metodei corelaţiei prin care se urmăreşte corelarea ieşirii din algoritm cu intrările acestuia. Tehnica de corelaţie se bazează pe calculul rapid al transformatei Walsh-Hadamard (vezi şi *Popescu* [56]).

7.4.1. Transformata Walsh-Hadamard

Definirea transformatei Walsh-Hadamard

Fie funcţia binară, scrisă în forma algebrică normală, $f : \mathbf{Z}_2^n \to \mathbf{Z}_2$. Vom defini transformata $\hat{f}$ prin formula:

$$\hat{f}(\mathbf{x}) = 1 - 2f(\mathbf{x}),$$

deci $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$. Vom prezenta teoria transformatei Walsh-Hadamard în funcţie de $\hat{f}$.

Observaţia 7.4.1. Funcţia $\hat{f}$ se poate defini similar prin formula $(-1)^{f(x)}$.

Pentru un şir binar $\mathbf{x}$ definim în mod similar transformata sa $\hat{\mathbf{x}}$ prin formula:

$$\hat{x}_i = 1 - 2x_i \text{ pentru orice } i.$$

Matricea Hadamard de ordinul 2^n este definită recursiv de formula:

$$H_{2^n} = \begin{pmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{pmatrix}$$

cu

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Matricea Hadamard este simetrică iar inversa sa este $\frac{1}{2^n} H_{2^n}$.

Definiţia 7.4.1. Pentru o funcţie $\hat{f}$: $\mathbf{Z}_2^n \to \{-1, 1\}$ vom defini transformata Walsh-Hadamard $\hat{F}$ prin formula:

$$\hat{F}(\boldsymbol{\omega}) = \sum_{\mathbf{x} \in \mathbf{Z}_2^n} \hat{f}(\mathbf{x})(-1)^{\boldsymbol{\omega}.\mathbf{x}} \text{ pentru orice } \boldsymbol{\omega} \in \mathbf{Z}_2^n$$

unde $\boldsymbol{\omega}.\mathbf{x}$ este produsul scalar al lui $\boldsymbol{\omega}$ cu $\mathbf{x}$. Să remarcăm faptul că transformarea Walsh-Hadamard se poate defini pentru orice funcţie reală.

Definiţia 7.4.2. Pentru un şir binar $\hat{\mathbf{x}}$ vom defini transformarea Walsh-Hadamard prin formula:

$$\hat{\omega}_i = \sum_{j=0}^{2^{m-1}} \hat{x}_j \, (-1)^{i.j} \text{ pentru orice } i$$

unde $i.j$ este produsul scalar dintre reprezentările binare ale lui i şi j.

Definiţia 7.4.3. Convoluţia a două funcţii $\hat{f}, \hat{g}$: $\mathbf{Z}_2^n \to \{-1, 1\}$ este definită ca

$$(\hat{f} * \hat{g})(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Z}_2^n} \hat{f}(\mathbf{y}) \, \hat{g}(\mathbf{x} \oplus \mathbf{y}).$$

similar cu definiţia a două funcţii vom defini convoluţia a două şiruri $\overset{\wedge}{\mathbf{x}}$ şi $\overset{\wedge}{\mathbf{y}}$ prin formula:

$$\hat{z}_i = \sum_{j=0}^{N-1} \hat{x}_j \hat{y}_{i-j} \text{ pentru } i = 0, ..., N-1.$$

Observaţia 7.4.2. Se observă faptul că transformata Walsh-Hadamard a unei funcţii $\hat{f}: \mathbf{Z}_2^n \to \{-1, 1\}$ se poate scrie ca produsul:

$$\hat{F} = H_{2^n} \hat{f}.$$

Avem o relaţie similară pentru transformata Walsh-Hadamard a şirurilor.

Observaţia 7.4.3. Termenul general al matricei Hadamard este $h_{ij} = (-1)^{\mathbf{i.j}}$ unde **i.j** reprezintă produsul scalar dintre reprezentările binare ale numerelor i şi j.

Observaţia 7.4.4. Inversa matricei H_{2^n} este $\frac{1}{2^n} H_{2^n}$ de unde rezultă faptul că:

$$\hat{f} = \frac{1}{2^n} H_{2^n} \hat{F}.$$

Observaţia 7.4.5. Legătura dintre transformatele Walsh-Hadamard $F(.)$ şi $\hat{F}(.)$ ale funcţiilor f respectiv $\hat{f}$ este:

$$\hat{F}(\boldsymbol{\omega}) = -2F(\boldsymbol{\omega}) + 2^n \delta(\boldsymbol{\omega}).$$

Observaţia 7.4.6. Transformata Walsh-Hadamard este liniară.

Proprietăţile transformatei Walsh-Hadamard

Vom prezenta principalele proprietăţi ale transformatei Walsh-Hadamard a funcţiei $\hat{f}$. Rezultate similare se obţin pentru transformata Walsh-Hadamard a şirurilor.

Teorema 7.4.1. *Operatorul $\wedge$ definit pentru funcţii reale este liniar. Transformata Walsh-Hadamard are un singur punct fix şi anume* $\mathbf{0}$.

Teorema 7.4.2. *Inversa transformatei Walsh-Hadamard pentru funcţia $\hat{f}: \mathbf{Z}_2^n \to \{-1, 1\}$ este dată de formula:*

$$\hat{f}(\mathbf{x}) = \frac{1}{2^n} \sum_{\boldsymbol{\omega} \in \mathbf{Z}_2^n} \hat{F}(\boldsymbol{\omega}) (-1)^{\boldsymbol{\omega}.\mathbf{x}}.$$

Teorema 7.4.3. *Pentru o funcţie* $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ *avem următoarea proprietate referitoare la suma valorilor transformatei Walsh-Hadamard:*

$$\sum_{\boldsymbol{\omega}\in\mathbf{Z}_2^n} \hat{F}(\boldsymbol{\omega}) = 2^n \hat{f}(\mathbf{0}).$$

Teorema 7.4.4. *Dacă variabila* x_j *este idle (nu apare în forma algebrică normală:* $f(x_1,...,x_i,...,x_n) = f(x_1,...,\bar{x}_i,...,x_n)$ *sau altfel spus* f *este degenerată) atunci*

$$\hat{F}(\boldsymbol{\omega}) = 0 \text{ pentru orice } \boldsymbol{\omega} : \ \omega_i = 1.$$

Teorema 7.4.5. *Dacă* $f : \mathbf{Z}_2^n \to \mathbf{Z}_2$ *este scrisă în forma algebrică normală:*

$$f(x) = a_0 + a_1x_1 + ... + a_nx_n + a_{12}x_1x_2 + ... + a_{12...n}x_1x_2...x_n$$

atunci:

$$a_{12...n} = 2^{n-k} \sum_{\boldsymbol{\omega}\in\mathbf{V}(i_1,...,i_k)} F(\boldsymbol{\omega}) \bmod 2$$

unde $\mathbf{V}(i_1,...,i_k) = \{\boldsymbol{\omega} \in \mathbf{Z}_2^n | \omega_i = 0 \text{ pentru } i \in \{i_1,...,i_k\}\}$.

Teorema 7.4.6. *Transformata Walsh-Hadamard a convoluţiei* $\hat{h}$ *a două funcţii* $\hat{f}, \hat{g}\colon \mathbf{Z}_2^n \to \{-1,1\}$ *este produsul convoluţiilor:*

$$\hat{H} = \hat{F} \cdot \hat{G}\,.$$

Demonstraţie. Folosind definiţia obţinem:

$$\begin{aligned}
\hat{F}(\boldsymbol{\omega}) \cdot \hat{G}(\boldsymbol{\omega}) &= \sum_{\mathbf{x}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})(-1)^{\boldsymbol{\omega}.\mathbf{x}} \sum_{\mathbf{y}\in\mathbf{Z}_2^n} \hat{g}(\mathbf{y})(-1)^{\boldsymbol{\omega}.\mathbf{y}} \\
&= \sum_{\mathbf{x}\in\mathbf{Z}_2^n} \sum_{\mathbf{y}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})\, \hat{g}(\mathbf{y})(-1)^{\boldsymbol{\omega}.(\mathbf{x}\oplus\mathbf{y})} \\
&= \sum_{\mathbf{x}\in\mathbf{Z}_2^n} \sum_{\mathbf{z}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})\, \hat{g}(\mathbf{x}\oplus\mathbf{z})(-1)^{\boldsymbol{\omega}.\mathbf{z}} \\
&= \sum_{\mathbf{z}\in\mathbf{Z}_2^n} \left(\sum_{\mathbf{x}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})\, \hat{g}(\mathbf{x}\oplus\mathbf{z}) \right) (-1)^{\boldsymbol{\omega}.\mathbf{z}}
\end{aligned}$$

$$= \sum_{\mathbf{z}\in\mathbf{Z}_2^n} \hat{h}(\mathbf{z})(-1)^{\boldsymbol{\omega}.\mathbf{z}}$$
$$= \hat{H}(\boldsymbol{\omega}).$$

Observaţia 7.4.7. i) Transformata Walsh-Hadamard a funcţiei $\hat{f}(\mathbf{x})$ definită de $f(x) = c$ (funcţia constantă) este vectorul $(2^n(1-2c), 0, ..., 0)^t$.

ii) Transformata Walsh-Hadamard a funcţiei $\hat{f}(\mathbf{x})$ definită de $f(x) = c + x_i$ (cea de a $i-$a funcţie proiecţie) este $(0, ..., 2^n(1-2c), ..., 0)^t$, termenul nenul fiind pe poziţia 2^{i-1}.

iii) Din observaţiile anterioare se observă faptul că termenul liber c are influenţă asupra semnului factorului nenul din transformata Walsh-Hadamard.

7.4.2. Testul statistic Walsh-Hadamard

Conceptul de test statistic

Conceptul de test statistic se poate găsi în orice carte de statistică fundamentală. Aplicaţii ale testelor statistice în criptografie se pot găsi în *Maurer* [46]. Sunt făcute două ipoteze statistice referitoare la secvenţele binare ce sunt testate:

H_0 : şirul $\mathbf{x}$ este produs de o sursă binară fără memorie: $Pr(X = 1) = p_0$ şi $Pr(X = 0) = 1 - p_0$, (în acest caz spunem că şirul nu prezintă nici o componentă predictibilă)

şi alternativa:

H_1 : şirul $\mathbf{x}$ este produs de o sursă binară fără memorie: $Pr(X = 1) = p_1$ şi $Pr(X = 0) = 1 - p_1$ cu $p_0 \neq p_1$,(în acest caz spunem că şirul prezintă o componentă predictibilă referitoare la probabilitatea p).

În testarea statistică sunt două tipuri de erori statistice: eroarea de ordinul 1, notată cu α (se mai numeşte şi nivel de semnificaţie al testului şi reprezintă probabilitatea apariţiei unui rezultat fals pozitiv) şi eroarea de ordinul 2, notată cu β (este probabilitatea apariţiei unui rezultat fals negativ). Aceste erori au următoarea interpretare:

$$\alpha = \Pr(\text{resping } H_0 | H_0 \text{ este adevărată}) = 1 - \Pr(\text{accept } H_0 | H_0 \text{ este adevărată})$$

respectiv

$$\beta = \Pr(\text{accept } H_0 | H_0 \text{ este falsă}) = 1 - \Pr(\text{resping } H_0 | H_0 \text{ este falsă}).$$

Aceste două erori nu pot fi minimizate simultan (testele Neymann-Pearson minimizează valorile lui β pentru un α fixat). Procedura de testare ce o vom prezenta

este următoarea: pentru o valoare fixă a lui α vom găsi un domeniu de acceptare pentru funcţia de test şi vom verifica dacă aceasta aparţine domeniului respectiv. Domeniul de acceptare se obţine din cuantilele de ordinul $\frac{\alpha}{2}$ şi $1-\frac{\alpha}{2}$. (Cuantila u_α de ordinul α este definită prin: $Pr(X < u_\alpha) = \alpha$.)

Funcţia de test

Vom începe prin a enunţa o serie de proprietăţi.

Fie $\mathbf{x}$ un şir binar pentru care vom realiza o testare referitoare la cele două ipoteze formulate. Vom presupune că lungimea şirului $\mathbf{x}$ este $n = 2^m$. Vom construi şirul transformat $\hat{\mathbf{x}}$ iar transformata sa Walsh-Hadamard o vom nota prin $\hat{\boldsymbol{\omega}}$.

Teorema 7.4.7. *Avem pentru prima componetă $\hat{\omega}_0$ a transformatei Walsh-Hadamard:*

i) valoarea medie a lui $\hat{\omega}_0$ este $m_0 = 2^m(1-2p)$.

ii) dispersia lui $\hat{\omega}_0$ este $\sigma_0^2 = 2^{m+2}p(1-p)$.

iii) distribuţia lui $\frac{\hat{\omega}_0 - m_0}{\sigma_0}$ este bine aproximată (pentru $m \geq 7$) de repartiţia normală $N(0,1)$.

Teorema 7.4.8. *Avem pentru cea de a i-a componentă $\hat{\omega}_i$ $(i \geq 1)$ a transformatei Walsh-Hadamard:*

i) valoarea medie a lui $\hat{\omega}_i$ este $m_i = 0$.

ii) dispersia lui $\hat{\omega}_i$ este $\sigma_i^2 = 2^{m+2}p(1-p)$.

iii) distribuţia lui $\frac{\hat{\omega}_i - m_i}{\sigma_i}$ este bine aproximată (pentru $m \geq 7$) de repartiţia normală $N(0,1)$.

Observaţia 7.4.8. i) Pentru $p = 0,5$ (sursă simetrică) valoarea medie a lui $\hat{\omega}_i$este egală cu 0, pentru orice i. Variabila aleatoare ω_i are aceiaşi distribuţie.

Teorema 7.4.9. *Pentru o sursă simetrică vectorul $\hat{\boldsymbol{\omega}}$ are o repartiţie normală multidimensională.*

Testul statistic este realizat cu ajutorul funcţiilor de test anterioare. De fapt, când se testează un şir binar de lungime 2^m cu ajutorul procedurilor anterioare, se efectuează 2^m teste statistice. Acest test este o baterie de teste de 2^{2^m} teste statistice (nu se vor efectua toate testele statistice).

Testul de aleatorism

Scopul acestui test este acela de a detecta autocorelaţiile din secvenţa testată.

Intrare:

-Secvenţa binară $\mathbf{x}$ de lungime n;

-Dimensiunea blocului de date de lungime 2^m;

-Limita de respingere α;

- p probabilitatea de apariţie a simbolului 1.

Ieşirea: Decizia referiatoare la aleatorism la nivelul de semnificaţie α raportată la cea de a $i-a$ funcţie de test (denumită autocorelaţia de tip i).

PASUL 1. Transformăm şirul binar $\mathbf{x}$ în şirul de 1 şi -1: $\overset{\wedge}{\mathbf{x}}= \mathbf{1} - 2\mathbf{x}$.

PASUL 2. Calculăm limita inferioară şi limita superiară de respingere a testului $u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$.

PASUL 3. Calculăm numărul de blocuri ce se procesează $\left[\frac{n}{2^m}\right]$. Vom diviza şirul $\overset{\wedge}{\mathbf{x}}$ în $\left[\frac{n}{2^m}\right]$ blocuri adicente.

PASUL 4. For $j = 0$ to $\left[\frac{n}{2^m}\right] - 1$ do

For $i = 0$ to $2^m - 1$ do

Calculează cea de a $i-a$ funcţie de test

$$t_{ij} = \frac{\overset{\wedge}{\omega}_{ij} - m_i}{\sigma_i},$$

unde $\overset{\wedge}{\omega}_{ij}$este cea de a $i-a$ componentă a transformatei Walsh-Hadamard a blocului j; valorile lui m_i şi σ_i sunt date de teoremele 7.4.7. respectiv 7.4.8.

PASUL 5. For $i = 0$ to $2^m - 1$ se ia decizia referitoare la cea corelaţia de ordinul i:

i) *Decizia majoritară:* dacă există o valoare

$$t_{ij} \notin [u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}],$$

(u_α este cunatila de ordinul α a distribuţiei normale) respingem ipoteza aleatorismului (raportată la cea de a i-a funcţie de test) a şirului $\mathbf{x}$ la nivelul de semnificaţie α şi vom afişa valorile lui i şi j.

ii) *Decizia maximum din* T: calculăm $T_i = \max\limits_j t_{ij}$, dacă

$$T_i \notin [u_{(\frac{\alpha}{2})^{\frac{1}{\left[\frac{n}{2^m}\right]}}}; u_{(1-\frac{\alpha}{2})^{\frac{1}{\left[\frac{n}{2^m}\right]}}}],$$

respingem ipoteza aleatorismului (raportată la cea de a $i-a$ funcţie de test) a şirului $\mathbf{x}$ la nivelul de semnificaţie α şi vom afişa valorile lui i.

iii) *Decizia* χ^2: calculăm

$$\chi_i^2 = \sum_{j=0}^{\left[\frac{n}{2^m}\right]-1} t_{ij}^2,$$

dacă $\chi_i^2 \notin [0, \chi^2(\left[\frac{n}{2^m}\right], \alpha)]$ respingem ipoteza aleatorismului (raportată la cea de a $i-a$ funcţie de test) a şirului $\mathbf{x}$ la nivelul de semnificaţie α şi vom afişa valorile lui i. ($\chi^2(\left[\frac{n}{2^m}\right], \alpha)$ este cuantila de ordinul α a distribuţiei $\chi^2(\left[\frac{n}{2^m}\right])$).

PASUL 6. Decizia finală: decizia majoritară, decizia maximum din T sau decizia χ^2 cu ajutorul statisticilor t_{ij}, T_i respectiv χ_i^2.

Observaţia 7.4.9. Să reţinem faptul că

$$\sum_{i=0}^{2^m-1} t_{ij} = \sum_{i=0}^{2^m-1} \frac{\overset{\wedge}{\omega}_{ij} - m_i}{\sigma_i} = 2^{\frac{m}{2}+1}(x_0^{(j)} - (1-2p)),$$

unde $x_0^{(j)}$ este prima componentă a celui de al $j-lea$ şir.

Observaţia 7.4.10. i) Dacă $t_{ij} \approx 0$ (dispersie mică) atunci decizia χ^2 nu detectează nici o abatere la uniformitate. Se va pune în evidenţă o abatere de la uniformitate cu ajutorul deciziei maximum din T.

ii) Regula de decizie majoritară este adecvată pentru valori mici ale lui $2^m - 1$ şi un număr mic de blocuri.

Observaţia 7.4.11. Prima valoare a funcţiei de test t_{0j} este statistica corespunzătoare testului frecvenţei. Cea de a doua funcţie de test t_{1j} este statistica corespunzătoare testului autocorelaţiei la distanţa $d = 2$. De fapt fiecare dintre puterile $d = 2^p$ ($p \leq m$) ale funcţiei de test Walsh-Hadamard au o componentă echivalentă cu testul autocorelaţiei la distanţa $d = 2^p$.

Observaţia 7.4.12. Procedurile de decizie sunt numite şi proceduri de agregare a rezultatelor şi sunt folosite mai ales când dispunem de mai multe realizări ale aceleiaşi funcţii de test.

7.4.3. Caracterizarea proprietăţilor criptografice

În cele ce urmează vom da o serie de caracterizări, prin intermediul transformatei Walsh-Hadamard (caracterizare spectrală), a principalelor caracteristici (balans, imunitate la corelaţie, propagare şi nedegenerare) ale unei funcţii criptografice (această funcţie poate fi un generator pseudoaleator, cifru bloc, funcţie de dispersie criptografică, etc.). Vom începe printr-o serie de definiţii.

Definiţia 7.4.4. Spunem despre funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ că îndeplineşte *criteriul de balans* (sau de *echilibrare*) dacă:

$$card\{\mathbf{x}|\ \hat{f}(\mathbf{x}) = 1\} = \frac{1}{2^{n-1}}.$$

Definiţia 7.4.5. Spunem despre funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ că îndeplineşte *criteriul de imunitate la corelaţie de ordin k* dacă $\hat{f}(\mathbf{X})$ este independentă statistic de orice submulţime de k poziţii de coordonate $(X_{i_1},...X_{i_k})$ ale lui $\mathbf{X} = (X_1,...X_n)'$, aceasta fiind o variabilă aleatoare uniform distribuită în $\mathbf{Z}_2^n$ ($\Pr(\mathbf{X} = i) = 2^{-n}$ pentu orice $i = 0,...,2^n - 1$). Spunem despre funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ că este *imună la corelaţie* dacă $\hat{f}$ este imună la corelaţie pentru orice ordin k. Vom nota această proprietate prin $IC(k)$.

Definiţia 7.4.6. Spunem despre funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ că îndeplineşte *criteriul propagării relativ la vectorul s* dacă $\hat{f}(\mathbf{x})\ \hat{f}(\mathbf{x} \oplus \mathbf{s})$ este echilibrată. Dacă $\hat{f}$ satisface criteriul propagării relativ la orice vector s atunci spunem că $\hat{f}$ satisface *criteriul propagării.* Funcţia $\hat{f}$ satisface *criteriul propagării de ordin k relativ la vectorul s* dacă $\hat{f}$ satisface criteriul propagării relativ la vectorul s cu $1 \leq w(s) \leq k$. Spunem despre funcţia $\hat{f}$ că satisface *criteriul propagării de ordin k* dacă $\hat{f}$ satisface criteriul propagării de ordin k relativ la orice vector s. Această proprietate se va nota prin $PC(k)$.

Definiţia 7.4.7. Spunem despre funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ că îndeplineşte *criteriul de avalaşă strictă* (SAC) dacă schimbarea unui bit la intrare produce schimbări, în proporţie de 50%, la ieşire.

Observaţia 7.4.13. *Criteriul de avalanşă strictă SAC* (schimbarea unui bit la intrare produce schimbări în proporţie de 50% la ieşire) este echivalent cu criteriul propagării de ordinul 1 $PC(1)$.

Avem următoarele caracterizări ale proprietăţilor funcţiilor $\hat{f}$ prin intermediul transformatei Walsh-Hadamard.

Teorema 7.4.10. *Funcţia* $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ *îndeplineşte criteriul de balans dacă*

şi numai dacă:

$$\sum_{\mathbf{x}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x}) = 0$$

sau echivalent

$$\hat{F}(\mathbf{0}) = 0.$$

Demonstraţia este evidentă (se foloseşte definiţia transformatei Walsh).

Teorema 7.4.11. *Funcţia* $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ *îndeplineşte criteriul de imunitate la corelaţie de ordin* k *dacă şi numai dacă:*

$$\hat{F}(\boldsymbol{\beta}) = 0 \text{ pentru orice } \boldsymbol{\beta} \in \mathbf{Z}_2^n, 1 \le w(\boldsymbol{\beta}) \le k,$$

$\hat{F}(.)$ *fiind transformata Walsh-Hadamard a lui* $\hat{f}$ *.*

Demonstraţie. Vom prezenta demonstraţia pentru cazul $k = 1$, pentru k arbitrar raţionamentul fiind similar. Presupunem că $\hat{f}$ îndeplineşte criteriul de imunitate la corelaţie de ordin 1. Fie $\boldsymbol{\beta} \in Z_2^n$ cu $w(\boldsymbol{\beta}) = 1$ ($\beta_i = 1,\ \beta_j = 0,\ i \neq j$). Succesiv putem scrie:

$$\begin{aligned}
\hat{F}(\boldsymbol{\beta}) &= \sum_{\mathbf{x}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})(-1)^{\boldsymbol{\beta}.\mathbf{x}} \\
&= \sum_{\mathbf{x}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})(-1)^{x_i} \\
&= \underset{\hat{f}(\mathbf{x})=1, x_i=0}{\sum_{\mathbf{x}\in\mathbf{Z}_2^n}} 1 + \underset{\hat{f}(\mathbf{x})=-1, x_i=1}{\sum_{\mathbf{x}\in\mathbf{Z}_2^n}} 1 + \underset{\hat{f}(\mathbf{x})=1, x_i=1}{\sum_{\mathbf{x}\in\mathbf{Z}_2^n}} (-1) + \underset{\hat{f}(\mathbf{x})=-1, x_i=0}{\sum_{\mathbf{x}\in\mathbf{Z}_2^n}} (-1) \\
&= 0 \text{ (utilizând definiţia).}
\end{aligned}$$

Demonstraţia implicaţiei reciproce se lasă, sub formă de exerciţiu, pe seama cititorului.

Teorema 7.4.12. *Funcţia* $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ *îndeplineşte criteriul de propagare de ordin* k *relativ la vectorul* s *dacă şi numai dacă:*

$$\sum_{\mathbf{x}\in\mathbf{Z}_2^n} \hat{f}(\mathbf{x})\,\hat{f}(\mathbf{x}\oplus\mathbf{s}) = 0 \text{ pentru orice } \mathbf{s} \text{ cu } 1 \le w(\mathbf{s}) \le k.$$

Demonstraţie. Deoarece $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ îndeplineşte criteriul de propagare de ordin k relativ la vectorul s (de pondere maxim k) dacă şi numai dacă $\hat{f}(\mathbf{x})\hat{f}(\mathbf{x}\oplus\mathbf{s})$ este echilibrată deci $\sum\limits_{\mathbf{x}\in\mathbf{Z}_2^n}\hat{f}(\mathbf{x})\hat{f}(\mathbf{x}\oplus\mathbf{s})=0$ pentru orice $\mathbf{s}$ cu $1\le w(\mathbf{s})\le k$.

Teorema 7.4.13. *Funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ îndeplineşte criteriul de avalanşă strictă dacă şi numai dacă:*

$$\sum_{\boldsymbol{\omega}\in\mathbf{Z}_2^n}(-1)^{\omega_i}\hat{F}^2(\boldsymbol{\omega})=0 \text{ pentru orice } \boldsymbol{\omega}\in\mathbf{Z}_2^n \text{şi } i=1,...,n.$$

Demonstraţie. Funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ îndeplineşte criteriul de avalanşă strictă dacă şi numai dacă $\sum\limits_{\mathbf{x}\in\mathbf{Z}_2^n}\hat{f}(\mathbf{x})\hat{f}(\mathbf{x}\oplus\mathbf{s})=0$ pentru orice $\mathbf{s}$ cu $w(\mathbf{s})=1$ adică $(\hat{f}*\hat{f})(\mathbf{s})=0$ (convoluţia lui $\hat{f}$ cu ea însăşi). Folosind definiţia inversei transformatei Walsh-Hadamard şi faptul că transformata convoluţiei a două funcţii este produsul transformatelor funcţiilor obţinem:

$$\begin{aligned}(\hat{f}*\hat{f})(\mathbf{s}) &= \frac{1}{2^n}\sum_{\boldsymbol{\omega}\in\mathbf{Z}_2^n}\hat{F}^2(\boldsymbol{\omega})(-1)^{\mathbf{s}.\boldsymbol{\omega}}\\ &= \frac{1}{2^n}\sum_{\boldsymbol{\omega}\in\mathbf{Z}_2^n}\hat{F}^2(\boldsymbol{\omega})(-1)^{\omega_i}.\end{aligned}$$

Din ultima relaţie rezultă concluzia teoremei.

Teorema 7.4.14. *O condiţie suficientă ca funcţia $\hat{f}\colon \mathbf{Z}_2^n \to \{-1,1\}$ să satisfacă criteriul de avalanşă strictă este ca:*

$$\hat{F}^2(\boldsymbol{\omega})=\hat{F}^2(\bar{\boldsymbol{\omega}}) \text{ pentru orice } \boldsymbol{\omega}\in\mathbf{Z}_2^n$$

Demonstraţie. Avem pentru orice $i=1,...,n$

$$\sum_{\boldsymbol{\omega}:\omega_i=0}\hat{F}^2(\boldsymbol{\omega})=\sum_{\boldsymbol{\omega}:\omega_i=1}\hat{F}^2(\bar{\boldsymbol{\omega}})$$

echivalent cu:

$$\sum_{\boldsymbol{\omega}\in\mathbf{Z}_2^n}(-1)^{\omega_i}\hat{F}^2(\boldsymbol{\omega})=0 \text{ pentru orice } \boldsymbol{\omega}\in Z_2^n \text{şi } i=1,...,n.$$

Ultima relaţie arată faptul că funcţia $\hat{f}$ satisface criteriul de avalanşă strictă (în conformitate cu teorema anterioară).

Observaţia 7.4.14. Îndeplinirea acestor proprietăţi constituie condiţii necesare dar nu şi suficiente ca o funcţie (sau algoritm) să fie sigură din punct de vedere criptografic.

Observaţia 7.4.15. Testarea criteriilor menţionate anterior se poate face prin demonstrarea efectivă a acestora în funcţie de proprietăţile funcţiei (algoritmului) ce se testează cu ajutorul testelor statistice. În literatura de specialitate există o serie de lucrări legate de tematica prezentată (vezi *Forré* [19], *Preneel* [63] şi [64]).

7.5. Atacul Siegenthaler

În multe generatoare, cheia-flux finală **b** se obţine combinând secvenţele de ieşire $\mathbf{a}_i, 1 \leq i \leq k$ ale mai multor sisteme. Între **b** şi fiecare $\mathbf{a}_i$ pot exista dependinţe statistice mai slabe sau mai puternice. Pentru a folosi aceste dependenţe, *Siegenthaler* [74] a dezvoltat un criteriu general de identificare a cheii pentru a conduce atacuri de tip "divide-et-impera" asupra unor astfel de sisteme. Atacurile se adreseaza fiecărei secvenţe constitutive a_i în mod individual şi se poartă prin căutări exhaustive. Nu vom intra în amănunte. Vrem doar să reţinem că, atât pentru criptograful care proiectează sistemele, cât şi pentru criptanalistul care le sparge, interdependenţele statistice între diversele părţi constitutive ale generatorului de chei-flux sunt extrem de importante.

7.6. Atacul consistenţei liniare

În contrast cu testul statistic al lui *Siegenthaler* [74], un test algebric orientat pentru a utiliza latenţa liniară existentă în diversele generatoare de chei-flux a fost dat de *Zeng*, *Yang* şi *Rao* [99]. Metoda este formulată pe baza unei estimări precise a probabilitaţii de consistenţă a unui sistem de ecuaţii liniare $Ax = b$ cu o $m \times n-$matrice de coeficienţi aleatoare $A, m > n$ şi un vector fixat, nenul, b. În multe generatoare de chei-flux, se poate găsi o subcheie comparativ mică K_1 din întreaga cheie secretă K şi se poate construi un sistem de ecuaţii liniare $Ax = b$, astfel încât:

1. matricea coeficienţilor A depinde în mare parte de cheia K_1;
2. vectorul b este determinat de un segment anumit al secvenţei de ieşire;
3. dacă subcheia K_1 este specificată greşit, sistemul corespunzator $Ax = b$ va fi inconsistent cu o probabilitate aproape de 1. Când un astfel de sistem este consistent se poate deduce din soluţia lui cealaltă parte a cheii K adică $K - K_1$. În acest caz,

se poate aplica căutarea exhaustivă pentru a determina subcheia K_1, cu aceeaşi consistenţă a sistemului liniar ca în cazul criteriului de identificare a cheii. Succesul unui astfel de atac înseamnă că cei $|K| - |K_1|$ biţi din partea necunoscută din K nu contribuie substanţial la tăria criptografică a întregului sistem. O oarecare formă de redundanţă a cheii a fost descoperită în sistem.

7.7. Metoda sindromului linear

Vom prezenta o clasă de metode de criptanaliză bazate pe corecţia iterativă a erorii pentru sindromul linear. Metodele se pot adapta pentru a fi folosite în diagnosticarea generatoarelor pseudoaleatoare ce au în componenţă registre lineare de fedback. Metoda va fi exemplificată pe generatoarele *Geffe* şi *Beth-Pipper* dar poate fi aplicată şi generatoarelor de tip cascadă *Gollmann* inclusiv generatorului *A5* (**GSM**). Metoda poate fi extinsă la aşa zisa metodă a corecţiei iterative a erorii pentru sindromul nelinear mai exact la registre de feedback care nu sunt lineare. Aceasta poate conduce la atacuri extrem de dure în cazul mai multor generatoare nelineare.

7.7.1. Formularea problemei

Fie A ieşirea unui **LFSR** cu *polinomul de feedback* $f(X)$ şi fie X un şir (text clar) cu proprietatea că:

$$\Pr(x(t) = 1) = s_0 < \frac{1}{2}.$$

Numărul s_0 se numeşte *rata iniţială a erorii* a şirului A în şirul $B = A + X$. Având acces la şirul B, cunoscând polinomul $f(X)$ (*algoritmul generator*) şi numărul s_0 (*statistica textului clar*) trebuie determinat şirul X (*text clar*) şi încărcarea iniţială a **LFSR** (*cheia*).

7.7.2. Preliminarii teoretice

Fie $r \geq 3, r \in \mathbf{Z}$ şi

$$g(X) = 1 + X^{i_1} + \ldots + X^{i_{r-1}}, \tag{7.2}$$

un multiplu al polinomului de feedback $f(X)$. Vom defini $2m + 1$ *sindromuri:*

$$\sigma_{i,k}(g(x)) = \sum_{p=0}^{r-1} b(i + i_p - i_k) \tag{7.3}$$

şi vom revizui semnalul $b(i)$ în semnalul $b^{'}(i)$ după regula deciziei majoritare (dacă jumătate plus unu din sindromuri sunt 1 $b^{'}(i) = \overline{b(i)}$ şi în caz contrar $b^{'}(i) = b(i)$.

S-a arătat în *Zeng* [98] şi [100] că rata erorii s_1 a şirului A în şirul $B^{'} = \{b^{'}(i)\}$ este:

$$s_1 = f_m(s_0) = p - (1-2p)\sum_{k=0}^{m-1} C_{2k+1}^{k}(pq)^{k+1}, \tag{7.4}$$

unde

$$p = p(s_0) = \frac{1-(1-2s_0)^{r-1}}{2}, \quad q = 1-p. \tag{7.5}$$

Mai mult există un număr întreg m_c numit *număr critic* astfel încât:

$$\lim_{k\to\infty} f_m^{(k)}(s_0) = \begin{cases} 0 \text{ dacă } m \geq m_c \\ \frac{1}{2} \text{ dacă } m < m_c. \end{cases} \tag{7.6}$$

unde $f_m^{(k)}$ este $k-$autocompunerea funcţiei f_m.

Aplicarea teoremei de convergenţă pentru relaţia 7.6 sugerează realizarea unor revizuiri iterate folosind un număr fix de $2m+1$ sindromuri şi se bazează pe presupunearea că pe parcursul fiecărei iteraţii comportamentul erorii este independent şi deci $s_{i+1} = f_m(s_i)$. Metoda amintită mai sus şi prezentată în *Zeng* [98] are două hibe:

1) nu minimizează distanţa soluţiei (lungimea minimă a interceptării pentru rezolvarea problemei).

2) nu garantează o probabilitate de succes.

Vom prezenta un nou algoritm (vezi şi *Zeng* [100]) în care se fac următoarele presupuneri:

1) $r = 3$.

2) $f(X)$ este polinom primitiv.

În [100] *Zeng* prezintă convergenţa metodei, probabilitatea de succes şi distanţa soluţiei.

7.7.3. Algoritmul Sindromului Linear

7.7.4. Numere critice şi numere de rafinare

Începem prezentarea algoritmului cu următoarea lemă.

Lema 7.7.1. *Funcţia $w(s) = f_1(s) - s$ are în $(0, \frac{1}{2})$ o unică rădăcină $\alpha \approx 0,1294$. Avem $w(s) < 0$ în $(0,\alpha), w(s) > 0$ în $(\alpha, \frac{1}{2})$ şi $\lim_{k\to\infty} f_1^{(k)}(s) = 0$ dacă $0 \leq s < \alpha$.*

Demonstraţie. Fie $u = 1 - 2s$, avem:

$$w(s) = p^2(3-2p) - s = \frac{(1-u^2)^2}{4}(3-(1-u^2)) - \frac{1-u}{2}$$

$$= \frac{u(u-1)(u^4+u^3+u^2+u-2)}{4} = \frac{u(u-1)h(u)}{4}.$$

Dar $h(u)$ este strict crescătoare în u şi $h(0) = -2, h(1) = 2$ deci $h(u)$ are un singur zero β în intervalul deschis $(0,1)$ care poate fi găsit cu metoda lui Newton a aproximaţiilor succesive ca fiind $\beta \approx 0,7412$. De aici obţinem concluzia despre zeroul α şi semnul lui $w(s)$. Mai mult dacă $s \in (0,\alpha)$ şi dacă definim $s_0 = s$, $s_k = f_1(s_{k-1})$ atunci şirul $s_k, k \geq 0$, descreşte strict la o limită $s^* \in [0,\alpha)$ şi $w(s^*) = 0$. Deci trebuie să avem $s^* = 0$. Aceast lucru demonstrează lema.

Ideea principală a algoritmului sindromului linear este de a face reviziuni iterate cu un număr reductibil de sindromuri cu lungimea segmentului în curs de procesare. Rolul central este jucat de conceptul de numere critice şi numere de rafinare.

Definiţia 7.7.1. *Numărul supercritic* m_{sc} corespunzător ratei iniţiale a erorii s_0 este cel mai mic număr întreg m care satisface ingalitatea:

$$s_k = f_{m-k+1}(s_{k-1}) < s_{k-1} \qquad \forall k = 1,\ldots,m. \tag{7.7}$$

Numărul de rafinare de ordinul t, notat cu l_t, corespunzător ratei iniţiale a erorii s_0 cel mai mic număr întreg l care satisface inegalitatea:

$$f_1^{(l)}(s_{m_{sc}}) < 10^{-t}. \tag{7.8}$$

Teorema 7.7.1. *Pentru orice* $s_0 \in (0,\frac{1}{2})$ *şi orice* $t \geq 0$ *există numerele supercritic* m_{sc} *şi de rafinare* l_t.

Demonstraţie. Avem

$$\lim_{m\to\infty} f_m(s_0) = p - (1-2p)\sum_{k=0}^{\infty} C_{2k+1}^k (pq)^{k+1} = 0.$$

Deci pentru m suficient de mare avem:

$$s_1 = f_m(s_0) < \min(\alpha, s_0).$$

Dar din relaţia 7.4 rezultă:

$$f_i(s) \leq f_1(s), \quad \forall i \geq 1.$$

Deci avem

$$\begin{aligned} s_2 &= f_{m-1}(s_1) < f_1(s_1) < s_1 < \alpha, \\ s_3 &= f_{m-2}(s_2) < f_1(s_2) < s_2 < \alpha, \end{aligned} \tag{7.9}$$

etc. Aceasta înseamnă că mulţimea numerelor întregi pozitive m pentru care condiţia 7.7 este satisfăcută nu este vidă şi deci existenţa numărului m_{sc} corespunzător lui s_0 rezultă din principiul binei ordonări a mulţimii numerelor naturale. Existenţa lui l_t este o consecinţă a lemei anterioare.

Teorema 7.7.1. este o justificare naturală a următorului algoritm de identificare a numerelor supercritice m_{sc}.

PASUL 1. $m_{sc} = 1$, $k = 1$, $s = s_0$.

PASUL 2. Dacă $k = 0$ **STOP**. Altfel calculează $s^{'} = f_k(s)$.

PASUL 3. Dacă $s^{'} < s$ atunci $k = k - 1$, $s = s^{'}$ şi **go to PASUL 2**.

PASUL 4. $m_{sc} = m_{sc} + 1$, $k = m_{sc}$, $s = s_0$ **go to PASUL 2**.

Algoritmul sindromului linear

Acum putem formula algoritmul sindromului linear.

Teorema 7.7.2. *Există un algoritm care să primescă la intrare un număr t şi un segment interceptat din B de lungime*

$$N(s_0, t) = (1 + 2l_t + 2\sum_{m=1}^{m_{sc}} L(m))n = c(s_0, t)n, \tag{7.10}$$

unde $n = grad(f(X))$ şi $L(m) = 2^{\lfloor \frac{4m-1}{6} \rfloor}$, iar ca ieşire furnizează starea **LFSR** *atacat la momentul i cu probabilitatea de succes:*

$$P_{succes} > (1 - 10^{-t})^n. \tag{7.11}$$

Complexitatea computaţională a algoritmului (numărul de operaţii per bit) este

$$\begin{aligned} Q(s_0, t) &= (6l_t^2 + 2m_{sc}(m_{sc} + 2)(2l_t + 1) + 4\sum_{j=1}^{m_{sc}}(2j + 1)D(j - 1))n & (7.12) \\ &= q(s_0, t)n & (7.13) \end{aligned}$$

unde

$$D(j) = L(1) + L(2) + \ldots + L(j),\ D(0) = 0. \tag{7.14}$$

Demonstraţie. Algoritmul este împărţit în două faze: faza de reducere îna care rata iniţială a erorii s_0 este redusă la $s_{m_{sc}} < \alpha$ şi faza de refinare în care eroarea remanentă $s_{m_{sc}}$ este redusă sub 10^{-t}.

La faza de reducere avem nevoie de $p = \lceil \frac{2m_{sc}+1}{3} \rceil$ multiplii trinomiali ai lui $f(x)$. Aceştia vor fi folosiţi pentru a forma sindromuri şi vor fi aleşi ca fiind

$$g_0(X) = f(X), \quad g_{i+1}(X) = g_i(X)^2, \quad 0 \leq i \leq p-2$$

Observăm că fiecare trinom $g(X) = 1 + X^{i_1} + X^{i_2}$ ne conduce la trei sindroame de forma:

$$\sigma_{i,k}(g(x)) = \sum_{p=0}^{2} b(i + i_p - i_k), \quad k = 0, 1, 2, \tag{7.15}$$

pentru acelaşi semnal criptografic $b(i)$. Vom aranja cele $3p$ sindromuri în următoarele două moduri

i) $\sigma_{i,0}(g_0), \sigma_{i,1}(g_0), \sigma_{i,2}(g_0), \ldots, \sigma_{i,0}(g_{p-1}), \sigma_{i,1}(g_{p-1}), \sigma_{i,2}(g_{p-1})$

ii) $\sigma_{i,2}(g_0), \sigma_{i,1}(g_0), \sigma_{i,0}(g_0), \ldots, \sigma_{i,2}(g_{p-1}), \sigma_{i,1}(g_{p-1}), \sigma_{i,0}(g_{p-1})$

Faza de reducere

PASUL 1. $N = c(s_0, t)n$, $m = m_{sc}$, $L = nL(m)$.

PASUL 2. Pentru $L \leq i \leq N - L - 1$, calculează sindromurile date de primele $2m + 1$ formule din i) sau ii) după cum $i \leq \frac{N}{2}$ sau $i > \frac{N}{2}$ şi atribuie $b(i) = \bar{b}\ (i)$dacă cel puţin $m + 1$ sindromuri sunt 1.

PASUL 3. $m = m - 1$. Dacă $m = 0$ **go to PASUL 5**.

PASUL 4. $L = L + nL(m)$ **go to PASUL 2**.

Faza de rafinare

PASUL 5. $m = l_t,\ L = L + n$

PASUL 6. $m = m - 1$. Dacă $m < 0$ **STOP**.

PASUL 7. Pentru $L \leq i \leq N - L - 1$, calculează sindromurile

$$\sigma_{i,0}(f), \sigma_{i,1}(f), \sigma_{i,2}(f)$$

şi atribuie $b(i) = \bar{b}\ (i)$dacă cel puţin două sindromuri sunt 1. **Go to PASUL 6**.

Observăm că la a $(m_{sc} - m + 1)-$ iteraţie a fazei de reducere semnalele $b(i)$ cu:

$$L \leq i \leq N - L - 1, \tag{7.16}$$

sunt revizuite după regula deciziei majoritare cu $2m + 1$ calcule dintr-o plajă mai mare de semnale $b(j)$ cu:

$$L - nL(m) \leq j \leq N - L + nL(m) - 1. \tag{7.17}$$

Prin inducţie matematică după m rezultă că după $(m_{sc} - m + 1)-$iteraţii avem:

$$\Pr(b(i) \neq a(i)) = s_m, \tag{7.18}$$

pentru toţi i care satisfac 7.16 demonstrând că la începutul iteraţiei avem în semnalele $b(j)$ cu j satisfăcând 7.17 toate datele necesare calculului celor $2m+1$ sindromuri necesare. Evident este suficient să verificăm punctul pentru cazul:

$$i = \lfloor \frac{N}{2} \rfloor, \quad 2m + 1 \equiv 1 \mod 3. \tag{7.19}$$

Folosind inegalitatea (care se verifică uşor) $\sum_{k=1}^{m} L(k) \geq 2^{\lceil \frac{2m+1}{3} \rceil}$ se demonstrează că cel mai mic j pentru care semnalul $b(j)$ este folosit în calculul sindroamelor este

$$j_{\max} = \lceil \frac{N}{2} \rceil + n2^{\frac{2m+1}{3}} \leq N - L + nL(m) - 1. \tag{7.20}$$

Aceasta înseamnă că restricţia este îndeplinită la fiecare iteraţie şi rata iniţială a erorii $s_0$va fi redusă sub α după faza de reducere. Acesta va fi redusă sub 10^{-t} după faza de rafinare. Deci după $m_{sc} + l_t$ iteraţii ale procesului de revizuire algoritmul va scoate un vector de dimensiune n :

$$(b(\frac{N-n}{2}), b(\frac{N-n}{2} + 1), \ldots, b(\frac{N-n}{2} + n - 1)), \tag{7.21}$$

care coincide cu starea vectorială $\frac{N-n}{2}$ a **LFSR** atacat cu probabilitatea 7.11

Formula 7.12 pentru complexitatea computaţională poate fi dedusă prin calcule directe.

7.8. Corecţia iterativă a erorii

7.8.1. Prezentare generală

Problema cea mai importantă din criptografie constă în recuperarea textului clar dintr-un mesaj cifrat. În aceast paragraf prezentăm trei algoritmi de refacere a textului cifrat în ipoteza că acesta este un **XOR** dintre ieşirea unui registru de deplasare şi textul clar. Metodele se pot extinde la combinaţii de registre de deplasare de *tip cascadă Gollmann*. În continuare formulăm trei principii pe care se bazează algoritmii ce vor fi prezentaţi. Metodele ce le prezentăm se mai numesc şi *metode*

de tip sindrom liniar. Cele trei principii pe baza cărora vom construi algoritmii de reconstrucţie sunt

P.1. Corecţia erorilor este bazată pe un număr satisfăcător de ecuaţii de control.

P.2. Corecţia erorilor se bazează pe estimarea probabilităţilor a posteriori obţinute prin folosirea ca probabilitate a priori în iteraţia curentă media estimaţiilor probabilitilor a posteriori din iteraţia precedentă.

P.3. Corecţia erorilor se bazează pe estimarea probabilităţilor a posteriori obţinute prin folosirea ca probabilitate a priori în iteraţia curentă estimaţia probabilităţii a posteriori din iteraţia precedentă.

7.8.2. Prezentarea algoritmilor de corecţie iterativă

În acest paragraf vom formula problema reconstrucţiei ieşirii unui **LFSR**. Vom prezenta cei trei algoritmi corespunzători celor trei principii formulate mai sus.

Fie $\{x_n\}_{n=1}^N$ ieşirea dintr-un **LFSR** de lungime L cu w reacţii. În modelul statistic se presupune că un şir de zgomot (binar) $\{e_n\}_{n=1}^N$ este realizarea unui şir de variabile binare i.i.r. $\{E_n\}_{n=1}^N$ astfel încât $P(E_n = 1) = p$ pentru $n = 1, \ldots, N$.

Fie $\{z_n\}_{n=1}^N$ versiunea perturbată a lui $\{x_n\}_{n=1}^N$ definit de:

$$z_n = x_n \oplus e_n, \quad n = 1, 2, \ldots, N \tag{7.22}$$

Se presupun cunoscute reacţiile **LFSR**, parametrul p şi un segment $\{z_n\}_{n=1}^N$. Se cere textul clar corespunzător $\{e_n\}_{n=1}^N$.

Fie $\Pi_n = \{\pi_k(n)\}_k$ o mulţime de *ecuaţii de control ortogonale* pentru bitul n care este generată cu ajutorul multiplilor polinomului caracteristic, $n = 1, 2, \ldots, N$.

Definim $c_k(n) = \sum\limits_{\substack{mod\ 2 \\ l \in \pi_k(n)}} z_l, k = 1, 2, \ldots, |\Pi_n|,\ n = 1, 2, \ldots, N$. $c_k(n)$ este o realizare a variabilei aleatoare binare $C_k(n)$, cu $k = 1, 2, \ldots, |\Pi_k|$. Notăm prin $\Pr(En, \{C_k(n)\}_{k=1}^N)$ *distribuţia comună de probabilitate* a variabilelor E_n şi $C_k(n),\ k = 1, 2, \ldots, |\Pi_n|$ şi prin $\Pr(E_n|\{C_k(n)\}_{k=1}^N)$ *probabilitatea a posteriori* corespunzătoare.

Descifrare majoritară

Iniţializare: $i = 1$, $I =$ constant (numărul de iteraţii), $p^{(0)} = p$.

PASUL 1: $i = i + 1$. Dacă $i > I$ **go to PASUL 5**.

PASUL 2: Calculează: $c_k(n),\ k = 1, 2, \ldots, |\Pi_k|,\ n = 1, 2, \ldots, N$.

PASUL 3: Calculează: $t_n = |\Pi_n| - 2 \sum\limits_{k=1}^{|\Pi_n|} c_k(n),\ n = 1, \ldots, N$ (t_n este funcţia de decizie majoritară).

PASUL 4: Dacă $t_n < 0$ atunci $z_n = z_n \oplus 1,\ n = 1, \ldots, N$.

go to PASUL 1.

PASUL 5: **STOP**.

Apriori - aposteriori medie

Iniţializare: $i = 1$, I= constant (numărul de iteraţii), $p^{(0)} = p$.
PASUL 1: $i = i + 1$. Dacă $i > I$ **go to PASUL 6**.
PASUL 2: Calculează: $c_k(n),\ k = 1, 2, \ldots, |\Pi_k|, n = 1, 2, \ldots, N$.
PASUL 3: Pentru $n = 1, \ldots, N$ calculează:

$$P_n^{(i)} = \Pr(E_n = 1 | \{C_k(n)\}_{k=1}^{|\Pi_n|} = \{c_k(n)\}_{k=1}^{|\Pi_n|}) = \tag{7.23}$$

$$= \frac{p^{(i)} p_w^{s_n} (1 - p_w)^{|\Pi_n| - s_n}}{p^{(i)} p_w^{s_n} (1 - p_w)^{|\Pi_n| - s_n} + (1 - p^{(i)})(1 - p_w)^{s_n} p_w^{|\Pi_n| - s_n}} \tag{7.24}$$

unde

$$s_n = \sum_{k=1}^{|\Pi_n|} c_k(n), \tag{7.25}$$

(numărul ecuaţiilor de control care nu se verifică) şi

$$p_w = \frac{1 - (1 - 2p^{(i)})^w}{2} \text{ (vezi 7.5).} \tag{7.26}$$

PASUL 4: Dacă $P_n^{(i)} > 0,5$ atunci $z_n = z_n \oplus 1$ şi $P_n^{(i)} = 1 - P_n^{(i)},\ n = 1, \ldots, N$.
PASUL 5: $p^{(i)} = \frac{1}{N} \sum\limits_{n=1}^{N} P_n^{(i)}$ **go to PASUL 1**.
PASUL 6: STOP.

Apriori - aposteriori

Iniţializare: $i = 1, I$= constant (numărul de iteraţii), $p_n^{(0)} = p, n = 1..N$.
PASUL 1: $i = i + 1$. Dacă $i > I$ **go to PASUL 6**.
PASUL 2: Calculează: $c_k(n),\ k = 1, 2, \ldots, |\Pi_k|,\ n = 1, 2, \ldots, N$.
PASUL 3: Pentru $n = 1, \ldots, N$ calculează:

$$P_n^{(i)} = \Pr(E_n = 1 | \{C_k(n)\}_{k=1}^{|\Pi_n|} = \{c_k(n)\}_{k=1}^{|\Pi_n|}) = \tag{7.27}$$

$$\frac{p_n^{(i)} \prod\limits_{l=1}^{|\Pi_n|} p_l(n)^{c_l(n)} (1 - p_l(n))^{\bar{c}_l(n)}}{p_n^{(i)} \prod\limits_{l=1}^{|\Pi_n|} p_l(n)^{c_l(n)} (1 - p_l(n))^{\bar{c}_l(n)} + (1 - p_n^{(i)}) \prod\limits_{l=1}^{|\Pi_n|} (1 - p_l(n))^{c_l(n)} p_l(n)^{\bar{c}_l(n)}} \tag{7.28}$$

unde $\bar{c}_l\ (n) = 1 - c_l(n)$ şi

$$p_l(n) = \frac{1 - \prod_{j=1}^{w} (1 - 2p_{m_j}^{(i)})}{2} \text{ (vezi 7.5)}, \tag{7.29}$$

unde $\{m_j\}_{j=1}^{w}$ este mulţimea indicilor biţilor implicaţi în ecuaţiile de control $\pi_l(n),\ l = 1, 2, \ldots, |\Pi_n|,\ n = 1, 2, \ldots, N$.

PASUL 4: Dacă $P_n^{(i)} > 0,5$ atunci $z_n = z_n \oplus 1$ şi $P_n^{(i)} = 1 - P_n^{(i)} n = 1, \ldots, N$.

PASUL 5: $p_n^{(i)} = P_n^{(i)}$ pentru $n = 1, 2, \ldots, N$ **go to PASUL 1**.

PASUL 6: **STOP**.

7.8.3. Rezultate experimentale

Prezentăm în figura 7.3 rezultatele obţinute în cazul unui **LFSR** de lungime 47 cu două reacţii la poziţiile 5 şi 47. Lungimea şirului observat este de 10^5 biţi. Conform rezultatelor toţi algoritmii sunt convergenţi dacă zgomotul este sub o anumit limită care depinde de lungimea şirului observat. Pentru zgomot mai mare primul algoritm care cade este P.1. iar ultimul este P.3.

7.8.4. Concluzii

1. Când zgomotul este mai mic decât o anumit limită (de exemplu. $p = 0,4$) toţi algoritmii sunt convergenţi la soluţia problemei. Algoritmul P.1. are costul de reconstrucţie cel mai mic.

2. În cazul în care avem un zgomot pentru care algoritmul P.1 este divergent algoritmii P.2. şi P.3. sunt convergenţi. Algoritmul P.2. are costul de reconstrucţie cel mai mic.

3. În cazul în care atât algoritmii P.1 şi P.2 sunt divergenţi algoritmul P.3 este convergent şi pentru a minimiza costul de reconstrucţie trebuie să procedăm în felul următor: realizăm iniţial corecţia erorilor ajutorul algoritmului P.3 şi după câteva iteraţi rulăm algoritmii P.1 şi P.2.

	Numărul de reziduri								
	Algoritm P.1.			Algoritm P.2.			Algoritm P.3.		
Iteraţia	p_1	p_2	p_3	p_1	p_2	p_3	p_1	p_2	p_3
1	40357	44440	45774	37728	41693	43077	37728	41693	43077
2	40383	45868	47301	35734	41397	43015	34462	40943	42712
3	39343	46758	47301	33477	41002	42934	30249	40194	42397
4	36610	47147	48388	30400	40659	42814	24943	39270	42211
5	31750	47468	48763	26130	40259	42821	15333	38191	41977
6	23614	47779	48626	19808	39827	42657	5719	36618	41796
7	13714	47610	48699	11850	39214	42522	1484	34849	41376
8	6246	47530	48817	6315	38544	42423	117	32711	41133
9	1820	47736	48667	3184	38935	42359	2	30097	40768
10	230	47606	48699	717	38661	42335	0	26603	40515
11	0	47528	48704	13	38432	42347		22190	40156
12		47574	48820	0	38216	42346		16766	39918
13		47478	48962		38028	42326		11810	39579
14		47532	48854		37870	42337		8403	39307
15		47551	48878		37688	42315		6110	39033
16		47466	48872		37505	42344		4006	38755
17		47578	48852		37320	42344		2198	38420
18		47613	48623		37127	42358		831	38079
19		.	48790		36940	42348		139	37718
20		.	48704		36661	42340		0	37277
21		.	48800		36304	42338			36800
22		.	48776		35838	42340			36235
23		.	48785		35225	42343			36655
24		.	48763		34429	42349			35003
25		.	48862		33569	42351			34262
26		.	48762		32504	42356			32350
27		.	48835		31189	42350			31183
28		.	48818		29703	42353			29750
29		.	48893		28146	42355			28273
30		.	48805		26409	42352			25309
31		.	48833		24191	42352			23818
32		.	48816		21280	42352			22280
33		.	48835		18105	42358			20518
34		.	48789		15042	42360			18441
35		.	48801		12245	42360			15922
36		.	.		9443	42360			12801
37		.	.		7080	42360			9685
38		.	.		5197	42360			7140
39		.	.		3446	42360			5337
40		.	.		1910	42360			3837
41		.	.		745	42360			2604
42		.	.		122	42360			1317
43		.	.		0	.			329
44		.	.			.			3
45		.	.			.			0

Figura 7.3: Studiu comparativ al eficienţei algoritmilor.

7.9. Algoritm de criptanaliză diferenţială

Descrierea metodei: Procedura prezentată este o procedură din clasa criptanalizei diferenţiale şi testează dacă anumite poziţii din cheie se regăsesc cu o probabilitate mai mare decât altele în criptogramă. Testul, numit test de criptanaliză diferenţială, poate fi efectuat la orice risc de ordinul 1. Acest test este de fapt o aproximare de ordinul 2 (*ponderea Hamming* a cuvântului diferenţelor cheilor este de 2) a unui proces de testare mult mai complex. Statistica testului este de tip normal.

Scopul testului: Testul pune în evidenţă punctele slabe (dacă există) ale algoritmului de cifrare.

Prezentăm în continuare schema în pseudocod a algoritmului.

Intrare. Se alege o cheie de bază bine precizată $K = (k_1, \ldots, k_n)$ cu $k_i \in \{0, 1\}$.

Ieşire. Punctele slabe ale algoritmului de cifrare (prin intermediul cheii) şi decizia referitoare la rezistenţa la criptanaliză diferenţială.

PASUL 0. Citeşte rata de respingere a testului α.

PASUL 1. (*Construcţie chei perturbate.*) Se construiesc n seturi de chei perturbate pornind de la cheia $\mathbf{K}$:

for $i = 1$ to n do $K^{(i)} = (\delta_{1i} \oplus k_1, \ldots, \delta_{ni} \oplus k_n)$:

$$\delta_{ji} = \begin{cases} 1 \text{ dacă } j \neq i, \\ 0 \text{ dacă } j = i. \end{cases}$$

pentru $i, j = 1, \ldots, n$. Altfel spus cheia i este obţinută din cheia de bază prin negarea bitului de pe poziţia i.

PASUL 2. (*Construcţie criptograme.*) Se construiesc $n+1$ criptograme pornind de la cheia de bază, cheile perturbate şi un text clar M. Notăm aceste criptograme cu $C^{(i)}, i = 1, \ldots, n+1$. Ca text clar M se poate opta pentru textul $0-$peste tot.

PASUL 3. (*Construcţie matrice de corelaţie.*) Se construieşte matricea $(n + 1) \times (n + 1)$ a valorilor de corelaţie $\mathbf{C}$:

$$c_{ij} = \text{corelaţia(criptograma } i\text{, criptograma } j),$$

corelaţia c_{ij} fiind de fapt valoarea statisticii testului de frecvenţă aplicată secvenţei (criptograma $i\oplus$ criptograma j). Matricea $\mathbf{C}$ este o matrice simetrică având 1 pe diagonala principală.

PASUL 4. (*calcul valori semnificative.*) Se numără valorile de corelaţie semnificative de deasupra diagonalei principale. O valoare se numeşte *semnificativă* dacă:

$$c_{ij} \notin [u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}].$$

Fie T numărul de valori semnificative (T este numărul de respingeri al testului de corelaţie).

PASUL 5. (*Decizia şi interpretarea rezultatelor.*) Dacă:

$$\frac{T-\alpha\frac{n(n+1)}{2}}{\sqrt{\alpha(1-\alpha)\frac{n(n+1)}{2}}} \notin [u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}],$$

atunci se decide nerezistenţa la criptanaliză diferenţială ($u_{\frac{\alpha}{2}}$ şi $u_{1-\frac{\alpha}{2}}$ sunt cuantilele repartiţiei normale de ordinul $\frac{\alpha}{2}$, respectiv $1-\frac{\alpha}{2}$) şi fişează elementele (i,j), cu $n \geq i > j \geq 1$, pentru care c_{ij} este semnificativ. Aceste elemente constituie puncte slabe pentru algoritm. În caz contrar nu putem preciza nimic despre rezistenţa la acest tip de atac.

Observaţie. Cheia iniţială $\mathbf{K}$ se poate alege $(0,\ldots,0)$.

7.10. Câteva tehnici de proiectare

Există numeroase tehnici care pot fi folosite pentru a garanta impredictibilitatea cheilor-flux (vezi *Rueppel* [65]). Vom investiga modul în care unele abordări sfârşesc prin a eşua sau modul în care slăbiciunea lor este expusă unor atacuri de tipul celor menţionate anterior. Pentru fiecare generator prezentat vom indica cele două caracteristici importante şi anume complexitatea linear echivalentă şi perioada sa. Reţinem că dacă avem n registre de deplasare notate prin $\mathbf{LFSR}_i$, $i = 1,\ldots,n$ de lungimi L_i astfel încât $L_i \neq L_j$ $\forall i \neq j$, care sunt combinate cu ajutorul unei funcţii nelineare f, adică ieşirea gereratorului este dată de formula $y = f(x_1,\ldots,x_n)$ unde x_i este ieşirea din $\mathbf{LFSR}_i$ iar f este în forma normală atunci complexitatea generatorului rezultant este $\bar{f}\ (L_1, L_2, \ldots, L_n)$ unde $\bar{f}$: $\mathbf{Z}^n \rightarrow \mathbf{Z}$ este extensia funcţiei f de la $\mathbf{Z}_2^n$ la mulţimea $\mathbf{Z}^n$.

7.10.1. Transformarea neliniară feed-forword

Multe generatoare de chei-flux combină două sau mai multe generatoare folosind o funcţie neliniară. Problema constă în alegerea unei astfel de funcţii suficient de puternică pentru efectul dorit. Vom menţiona în continuare două tehnici.

7.10.2. Generatorul Geffe

Unul din cele mai simple moduri de a combina trei **LFSR**-uri este folosirea unui multiplexer doi-la-unu (vezi figura 7.4). Dacă ieşirea celor trei **LFSR**-uri la momentul t este $a_1(t), a_2(t)$, respectiv $a_3(t)$, ieşirea multiplexer-ului va fi:

$$b(t) = a_1(t)a_3(t) \oplus a_1^{'}(t)a_2(t) = a_2(t) \oplus a_1(t)(a_2(t) \oplus a_3(t)).$$

Dacă polinoamele primitive de feedback ale celor trei **LFSR**-uri au gradele n_1, n_2, respectiv n_3, generatorul va avea complexitatea liniară:

$$LC = n_3 n_1 + (n_1 + 1)n_2$$

şi perioada (dacă n_1, n_2 şi n_3 sunt prime între ele):

$$T = (2^{n_1} - 1)(2^{n_2} - 1)(2^{n_3} - 1).$$

Slăbiciunea acestui generator provine din faptul că probabilitatea de coincidenţă este destul de mare:

$$p = \Pr(b(t) = a_2(t)) = \Pr(a_1(t) = 0) + \Pr(a_1(t) = 1)\Pr(a_3(t) = a_2(t)) = 0,75.$$

Probabilitatea de coincidenţă între $b(t)$ şi $a_3(t)$ poate fi estimată analog. În consecinţă, dacă trinoamele primitive de feedback ale **LFSR**-urilor sunt cunoscute, având fiecare gradul mai mic sau egal cu n, sistemul poate fi uşor spart de un atac care foloseşte metoda sindromului liniar. Stările curente ale celor trei **LFSR**-uri pot fi depistate dintr-un segment de lungime $N = 37n$ al secvenţei de ieşire. Costul de calcul este de doar 896 operaţii pe n biţi.

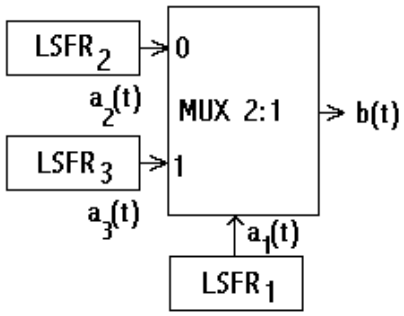

Figura 7.4: Generatorul Geffe.

7.10.3. Generatorul Jennings

Şi aceasta metodă foloseşte un multiplexer pentru a combina două LFSR-uri de lungime l şi n. Metode similare acesteia au fost recomandate de *EBU* (*European Broadcasting Television*) pentru codificarea transmisiunilor de televiziune. Generatorul produce un semnal de ieşire $c(t),\ t \geq 0$, în felul următor: fie două **LFSR**-uri

cu secvenţele de ieşire $a(t),\ t < l$, respectiv $b(t),\ t < n$. Pentru început, fixăm un număr natural $h \leq \min(l, [log2n])$ şi un şir de poziţii pe banda primului **LFSR:**

$$0 \leq i_0 < i_1 < \ldots < i_{h-1} \leq l - 1.$$

La fiecare moment $t \geq 0$, calculăm numărul:

$$u(t) = a(t + i_0) + a(t + i_1)2 + \ldots + a(t + i_{h-1})2^{h-1},$$

şi îl transformăm în:

$$\theta(u(t)) = s_0(t) + s_1(t)2 + \ldots + s_{k-1}(t)2^{k-1}, \quad k = [log2n],$$

unde $\theta : \{0, 1, \ldots, 2^{h-1}\} \to \{0, 1, \ldots, n - 1\}$ este o aplicaţie injectivă. Dacă polinoamele primitive de feedback ale celor două **LFSR**-uri sunt cunoscute, atunci aplicaţia θ împreună cu stările iniţiale K_1, K_2 ale celor două **LFSR**-uri formează cheia secretă a acestui sistem de criptare. Semnalul de ieşire este definit prin:

$$c(t) = b[t + \theta(u(t))].$$

Dacă $(l, n) = 1$, secvenţa de ieşire are perioada $(2^l - 1)(2^n - 1)$ şi complexitatea liniară:

$$LC \leq n(1 + \sum_{i=1}^{h} C_i^l),$$

cu egalitate dacă poziţiile alese pe banda primului **LFSR** sunt la distanţe egale. Dacă privim, însă, cu atenţie definiţia lui $c(t)$ observăm că, indiferent de forma lui θ, semnalul poate fi exprimat ca o combinaţie liniară a biţilor $b(t)$, cu coeficienţi care depind doar de K_1. Aceasta slăbiciune poate fi exploatată printr-un atac al consistenţei liniare. În *Zeng* [99], se arată că, dacă polinoamele de feedback sunt cunoscute, generatorul Jennings poate fi spart pe o secvenţă de ieşire de lungime $N \geq l + n2^h$ cu 2^{h+1} teste de consistenţă şi având ca scop al căutării exhaustive doar pe K_1. Deşi familia posibilelor aplicaţii θ este foarte mare, contribuţia acestei chei la puterea de criptare a sistemului este neglijabilă. La fel putem spune şi despre K_2.

7.10.4. Generatorare cu tact controlat

Pe lângă a supune secvenţele **LFSR**-urilor la diverse transformari feedforward neliniare, un mijloc important de întarire a sistemului îl reprezintă controlul tactului **LFSR**-urilor, cu ajutorul unei secvenţe de control. Aceasta se poate realiza prin diverse tehnici.

Generatorul Beth-Piper

În generatoarele intermitente informaţia de pe fiecare nivel nu se mai deplasează la fiecare moment t cu o poziţie. De aceasta dată, fiecare generator din sistem are un ceas care decide dacă informaţia se deplasează sau ramâne pe aceleaşi poziţii. Generatorul Beth-Piper (vezi figura 7.5) conţine trei **LFSR**-uri, cu secvenţele de ieşire a_1, a_2, respectiv a_3, în care $\mathbf{LFSR_1}$ şi $\mathbf{LFSR_3}$ folosesc acelaşi ceas, în timp ce ceasul $\mathbf{LFSR_2}$ este controlat de $\mathbf{LFSR_1}$, astfel încât $\mathbf{LFSR_2}$ se deplasează la momentul t doar dacă $a_1(t-1) = 1$. Impunând condiţii corespunzătoare pentru gradele n_1, n_2, n_3 ale celor trei **LFSR**-uri, secvenţa de ieşire $b(t)$ va avea complexitatea liniară:

$$LC = (2^{n_1} - 1)n_2 + n_3$$

şi perioada:

$$T = (2^{n_1} - 1)(2^{n_2} - 1)(2^{n_3} - 1),$$

Deşi secvenţa de ieşire are o complexitate liniară foarte mare, siguranţa acestui generator este foarte slabă. Să presupunem că $a_2'(t)$ reprezintă semnalul de ieşire la momentul t al generatorului $\mathbf{LFSR_2}$ în cazul în care el funcţionează fără intermitenţe. Atunci, avem probabilitatea de coincidenţă:

$$p = \Pr[b(t) \oplus b(t+1) = a_3(t) \oplus a_3(t+1)] =$$

$$= \Pr(a_1(t) = 0) + \Pr(a_1(t) = 1) \Pr(a_2'(t)) = a_2'(t+1)) = \frac{1}{2} + \frac{1}{4}.$$

Deci, dacă polinoamele de feedback ale lui $\mathbf{LFSR_1}$ şi $\mathbf{LFSR_3}$ sunt cunoscute, putem aplica metoda sindromului liniar pentru a descoperi mai întâi secvenţa a_3 din b şi apoi secvenţa a_1 din a_2. Pentru aceasta, nu avem nici măcar nevoie să cunoaştem polinomul de feedback al lui $\mathbf{LFSR_2}$.

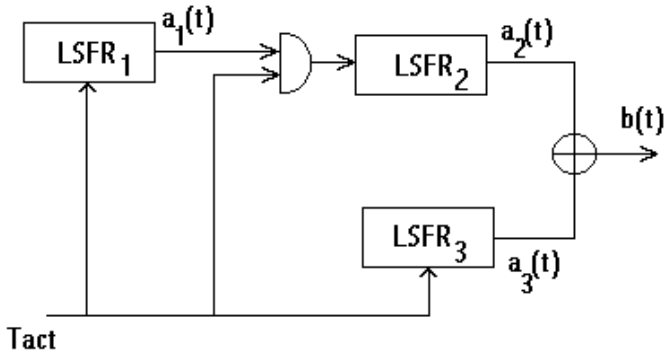

Figura 7.5: Generatorul Beth-Piper.

Generatorul Gollmann

Figura 7.6 arată o versiune îmbunătaţită a generatorului Beth-Piper. Ea constă într-o serie de l **LFSR**-uri, având polinoame primitive de feedback de acelaşi grad n. Ceasul $\mathbf{LFSR_i}$ este controlat de toate $\mathbf{LFSRj}$, $j < i$, la fel ca în generatorul Beth-Piper. S-a arătat că secvenţa de ieşire are perioada:

$$T = (2^n - 1)^l,$$

şi o complexitate liniară foarte mare

$$LC \geq n(2^{n-1} - 1)^{l-1}.$$

Există însă unele slăbiciuni ale sistemului care au fost relevate în literatura de specialitate.

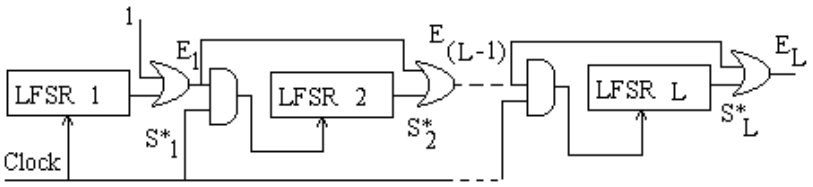

Figura 7.6: Generatorul Gollman.

Generator cu control intermitent bilateral

O altă metodă se bazează pe observaţia că dacă o secvenţă binară $\mathbf{b}$ are ca perioadă un număr prim impar T, complexitatea sa liniară $LC(\mathbf{b})$ va fi marginită inferior de ordinul $Ord_T(2)$ al numărului 2 modulo T, adică:

$$LC(b) \geq Ord_T(2).$$

O secvenţă având perioadă primă este de preferat, deoarece ea poate fi supusă la diverse transformări criptografice ulterioare fără a-i influenţă marginea inferioară a complexităţii sale liniare, ceea ce arată că transformările nu o vor transforma într-o secvenţă constantă. Totuşi, dacă $T = 2^n - 1$ este număr prim *Mersenne*, vom avea $Ord_T(2) = n$. Deci T trebuie ales ca fiind un număr prim diferit de termenii progresiei $2^n - 1$. Secvenţe cu perioada de forma $q2^n - 1$, cu $q \geq 3$ impar, au fost

construite cu o pereche de **LFSR**-uri cu $n-$nivele. De exemplu, pentru un generator al unei secvenţe de perioadă $5 \times 2^{n-2} - 1$, controlul tactului se poate realizeaza astfel:

1. dacă $(a(t+n-1), a(t+n-2)) = (0,1)$, atunci tactul generatorului $\mathbf{LFSR_2}$ este blocat;

2. dacă $(b(t+n-1), b(t+n-2)) = (0,1)$, dar $(a(t+n-1), a(t+n-2)) \neq (0,1)$, atunci tactul generatorului $\mathbf{LFSR}_1$ este blocat.

Complexitatea liniară a acestui generator are un ordin de mărime aproximativ egal cu cel al perioadei. Nu s-a observat nici o redundanţă evidentă a cheii.

7.10.5. Generatoare cu ceasuri multiple

Sisteme cu mai multe ceasuri

Tehnicile discutate pâna aici pornesc de la presupunerea ca **LFSR**-urile studiate lucrează cu aceeaşi viteză ca cea a ieşirii finale. Câţiva cercetători au propus ca **LFSR**-urile să lucreze la viteze mai mari decât cea a ieşirii finale. Deşi presupune utilizarea a mai mulţi cicli ai ceasului sistemului pentru obţinerea unui singur bit, printr-o folosire judicioasă metoda poate deschide noi posibilităţi în proiectarea unor generatori interesanţi.

Generatorul Massey-Rueppel cu mai multe viteze

Acest generator utilizează două **LFSR** puse să lucreze la viteze diferite. Al doilea **LFSR** este proiectat să lucreze la viteza $d \leq 2 \times$ viteza primului **LFSR**, iar semnalul de ieşire este produs după regula:

$$c(t) = \sum_{1=0}^{l-1} a(t+i)b(dt+1).$$

Factorul de viteză este variabil şi este utilizat ca parte a cheii. Dacă amândouă **LFSR**-urile au polinoame primitive de feedback şi în plus $(l,n) = 1$, $(d, 2n-1) = 1$ atunci secvenţa de ieşire va avea o complexitate liniară $LC = ln$ cu perioada $T = (2^l - 1)(2^n - 1)$ şi cu excelente proprietăţi statistice. Din nou, caracterul biliniar al definiţiei semnalului $c(t)$ pentru un d fixat lasă destul spaţiu pentru ca atacul consistenţei liniare să fie aplicat aici. Se poate arăta că dacă polinoamele de feedback sunt cunoscute criptanalistul poate determina factorul de viteza d şi stările iniţiale ale celor două **LFSR**-uri prin efectuarea a $(d_{\max} - 1)(2^l - 1)$ teste de consistenţă aplicate unui segment de ieşire de lungime $N \geq l + n + \log_2 d_{\max}$.

7.10.6. Generatoare autodecimate

Este folosit un **LFSR** cu un polinom primitiv de feedback care îşi autocontrolează ceasul în felul urmator: când semnalul de ieşire este 0 sunt aplicate d tacturi de ceas **LFSR**-ului; altfel numărul tacturilor aplicate este k. Efectul este că la formarea secvenţei de ieşire anumite semnale ale $m-$secvenţei generate sunt sărite sau decimate dupa o regulă pseudoaleatoare determinată chiar de acea $m-$secvenţă. După cum se remarcă schema aceasta nu are şanse de utilizare în criptografie. Deşi schemele propuse par a fi nesatisfăcătoare, ideea autodecimării este interesantă; multe ramân de făcut până ce un generator cu bune calitaţi criptografice va putea fi propus. Clasa generatoarelor de tip **LFSR** poate fi extinsă la clasa generatoarelor tip **FSR**. Un **FSR** este un registru de deplasare în care reacţiile sunt nelineare. Ieşirea unui registru **FSR**, de perioadă maximă, se numeşte şir *de Bruijn*.

7.11. Exemplu de atac criptanalitic

Tehnici de programare în criptanaliză. Reamintim că în cadrul operaţiei de recuperare a criptogramelor se folosesc de regulă algoritmi de tip:

-liniar/neliniar: funcţia ce decide corectitudinea unei decriptări este liniară, respectiv neliniară;

-dinamic/stochastic: se ţine cont de structura probabilistă a literelor din limba respectivă;

-divide et impera: problema se sparge în subprobleme a căror rezolvare este mult mai uşoară;

-backtracking: se explorează în mod sistematic soluţiile posibile cu ajutorul unor condiţii de continuare;

-greedy: este o tehnică ce furnizează optimul local care poate să nu fie optimul global;

-brut: se explorează toate posibilităţile.

Descrierea acestor tehnici este prezentată în ultimul capitol.

În cele ce urmează se prezintă un exemplu de aplicare a tehnicii divide et impera.

Prezentarea generală a sistemelor de cifrare atacate. Vom prezenta, pe scurt criptanaliza unor sistemelor de cifrare în care operaţia de cifrarea este însumarea mod 2 a textului clar cu un generator de perioadă P, deci $C = T \oplus G$, unde prin T am notat textul clar şi prin G generatorul pseudoaleatoriu. Se presupune că $P \ll \dim(T)$.

Situaţia in care atacul poate avea loc. Atacul este eficient în cazul în care cunoaştem structura statistică a textului clar (de exemplu, este un text dintr-o anumită limbă).

Algoritmul de atac. Procedura de atac este următoarea:

Intrare: Criptograma c codificată pe $L-$biţi, lungimea perioadei P a generatorului pseudoaleatoriu.

Ieşire: cheia de cifrare g de perioadă P.

PASUL 0. Defineşte mulţimea cea mai probabilă S (aceasta se poate estima pe baza unui text clar numit text de referinţă) şi probabilităţile a priori de apariţie a literelor din limba respectivă. Vom nota cu $p_0, \ldots, p_{2^L-1}$ valorile acestor probabilităţi.

Pentru determinarea fiecărei componente $g[i]$ $(i = 1, \ldots, P)$ a lui g se procedează astfel:

PASUL 1. $k = 0$;

PASUL 2. Se determină mulţimea:

$$A[i;k] = \{c[i] \oplus k, c[i+P] \oplus k, .., c[i+P \cdot j] \oplus k, \ldots\}$$

PASUL 3. $k = k + 1$;

PASUL 4. Dacă $k < 2^L$ **goto PASUL 2.**

PASUL 5. $g[i]$ este acel element care maximizează expresia:

$$\sum_{j=0}^{2^L-1} \frac{card\{f | f \in A[i;k], f = j\}}{n} p_j,$$

deci:

$$A[i; g[i]] = \max_{k=0,\ldots,2^L-1} \sum_{j=0}^{2^L-1} \frac{card\{f | f \in A[i;k], f = j\}}{n} p_j.$$

sau altfel spus probabilitatea ca $g(k)$ să fie cheia reală este mai mare ca probabilitatea ca k sa fie cheia reală.

Observaţii. i) În această metodă de atac se foloseşte entropia textului (redundanţa caracterelor este mare în cazul fişierelor text).

ii) Se mai poate adăuga o buclă pentru contorizarea lui P găsind deci şi lungimea cea mai probabilă de cheie notată cu P_{opt}.

iii) Algoritmii de cifrare, bazaţi pe sumare repetitivă mod 2 a textului cu un generator pseudoaleatoriu, nu sunt siguri dacă textul clar este comparabil cu mărimea perioadei generatorului.

iv) Randamentul metodei descrise este direct proporţional cu fidelitatea textului de referinţă.

v) Prezentul algoritm este aplicat cu succes pentru utilitarele de cifrare implementate în programele Wincrypt, produsul Word (din pachetul Office 95) şi utilitarul Norton Navigator: în funcţie de parola citită se iniţializează un generator pseudoaleatoriu de perioadă P (512 bytes pentru WinCrypt, 16 bytes pentru Word 95 şi 1 byte pentru Norton Navigator) cu ajutorul căruia se face o sumare modulo 2 a textului clar, obţinându-se astfel criptograma.

7.12. Aplicaţii

Exerciţiul 7.12.1. Fie $\mathbf{LFSR}_i$, $i = 1, 2$ două registre de deplasare cu polinoamele de feedback asociate primitive şi de lungimi $L_1 \neq L_2$. Notăm cu $x_i(t), i = 1, 2$ ieşirile din aceste registre.

a) Considerăm *sistemul sumă modulo* 2 dat de ieşirea $y(t) = x_1(t) \oplus x_2(t)$. Complexitatea liniar echivalentă a sistemului sumă modulo 2 va fi $L_1 + L_2$.

b) Considerăm *sistemul produs* dat de ieşirea $y(t) = x_1(t) \times x_2(t)$. Complexitatea liniar echivalentă a sistemului sumă modulo 2 va fi $L_1 \times L_2$.

Exerciţiul 7.12.2. Fie n registre de deplasare notate prin $\mathbf{LFSR}_i$, $i = 1, \ldots, n$ de lungimi L_i astfel încât $L_i \neq L_j \ \forall i \neq j$, care sunt combinate cu ajutorul unei funcţii neliniare f, adică ieşirea gereratorului este dată de formula $y = f(x_1, \ldots, x_n)$ în care x_i este ieşirea din $\mathbf{LFSR}_i$ iar f este în forma normală. Atunci complexitatea generatorului rezultant este $\bar{f}\ (L_1, L_2, \ldots, L_n)$ unde $\bar{f}$: $\mathbf{Z}^n \to \mathbf{Z}$ este extensia funcţiei f de la $\mathbf{Z}_2^n$ la mulţimea $\mathbf{Z}^n$.

Răspuns. Demonstraţia acestei afirmaţii rezultă din aplicarea succesivă a exerciţiului 1 punctele a) şi b).

Exerciţiul 7.12.3. Studiaţi complexitatea unui algoritm de cifrare obţinut prin:

i) decimarea cu o anumită rată a ieşirii unui registru de deplasare liniar;

ii) interclasarea cu ratele de eşantionare m şi n a două registre de deplasare liniare de complexităţi L_1 respectiv L_2.

iii) aplicarea unei transpoziţii de lungime n asupra ieşirii unui registru de deplasare liniar.

iv) aplicarea unei permutări de lungime n asupra ieşirii unui registru de deplasare liniar.

Exerciţiul 7.12.4. Problemă similară pentru perioada algoritmului obţinut prin aceleaşi reguli impuse la exerciţiul 7.12.3.

Exerciţiul 7.12.5. Generatorul Geffe, numit şi multiplexor 2 : 1, este descris de trei registre de deplasare $\mathbf{LFSR}_i$ iar ieşirea de formula: $y(t) = x_1(t) \cdot x_3(t) \oplus \bar{x}_1(t) \cdot x_2(t)$. Care este complexitatea şi perioada acestui generator?

Răspuns. Se aplică exerciţiul 2. Complexitatea liniar echivalentă a acestui generator va fi $L_1 \times L_3 + (1 + L_1) \times L_2$.

Exerciţiul 7.12.6. Aplicaţi tehnica sindromului liniar pentru generatorul Geffe.

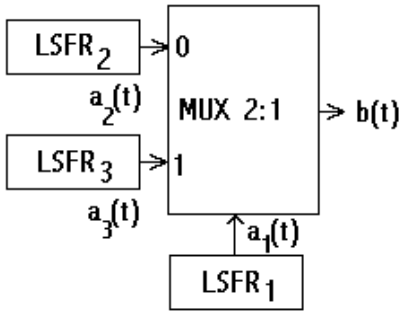

Figura 7.7: Generatorul Geffe.

Răspuns. Generatorul *Geffe* (multiplexor 2:1) este descris de trei registre de deplasare $\mathbf{LFSR}_i$ iar ieşirea de formula:

$$b(t) = a_1(t)a_3(t) + \bar{a}_1\ (t)a_2(t).$$

Ieşirea din generatorul G la momentul t va fi:

$$b(t) = a_1(t)a_3(t) + \bar{a}_1\ (t)a_2(t) = a_2(t) + a_1(t)(a_2(t) + a_3(t)).$$

Dar

$$s_0 = \Pr(b(t) \neq a_2(t)) = \frac{1}{4}.$$

Aceasta este punctul slab al sistemului. De exemplu pentru $n \leq 100$ şi pentru $t = 4$ şirul $A_2 = \{a_2(t)\}$ poate fi recuperat cu probabilitate de succes de $0,99$ dacă lungimea şirului interceptat este de $37n$.

Să presupunem că am recuperat A_2. Comparăm semnalele $a_2(t)$ cu $b(t)$ pentru $0 \leq t \leq 37n$ şi reţinem acele momente t_i pentru care

$$a_2(t_i) + b(t_i) = 1.$$

Se observă că la aceste momente:

$$a_1(t_i) = 1, \quad a_3(t_i) = b(t_i).$$

Împărţim fiecare putere X^{t_i} cu $f_1(X)$ şi $f_3(X)$ şi obţinem resturile

$$X^{t_i} = r_{i,0} + r_{i,1}X + \ldots + r_{i,n_1-1}X^{n_1-1}\ mod\ f_1(X),$$

$$X^{t_i} = s_{i,0} + s_{i,1}X + \ldots + s_{i,n_3-1}X^{n_3-1} \; mod \; f_3(X).$$

Avem apoi două sisteme lineare de aproximativ $9n$ ecuaţii:

$$r_{i,0}a_1(0) + r_{i,1}a_1(1) + \ldots + r_{i,n_1-1}a_1(n_1-1) = 1,$$

$$s_{i,0}a_3(0) + s_{i,1}a_3(1) + \ldots + s_{i,n_3-1}a_3(n_3-1) = b(t_i).$$

Dacă $n_1, n_2 << 9n$ atunci aceste sisteme vor determina cu probabilitate aproape de 1 stările iniţiale ale registrelor $\mathbf{LFSR}_1$ şi $\mathbf{LSFR}_3$.

Exerciţiul 7.12.7. Aplicaţi tehnica sindromului liniar pentru generatorul *Beth-Piper*. Care este complexitatea şi perioada acestui generator?

Răspuns. Generatorul *Beth-Piper* este descris tot de trei registre de deplasare $\mathbf{LFSR}_i$ iar ieşirea de formula

$$b(t) = a_3(t) \oplus a_2^{'}(t).$$

unde $a_2^{'}(t)$ este ieşirea din $\mathbf{LFSR}_2$ (cu tactul controlat de $\mathbf{LFSR}_1$) care are tact dacă şi numai dacă $a_1(t) = 1$.

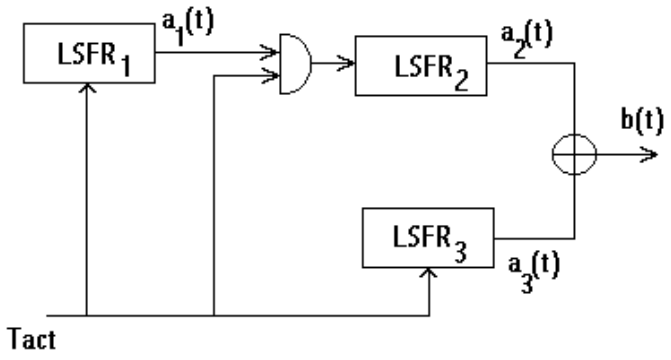

Figura 7.8: Generatorul Beth-Piper.

Dacă polinoamele de feedback $f_1(X)$ şi $f_3(X)$ sunt trinoame de grad cel mult n iar $f_2(X)$ este arbitrar necunoscut atunci sistemul Beth-Piper poate fi spart dintr-un segment interceptat de lungime $N = 37n$ iar costul computaţional va fi $Q = 1792n$ operaţii pe bit. Nu prezentăm aici metoda de criptanaliză ea fiind identică cu cea de la teorema anterioară.

Exerciţiul 7.12.8. Se consideră un algoritm generator pseudoaleator compus din opt registre de feedback (reacţiile necunoscute) cu tactul condiţionat de ieşirea anteriorului. Ieşirile celor opt registre sunt concatenate pentru a forma un byte. Dezvoltaţi o procedură eficientă de atac asupra generatorului. Care este complexitatea de spargere a algoritmului propus? Care este complexitatea liniar echivalentă a algoritmului pseudoaleator?

Răspuns. Se va decima cu o rată de $1:7$ (citesc din 8 în 8) ieşirea generatorului. Şirul binar obţinut va indica după $2L$ observaţii (cu ajutorul algorimtului Berlekamp-Massey) starea iniţială şi reacţiile primului registru de deplasare. Vom afla apoi tactul celui de al doilea registru. Vom decima cu aceeiaşi rată ieşirea generatorului (punctul de plecare va fi bitul 2). Cu ajutorul algoritmului Berlekamp-Massey vom reconstrui starea iniţială şi reacţile celui de al doilea registru., etc.

Exerciţiul 7.12.9. Care este complexitatea de spargere a sistemului de protecţie a datelor oferit de produsul *Norton Navigator 95*?

Răspuns. Sistemul de cifrare constă în sumarea mod 2 cu un bytes obţinut din mixarea caracterelor din parolă. Complexitatea de spargere este $O(1)$.

Exerciţiul 7.12.10. Scrieţi un program care să rezolve probelma accesului la datele cifrate cu programul de tehnoredactare *Word 95*. Problemă similară pentru programul *Wincrypt*.

Răspuns. La utilitarul Word se face XOR cu o secvenţă de 16 bytes iar la utilitarul Wincrypt se face XOR cu o secvenţă de 512 bytes. Complexitatea atacului este $O(n)$.

Exerciţiul 7.12.11. Studiaţi metoda de protecţie prin parolă oferită de utilitarele *Microsoft Access* şi *Microsoft Schedule*.

Exerciţiul 7.12.12. Utilitarele *Lotus Organizer 1.0* şi *Lotus Organizer 2.0* oferă o facilitate de protecţie a datelor prin intermediul unei parole. Care este schema de realizare a semnăturii parolei şi care este forma algoritmului de cifrare a datelor?

Exerciţiul 7.12.13. Studiaţi protecţia prin parolă oferită de sistemul *WP*.

Exerciţiul 7.12.14. Care este diferenţa dintre sistemele de protecţie folosite de programul *Word 2000* în versiunea engleză şi versiunea franceză? Studiu similar pentru produsul *Excel 2000*.

Exerciţiul 7.12.15. Ce complexitate de spargere are sistemul de protecţie oferit de utilitarul de arhivare *arj*? Dar sistemul de protecţie de la utilitarele de arhivare *zip* şi *rar*?

Răspuns. Indicaţii privind descrierea sistemelor de cifrare menţionate în exerciţiile 7.12.9-7.12.15 se pot găsi, spre exemplu, la adresa www.password-crackers.com sau la siteul oficial corespunzător produsului analizat.

Exerciţiul 7.12.16. Fie (x_n) o secvenţă binară produsă de polinomul de feedback $f(X)$ (de grad L) care este primitiv. Care este numărul de biţi de 0 şi numărul de biţi de 1 din (x_n) pe o perioadă de repetiţie a recurenţei liniare ataşate?

Răspuns. $n_0 = 2^{L-1} - 1, n_1 = 2^{L-1}$.

Exerciţiul 7.12.17. Fie (x_n) şi (y_n) două secvenţe binare produse de polinoamele de feedback $f(X)$ (de grad L_1) respectiv $g(X)$ (de grad L_2) care sunt primitive. Care este perioada şi complexitatea secvenţei binare produse prin decimarea lui (x_n) de (y_n).

Exerciţiul 7.12.18. Polinomul $1+X+X^4$ este ireductibil? Dar primitiv? Este adecvat utilizării ca polinom generator pentru un $LFSR$ criptografic?

Capitolul 8

CRIPTANALIZA CIFRURILOR BLOC

Nearly every inventor of a cipher system was been convinced of the unsolvability of his brain child.
David Kahn

8.1. Introducere şi concepte generale

Prezentul capitol face o prezentare a noţiunii de cifru bloc, a modurilor de operare precum şi a principalelor caracteristici ale acestora. În finalul capitolului se prezintă o serie de tehnici şi metode de criptanaliză a cifrurilor bloc care vor fi exemplificate pe noul standard de cifrare bloc AES (*Advanced Encryption Standard*).

Cifrurile bloc procesează informaţia pe blocuri de o lungime stabilită apriori. În cele ce urmează vom nota prin n lungimea blocului procesat (exprimată în biţi), V_n spaţiul vectorilor n dimensionali şi prin $\mathcal{K}$ spaţiul cheilor. Un bloc de text clar se va nota prin M iar un bloc de text cifrat se va nota prin C.

Definiţia 8.1.1. Un cifru bloc pe n biţi este o funcţie $E : V_n \times K \to V_n$ astfel încât pentru orice cheie $k \in K$ funcţia $E(\cdot, k)$ este o funcţie inversabilă (funcţia de cifrare cu ajutorul cheii k) din V_n în V_n. Funcţia inversă este funcţia de decifrare şi va fi notată prin $D_K(\cdot) = E_K^{-1}(\cdot)$.

Reamintim că a *sparge un cifru* nu înseamnă în mod obligatoriu de a găsi o cale practică astfel încât un interceptor să recupereze textul clar numai din criptograme. În cadrul criptografiei academice, regulile sunt relaxate considerabil. A sparge un

cifru înseamnă a găsi o slăbiciune care poate fi exploatată pentru recuperare cheii şi/sau a textului cu o complexitate mai mică decât atacul brut.

8.2. Securitatea şi complexitatea atacurilor

Obiectivul unui cifru bloc este să asigure confidenţialitatea. Obiectivul corespunzător al adversarului este să descopere textul original din textul cifrat.

Un cifru bloc este *total spart* dacă a fost descoperită cheia şi spart parţial dacă adversarul poate reconstitui o parte din textul clar, dar nu şi cheia din textul cifrat.

Pentru a evalua securitatea unui cifru bloc se obişnuieşte ca întotdeauna să se presupună că adversarul:

i) are acces la toate datele transmise prin canalul de text cifrat;

ii) (presupunerea lui Kerckhoff): ştie toate detaliile despre funcţia de cifrare, mai puţin cheia secretă.

Cele mai importante clase de atacuri pentru cifrurile cu chei simetrice sunt:

- atac pe baza textului cifrat;
- atac pe baza textului clar/cifrat;
- atac pe baza textului clar/cifrat ales de criptanalist;
- atac pe baza cheilor.

8.3. Criterii de evaluare a cifrurilor bloc

Următoarele criterii pot fi folosite pentru a evalua cifrurile bloc:

- nivelul de securitate estimat (pe baza rezistenţei la anumite tipuri de atacuri);
- mărimea cheii;
- mărimea blocului de date care se cifrează;
- complexitatea funcţiei de cifrare;
- expansiunea datelor;
- propagarea erorilor;
- viteza de cifrare a datelor.

8.4. Moduri de operare

Există două moduri principale de utilizare în practică a algoritmilor simetrici: cifrarea bloc şi cifrarea secvenţială. Cifrarea bloc operează cu blocuri de text clar şi cifrat-de regulă de 64 de biţi, uneori chiar mai mari. Cifrarea secvenţială operează cu secvenţe de text clar şi cifrat de un bit sau octet. În cazul cifrării bloc, acelaşi bloc de text clar va fi cifrat de fiecare dată în acelaşi bloc de text cifrat, folosind

aceeaşi cheie. În cazul cifrării secvenţiale, secvenţele similare de text clar vor fi cifrate diferit în cazul unor cifrări repetate.

Modurile de cifrare constituie combinaţii ale celor două tipuri de bază, unele folosind metode feedback, altele realizând simple operaţii. Aceste operaţii sunt simple, deoarece securitatea este atributul cifrării şi nu al modului în care se realizează schema de cifrare. Mai mult, modul de realizare a cifrării nu duce la compromiterea securităţii date de algoritmul de bază.

Un cifru bloc cifrează textul în blocuri de n biţi de mărimi fixe. În general este folosită valoarea $n = 64$ biţi. Cele mai cunoscute moduri de operare sunt ECB (*electronic code book*), CBC (*cipher block chaining*), CFB (*cipher feedback block*) şi OFB (*output feedback block*). În funcţie de modul de operare al cifrului bloc se vor aplica atacuri specifice. Vom nota în cele ce urmează, cu E_k se notează funcţia de cifrare a blocului în timp ce cu D_k notăm funcţia de descifrare.

8.4.1. Modul ECB (electronic code-book)

Modul ECB este cea mai obişnuită formă de cifrare bloc: un bloc de text clar este transformat într-un bloc de text cifrat, fiecare bloc fiind cifrat independent şi fiecare cheie fiind diferită de celelalte. Dacă acelaşi bloc de text clar se cifrează întotdeauna în acelaşi bloc de text cifrat, teoretic este posibilă o carte de coduri în care să se facă asocierea text clar-text cifrat. Pentru blocuri de 64 de biţi rezultă un număr de 2^{64} intrări în cartea de coduri- mărime prea mare pentru a permite memorarea şi manipularea.

Modul ECB este cel mai simplu mod de lucru, fiecare bloc de text clar fiind cifrat independent. Cifrarea se poate face luând aleator blocuri din cadrul fişierului. Acest mod de cifrare este indicat pentru cifrarea documentelor care sunt accesate aleator, ca de exemplu baze de date, unde fiecare înregistrare poate fi adăugată, ştearsă, cifrată sau descifrată independent de celelalte.

Problema ridicată de ECB este aceea că, dacă un criptanalist are pentru câteva mesaje şi textul clar şi pe cel cifrat, poate afla codul, fără a deţine şi cheia de cifrare.

Padding-ul

Completarea blocurilor (*padding*) este folosită atunci când mesajele nu se împart exact în blocuri de 64 de biţi. Se completează ultimul bloc cu un model zero-unu, alternând cifrele de 0 şi 1, astfel încât blocul să devină complet.

Algoritmul ECB

Intrare: Cheia K de k biţi, mesajul clar $M = M_1, \ldots, M_t$ pe blocuri de n biţi.

Ieşire: Textul cifrat $C = C_1, \ldots, C_t$ care ulterior se descifrează pentru a descoperi textul original.

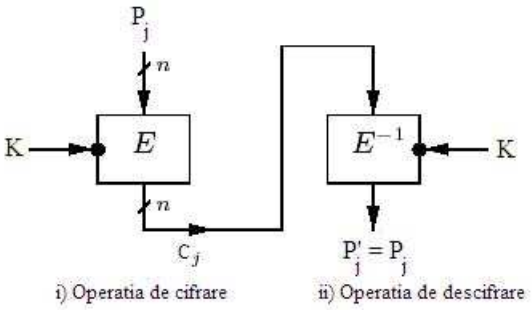

Figura 8.1: Modul de lucru ECB.

1. *Cifrarea*: pentru orice $i = 1, \ldots, t$ avem: $C_i = E_k(M_i)$.
2. *Descifrarea*: pentru orice $i = 1, \ldots, t$ avem: $M_i = D_k(C_i)$.

Proprietăţi ale modului de operare ECB

Următoarele proprietăţi rezultă din modul de construcţie al cifrului bloc tip ECB.
1. Din texte identice rezultă texte cifrate identice.
2. Dependenţa în lanţ: blocurile sunt cifrate independent de alte blocuri.
3. Propagarea erorilor: una sau mai multe erori într-un singur bloc de texte cifrate afectează descifrarea numai a acelui bloc.

8.4.2. Modul CBC (cipher-block chaining)

Acest mod foloseşte un mecanism *feedback*, deoarece rezultatul cifrării unui bloc anterior revine prin buclă şi intervine în cifrarea blocului curent. Cu alte cuvinte, blocul anterior este folosit pentru a modifica cifrarea următorului bloc. În acest fel, textul cifrat nu mai depinde doar de textul clar, ci şi de modul de cifrare al blocului anterior.

În modul CBC, textul clar, înainte de a intra în modul de cifrare propriu-zis, este însumat mod 2 cu blocul de text cifrat anterior. Figura 8.2 prezintă operaţia de cifrare/descifrare în modul CBC.

După ce blocul de text clar este cifrat, textul cifrat rezultat este stocat într-un registru al buclei de reacţie. Înainte ca următorul text clar să fie cifrat, el este sumat mod 2 cu blocul din registrul de reacţie şi devine următoarea intrare în rutina de cifrare. După cifrare , conţinutul registrului este înlocuit cu blocul cifrat. În acest fel, cifrarea blocului i depinde de toate cele $i - 1$ blocuri anterioare.

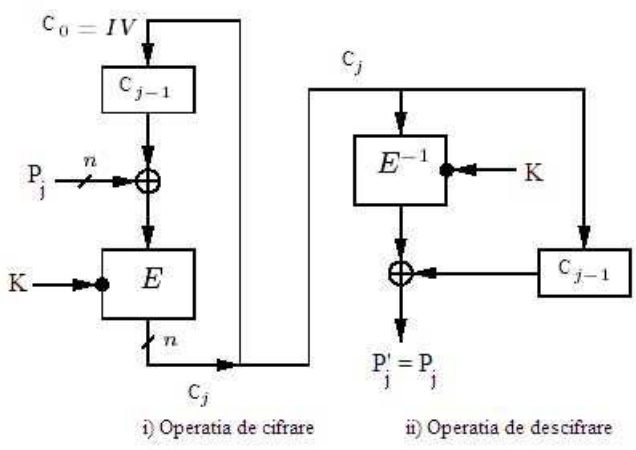

Figura 8.2: Modul de lucru CBC.

În procesul de descifrare (care este exact procesul invers cifrării), textul cifrat este descifrat normal şi depozitat în registrul de reacţie. După ce următorul bloc este descifrat, el este făcut sumă mod 2 cu conţinutul registrului de reacţie.

Din punct de vedere matematic, procesul de cifrare arată astfel:

$$C_i = E_k(P_i \oplus C_{i-1}).$$

Ecuaţiile corespunzătoare operaţiei de descifrare sunt:

$$P_i = C_{i-1} \oplus D_k(C_i).$$

Vectorul de iniţializare

Modul de lucru CBC face ca acelaşi text să se transforme în blocuri diferite de text cifrat. Două mesaje identice se vor transforma în acelaşi mesaj cifrat. Mai mult, două mesaje care încep la fel vor fi cifrate identic până la prima diferenţă. Prevenirea acestui lucru se poate face cifrând primul bloc cu un vector de date aleator. Acest bloc de date aleatoare se numeşte vector de iniţializare notat cu IV, numit şi variabilă de iniţializare sau valoare iniţială pentru înlănţuire. Vectorul IV nu are nici un înţeles de sine stătător, el doar face cifrarea oricărui mesaj, unică. Când receptorul descifrează acest bloc, el îl utilizează pentru iniţializarea registrului buclei de reacţie.

Vectorul de iniţializare nu trebuie să fie secret acesta poate fi transmis în clar împreună cu mesajul cifrat. Dacă acest lucru pare greşit, să considerăm că avem

un mesaj din câteva blocuri, $B_1, B_2, \ldots, B_i$, astfel încât B_1 este cifrat cu IV, B_2 este cifrat utilizând ca vector de iniţializare textul cifrat de la B_1,etc. Dacă avem n blocuri, sunt $n-1$ vectori de iniţializare expuşi, chiar dacă vectorul original este ţinut secret.

Padding-ul

Completarea blocurilor este analoagă celei de la modul ECB. Există însă aplicaţii în care textul cifrat trebuie să aibă aceeaşi mărime ca textul clar. Este foarte probabil ca fişierul ce urmează să fie cifrat să fie pus în aceeaşi locaţie de memorie. În acest caz rămâne un ultim bloc, mai mic, a cărui cifrare se va face diferit. Se presupune că ultimul bloc are j biţi. După cifrare, ultimul bloc întreg, se mai cifrează încă o dată, se selectează ultimii j biţi din rezultat şi se face suma mod 2 cu blocul incomplet.

Algoritmul CBC

Intrare: Cheia K pe k biţi, vectorul iniţial IV de n biţi, mesajul clar $M = M_1, \ldots, M_t$ pe blocuri de n biţi.

Ieşire: Textul cifrat $C = C_1, \ldots, C_t$ care ulterior se descifrează pentru a descoperi textul original.

1. *Cifrarea*: $C_0 = IV$, şi recursiv avem

$$C_j = E_k(C_{j-1} \oplus M_j).$$

2. *Descifrarea*: $C_0 = IV$, pentru orice $j = 1, \ldots, t$ avem

$$M_j = C_{j-1} \oplus D_k(C_j).$$

Proprietăţi ale modului de operare CBC

Au loc următoarele proprietăţi ale modului de lucru CBC.

1. Texte identice: blocuri de texte cifrate, identice, rezultă când acelaşi text este cifrat sub aceeaşi cheie şi cu acelaşi vector IV. Schimbând IV, cheia sau primul bloc de text, rezultă un text cifrat diferit.

2. Dependenţa în lanţ: mecanismul în lanţ face ca textul cifrat c_j să depindă de x_j şi de blocurile de text precedente.

3. Propagarea erorilor: o singură eroare de un bit în blocul textului cifrat c_j afectează descifrarea blocurilor c_j şi c_{j+1}.

4. Dacă o eroare (inclusiv pierderea unuia sau mai multor blocuri) apare în blocul c_j, dar nu în c_{j+1}, c_{j+2} este corect descifrat din x_{j+2}.

8.4.3. Modul CFB (cipher feedback)

În timp ce modul CBC procesează textul de n biţi odată, unele aplicaţii cer ca unităţi de r biţi ale textului să fie cifrate şi transmise fără întârziere pentru un număr fixat $r \leq n$. În acest caz se foloseşte modul CFB.

Modul CFB poate fi privit ca un mod secvenţial cu autosincronizare, însă la nivel de blocuri de text cifrat. Algoritmul CFB la nivel de bloc operează cu o coadă de mărime egală cu cea a blocului. În primă fază aceasta este iniţializată cu VI , analog modului CBC.

Dacă mărimea blocului de cifrat este n , atunci modul CFB pe n biţi are forma pentru cifrare

$$C_i = P_i \oplus E_k(C_{i-1}),$$

respectiv pentru descifrare

$$P_i = C_i \oplus E_k(C_{i-1}).$$

Figura 8.3 ilustrează această formă.

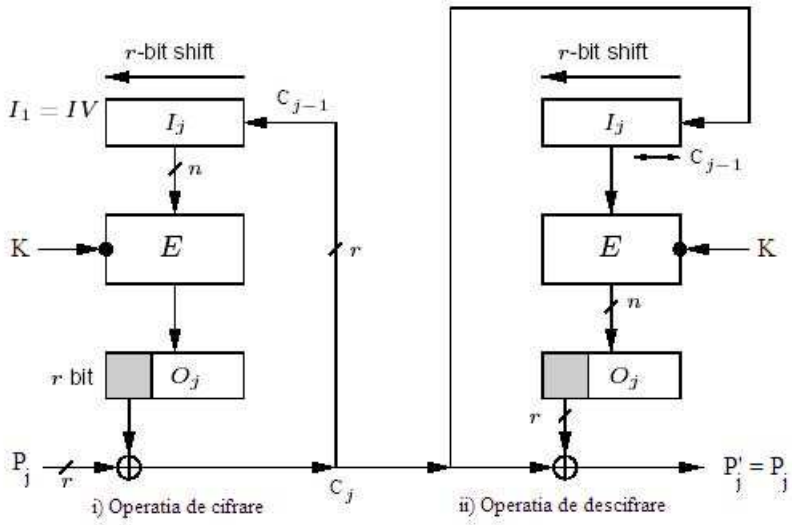

Figura 8.3: Modul de lucru CFB.

Algoritmul CFB

Intrare: Cheia K pe k biţi, vectorul iniţial IV de n biţi, textul clar pe blocuri de r biţi $M = M_1, \ldots, M_t$ cu $r \leq n$.

Ieşire: Blocuri de text cifrate pe r biţi care se descifrează pentru a descoperi textul clar.

1. *Cifrarea*: $I_1 = IV$ (I_j este valoarea de intrare a unui registru de deplasare). Fie blocul de r biţi: $x_1, \ldots, x_r$. Pentru $1 \leq j \leq r$ se execută:

a) $O_j = E_k(I_j)$ (calculează blocul cifrat de ieşire);

b) $t_j =$"cei mai la stânga" r biţi ai lui O_j;

c) $c_j = x_j \oplus t_j$;

d) $I_{j+1} = 2^r I_j + c_j \bmod 2^n$.

2. *Descifrarea*: $I_1 = IV$, pentru $1 \leq j \leq r$ calculează: $x_j = c_j \oplus t_j$, unde t_j, O_j, I_j sunt calculaţi ca mai sus.

Observaţia 8.4.1. Dacă $r = n$ atunci recurenţa corespunzătoare cifrării este:

$$C_j = M_j \oplus E_k(M_{j-1}),$$

cu

$$M_0 = IV.$$

8.4.4. Modul OFB (output feedback)

Modul de operare OFB poate fi folosit în aplicaţiile în care toate erorile de propagare trebuie să fie evitate. Este asemănător cu modul CFB şi permite cifrarea blocurilor cu diverse mărimi, dar diferă în sensul că rezultatul funcţiei de cifrare E serveşte ca feedback.Acest mod se numeşte, uneori, cu reacţie internă, deoarece bucla de reacţie este independentă atât de textul clar, cât şi de cel cifrat.

Dacă mărimea blocului este n, modul OFB pe n biţi se scrie pentru cifrare:

$$C_i = P_i \oplus S_i;$$

iar pentru descifrare:

$$P_i = C_i \oplus S_i;$$

unde

$$S_i = E_k(S_{i-1}),$$

reprezintă starea care este independentă de textul clar şi de cel cifrat.

Modul OFB pe n biţi este prezentat în figura 8.4.

Există două versiuni:

-ISO (necesită un feedback de n biţi şi este mai sigură);

-FIPS (permite $r < n$ biţi de feedback).

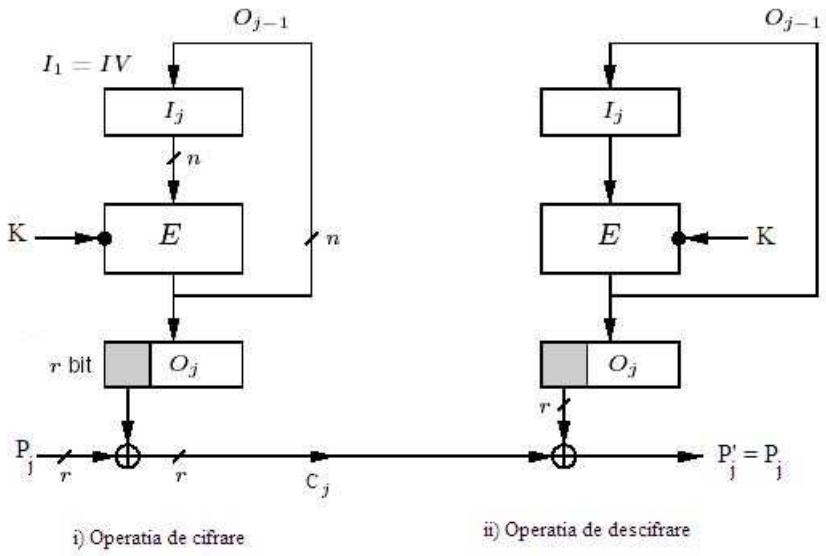

Figura 8.4: Modul de lucru OFB pe n biţi.

Algoritm OFB (pentru ISO):

Intrare: Cheia k, vectorul IV pe n biţi, blocuri de text clar pe r biţi $x_1, \ldots, x_r$ $(1 \leq r \leq n)$

Ieşire: Blocuri de text cifrate $c_1, \ldots, c_r$ pe r biţi.

1. *Cifrarea*: $I_1 = IV$. Pentru $1 \leq j \leq r$, dându-se x_j

a) $O_j = E_k(I_j)$.

b) t_j sunt cei mai de la stânga r biţi ai lui O_j.

c) $c_j = x_j \oplus t_j$.

d) $I_{j+1} = O_j$.

2. *Descifrarea*: $I_1 \leftarrow IV$. Pentru $1 \leq j \leq r$ calculăm: $x_j = c_j \oplus t_j$, unde t_j, O_j, I_j sunt calculaţi ca mai sus.

Algoritm OFB (cu feedback de r biţi) (pentru FIPS 81) :

Intrare: Cheia k pe k biţi, IV pe n biţi, blocurile de text pe r biţi $x_1, \ldots, x_r$ cu $1 \leq r \leq n$:

Ieşire: Blocurile de text cifrate $c_1, \ldots, c_r$.

Ca şi în algoritmul OFB, dar înlocuind relaţia

$$I_{j+1} = O_j$$

cu

$$I_{j+1} = (2^r I_j + t_j) \bmod 2^n.$$

Observaţia 8.4.2. O variaţie a modului de lucru OFB este *modul de lucru contor* în care trecerea de la o stare la alta este dată de funcţia de incrementare: $C_i = P_i \oplus S_i$; unde $S_i = E_k(contor[i])$, $contor[i+1] = contor[i] + 1$.

8.4.5. Modul BC (block chaining)

A folosi un algoritm bloc în modul BC înseamnă a face suma mod 2 între intrarea în blocul de cifrare şi suma mod 2 a tuturor blocurilor cifrate anterior. În acelaşi fel ca în CBC, procesul va fi declanşat de un IV.

Ecuaţiile care descriu comportarea modului de operare BC sunt, pentru cifrare

$$C_i = E_k(M_i \oplus F_i),$$

unde

$$F_{i+1} = F_i \oplus C_i,$$

cu

$$F_1 = IV.$$

Operaţia de descifrare are forma:

$$P_i = F_i \oplus D_k(C_i).$$

8.4.6. Modul BC cu sumă de control (BC-checksum)

Modul BC cu sumă de control este similar modului BC simplu, deosebirea făcându-se la ultimul bloc în care se face suma mod 2 a blocurilor clare anterioare. Modul BC cu sumă de control ne asigură de faptul că orice modificare în structura cifrată ablocurilor se va evidenţia cu ocazia descifrării ultimului bloc. Dacă acest ultim bloc are în componenţă şi elemente de integritate atunci testul de integritate se va face foarte simplu.

8.4.7. Modul OFBNLF (output feedback block with a nonlinear function)

OFBNLF este o variantă atât a modului OFB, cât şi a celui ECB, în care cheia se schimbă la fiecare bloc.

Ecuaţiile care descriu comportarea acestui mod de operare sunt, pentru cifrare:

$$C_i = E_{k_i}(P_i),$$

pentru descifrare:

$$P_i = D_{k_i}(C_i),$$

unde:

$$K_i = E_k(K_{i-1}).$$

8.4.8. Cascade de cifruri şi cifrări multiple

Există mai multe modalităţi de a combina algoritmi bloc pentru a obţine noi algoritmi de cifrare. Scopul urmărit este de a încerca să se îmbunătăţească securitatea şi prin alte mijloace decât prin scrierea unui nou algoritm. Cifrarea multiplă este una din tehnicile combinării: se utilizează un algoritm pentru a cifra acelaşi text clar de mai multe ori, cu mai multe chei.Cascadarea are acelaşi principiu, dar utilizează mai multe chei.

Definiţia 8.4.1. O cascadă de cifruri este concatenarea a $L \geq 2$ blocuri de cifruri (numite etape), fiecare cu chei independente.

Textul este intrarea primei etape iar ieşirea (rezultatul) etapei i este intrarea etapei $i+1$ şi ieşirea etapei L este ieşirea cascadei textului cifrat.

Definiţia 8.4.2. Cifrarea multiplă este similară unei cascade de L cifruri identice, dar cheile etapei nu trebuie să fie independente şi cifrurile etapei pot fi, fie un cifru bloc E, fie funcţie de descifrare corespunzătoare $D = E^{-1}$.

Cifrarea dublă

Un mod simplu de îmbunătăţire a securităţii unui algoritm bloc este de a cifra un bloc cu două chei diferite. Prima dată se cifrează blocul cu prima cheie, apoi întregul bloc de text cifrat rezultat se cifrează cu cea de a doua cheie. Descifrarea este procesul invers.

Definiţia 8.4.3. Cifrarea dublă este definită ca $E(x) = E_{K_2}(E_{K_1}(x))$.

Blocul de text cifrat rezultat din dubla cifrare ar trebui să fie mult mai greu de spart utilizându-se o căutare exhaustivă. Pentru o lungime a cheii de n biţi, faţă de 2^n variante posibile de chei la o singură cifrare , avem 2^{2^n} variante posibile, ceea ce pentru un algoritm de 64 de biţi înseamnă 2^{128} de chei posibile.

Metoda Davies-Price. Este o variantă de CBC, a cărei reprezentare matematică este următoarea:

-pentru cifrare:

$$C_i = E_{k_1}(P_i \oplus E_{k_2}(C_{i-1})),$$

-respectiv pentru descifrare:

$$P_i = D_{k_1}(C_i) \oplus D_{k_2}(C_{i-1}).$$

Double OFB/Counter. Această metodă utilizează un algoritm bloc pentru a genera două şiruri de chei ce se utilizează la cifrarea unui text clar.

Astfel putem scrie pentru variabila internă S_i :

$$S_i = E_{k_1}(S_{i-1} \oplus I_1),$$

unde (I_1 este variabilă tip contor)

$$I_1 = I_1 + 1,$$

şi respectiv pentru variabila internă T_i :

$$T_i = E_{k_2}(T_{i-1} \oplus I_2),$$

unde (I_2 este variabilă tip contor)

$$I_2 = I_2 + 1.$$

Funcţia de cifrare este:

$$C_i = P_i \oplus S_i \oplus T_i,$$

Cifrarea triplă

Definiţia 8.4.4. Cifrarea triplă este definită ca:

$$E(x) = E^{(3)}_{K_3}(E^{(2)}_{K_2}(E^{(1)}_{K_1}(x)))$$

unde $E^{(j)}_K$ este E_k sau $D_K = E^{-1}_K$.

Cifrarea triplă cu două chei. O idee mai bună, propusă de Tuchman, prelucrează un bloc de trei ori cu două chei: mai întâi cu prima cheie, apoi cu a doua, apoi din nou cu prima. El sugerează ca expeditorul să cifreze de trei ori, aşa cum specifică algoritmul, iar destinatarul, la descifrare, va urma algoritmul invers: descifrare cu prima cheie, cifrare cu a doua, descifrare cu prima.

$$C = E_{k_1}(D_{k_2}(E_{k_1}(P))),$$

$$P = D_{k_1}(E_{k_2}(D_{k_1}(C))).$$

Această metodă se mai numeşte şi EDF (*encrypt-decrypt-encrypt).*Dacă algoritmul are o cheie de n biţi, atunci schema va avea o cheie de $2n$ biţi. Modelul cifrare-descifrare a fost proiectat de IBM pentru a proteja implementările convenţionale ale algoritmului şi a fost adăugat la DES şi în standardele ISO.

Cifrarea triplă cu trei chei. În cazul cifrării triple, numărul de trei chei este cel mai potrivit. Lungimea cheii este mare, iar memorarea sa nu poate fi o problemă.

$$C = E_{k_3}(D_{k_2}(E_{k_1}(P))),$$

$$P = D_{k_1}(E_{k_2}(D_{k_3}(C))).$$

Alte variante de cifrare triplă.

Inner CBC

Se cifrează întregul mesaj în modul CBC, de trei ori diferit. Sunt necesari trei vectori de iniţializare diferiţi.

Operaţia de cifrare are următoarea formă:

$$C_i = E_{k_3}(S_i \oplus C_{i-1}),$$

unde

$$S_i = D_{k_2}(T_i \oplus S_{i-1}),$$

şi

$$T_i = E_{k_1}(P_i \oplus T_{i-1}).$$

Pentru descifrare se foloseşte următoarea formulă:

$$P_i = T_{i-1} \oplus D_{k_1}(T_i),$$

unde

$$T_i = S_{i-1} \oplus E_{k_2}(S_i),$$

şi

$$S_i = C_{i-1} \oplus D_{k_3}(C_i).$$

iar C_0, S_0 şi T_0 sunt vectori de iniţializare.

Outer CBC

Se cifrează întregul fişier în modul CBC, de trei ori diferit. Este necesar un singur vector de iniţializare.

Pentru cifrare:

$$C_i = E_{k_3}(D_{k_2}(E_{k_1}(P_i \oplus C_{i-1}))),$$

respectiv pentru descifrare:

$$P_i = C_{i-1} \oplus D_{k_1}(E_{k_2}(D_{k_3}(C_i))).$$

Ambele moduri necesită mai multe resurse decât în cazul unei singure cifrări: mai mult *hardware* sau mai mult timp.

8.5. Generarea tabelelor de substituţie

Tabele de substituţie, cunoscute sub acronimul de $S-box$, joacă un rol important în proiectarea unui cifru bloc. O tabelă de substituţie S este o funcţie bijectivă de la $GF(2^n)$ la $GF(2^n)$ (uzual $n = 8$) cu următoarele proprietăţi:

-grad mare de neliniaritate;

-rezistenţă la criptanaliza diferenţială;

-costrucţie şi computabilitate eficientă.

Următorul algoritm generează tabele de substituţie din $GF(2^n)$ cu proprietăţile menţionate mai sus. Algoritmul îşi are originea în *Nyberg* [52].

PAS 0. Se alege un polinom ireductibil $m(x)$ generatorul lui $GF(2^n)$, $\mathbf{A}$ matrice inversabilă din $M_{n\times n}(\mathbf{Z}_2)$, $\mathbf{b}$ vector din $\mathbf{Z}_2^n$ cu ponderea Hamming $\frac{n}{2}$.

PAS 1. Se alege $F(x)$ din $GF(2^n)$ de forma x^{-1} sau x^{2^k+1}, $k \in \mathbf{N}$.

PAS 2. Tabela S este definită prin formula:

$$S(x) = \mathbf{A}\cdot F(x) + \mathbf{b}.$$

Observaţia 8.5.1. Algoritmul AES utilizează funcţia $F(x) = x^{-1}$.

8.6. Criptanaliza diferenţială

Criptanaliza diferenţială a fost introdusă iniţial de *E. Biham* şi *A. Shamir* la începutul anilor 90. Cifrul pe care s-a aplicat, de către aceaştia, a fost standardul american de cifrare a datelor *DES* (Data Encryption Standard). În esenţă tehnica criptanalizei diferenţiale constă într-o formă particulară a tehnicii *atacului cu text clar cunoscut*: se analizează perechi de text clar cunoscut (C_i, C_j), care sunt într-o anumită relaţie (de exemplu ponderea *Hamming* a sumei mod 2 este o constantă dată sau un vector dat: $C_i \oplus C_j = D_{ij}$) şi efectul asupra textelor cifrate cu aceeiaşi cheie (de exemplu vectorul sumei mod 2 a textelor cifrate: $M_i \oplus M_j$) . Scopul urmărit este acela de a recupera (cel puţin probabilist) cheia de cifare K.

8.7. Criptanaliza liniară

Criptanaliza liniară este o tehnică prin care se urmăreşte construcţia unui sistem de ecuaţii liniare între biţii textului clar, ai textului cifrat şi ai cheii. Rezolvarea acestui sistem de ecuaţii duce la aflarea cheii de cifrare. Sistemul de ecuaţii ce se construieşte poate fi chiar un sistem probabilist, în sensul că o ecuaţie este verificată cu o anumită probabilitate.

8.8. Alte metode

În studiul cifrurilor bloc se mai pot aplica următoarele metode şi tehnici criptografice (vezi *Schneier* [70]):

-Metoda caracteristicilor diferenţiale condiţionate,

-Criptanaliza cu ajutorul cheilor deplasate,

-Criptanaliza diferenţial-liniară,

-Metoda combinată diferenţial-liniară,

-Metoda criptanalizei diferenţiale de ordin superior,

-Metoda criptanalizei diferenţiale de ordin superior a cifrului KN,

-Metoda aproximării multi-lineare,

-Metoda diferenţialelor trunchiate,

-Metoda generalizată a criptanalizei lineare,

-Metoda de atac prin interpolare,

-Metoda de atac prin funcţii non-surjective,

-Tehnica transformatei Walsh-Hadamard.

8.9. Implementări şi rezultate experimentale

8.9.1. Implementarea standardului de cifrare A.E.S.

Cifrul bloc AES (Advanced Encryption Standard) este noul standard de cifrare a datelor care înlocuieşte standardul DES. Standardul AES este un caz particular al algoritmului de cifrare RIJNDAEL (proiectat de Joan Daemen şi Vincent Rijmen) în sensul că are setate lungimea cheii de cifrare la 128 biţi iar dimensiunea blocului de date care se cifrează la 128, 192 sau 256 biţi. În figura 8.5 se prezintă ultimii 10 finalişti (algoritmii) care au participat la selecţia standardului. Implementarea algoritmului RIJNDAEL s-a realizat în limbajul de programare C.

Cifru	Sursă documentare
CAST-256	Entrust Technologies, Inc. (represented by Carlisle Adams)
CRYPTON	Future Systems, Inc. (represented by Chae Hoon Lim)
DEAL	Richard Outerbridge, Lars Knudsen
DFC	CNRS - Centre National pour la Recherche Scientifique - Ecole Normale Supérieure (represented by Serge Vaudenay)
E2	NTT - Nippon Telegraph and Telephone Corporation
FROG	TecApro Internacional S.A.
HPC	Rich Schroeppel
LOKI97	Lawrie Brown, Josef Pieprzyk, Jennifer Seberry
MAGENTA	Deutsche Telekom AG (represented by Dr. Klaus Huber)
MARS	IBM (represented by Nevenko Zunic)
RC6	RSA Laboratories (represnted by Matthew Robshaw)
RIJNDAEL	Joan Daemen, Vincent Rijmen
SAFER+	Cylink Corporation (represented by Dr. Lily Chen)
SERPENT	Ross Anderson, Eli Biham, Lars Knudsen
TWOFISH	Bruce Schneier, John Kelsey, Doug Whiting, David

Figura 8.5: Principalii algoritmi implicaţi în procesul de selectare a standardului A.E.S.

8.9.2. Testarea algoritmului AES

Pentru fixarea ideilor vom nota, în cele ce urmează, prin n dimensiunea blocului de date al cifrului bloc iar prin k lungimea cheii acestuia. Algoritmii candidaţi pentru standardul de cifrare bloc *AES* au fost supuşi testelor statistice (vezi *NIST 800-22* [107]), construcţia eşantioanelor (vezi *Soto* [87]) fiind realizată după cum urmează:

-*avalanşa cheii*: se fizează în mod aleatoriu un set de chei, se setează la 0−peste tot textul (text de referinţă) şi se urmăreşte efectul cifrării textului de referinţă cu ajutorul setului de chei perturbate într-un singur bit;

-*avalanşa textului de intrare*: se fizează în mod aleatoriu o mulţime de texte, se setează la 0−peste tot cheia (cheia de referinţă) şi se urmăreşte efectul produs prin cifrarea, cu cheia de referinţă, a mulţimii textelor perturbate într-un singur bit;

-*corelaţie intrare/ieşire*: se urmăreşte detectarea corelaţiilor dintre intrare şi ieşire prin utilizarea a r chei de cifrare pseudoaleatoare (de referinţă) şi a s texte de intrare pseudoaleatoare (de referinţă), cifrându-se, în modul ECB, toate cele s texte cu toate cele r chei.

-*corelaţie în modul de lucru CBC*: se fixează, în mod pseudoalatoriu un set de chei de cifrare (de referinţă), se setează la 0−peste tot vectorul de iniţializare precum şi textul clar (suficient de lung) şi se urmăreşte detecţia corelaţiei din concatenarea tuturor textelor cifrate rezultate.

-*text clar cu densitate redusă*: se testează aleatorismul statistic al cifrării tuturor textelor clare de pondere Hamming 0, 1 şi 2 cu ajutorul unui set de chei generate pseudoaleator.

-*chei cu densitate redusă*: se testează aleatorismul statistic al cifrării textelor clare generate pseudoaleator cu toate cheile de pondere Hamming 0, 1 şi 2.

-*text clar cu densitate mare*: se testează aleatorismul statistic al cifrării tuturor textelor clare de pondere Hamming n, $n-1$ şi $n-2$ cu ajutorul unui set de chei generate pseudoaleator.

-*chei cu densitate mare*: se testează aleatorismul statistic al cifrării textelor clare generate pseudoaleator cu toate cheile de pondere Hamming n, $n-1$ şi $n-2$.

8.9.3. Rezultate experimentale

Vom prezenta în cele ce urmează o serie de rezultate experimentale obţinute în cazul aplicării testelor de avalanşă strictă, imunitate la corelaţie, balans, nedegenerare precum şi a unor variante de criptanaliză diferenţială asupra algoritmului de cifrare bloc RIJNDAEL (s-au folosit pentru testare numai cheile de pondere Hamming 0 şi 1 iar ca text de referinţă textul nul).

1. Rezultatele testului de avalanşă strictă asupra algoritmului RIJNDAEL

Cheia de referinţă: cheia 0-peste tot
Textul de referinţă: 0-peste tot
Acţiune efectuată: perturbarea cu un bit a cheii de referinţă.
Dimensiunea cheii: 128 biţi
Volumul eşantionului: 128 biţi
Riscul de efectuare al testării statistice: $0,05$
Deplasarea iniţială (cuvinte ignorate): 0
Parametrul de codificare al datelor 8 biţi
Sunt afişate procentele de schimbări pe poziţiile slabe ale cheii de bază:
$60,156\%$ poziţia 31,
$60,938\%$ poziţia 37,
$60,156\%$ poziţia 51,
$40,602\%$ poziţia 125.
Numărul de poziţii slabe ale cheii 4
Decizie: algoritmul implementat indeplineşte criteriul de avalanşă strictă.
Statistica testului (procentul de respingeri) este: $-0,973329$
Indicaţii asupra atacului optimal cu chei de pondere 1: probabilităţile extreme sunt $40,625$ poziţia 125 respectiv $60,938$ poziţia 37.

2. Rezultatele testului de imunitate la corelaţie asupra algoritmului RIJNDAEL

Cheile de referinţă sunt cheile de pondere 0 şi 1
Textul de referinţă: 0-peste tot
Acţiune efectuată: perturbarea cu un bit a cheii de referinţă.
Dimensiunea cheii: 128 biţi
Deplasarea iniţială (cuvinte ignorate) 0
Parametrul de codificare al datelor 8 biţi
Volumul eşantionului: 128 biţi
Riscul de efectuare al testarii statistice 0,05
Sunt afişate procentele de schimbări de la cheile slabe:
Dezizie: cheia de pondere 0 este imună la corelaţie.
Rezultatele pentru cheile de pondere 1 sunt următoarele:
$60,156\%$ poziţia 43
$59,375\%$ poziţia 74
Numărul de corelaţii ale cheilor de pondere 1 este 2
Decizie: Algoritmul implementat indeplineşte criteriul de imunitate la corelaţie pentru cheile de pondere Hamming 1.
Statistica testului (procentul de respingeri) este: $-1,784436$
Indicaţii asupra atacului optimal cu chei de pondere 1: Probabilităţile extreme sunt $41,406\%$ poziţia 82 respectiv $60,156\%$ poziţia 43.

3. Rezultatele testului de balans asupra algoritmului RIJNDAEL
Cheile de referinţă sunt cheile de pondere 1
Textul de referinţă: 0-peste tot
Dimensiunea spaţiului de intrare este de 128 biţi
Dimensiunea spaţiului de ieşire este de 8 biţi
Depalasarea iniţială (ture in gol) 0
Parametrul de codificare al datelor: 8 biţi
Riscul de efectuare al testarii statistice $0,05$
Probabilitatea de balans este $0,5$
Sunt afisate numărul de intrări care realizează fiecare byte:
Se testează funcţia 0 : $N[0] = 62,\ N[1] = 66$
Statistica testului este $0,5156$
Se testează funcţia 1 : $N[0] = 60,\ N[1] = 68$
Statistica testului este $0,5312$
Se testează funcţia 2 : $N[0] = 64,\ N[1] = 64$
Statistica testului este $0,50$
Se testează funcţia 3 : $N[0] = 61,\ N[1] = 67$
Statistica testului este $0,5234$
Se testează funcţia 4 : $N[0] = 64,\ N[1] = 64$
Statistica testului este $0,5$
Se testează funcţia 5 : $N[0] = 59,\ N[1] = 69$
Statistica testului este $0,539$
Se testează funcţia 6 : $N[0] = 62,\ N[1] = 66$
Statistica testului este $0,5156$
Se testează funcţia 7 : $N[0] = 54,\ N[1] = 74$
Statistica testului este $0,5781$
Decizie: algoritmul implementat indeplineşte criteriul de balans.

4. Rezultatele testului de nedegenerare asupra algoritmului RIJNDAEL
Cheia de referinţă: cheia 0−peste tot
Textul de referinţă: 0-peste tot
Dimensiunea cheii: 128 biţi
Volumul eşantionului: 128 biţi
Deplasare iniţială (cuvinte ignorate): 0
Parametrul de codificare al datelor 8 biţi
Sunt afişate punctele de degenerare ale algoritmului: nu este cazul
Numărul de puncte degenerate ale algoritmului 0
Decizie: algoritmul implementat îndeplineşte criteriul de nedegenerare.

5. Rezultatele testului de criptanaliză diferenţială asupra algoritmului RIJNDAEL

Dimensiunea cheii 128 biţi

Dimensiunea bloc de date iniţial: 128 biţi

Bloc de date iniţial 16 de 0− peste tot

Riscul de efectuare al testării statistice: $0,05$

Probabilitatea de corelare: $0,5$;

Iteraţii efectuate: 1000;

Sunt afişate cele mai semnificative cheii de bază (indice de coincidenţă extram global) pentru cheile de pondere Hamming 0 şi 1:

Cheia nulă are patternul:

Indice de coincidenţă maxim între două cifrări succesive este 0.640625 obţinut la iteraţia 914;

Indice de coincidenţă minim între două cifrări succesive este 0.343750 obţinut la iteraţia 311;

Indice de coincidenţă maxim cu textul iniţial este 0.640625 obţinut la iteraţia 890;

Indice de coincidenţă minim cu textul iniţial este 0.335938 obţinut la iteraţia 65;

Cheia cea mai slabă (corelaţie directă) este 17 cu indicele maxim de coincidenţă între două cifrări succesive 0.695313 obţinut la iteraţia 667;

Cheia cea mai slabă (corelaţie indirectă) este 13 cu indice minim de coincidenţă între două cifrări succesive 0.328125 obţinut la iteraţia 419;

Cheia cea mai slabă (autocorelaţie directă) este 88 cu indice maxim de coincidenţă cu textul iniţial 0.679688 obţinut la iteraţia 281;

Cheia cea mai slabă (autocorelaţie indirectă) este 43 cu indice minim de coincidenţă cu textul iniţial 0.320313 obţinut la iteraţia 168.

8.9.4. Interpretarea rezultatelor

După cum se observă algoritmul de cifrare RIJNDAEL îndeplineşte cerinţele formulate în cadrul capitolului 3. Acest lucru nu însemnă faptul că algoritmul este absolut sigur: *în criptografie nici o regulă nu este absolută* (Étienne Bazeries, 1901). O descriere completă a interpretării rezultatelor testelor statistico-informaţionale poate fi găsită în *Simion* [84].

8.10. Concluzii

Cifrurile bloc pot fi programate să funcţioneze ca un cifru flux şi reciproc (a se vedea modul de lucru OFB). Cea mai bună definiţie, care pune în evidenţă diferenţele dintre aceste două structuri de cifrare, este următoarea:

Definiţia 8.10.1. Cifrurile bloc operează asupra datelor cu transformări (fixe) asupra blocurilor de date clare; cifrurile flux operează asupra datelor cu transformări (ce variază în timp) asupra cuvintelor (biţi, octeţi, etc.) din textul clar.

În cazul aplicaţiilor practice, cifrurile bloc par mai generale (pot funcţiona în unul dintre cele patru moduri principale) iar cifrurile flux sunt mai simplu de analizat din punct de vedere matematic. O altă diferenţă este aceea că cifrurile flux cifrează sau descifrează un singur cuvânt de date la un tact, deci nu sunt optime pentru implementările software. Cifruile bloc sunt mai simplu de implementat din punct de vedere soft deoarece acestea operează cu blocuri de cuvinte de procesor deci sunt mai rapide. Pe de altă parte cifrurile flux sunt mai uşor de implementat în aplicaţiile software.

8.11. Aplicaţii

Deoarece nu există o formulă matematică universală care să poată fi aplicată în operaţia de criptanaliză, am propus ca exerciţii la acest capitol modificări ale unor algoritmi de cifruri bloc consacrate. Sunt date o serie de indicaţii precedate de o scurtă descriere a algoritmilor propiu-zişi.

Exerciţiul 8.11.1. Studiaţi următorele simplificări ale algoritmului RC5:
-RC5 cu 8 iteraţii dar fără rotaţii;
-RC5 cu 8 iteraţii iar numărul de rotaţii egal cu numărul de iteraţii.

Răspuns. În cele ce urmează facem o scurtă descriere a cifrului RC5 cu r iteraţii. Acesta are lungimea blocului de date variabilă dar vom considera în cele ce urmează că aceasta a fost setată la 64 biţi. Operaţia de cifrare foloseşte $2r+2$ chei dependente de cuvintele pe 32 biţi $S_0, S_1, S_2, \ldots, S_{2r+2}$ unde r este numărul de iteraţii. Pentru cifrare blocul de date se împarte în două părţi de 32 biţi notate cu L respectiv R (RC5 face apel la codificarea *little-endian* pentru împachetarea octeţilor în cuvinte: primul octet se transformă în cele mai puţin semnificative poziţii ale lui L, etc.). Apoi avem:

$$\begin{cases} L = L + S_0, \\ R = R + S_1. \end{cases}$$

Pentru $i = 1, \ldots, r$ se execută:

$$\begin{cases} L = ((L \oplus R) << R) + S_{2i}, \\ R = ((R \oplus L) << L) + S_{2i+1}. \end{cases}$$

Ieşirea constă în registrele L şi R. Simbolul $\oplus$ are semnificaţia sumei mod 2, simbolul $<<$ semnifică rotire circulară şi în fine simbolul $+$ are semnificaţia sumei mod 2^{32}. Operaţia de descifrare este similară (intervin operatorii $\oplus, >>$ şi $-$). Modul de construcţie al secvenţei S (care derivă din cheie) nu este esenţial în cadrul acestui exerciţiu.

Dacă setăm numărul de iteraţii $r = 8$ şi nu facem nici un fel de rotaţii atunci pentru $i = 1, \ldots, 8$ se execută:

$$\begin{cases} L = (L \oplus R) + S_{2i}, \\ R = (R \oplus L) + S_{2i+1}. \end{cases}$$

Algoritmul astfel setat nu îndeplineşte criteriul de avalanşă strictă (schimbarea unui bit în blocul de text clar produce, în medie, schimbări de 50% la ieşire). Schema de mai sus permite atacul cu ajutorul tehnicii criptanalizei liniare pentru aflarea lui S, deci a cheii efective.

Dacă setăm numărul de iteraţii $r = 8$ şi numărul de rotaţii egal cu r atunci pentru $i = 1, \ldots, 8$ se execută:

$$\begin{cases} L = ((L \oplus R) << 8) + S_{2i}, \\ R = ((R \oplus L) << 8) + S_{2i+1}. \end{cases}$$

Algoritmul astfel setat nu îndeplineşte criteriul de avalanşă strictă. Schema de mai sus permite atacul cu ajutorul tehnicii criptanalizei diferenţial/liniare pentru aflarea lui S.

Exerciţiul 8.11.2. Studiaţi următorele simplificări ale algoritmului DES:
-DES cu 12 iteraţii dar fără aplicaţiile S;
-DES cu 4 iteraţii;
-DES cu 6 iteraţii.

Răspuns. Cifrul bloc DES (proiectat în 1977) este sub controlul unei chei efective de 56 biţi (cheia de bază este de 64 biţi, 8 biţi fiind pentru detecţia erorilor) iar mărimea blocului de date este de 64 biţi. Textul clar este permutat iar apoi este împărţit în două blocuri L şi R de lungime 32 biţi. Se execută apoi iterativ operaţiile (pentru $i = 1, \ldots, \textit{numărul de iteratii}$):

$$\begin{cases} L_i = R_i, \\ R_i = L_i \oplus f(R_{i-1}, K_i). \end{cases}$$

În final textul este supus permutării inverse. Ne concentrăm asupra descrierii funcţiei $f : \mathbf{Z}_2^{32} \times \mathbf{Z}_2^{48} \to \mathbf{Z}_2^{32}$. Iniţial blocul R (32 biţi) este extins cu ajutorul

funcţiei E la un bloc pe 48 biţi care este sumat mod 2 cu cheia K (extinsă la 48 biţi cu ajutorul algoritmului de producere a subcheilor). Opt aplicaţii $S : \mathbf{Z}_2^6 \to \mathbf{Z}_2^4$ produc o ieşire pe 32 biţi care este permutată pentru a produce ieşirea finală dintr-o iteraţie. Dacă aplicaţiile S sunt fixe (se selectează 4 biţi din 6 în mod fix) atunci se poate aplica tehnica criptanalizei diferenţiale (biţii de la ieşire sunt biţii de la intrare (sumaţi mod 2 cu cheia K) dar într-o altă ordine).

Algoritmul DES cu 4 cât şi cu 6 iteraţii poate fi spart cu ajutorul tehnicii atacului cu text clar cunoscut.

Exerciţiul 8.11.3. Studiaţi regula B a algoritmului *Skipjack* cu 8 iteraţii.

Exerciţiul 8.11.4. Ce defect are un algoritm de cifrare care este închis (un algoritm de cifrare se numeşte *închis* dacă pentru orice chei k_1 şi k_2 există o cheie k_3 astfel încât pentru orice text clar M avem $E_{k_1}E_{k_2}(M) = E_{k_3}(M)$)?

Răspuns. Ca metodă de atac generică se poate opta pentru cifrarea repetitivă.

Exerciţiul 8.11.5. Aplicaţi tehnica criptanalizei diferenţiale şi criptanalizei liniare asupra algorimului FEAL.

Exerciţiul 8.11.6. Studiaţi tehnica criptanalizei diferenţiale în cazul algoritmului DES cu 16 iteraţii.

Exerciţiul 8.11.7. Aplicaţi tehnica criptanalizei liniare în cazul algoritmului DES cu 16 iteraţii.

Exerciţiul 8.11.8. Având la dispoziţie un cifru bloc $E_k(.)$ proiectaţi un cifru flux şi viceversa.

Exerciţiul 8.11.9. Scrieţi funcţia analitică a celor opt funcţii de substituţie S ale cifrului DES.

Exerciţiul 8.11.10. Fie $E_M(.)$ şi $D_K(.)$ funcţiile de cifrare respectiv descifrare ale unui cifru. Care este valoarea lui $D_K(E_K(M))$?

Notă. Descrierea algoritmilor RC5, DES, Skipjack şi FEAL poate fi găsită în *Schneier* [69] sau *Menenzes* [50].

Exerciţiul 8.11.11. Implementaţi modalităţi de testare a cifrurilor bloc.

Exerciţiul 8.11.12. Implementaţi modalităţi de generare a tabelelor de substituţie.

Exerciţiul 8.11.13. Fie $E(\cdot, \cdot)$ o funcţie de cifrare pe m biţi de cheie şi n biţi de date. Care este valoarea maximă a lui m astfel încât cheia efectivă a cifrului să fie m?

Capitolul 9

CRIPTANALIZA CIFRURILOR CU CHEI PUBLICE

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

9.1. Principii de bază

9.1.1. Introducere

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

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

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

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

9.1.2. Securitatea algoritmilor cu cheie publică

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

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

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

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

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

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

9.1.3. Comparaţii ale algoritmilor asimetrici şi a algoritmilor simetrici

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

9.2. Algoritmi de tip rucsac

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

9.2.1. Algoritmi rucsac supercrescător

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

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

Problema rucsacului supercrescător este uşor de rezolvat folosind următoarea schemă:

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

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

9.2.2. Crearea cheii publice din cheia privată

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

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

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

9.2.3. Cifrarea

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

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

9.2.4. Descifrarea

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

9.2.5. Implementarea efectivă

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

9.3. Algoritmul RSA

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

9.3.1. Descrirea principiilor de cifrare şi descifrare

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

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

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

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

Cu alte cuvinte

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

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

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

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

Pentru a descifra un mesaj se calculează:

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

deoarece

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

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

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

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

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

9.3.2. Viteza algoritmilor tip RSA

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

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

Utilizarea CRT

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

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

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

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

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

respectiv

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

Utilizând CRT vom avea:

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

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

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

În **faza de calcul** se execută:

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

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

9.3.3. Securitatea RSA-ului

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

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

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

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

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

9.3.4. Tipuri de atacuri asupra algoritmilor RSA

Atac cu text cifrat ales

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

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

$$m = c^d.$$

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

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

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

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

Interceptorul E va calcula:

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

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

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

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

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

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

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

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

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

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

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

Atac cu ajutorul modulelor comune

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

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

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

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

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

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

$$re_1 + se_2 = 1.$$

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

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

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

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

Atac asupra exponenţilor de cifrare de dimensiuni mici

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

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

Atac asupra exponenţilor de descifrare de dimensiuni mici

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

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

Atac în cazul numerelor prime apropiate

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

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

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

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

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

Atac de tip eroare hardware

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

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

unde

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

şi

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

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

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

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

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

deci modulul n poate fi factorizat fără probleme.

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

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

Atacul asupra cifrării şi semnării cu algoritmul RSA

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

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

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

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

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

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

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

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

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

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

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

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

-exponentul de descifrare trebuie să fie mare.

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

9.3.5. Trape în generarea cheilor RSA

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

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

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

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

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

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

Ascunderea exponentului privat mic d

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

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

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

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

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

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

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

9.4. Algoritmul Pohlig-Hellman

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

ca în algoritmul RSA:

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

unde

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

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

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

9.5. Algoritmul Rabin

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

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

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

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

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

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

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

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

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

9.6. Algoritmul ElGamal

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

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

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

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

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

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

Pentru a descifra (a, b) calculăm:

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

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

9.7. Curbe eliptice

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

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

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

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

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

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

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

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

cu

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

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

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

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

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

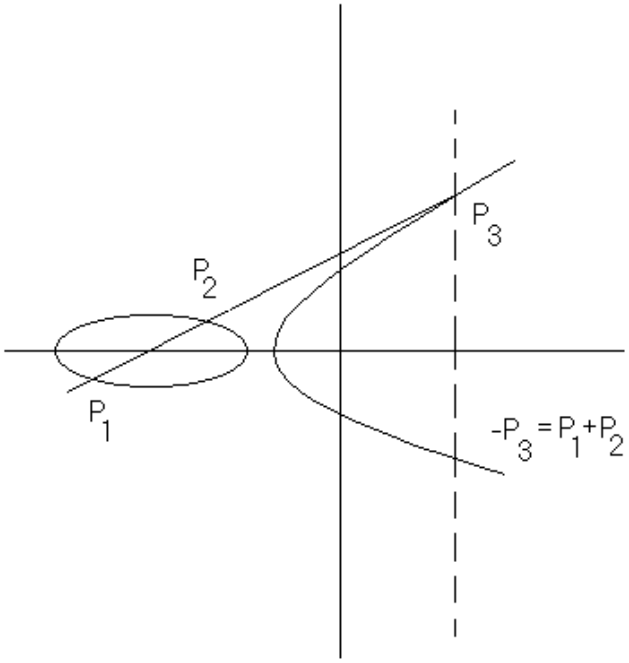

Figura 9.1: Operaţia de adunare pe o curbă eliptică.

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

9.8. Aplicaţii practice

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

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

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

-smart-carduri folosite în sistemul de plăţi electronice;

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

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

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

9.9. Teste de primalitate şi metode de factorizare

9.9.1. Teste de primalitate

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

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

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

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

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

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

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

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

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

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

Testul Fermat

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

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

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

Testul Solovay-Strassen

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

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

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

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

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

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

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

Testul Miller-Rabin

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

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

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

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

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

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

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

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

9.9.2. Metode de factorizare

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

-metoda Dixon;

-metoda Pomerace-Silverman-Montgomery.

9.9.3. Metode de generare a numerelor prime

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

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

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

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

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

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

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

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

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

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

Ieşire. Un număr prim p tare.

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

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

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

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

PASUL 5. Se returnează numărul prim tare p.

9.10. Infrastructura Cheilor Publice (PKI)

9.10.1. Elementele PKI

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

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

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

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

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

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

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

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

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

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

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

9.10.2. Ghid de folosire a tehnologiei PKI în reţele deschise

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

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

-optimizarea resursei timp;

-optimizarea costurilor;

-disponibilitate marită a tipului de servicii furnizate;

-integritatea datelor;

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

-nivelul de încredere solicitat;

-integritatea cheilor publice şi a cheilor private;

-cuantificarea consecinţelor potenţialelor riscuri;

-politici, practici şi proceduri;

-conectarea la o infrastructură existentă;

-interoperabilitatea cu alte infrastructuri;

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

9.10.3. Riscuri ale utilizării tehnologiei PKI

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

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

9.10.4. Standarde ale PKI

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

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

9.11. Aplicaţii

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

Răspuns. Criptograma este:

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

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

Exponentul de descifrare va fi:

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

Descifrarea mesajului E va fi:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Operaţia de descifrare pentru a obţine M este:

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

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

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

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

Capitolul 10

CRIPTANALIZA SEMNĂTURILOR DIGITALE

Cryptography+loose of discipline=haos.
Equation of russian cryptologist

10.1. Prezentare generală

După ce dispunem de un algoritm de cifrare şi un sistem de generare a cheilor trebuie să organizăm transferul cheilor generate către beneficiarii sistemului criptografic pentru ca aceştia să poată comunica în condiţii de siguranţă. Această distribuţie de chei se poate face prin curier (operaţie costisitoare) sau folosind chiar canalul de comunicaţie (dar în acest caz trebuie să demonstrăm matematic rezistenţa unui astfel de protocol).

În timp ce criptografia clasică se ocupa doar de asigurarea secretului comunicaţiilor, criptologia modernă se ocupă şi de problema autenticităţii. Autentificarea este un *operator dual* operatorului de criptare: în cadrul operaţiei de autentificare există un anume grad de redundanţă a mesajului, pe când în cadrul operaţiei de criptare această redundanţă este minimă.

Autenticitatea se *referă* la două aspecte:

-autenticitatea mesajelor, când problema este de a asigura destinatarul că mesajul recepţionat este cel trimis de expeditor;

-autenticitatea utilizatorului, când problema este de a verifica faptul că utilizatorul este persoana ce pretinde a fi.

Asigurarea autenticităţii a fost impusă, în special, de:

-existenţa interceptării active, când atacatorul poate interveni asupra conţinutului mesajelor interceptate;

-apariţia poştei electronice, în care puţine mesaje necesită a fi secretizate dar pentru foarte multe mesaje este necesară autentificarea.

În afara faptului că destinatarul trebuie să fie sigur că mesajul recepţionat este cel trimis de expeditor, un sistem de autentificare trebuie să asigure şi soluţionarea eventualelor *dispute* între cei doi corespondenţi.

În cazul utilizării poştei clasice, autenticitatea şi eventualele dispute sunt soluţionate prin semnarea documentului de către expeditor. Această semnătură ar trebui să îndeplinească următoarele funcţii (ideale):

1. semnătura este *nefalsificabilă*: semnătura este o *dovadă* că semnatarul a semnat în mod deliberat documentul;

2. semnătura este *autentică*: semnătura *convinge destinatarul* că semnatarul a semnat în mod deliberat documentul;

3. semnătura este *nereutilizabilă*: semnătura face parte din document şi *nu poate fi mutată pe alt document*;

4. documentul semnat este *inalterabil*: după semnare *documentul nu poate fi modificat*;

5. semnătura este *nerepudiabilă*: semnatarul nu poate pretinde mai târziu că nu a semnat documentul respectiv.

În cazul utilizării poştei electronice (precum şi a altor instrumente asemănătoare) s-au inventat sisteme criptografice care să satisfacă aceste cerinţe; au apărut astfel algoritmii pentru *SEMNĂTURI DIGITALE*.

Menţionăm că, în multe din lucrările de specialitate, prin autentificare se înţelege doar autentificarea utilizatorului (care se poate face şi cu ajutorul semnăturilor digitale), în timp ce semnăturile digitale sunt destinate asigurării autenticităţii mesajelor şi rezolvării eventualelor dispute dintre corespondenţi.

10.2. Noţiuni preliminare

Funcţii unidirecţionale (One-Way Functions). Începem printr-o serie de definiţii.

Definiţia 10.2.1. O funcţie f se numeşte funcţie unidirecţională dacă:
a) fiind dat x, este uşor de calculat $f(x)$;
b) fiind dat $f(x)$, este greu de calculat x.

Definiţia 10.2.2. O funcţie f se numeşte funcţie unidirecţională cu trapă (trapdoor) dacă:
a) fiind dat x, este uşor de calculat $f(x)$;
b) fiind dat $f(x)$, este greu de calculat x;
c) pe baza unei informaţii secrete y, este uşor de calculat x din $f(x)$.

Deci funcţiile unidirecţionale cu trapă sunt funcţii unidirecţionale speciale care permit calculul uşor al lui x din $f(x)$ când este cunoscută o anumită informaţie, fără această informaţie calculul lui x din $f(x)$ fiind imposibil din punct de vedere computaţional.

Notă. Din punct de vedere strict matematic, nu există nici o demonstraţie de existenţă a funcţiilor unidirecţionale şi nici un argument real că ele pot fi construite.

Cu toate acestea, există multe funcţii care par a fi unidirecţionale: $f(x)$ se poate calcula uşor şi, cel puţin până în prezent, nu s-a găsit o cale uşoară de a obţine x din $f(x)$.

Atât funcţiile unidirecţionale cât şi funcţiile unidirecţionale cu trapă sunt folosite intens în criptografia cu chei publice.

Din multitudinea de funcţii care par unidirecţionale, marea majoritate a algoritmilor cu chei publice apelează doar la trei:

• *logaritmul discret*: dându-se numerele p, g şi y, p fiind un număr prim, să se găsească x astfel încât $g^x = y \bmod p$. Un exemplu de algoritm ce apelează la această tehnică este *algoritmul Diffie-Hellman* de schimb al cheilor între două entităţi A şi B care cad de comun acord pe un canal public asupra unui *număr prim* p (de cel puţin 512 biţi şi $(p-1)/2$ prim) şi asupra unui *element primitiv* g (în algebra mod$p$), adică $ord(g) = p$. Protocolul *Diffie-Hellman* este descris în continuare:

PAS 1. A alege un element x, calculează $X = g^x \bmod p$, pe care-l trimite către B.

PAS 2. B alege un element y, calculează $Y = g^y \bmod p$, pe care-l trimite către A.

PAS 3. A şi B calculează cheia prin formulele X^y respectiv Y^x.

Sintaxa MAPLE pentru rezolvarea problemei logaritmului discret este:

$$> msolve(ecuatia, variabila, modul);$$

• *factorizarea*: dacă N este produsul a două numere prime atunci:

a) să se factorizeze N;

b) fiind date e şi C, să se găsească M astfel încât $M^e = C \bmod N$;

c) fiind date M şi C să se găsească d astfel încât $C^d = M \bmod N$;

d) fiind dat x, să se decidă dacă există y astfel încât $x = y^2 \bmod N$.

Procedura de cifrare cu ajutorul acestei tehnici este descrisă pentru următorul scenariu general de comunicare confidenţială între A şi B care dispun de un număr natural N (compozit $N = pq$, factorii primi p şi q fiind factori privaţi şi suficient de mari). Pentru a primi mesaje private B execută următoarele:

PAS 0 (*alegere exponent de criptare şi de decriptare*). *B alege exponentul de criptare s_B* (numit şi *cheie publică*, pe care-l face public, adică îl comunică lui A) relativ prim cu $\varphi(N)$.

B calculează exponentul de decriptare t_B (numit şi cheie secretă) ca fiind inversul multiplicativ al lui s_B modulo $\varphi(N)$.

Deoarece B dispune de factorizarea $N = pq$ acesta poate *calcula eficient* $\varphi(N)$ prin formula $\varphi(N) = (p-1)(q-1)$, deci cu ajutorul teoremei lui Euler ($s_B, \varphi(N)$ fiind relativ prime) avem $s_B^{\varphi(\varphi(N))} = 1 \bmod \varphi(N)$, de unde $t_B = s_B^{\varphi(\varphi(N))-1} \bmod \varphi(N)$. Calculul efectiv al lui t_B se face cu ajutorul algoritmului lui *Euclid extins* (se calculează t_B din congruenţa $s_B t_B = 1 \bmod \varphi(N)$, cunoscându-se s_B şi $\varphi(N)$: există un unic k, număr întreg astfel încât $s_B t_B = 1 + k\varphi(N)$).

PAS 1. A trimite mesajul criptat pentru B: $M^{s_B} \bmod N$.

PAS 2. B descifrează mesajul recepţionat prin ridicarea numărului $M^{s_B} \bmod N$ la puterea t_B obţinând pe M (se aplică teorema lui *Euler*: $M^{s_B t_B} = M^{\varphi(N)k+1} = M \bmod N$).

Exponentul de decriptare t_B nu poate fi dedus (eficient) din s_B şi N ci numai din s_B şi *factorii* lui N, deci atâta timp cât se păstrează secretul asupra factorizării lui N schema anterioară este sigură computaţional (la ora actuală se consideră suficient de sigur (nu poate fi factorizat în câteva minute) un număr de 200 cifre care este produsul a două numere de 100 de cifre). Evident $(M, \varphi(N)) = 1$, deci anumite mesaje trebuie evitate. Pentru calculul eficient al lui $\varphi(\varphi(N)) = \varphi((p-1)(q-1))$ trebuie ca $p-1$ şi $q-1$ să fie construite din numere prime. Pentru a comunica B cu A se procedează similar.

Programul MAPLE are proceduri de testare a primalităţii unui număr mare (algoritm probabilist) precum şi de factorizare a acestuia. Sintaxele acestor proceduri sunt:

$$> isprime(numar);$$

respectiv

$$> ifactor(numar);$$

• *problema rucsacului*: fiind dată o mulţime de numere întregi să se găsească o submulţime cu suma S.

Desigur, nu orice algoritm cu chei publice bazat pe una din aceste trei probleme, greu rezolvabile matematic, este în mod automat rezistent în faţa criptanalizei, el trebuind să îndeplinească şi alte condiţii; în schimb, în ciuda numeroaselor încercări aproape toţi algoritmii cu chei publice bazaţi pe alte funcţii care par unidirecţionale s-au dovedit a nu fi rezistenţi din punct de vedere criptografic.

10.3. Funcţii hash

10.3.1. Generalităţi

Problematica funcţiilor hash fiind deosebit de vastă, în cele ce urmează ne vom

opri numai asupra aspectelor strict necesare înţelegerii utilizării acestor funcţii în cadrul algoritmilor de semnătură digitală.

Definiţia 10.3.1. *Funcţia hash* este o funcţie care se aplică unui şir de lungime oarecare obţinându-se un şir de lungime fixată (de obicei, mai mică decât lungimea şirului de intrare).

Definiţia 10.3.2. O funcţie H se numeşte *funcţie hash unidirecţională* dacă:
a) H este funcţie hash;
b) H este funcţie unidirecţională.

Pentru a putea fi folosite pentru semnături digitale, funcţiile hash unidirecţionale trebuie să mai îndeplinească, printre altele una din următoarele două condiţii:
1) oricare ar fi M dat, este greu de găsit $M^{'}$ astfel încât $H(M^{'}) = H(M)$;
2) este greu de găsit o pereche oarecare M, $M^{'}$ astfel încât $H(M) = H(M^{'})$.

Funcţiile hash unidirecţionale care îndeplinesc condiţia (1) se numesc funcţii hash unidirecţionale slabe (sau universale), iar cele care îndeplinesc condiţia (2) se numesc funcţii hash unidirecţionale tari (sau fără coliziuni).

Prima condiţie este uşor de justificat: dacă A a semnat mesajul M cu $H(M)$, iar B obţine $M^{'}$ astfel încât $H(M^{'}) = H(M)$, atunci B ar putea pretinde că A ar fi semnat mesajul $M^{'}$.

A doua condiţie este justificată de existenţa atacului *birthday*, metodă generală de atac aplicabilă oricărei funcţii hash, atac inspirat de paradoxul matematic al zilei de naştere.

Datorită atacului birthday, pentru o funcţie hash care are la ieşire un şir cu o lungime de m biţi (2^m posibilităţi) se pot găsi coliziuni generând doar $2^{m/2}$ perechi de mesaje-valori hash.

În aceste condiţii, algoritmii hash care produc valori hash de 64 biţi se consideră nesiguri deoarece, cu tehnologia actuală, se pot genera $2^{64/2} = 2^{32}$ mesaje şi deci este posibilă găsirea de mesaje care să intre în coliziune. De aceea se recomandă ca valoarea hash să fie de lungime de cel puţin 128 biţi.

10.3.2. Algoritmi hash

Există foarte multe moduri de proiectare a funcţiilor hash unidirecţionale (vezi *Pieprzyk* [55]). Majoritatea funcţiilor hash unidirecţionale utilizează funcţii unidirecţionale care produc o valoare hash de lungime n pe baza a două intrări de lungime n. În general, intrarea este constituită dintr-un bloc de text şi valoarea hash a blocului de text precedent, adică:

$$h_i = f(M_i, h_{i-1}),$$

valoarea hash finală fiind considerată valoarea hash a întregului mesaj. În acest fel funcţia hash unidirecţională produce o valoare de lungime fixă (n), oricare ar

fi lungimea mesajului. Pentru a evita obţinerea unor valori hash provenind de la mesaje de lungimi diferite, este indicat ca mesajele să conţină şi informaţii privind lungimea lor. Se ştie că pentru un cifru bloc sigur este greu să se afle cheia dacă se cunoaşte textul clar şi criptograma. De aceea foarte mulţi algoritmi hash utilizează drept funcţie hash unidirecţională un cifru bloc. În aceste condiţii, din multitudinea de clasificări ale funcţiilor hash vom apela la clasificarea care împarte aceste funcţii în două categorii: funcţii hash bazate pe cifruri bloc şi funcţii hash nebazate pe cifruri bloc.

În cele ce urmează vom da câteva exemple de funcţii hash.

10.3.3. Funcţii hash bazate pe cifruri bloc

Convenim să facem următoarele notaţii: $E(K, M)$-algoritmul de cifrare bloc, utilizând cheia K, aplicat mesajului M.

IV-un bloc de iniţializare.

t-numărul de blocuri în care a fost împărţit mesajul, aceste blocuri putând juca fie rolul lui K, fie rolul lui M.

În exemplele următoare avem, pentru toate, $H_0 = IV$ şi $H(M) = H_t$; de aceea vom specifica doar relaţia de iterare, pentru $i = 1, 2, \ldots, t$.

1) $H_i = E(M_i, H_{i-1})$ -Rabin;
2) $H_i = E(K, M_i \oplus H_{i-1})$ -Cipher Block Chaining;
3) $H_i = E(K, M_i \oplus M_{i-1} \oplus H_{i-1})$ -Combined Plaintext-Ciphertext Chaining;
4) $H_i = E(M_i \oplus H_{i-1}, H_{i-1})$ -Key Chaining;
5) $H_i = E(M_i, H_{i-1}) \oplus H_{i-1}$ -Davies-Meyer;
6) $H_i = E(H_{i-1}, M_i) \oplus M_i$ -Matyas;
7) $H_i = E(M_i, H_{i-1}) \oplus M_i \oplus H_{i-1}$-N hash;
8) $H_i = E(H_{i-1}, M_i) \oplus M_i \oplus H_{i-1}$-Miyaguchi;
9) $H_i = E(M_i, M_i \oplus H_{i-1}) \oplus H_{i-1}$;
10) $H_i = E(H_{i-1}, M_i \oplus H_{i-1}) \oplus M_i$;
11) $H_i = E(M_i, M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$;
12) $H_i = E(H_{i-1}, M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$;
13) $H_i = E(M_i \oplus H_{i-1}, M_i) \oplus M_i$;
14) $H_i = E(M_i \oplus H_{i-1}, M_{i-1}) \oplus H_{i-1}$.

Menţionăm că există şi alţi algoritmi hash bazaţi pe cifruri bloc, mult mai complecşi în ceea ce priveşte descrierea lor. Se recomandă cititorului implementarea schemelor amintite.

10.3.4. Funcţii hash nebazate pe cifruri bloc

Vom prezenta câteva exemple importante.

1) De tip **RSA**:
$H_i = (H_{i-1} \oplus M_i)^e \bmod N$.
2) De tip pătratic:
H_i extrage m biţi din $(00111111||H_i||M_i)^2 \bmod N$.

Există şi scheme bazate pe utilizarea automatelor celulare, transformarea Fourier discretă etc., dar descrierea lor ar lua prea mult spaţiu.

Dintre funcţiile hash nebazate pe cifruri bloc reamintim funcţiile **MD2**, **MD4** şi **MD5** proiectate de *Ron Rivest*, **SHA** concepută de *NSA* (care este şi standard guvernamental SUA), **RIPEMD** proiectată de *den Boer* în cadrul proiectului European RACE de evaluare a primitivelor criptografice precum şi funcţia **MDC2** utilizată de *IBM.*

10.4. Modalităţi de realizare a semnăturilor digitale

10.4.1. Aplicarea criptosistemelor simetrice

Cifrarea mesajelor cu sistemele de cifrare simetrice asigură autenticitatea mesajelor transmise (se presupune că sistemul este indecriptabil iar cheia este cunoscută de cei doi corespondenţi), dar nu poate rezolva eventualele dispute dintre expeditor şi destinatar.

O cale de a utiliza criptosistemele simetrice pentru realizarea semnăturilor digitale este de a se apela la un arbitru notat în cele ce urmează cu T. Cei doi corespondenţi A şi B îşi aleg fiecare câte o cheie secretă de cifrare, K_A şi K_B pe care o transmit lui T. Cheia K_A este cunoscută doar de A şi T iar cheia K_B doar de B şi T. T este o persoană în care A şi B au încredere absolută. Corespondenţa dintre A şi B se va desfăşura conform următorului protocol:

1. A cifrează mesajul pentru B cu cheia K_A şi-l trimite lui T: $C = E_A(M)$.
2. T descifrează mesajul primit de la A cu cheia K_A: $D_A(C) = M$.
3. T concatenează mesajul descifrat (M) cu o propoziţie în care specifică faptul că a primit acest mesaj de la A şi, în continuare concatenează criptograma (C): $M||P||C$.
4. T cifrează întregul pachet cu cheia K_B şi-l trimite lui B:
$C' = E_B(M||P||C)$.
5. B descifrează pachetul primit cu cheia K_B: $D_B(C') = M||P||C$.

În acest fel B a intrat în posesia mesajului M, care este certificat de T că a fost expediat de A.

Deşi acest protocol poate funcţiona, el are următoarele dezavantaje:
-mesajul se lungeşte la mai mult decât dublu;

-mai ales într-o reţea, T va fi foarte aglomerat;
-toţi corespondenţii trebuie să aibă încredere absolută în T.

10.4.2. Aplicarea criptosistemelor asimetrice

Pentru a se transmite un *mesaj cifrat* de la A la B se procedează astfel:
1. A cifrează mesajul M cu cheia publică a lui B: $C = E_B(M)$.
2. A trimite criptograma C lui B.
3. B descifrează criptograma C cu cheia sa secretă:
$D_B(C) = D_B(E_B(M)) = M$.
Pentru a se transmite un *mesaj semnat* de la A la B se procedează astfel:
1. A aplică mesajului M cheia sa secretă: $C' = D_A(M)$.
2. A transmite documentul C' lui B.
3. B aplică documentului C' cheia publică a lui A:
$E_A(C') = E_A(D_A(M)) = M$.

Se observă că în protocolul anterior operaţiile de cifrare şi descifrare au fost aplicate *doar* pentru semnarea documentului, mesajul M fiind în realitate transmis *în clar:* toţi corespondenţii cunosc cheia publică a lui A şi deci pot obţine mesajul M.

Pentru a se asigura şi cifrarea mesajelor se procedează astfel:
1. A semnează mesajul cu cheia sa secretă $C = D_A(M)$.
2. A cifrează mesajul semnat cu ajutorul cheii publice a lui B: $C' = E_B(C)$.
3. A transmite documentul C' lui B.
4. B descifrează documentul C' folosind cheia sa secretă:
$D_B(C') = D_B(E_B(C)) = C$.
5. B verifică autenticitatea şi obţine mesajul cu ajutorul cheii publice a lui A:
$E_A(C) = E_A(D_A(M)) = M$.

Dezavantajul major al aplicării criptosistemelor asimetrice pentru obţinerea semnăturii digitale constă în faptul că aceşti algoritmi sunt foarte lenţi şi deci, atunci când se aplică întregului mesaj (ca în exemplele anterioare), se consumă prea mult timp.

10.4.3. Apelarea la funcţii hash unidirecţionale

Datorită dezavantajelor, atât din punct de vedere al memoriei cât şi al timpului de calcul, modalităţile de realizare a semnăturii digitale prezentate anterior sunt foarte puţin utilizate în practică. Expunerea lor a fost făcută mai mult cu scopuri didactice, datorită simplităţii algoritmilor utilizaţi putându-se verifica mai uşor proprietăţile semnăturilor digitale. În majoritatea cazurilor practice se aplică o funcţie hash unidirecţională, ceea ce conduce (în special pentru mesajele lungi) la obţinerea unei

semnături digitale de o lungime mai mică decât a mesajului şi cu un consum de timp de calcul rezonabil.

Funcţiile hash unidirecţionale sunt întâlnite în literatura de specialitate sub diferite denumiri: compression function, contraction function, message digest, fingerprint, cryptographic checksum, data integrity check (**DIC**), manipulation detection code (**MDC**), message authentification cod (**MAC**), data authentification code (**DAC**). Funcţiile hash unidirecţionale pot utiliza sau nu o cheie pentru obţinerea valorii hash.

În cazul funcţiilor hash unidirecţionale cu chei (de tip **MAC**), A trimite lui B următorul document:

$M||H_K(M)$ dacă este necesară doar autentificarea mesajului,

sau

$E(M||H_K(M))$ dacă este necesară şi secretizarea mesajului.

Se observă că este posibilă asigurarea autenticităţii fără a fi necesară şi asigurarea confidenţialităţii.

În cazul funcţiilor hash unidirecţionale fără cheie (de tip **MDC**), A trimite lui B următorul document: $E(M||H(M))$. Se observă că asigurarea autenticităţii depinde de rezistenţa sistemului de cifrare utilizat.

Există foarte multe variante de obţinere a semnăturii digitale, una simplă fiind următoarea: A trimite lui B documentul $M||D_A(H(M))$ adică mesajul concatenat cu valoarea hash a mesajului, această valoare fiind cifrată cu cheia secretă a lui A.

10.4.4. Semnături digitale convenţionale (normale)

În cazul unei semnături digitale convenţionale sunt utilizaţi trei algoritmi:

a) algoritmul de generare a cheii publice şi a cheii private;

b) algoritmul de semnare, derulat de expeditor, apelând la cheia privată proprie;

c) algoritmul de verificare, derulat de destinatar, apelând la cheia publică a expeditorului.

În cele ce urmează prezentăm trei dintre cele mai cunoscute scheme de semnături digitale. Precizăm că schema **DSA** (*Digital Signature Algorithm*) a fost creată de **NSA** utilizând idei ale celorlalte două scheme, iar funcţia hash încorporată (**SHA**) a fost creată tot de **NSA**, fiind foarte asemănătoare cu funcţiile hash **MD4** şi **MD5** proiectate de *Ron Rivest*.

Semnătura ElGamal

Cheia publică:

p prim (poate fi folosit de un grup de utilizatori);

$g < p$ (poate fi folosit de un grup de utilizatori);

$y = g^x \bmod p$.
Cheia privată: $x < p$.
Semnarea:
k număr aleator, relativ prim cu $p-1$;
$r = g^k \bmod p$ (*semnătură*);
s astfel încât $M = xr + ks \bmod (p-1)$ (*semnătură*).
Verificarea: $y^r r^s \bmod p \stackrel{?}{=} g^M \bmod p$.

Semnătura Schnorr

Cheia publică:
p prim (poate fi folosit de un grup de utilizatori);
q prim, q divizor al lui $p-1$ (poate fi folosit de un grup de utilizatori);
a astfel încât $a^q = 1 \bmod p$ (poate fi folosit de un grup de utilizatori);
$v = a^{-x} \bmod p$.
Cheia privată: $x < q$.
Semnarea:
k număr aleator, $k < q$;
$r = a^k \bmod p$ (*semnătură*);
$e = H(M||r)$ (*semnătură*);
$s = k + xe \bmod q$.
Verificare:
$z = a^s v^e \bmod p$;
$e \stackrel{?}{=} H(M||z)$.

Semnătura DSA

Cheia publică:
p prim de 512-1024 biţi (poate fi folosit de un grup de utilizatori);
q prim de 160 biţi, q divizor al lui p-1 (poate fi folosit de un grup de utilizatori);
$g = h^{\frac{p-1}{q}}$, unde $h < p-1$ şi $h^{\frac{p-1}{q}} \bmod q > 1$ (poate fi folosit de un grup de utilizatori);
$y = g^x \bmod p$.
Cheia privată: $x < q$.
Semnarea:
k număr aleator, $k < q$;
$r = (g^k \bmod p) \bmod q$(*semnătură*);
$s = (k^{-1}(H(M) + xr) \bmod q$ (*semnătură*).
Verificare:
$w = s^{-1} \bmod q$;

$u_1 = (H(M)w) \bmod q$;
$u_2 = (rw) \bmod q$;
$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$;
$v \stackrel{?}{=} r$.

10.5. Alte tipuri de semnături digitale

10.5.1. Semnătura invizibilă

Există situaţii când expeditorul doreşte ca semnătura sa să poată fi verificată doar de destinatar, dar nu şi de o terţă persoană. În acest scop au fost concepute semnăturile invizibile, cunoscute şi sub denumirea mai puţin adecvată de *undiniable signature*. În acest caz, procesul de verificare a semnăturii nu mai poate fi executat doar de destinatar, ci este un proces interactiv la care ia parte şi expeditorul. Aceste scheme conţin, în general, şi un protocol de dezavuare prin care se poate proba dacă semnătura aparţine sau nu expeditorului.

10.5.2. Semnături fail-stop

Semnătura digitală protejează expeditorul cât timp aceasta nu poate fi falsificată, adică cât timp metoda criptologică pe care se bazează obţinerea cheii private rămâne invulnerabilă în faţa criptanalizei. Dacă această metodă nu a rezistat, semnătura poate fi falsificată, iar expeditorul este lipsit de apărare. În cazul semnăturii fail-stop, expeditorul poate aduce dovezi că semnătura sa a fost falsificată. Ideea de bază este ca pentru orice cheie publică să existe mai multe chei private, compatibile cu acea cheie publică, fiecare producând o semnătură posibilă. Dintre aceste chei private, A cunoaşte şi foloseşte doar una. Chiar dacă un atacator reuşeşte să deducă una din cheile private pretabile a fi folosite de A, setul de chei posibile se poate mări suficient de mult astfel încât probabilitatea ca această cheie să fie cheia reală utilizată de A să fie oricât de mică.

10.6. Legislaţia în domeniu

La sfârşitul anului 1999 a fost adoptată *directiva Uniunii Europene* referitoare a semnătura digitală. Adoptarea acestei directive implica amornizarea legislaţiilor naţionale. Legea Germană referitoare la semnătura electronică a fost revizuită în conformitate cu directivele UE şi are efect din anul 2001. În anul 2001 a fost adoptată în România legea 455 a semnăturii electronice, precum normele tehnice şi

metodologice de aplicare ale acesteia (HG 1259/2001, modificat prin HG 2303/2004). În SUA legea semnăturii electronice (Electronic Signature Act) este în vigoare încă din octombrie 2000. La ora actuală nu sunt amornizate încă legislaţiile în domeniul semnăturii electronice între SUA şi UE.

10.7. Aplicaţii

Exerciţiul 10.7.1. Setaţi (cât mai simplu) parametrii de securitate ai semnăturiilor ElGamal, Schnorr şi DSA.

Exerciţiul 10.7.2. Semnaţi mesajul *SEMNATURA DIGITALĂ* cu ajutorul parametrilor setaţi la exerciţiul anterior.

Exerciţiul 10.7.3. Implementaţi un algoritm de atac asupra funcţie de dispersie criptografică $MD4$ (vezi *Wang* [92] şi [93]).

Exerciţiul 10.7.4. Implementaţi algoritmi de atac asupra funcţiilor de dispersie criptografică $SHA0$ şi $SHA1$ (vezi *Wang* [94]).

Exerciţiul 10.7.5. Implementaţi o procedură care să producă coliziuni asupra primilor 40 de biţi ai algoritmului MD4/MD5.

Exerciţiul 10.7.6. Ce deosebeşte algoritmul $SHA-0$ de $SHA-1$?

Exerciţiul 10.7.7. Care sunt principalele metode de atac asupra funcţiilor bazate pe cifruri bloc simetrice?

Capitolul 11

CRIPTANALIZA PROTOCOALELOR CRIPTOGRAFICE

Insufficient cooperation in the development of one's own procedures, faulty production and distribution of key documents, incomplete key procedures, overlooked possibilities for compromise during the introduction of keying procedures provide the unauthorized decryption.
Erich Hüttenhain, 1978.

11.1. Protocoale elementare

În cadrul acestei secţiuni sunt referite, pe scurt, teoria protocoalelor criptografice şi vom prezenta o serie de protocoale de identificare/autentificare sigure din punct de vedere criptografic. Un protocol criptografic este un algoritm pentru realizarea comunicaţiilor sigure între diferiţi parteneri, aliaţi sau adversari. Protocoalele criptografice sunt în strânsă legătură cu conceptul de semnătură digitală. O serie de protocoale criptografice se bazează pe protocoale elementare, descrise în secţiunile ce urmează.

11.1.1. Protocoale de schimb de chei

Această clasă de protocoale are rolul de a stabili o cheie secretă de comunicaţie K între doi sau mai mulţi utilizatori ai reţelei. Fără restrângerea generalităţii, putem

presupune că trebuie să rezolvăm problema pentru doi utilizatori notaţi generic cu A şi B.

Schimbul de chei cu ajutorul criptografiei simetrice

Acest protocol presupune faptul că cei doi utilizatori ai reţelei A şi B au realizată o schemă de partajare a cheii secrete cu *Centrul de Distribuţie a Cheilor*, notat prin CDC. A, B şi CDC dispun de un sistem de cifrare simetric. Protocolul decurge după cum urmează:

PAS 1. A face o cerere de cheie secretă de sesiune, către CDC, pentru a comunica cu B.

PAS 2. CDC generează o cheie secretă de sesiune şi va realiza două copii ale acesteia, cifrate o dată cu ajutorul cheii lui A şi pe cea de a doua cu ajutorul cheii lui B, pe care apoi le va trimite lui A.

PAS 3. A descifrează, cu ajutorul cheii sale secrete, cheia de sesiune cu ajutorul căreia va comunica cu B.

PAS 4. A trimite lui B copia cifrată a cheii (cu cheia secretă a lui B), primită de la CDC.

PAS 5. B descifrează, cu ajutorul cheii sale secrete, cheia secretă de sesiune.

PAS 6. A şi B folosesc cheia secretă de sesiune pentru a comunica.

Acest protocol presupune o încredere absolută în CDC, care de cele mai multe ori este un computer sau o entitate de încredere. Dacă un adversar corupe $CDC-$ul, atunci întreaga reţea de comunicaţii este compromisă: se pot citi toate comunicaţiile din reţea. O altă problemă este aceea că CDC poate fi foarte aglomerat, deci vor apare perturbaţii ale comunicaţiilor.

Schimbul de chei cu ajutorul criptografiei asimetrice (chei publice)

În cazul acestui protocol utilizatorii A şi B folosesc o infrastructură de chei publice, denumită generic CDC, pentru a stabili o cheie secretă de sesiune. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A primeşte cheia publică a lui B de la CDC.

PAS 2. Utilizatorul A generează cheia secretă de sesiune, o cifrează cu ajutorul cheii publice a lui B şi o trimite apoi acestuia.

PAS 3. Utilizatorul B descifrează cheia secretă de sesiune cu ajutorul cheii sale private.

PAS 4. A şi B folosesc cheia secretă de sesiune pentru a comunica cifrat.

Atacul prin a treia persoană

Atacul prin intermediul celei de a treia persoane este unul dintre cele mai puternice atacuri. Acest tip de atac poate fi *pasiv* (interceptarea canalului de comunicaţie şi decriptarea mesajelor folosind numai textul cifrat) sau *activ* (interceptarea canalului de comunicaţie, modificare de mesaje, stergere de mesaje, inserţie de mesaje şi decriptarea mesajelor folosind atacul cu text clar cunoscut sau text clar ales). Interceptorul pasiv se va nota cu E iar interceptorul activ cu M. Interceptorul M se poate substitui lui B când comunică cu A şi viceversa. Atacul decurge după cum urmează:

PAS 1. Utilizatorul A trimite lui B cheia sa publică. Interceptorul M activ interceptează această cheie, o substituie cu cheia sa publică pe care o trimite lui B.

PAS 2. Utilizatorul B trimite lui A cheia sa publică. Interceptorul M activ interceptează această cheie, o substituie cu cheia sa publică pe care o trimite lui A.

PAS 3. Când utilizatorul A trimite un mesaj lui B, acesta va fi cifrat cu cheia publică a interceptorului M, care va descifra mesajul cu ajutorul cheii sale private (are deci acces la textul clar) pe care îl va cifra ulterior cu ajutorul cheii publice a lui B.

PAS 4. Când utilizatorul B trimite un mesaj lui A, acesta va fi cifrat cu cheia publică a interceptorului M, care va descifra mesajul cu ajutorul cheii sale private (are deci acces la textul clar) pe care îl va cifra ulterior cu ajutorul cheii publice a lui A.

Atacul funcţionează chiar dacă cheile publice sunt stocate într-o bază de date. Acest tip de atac este posibil datorită faptului că atât A cât şi B nu au posibilitatea de a verifica dacă, într-adevăr, comunică unul cu altul. Dacă interceptorul M nu cauzează funcţionării reţelei şi nu induce întârzieri în reţea atunci atacul este nedectat.

Protocolul mesajelor interclasate

Pentru evitarea atacului prin a treia persoană, se poate folosi protocolul mesajelor interclasate propus de *Ron Rivest* şi *Adi Shamir.* Protocolul decurge după cum urmează:

PAS 1. A trimite cheia sa publică lui B.

PAS 2. B trimite cheia sa publică lui A.

PAS 3. A cifrează mesajul folosind cheia publică a lui B. Trimite o parte din mesajul cifrat lui B.

PAS 4. B cifrează mesajul folosind cheia publică a lui A. Trimite o parte din mesajul cifrat lui A.

PAS 5. A trimite cealaltă parte de mesaj cifrat lui B.

PAS 6. B formează mesajul cifrat (primit din cele două părţi) pe care îl descifrează cu ajutorul cheii sale private. B trimite lui A cealaltă parte de mesaj cifrat.

PAS 7. A formează mesajul cifrat (primit din cele două părţi) pe care îl descifrează cu ajutorul cheii sale private.

Cheia acestui protocol este aceea că o parte de mesaj este nefolositoare fără cealaltă parte: nu poate fi descifrat. Utilizatorul B nu poate citi nimic din mesajul lui A până la pasul 6 iar utilizatorul A nu poate citi nimic până la pasul 7. Sunt mai multe posibilităţi de a realiza acest lucru:

-dacă algoritmul de cifrare este bloc, atunci orice parte a blocului poate fi transmisă în orice parte a mesajului;

-descifrarea mesajului poate să fie dependentă de vectorul de iniţializare ce se poate transmite în orice parte a mesajului;

-prima parte a mesajului poate fi o funcţie hash a mesajului cifrat iar mesajul cifrat se poate transmite în cea de a doua parte.

Schimbul de chei cu ajutorul semnăturilor digitale

Evitarea atacului prin a treia pesoană se poate face şi prin implementarea noţiunii de semnătură digitală. Acest protocol face apel la o entitate de încredere care va semna cheile publice ale utilizatorilor împreună cu o semnătură a proprietarului. Acum se poate verifica dacă cheile aparţin persoanei care pretinde a fi. Protocolul decurge în mod similar. Interceptorul activ M nu poate să se substituie nici lui A nici lui B deoarece acesta nu dispune de cheile private ale acestora. M nu poate substitui cheia reală cu cheia sa publică deoarece aceasta va fi semnată de către entitate de încredere TP ca aparţinând lui M. Tot ceea ce poate face M este aceea de a perturba reţeaua de comunicaţii. Acest protocol foloseşte pe TP iar riscul de a compromite $CDC-$ul este mult mai mic decât în primul protocol. Dacă M compromite pe TP (deci sparge CDC) atunci acesta dispune de cheia privată a lui TP. Acest lucru îi va permite să semneze noi chei, nu să descifreze cheile de sesiune sau să citescă mesajele din trafic. Interceptorul M va trebui să substituie cheile publice ale utilizatorilor din reţea şi deci va lansa un atac de tip cea de a treia persoană.

Transmiterea simultană a cheii şi a mesajului

Utilizatorii reţelei de comunicaţii A şi B nu trebuie, în mod neapărat, să realizeze un protocol de schimb de chei apriori transmiterii mesajelor. Protocolul următor permite transmiterea unui mesaj notat prin M fără a fi avut loc un schimb de chei:

PAS 1. A generează K cheie secretă de sesiune şi va cifra mesajul M cu ajutorul cheii $K : E_K(M)$.

PAS 2. A citeşte cheia publică a lui B din baza de date.

PAS 3. A cifrează cheia secretă de sesiune cu ajutorul cheii publice a lui B : $E_B(K)$.

PAS 4. A transmite lui B atât mesajul cifrat cât şi cheia cifrată: $E_K(M), E_B(K)$. (pentru a evita atacul prin a treia persoană A poate semna electronic transmisia).

PAS 5. B descifrează, cu ajutorul cheii sale private, cheia de sesiune K generată de A.

PAS 6. B descifrează, cu ajutorul cheii de sesiune, mesajul transmis de către A.

Acest sistem hibrid demonstrează cum se foloseşte criptografia cu chei publice în sisteme de comunicaţie. Aceasta se poate combina cu semnături digitale, identificatori dinamici şi alte protocoale de securitate.

Transmiterea cheii şi a mesajului către mai mulţi utilizatori

Realizarea transmisiei, de către A, a unui mesaj către mai mulţi utilizatori B, C, D,etc. se realizează după următorul protocol:

PAS 1. A generează K cheie secretă de sesiune şi va cifra mesajul M cu ajutorul cheii $K : E_K(M)$.

PAS 2. A citeşte cheile publice a lui B, C, D,etc. din baza de date.

PAS 3. A cifrează cheia secretă de sesiune cu ajutorul cheilor publice a lui B, C, D,etc.: $E_B(K), E_C(K), E_D(K), etc.$

PAS 4. A transmite atât mesajul cifrat cât şi cheile cifrate:

$$E_K(M), E_B(K), E_C(K), E_D(K), etc.$$

(pentru a evita atacul prin a treia persoană A poate semna electronic transmisia datelor).

PAS 5. Numai utilizatorii B, C, D etc. pot descifra, folosind cheile corespunzătoare private, cheia secretă de sesiune.

PAS 6. Numai utilizatorii B, C, D etc. pot descifra, folosind cheia secretă de sesiune, mesajele cifrate primite de la A.

11.1.2. Protocoale de autentificare

Scopul protocoalelor de autentificare este acela de a realiza o confirmare a identităţii într-o reţea de calculatoare. Vom prezenta o serie de protocoale care realizează acest lucru precum şi principalele tipuri de vulnerabilităţi ale acestora.

Autentificarea cu ajutorul funcţiilor hash

O metodă de autentificare a lui A de către B este aceea a parolelor care se realizează cu ajutorul unei funcţii hash:

PAS 1. A trimite lui B parola sa.

PAS 2. B calculează hashul parolei.

PAS 3. B compară hashul calculat cu valoarea stocată anterior. Dacă aceste două valori coincid atunci, B autentifică pe A.

Dacă se corupe hashul parolei atunci nu se poate afla din acesta parolă. Totuşi metoda atacului de tip dicţionar poate fi de folos în acest caz:

PAS 1. Se creează un dicţionar de posibile parole.

PAS 2. Se realizează a căutare exhaustivă a hashului parolei printre hashul cuvintelor din dicţionar.

Metoda de *atac de tip dicţionar* este foarte eficientă în cazul parolelor. Pentru a se evita un astfel de atac trebuie realizată o transmitere dinamică a parolei dar şi aşa mai rămâne riscul compromiterii bazei de date care stochează hashurile.

Autentificarea cu ajutorul criptografiei cu chei publice

Prezentăm o formă simplificată a unei metode de autentificare a lui A de către B cu ajutorul criptografiei cu chei publice.

PAS 1. A trimite lui B o secvenţă aleatoare.

PAS 2. A cifrează secvenţa aleatoare cu ajutorul cheii sale private pe care o trimite lui B împreună cu identificarea sa.

PAS 3. B descifrează secvenţa primită cu ajutorul cheii publice a lui A (accesul la cheia publică se face cu ajutorul identificatorului primit de la A).

PAS 4. B compară ce a descifrat cu ceea ce a primit la pasul 1 al protocolului. Dacă cele două secvenţe sunt egale, atunci B autentifică pe A.

Remarcăm faptul că nu avem acces la cheia privată a lui A deci nu se poate realiza impersonalizarea. Mult mai important este faptul că A nu va transmite niciodată cheia sa privată. Interceptorul pasiv sau activ nu va putea deduce cheia privată a lui A sau substitui pe A.

Autentificarea reciprocă

Se poate pune problema autentificării reciproce între doi utilizatori ai reţelei, notaţi generic cu A şi B, care dispun fiecare de parola celuilalt (P_A respectiv P_B). Vom prezenta un protocol ce *nu* va funcţiona:

PAS 1. A şi B fac schimb de cheile publice.

PAS 2. A cifrează P_A cu ajutorul cheii publice a lui B şi trimite rezultatul lui B.

PAS 3. B cifrează P_B cu ajutorul cheii publice a lui A şi trimite rezultatul lui A.

PAS 4. B descifrează ceea ce a primit de la A în cadrul pasului 2 şi verifică dacă a primit ceea ce trebuia.

PAS 5. A descifrează ceea ce a primit de la B în cadrul pasului 3 şi verifică dacă a primit ceea ce trebuia.

Atacul prin intermediul interceptorului activ decurge în mod natural: substituirea cheilor publice, descifrarea mesajelor (cu ajutorul cheii private a interceptorului) recuperarea parolelor P_A şi P_B, cifrarea mesajelor (cu ajutorul cheilor publice ale utilizatorului), etc. Evitarea atacului prin intermediul celei de a treia persoane se rezolvă apelând la un protocol de autentificare reciprocă bazat pe mesaje interclasate.

Protocoalele SKID (Secret Key Identification)

Protocoalele *SKID* se bazează pe criptografia simetrică (existenţa unui cod MAC notat prin H) şi rezolvă următoarele probleme:

-$SKID2$ autentificarea lui B de către A;

-$SKID3$ autentificarea reciprocă.

Protocolul SKID2 presupune faptul că cei doi utilizatori au cunoştinţă de cheia secretă K. Protocolul decurge după cum urmează:

PAS 1. A generează un număr aleatoriu, notat prin R_A, pe care îl comunică lui B.

PAS 2. B generează un număr aleatoriu, notat prin R_B, şi va trimite lui A:

$$R_B, H_K(R_A, R_B, B).$$

PAS 3. A va calcula $H_K(R_A, R_B, B)$ şi va compara cu ceea ce a primit la pasul 2. Dacă secvenţele sunt identice atunci B este autentificat de A.

Protocolul SKID3 decurge iniţial ca *SKID2* după care intervin paşii adiţionali:

PAS 4. A trimite lui B: $H_K(R_B, A)$.

PAS 5. B calculează $H_K(R_B, A)$ şi compară cu ceea ce a primit anterior. Dacă secvenţele sunt identice atunci A este autentificat de B.

Autentificarea mesajelor

Mesajele transmise într-o reţea de comunicaţii se pot autentifica prin intermediul semnăturii digitale.

11.1.3. Autentificarea şi schimbul de chei

Această clasă de protocoale combină protocolalele autentificare cu protocoalele de schimb de chei pentru a rezolva următorul scenariu general: A şi B sunt utilizatori ai unei reţele de comunicaţii şi doresc să comunice securizat. Cele mai multe protocoale fac apel la o terţă persoană notată cu TP (trusted person). Fiecare utilizator are o cheie secretă cunoscută numai de către TP. În tabelul de mai jos sunt specificate principalele elemente ale protocoalelor ce se vor prezenta în secţiunile următoare.

A	identificatorul utilizatorului A
B	identificatorul utilizatorului B
E_A	operaţia de cifrare cu o cheie cunoscută de TP şi A
E_B	operaţia de cifrare cu o cheie cunoscută de TP şi B
I	index
K	cheie de sesiune
L	timp de viaţă
T_A, T_B, T	ştampile de timp
R_A, R_B	numere aleatoare alese de A şi B

Protocolul bazat pe server de încredere

Cel mai simplu protocol care se bazează pe criptografia simetrică şi care face apel la un server de încredere TP. Utilizatorii A şi B au cheie secretă cunoscută numai de către TP. Aceste chei sunt folosite pentru distribuţia cheilor şi nu sunt folosite pentru cifrarea mesajelor. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A concatenează o ştampilă de timp, identificatorul lui B, cheia secretă de sesiune pe care le cifrează cu cheia cunoscută de TP şi A. Trimite apoi lui TP identificatorul său A şi mesajul cifrat:

$$A, E_A[T_A, B, K].$$

PAS 2. TP va descifra mesajul primit de la A, recuperează cheie secretă de sesiune, generează o nouă ştampilă de timp, adjuncţionează identificatorul lui B şi va cifra mesajul cu cheia lui B :

$$E_B[T_B, A, K].$$

Un punct vulnerabil este generarea numerelor aleatoare de către utilizatorul A : generarea de numere aleatoare nu este un lucru simplu.

Protocolul Yahalom

Acest protocol presupune faptul că fiecare dintre utilizatorii A şi B au cheie secretă cunoscută numai de către TP. Protocolul decurge după cum urmează:

PAS 1. A generează R_A şi transmite lui B următorul mesaj:

$$A, R_A.$$

PAS 2. B generează R_B şi transmite lui TP următorul mesaj:

$$B, E_B[A, R_A, R_B].$$

PAS 3. TP generează cheia secretă de sesiune şi transmite lui A mesajul:

$$E_A[B, K, R_A, R_B], E_B[A, K].$$

PAS 4. A descifrează prima parte a mesajului cu ajutorul cheii sale secrete, recuperează K şi confirmă dacă este R_A-ul trimis iniţial. Trimite lui B mesajul:

$$E_B[A, K], E_K[R_B].$$

PAS 5. B descifrează cifrat cu cheia sa secretă, extrage K, descifrează mesajul cifrat cu cheia secretă de sesiune şi confirmă dacă R_B este cel generat la pasul 2.

La finalul protocolului A şi B sunt convinşi că fiecare discută cu cine trebuie şi nu cu o terţă parte. În aceelaşi timp aceştia dispun de cheia secretă de sesiune K. Noutatea acestui protocol este aceea că B este primul cel contactează pe TP iar acesta trimite un singur mesaj lui A.

Protocolul Needham-Schroeder

Acest protocol este elaborat de *Roger Needham* şi *Michael Schroeder* se bazează pe existenţa unei terţe părţi de încredere TP şi face apel la conceputul de criptografie simetrică. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A trimite lui TP mesajul:

$$A, B, R_A.$$

PAS 2. TP va genera cheia secretă de sesiune şi va trimite lui A mesajul:

$$E_A[R_A, B, K, E_B[K, A]].$$

PAS 3. Utilizatorul A va recupera cheia secretă de sesiune K, va confirma starea lui R_A şi va trimite lui B mesajul:

$$E_B[K, A].$$

PAS 4. Utilizatorul B va recupera cheia secretă de sesiune K şi va trimite lui A mesajul:

$$E_K[R_B].$$

PAS 5. Utilizatorul A va recupera R_B şi va trimite lui B mesajul:

$$E_K[R_B - 1].$$

PAS 6. Utilizatorul B verifică starea lui $R_B - 1$.

Utilizarea numerelor R_A, R_B şi $R_B - 1$ are scopul de a preveni *atacurile prin redare*: interceptorul activ va stoca valorile lui înregistrate în cadrul protocolului pentru o utilizare ulterioară.

O slăbiciune a acestui protocol constă în faptul că *vechile chei de sesiune pot fi folosite* pentru a lansa un atac:

PAS 0. Interceptorul activ M va înregistra toate mesajele din pasul 3 al protocolului.

PAS 1. Interceptorul activ M va trimite lui B mesajul:

$$E_B[K, A].$$

PAS 2. Utilizatorul B va extrage cheia secretă de sesiune K, va genera R_B şi va trimite lui A mesajul:

$$E_K[R_B].$$

PAS 3. M interceptează mesajul, va recupera R_B, şi va trimite lui B:

$$E_K[R_B - 1].$$

PAS 4. Utilizatorul B verifică veridicitatea lui $R_B - 1$ (deci va "confirma" identitatea lui A).

Pentru evitarea acestui tip de atac trebuie folosite ştampilele de timp (în pasul 2 al protocolului se va utiliza: $E_B[K, A, T]$).

Compromiterea cheii secrete a lui A are consecinţe dramatice: interceptorul activ M o poate folosi pentru a obţine chei secrete de sesiune pentru a comunica cu B. Atacul iniţiat de interceptorul M poate continua chiar dacă A schimbă cheia secretă de sesiune. Protocolul descris în secţiunea următoare rezolvă această problemă.

Protocolul Otway-Rees

Protocolul Otway-Rees utilizează deasemenea conceptul de criptografie simetrică. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A va transmite lui B următorul mesaj:

$$I, A, B, E_A[R_A, I, A, B].$$

PAS 2. Utilizatorul B va trimite lui TP mesajul:

$$I, A, B, E_A[R_A, I, A, B], E_B[R_B, I, A, B].$$

PAS 3. TP va genera o cheie secretă de sesiune K şi va trimite lui B mesajul:

$$I, E_A[R_A, K], E_B[R_B, K].$$

PAS 4. Utilizatorul B recuperează cheia secretă de sesiune, va confirma starea numărului R_B şi va trimite lui A mesajul:

$$I, E_A[R_A, K].$$

PAS 5. Utilizatorul A va recupera cheia secretă de sesiune şi va confirma starea numărului R_A.

Dacă protocolul s-a încheiat şi dacă indexul I, numerele R_A şi R_B sunt nemodificate, atunci utilizatorii A şi B sunt în posesia cheii K şi sunt convinşi de veridicitatea acesteia.

Protocolul Kerberos

Protocolul Kerberos este o variantă a protocolului Needham-Schroeder. Versiunea 5 a protocolui Kerberos face presupunerea că fiecare din utilizatorii A şi B au câte o cheie secretă cunoscută numai de către TP. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A trimite lui TP mesajul:

$$A, B.$$

PAS 2. TP va trimite lui A mesajul:

$$E_A[T, L, K, B], E_B[T, L, K, A]$$

unde T este timerul, L perioada de valabilitate a acestuia iar K cheia de sesiune.

PAS 3. Utilizatorul A va recupera cheia secretă de sesiune K şi va trimite lui B mesajul:

$$E_K[A, T], E_B[T, L, K, A].$$

PAS 4. Utilizatorul B va trimite lui A mesajul:

$$E_K[T + 1]$$

care decide valabilitatea timerului T.

Protocolul funcţionează în baza presupunerii că timerii lui A şi B sunt sincronizaţi cu timerul lui TP. În aplicaţiile practice, efectul este obţinut prin sincronizarea timerului la interval de câteva minute.

11.1.4. Protocoale de transfer orb

Să considerăm următoarea variantă de comunicare confidenţială: A doreşte să îi comunice lui B *un bit de informaţie* astfel încât după finalizarea protocolului A să nu ştie dacă B a primit bitul de informaţie, dar B ştie. Următorul exemplu va exemplifica cele de mai sus: A este un deţinător de secrete, iar B un cumpărător. A comunică lui B lista (sumarul) informaţiilor deţinute. B doreşte să cumpere unul dintre secrete, dar nu vrea ca nici măcar A să ştie care a fost cumpărat (în caz contrar B va dezvălui lui A un alt secret şi anume ce a cumpărat acesta).

Protocolul de transfer orb constă din următorul scenariu general: A doreşte să îi comunice, prin intermediul unui protocol de transfer în orb, lui B factorizarea numărului $n = p \times q$ (p şi q sunt numere foarte mari).

PAS 1. B alege un număr x şi îi comunică lui A numărul $x^2 \bmod n$.

PAS 2. A care cunoaşte factorizarea poate calcula (eficient) cele patru rădăcini pătrate $\pm x, \pm y$ ale lui $x^2 \bmod n$ şi comunică una din acestea lui B. (Observăm că A cunoaşte doar pătratul şi deci el nu are nici o cale să afle care dintre rădăcinile pătrate este x.)

PAS 3. B verifică dacă rădăcina primită este congruentă cu $\pm x \bmod n$. În caz afirmativ, B nu a primit nici o informaţie. În caz contrar B poate calcula (eficient) factorizarea lui n deoarece cunoaşte două rădăcini pătrate diferite ale aceluiaşi număr $(\mathrm{mod} n)$. A nu ştie care din cele două situaţii a avut loc.

Evident, pentru creşterea siguranţei criptografice, protocolul se poate itera. Există variante de protocoale de transfer orb bazate pe generatoare pseudoaleatoare şi pe funcţii de dispersie criptografice.

11.1.5. Analiza formală a protocoalelor de autentificare şi a protocoalelor de schimb de chei

Prin tehnica analizei formale a protocoalelor criptografice se urmăreşte, în special, evidenţierea vulnerabilităţilor şi analiza riscurilor ce decurg din acestea. În general sunt patru abordări ale analizei protocoalelor criptografice:

1. Modelarea şi verificarea protocolului prin folosirea limbajelor de specificaţie, verificarea tehnicilor şi metodelor ce nu sunt proiectate pentru analiza protocoalelor criptografice;

2. Dezvoltarea de sisteme expert prin care proiectantul protocolului criptografic poate dezvolta şi investiga diverse scenarii;

3. Modelarea cerinţelor familiei protocolului criptografic folosind logica analizei cunoştinţelor şi a presupunerilor probabiliste;

4. Dezvoltarea de metode formale bazate pe proprietăţi algebrice ale sistemelor criptografice.

O discuţie a paşilor de mai sus ar depăşi cadrul alocat acestei cărţi.

11.1.6. Divizarea secretului

Operaţia prin care un secret se reconstituie din două sau mai multe părţi componente se numeşte *protocol de divizare a secretului*. Iată un astfel de protocol pentru două părţi implicate în operaţiune notate generic cu A şi B. Se notează proprietarul secretului prin TP (cel ce realizează divizarea efectivă).

PAS 1. TP generează o secvenţă aleatoare R de aceeaşi lungime ca mesajul M.

PAS 2. TP $XOR-$ează M cu R pentru a obţine secvenţa S.

PAS 3. TP va distribui R lui A şi S lui B.

Reconstrucţia lui M se va obţine din sumarea modulo 2 a secvenţelor distribuite lui A şi B. Părţile implicate în acest protocol nu pot reconstitui nici o parte din secvenţa secretă. Scheme similare se pot realiza pentru mai mulţi utilizatori.

11.1.7. Partajarea secretului

Operaţia prin care un secret se reconstituie prin *concatenarea* a două sau mai multe părţi componente se numeşte *protocol de partajare a secretului*. Un astfel de protocol este protocolul de tip logică majoritară. Spre deosebire de schemele de divizare a secretului, părţile implicate în acest proces deţin o parte din secvenţa secretă. Probleme de atac a acestor clase de protocoale sunt complexe: partajarea secretului în care unul dintre utilizatori este corupt, partajarea secretului fără centrul de distribuţie etc.

11.2. Protocoale avansate

11.2.1. Protocol de tip demonstraţie convingătoare fără detalii

Scenariul general în care se aplică acest tip de protocol este următorul: A doreşte să îl convingă pe B că deţine o anumită informaţie, dar pe de altă parte nu doreşte ca B să afle acea informaţie. Cerinţele de bază ale protocolului sunt:

1. A nu poate să-l înşele pe B. Dacă A nu cunoaşte informaţia, probabilitatea ca să-l înşele pe B să fie mică.

2. B nu află nimic de la A. Deci B nu se poate substitui ulterior lui A.

Acest tip de protocoale se aplică la autentificarea cărţilor de credit, parolelor pentru calculator (inclusiv reţele) precum şi la autentificarea în reţelele de comunicaţii. O variantă de astfel de protocol, care poate fi modificat corespunzător, pentru a nu

mai face apel la noţiunea de eficienţă computaţională este cea de tip cunoaştere zero ale identităţii.

11.2.2. Protocol de tip dezvăluire minimă

Scenariul general este următorul A : doreşte să-l convingă pe B că ştie factorizarea numărului $n = p \times q$(p şi q sunt numere foarte mari).

PAS 1. B alege aleatoriu numărul întreg b şi îi comunică lui A : $b^4 \bmod n$.

PAS 2. A (care ştie factorizarea lui n) îi comunică lui B : $b^2 \bmod n$.

B nu află decât că A ştie să factorizeze numărul n, dar nu află nimic despre această factorizare. Pentru creşterea siguranţei criptografice, protocolul se poate itera.

11.2.3. Protocol de tip dezvăluire zero

Un exemplu de protocol de autentificare între două părţi A şi B, care dispun de informaţia comună I_A(identificatorul lui A) şi I_B(identificatorul lui B). Este un exemplu de protocol de tip *cunoaştere zero a identităţii* (B îi demonstrează lui A că deţine o anumită informaţie fără să dezvăluie nici un bit din informaţie). Această informaţie comună se presupune secretă, deci compromiterea ei conduce la spargerea protocolului.

Protocol de autentificare lui B de către A.

PAS 1. A generează aleatoriu un şir de biţi $a_1, \ldots, a_n$ şi îi comunică lui B.

PAS 2. B generează aleatoriu un şir de biţi $b_1, \ldots, b_m$ şi îi comunică lui A.

PAS 3. B calculează:

$$k^B = g(\phi(I_B, \mathbf{a}, \mathbf{b})),$$

unde g este un generator pseudoaleatoriu, iar ϕ o funcţie de dispersie criptografică. B comunică lui A k_p^B(primii p biţi din ieşirea lui g).

Pas 4. A calculează $g_p(\phi(I_B, \mathbf{a}, \mathbf{b}))$ şi compară cu ceea ce a primit de la B în pasul anterior. Dacă cei doi vectori coincid, atunci A *autentifică* pe B.

Pentru creşterea securităţii protocolului acesta se poate *itera*.

Pentru autentificarea lui A de către B protocolul se modifică corespunzător. El se poate implementa foarte uşor. Tăria acestuia constă în tăria generatorului pseudoaleatoriu g şi a funcţiei de dispersie criptografică.

11.2.4. Protocoale de tip transfer bit şi aplicaţii

Introducere. Protocolul de tip *transfer bit* este o componentă de bază pentru mai multe protocoale criptografice. Una dintre părţile implicate A doreşte să-i comunice altei părţi B un bit b astfel încât B să nu ştie ce bit a primit. La un pas

ulterior A poate face public bitul trimis, iar B poate decide dacă ce a fost făcut public este ceea ce a primit el de la A. O metodă practică de prezentare a protocolului este aceea de *sigilare* a bitului cu ştampila de timp. Aceste scheme folosesc la protocoale de tip cunoaştere zero, scheme de identificare, protocoalele cu mai mulţi participanţi şi pot implementa protocolul de dat cu zarul prin telefon.

Definiţia 11.2.1. Un *protocol de tip transfer de bit* este compus din două faze şi anume:

1. *Faza de transfer,* în care A are un bit b pe care doreşte să-l comunice lui B. A şi B schimbă mesaje. La sfârşitul acestei faze B este în posesia unei informaţii despre ceea ce reprezintă b.

2. *Faza de dezvăluire* la finalul căreia B cunoaşte bitul b.

Protocolul trebuie să îndeplinească următoarele: Pentru orice B probabilist cu timp polinomial, pentru orice polinom p şi pentru un parametru de securitate n suficient de mare:

a) după faza de transfer B nu poate ghici b cu o probabilitate mai mare decât $\frac{1}{2} + \frac{1}{p(n)}$;

b) A nu poate dezvălui decât o singură valoare b. Dacă se încearcă dezvăluirea altei valori, acest lucru va fi detectat cu probabilitate de cel puţin $1 - 1/p(n)$.

În definiţia proprietăţilor protocolului am presupus că, apriori, B nu poate ghici b cu o probabilitate mai mare decât $0,5$.

Definiţia se poate extinde natural la transferul de mai mulţi biţi. Schema propusă se bazează pe noţiunea de generator pseudoaleatoriu.

Definiţia 11.2.2. Fie $m(n)$ o funcţie astfel încât pentru $m(n) > n$, aplicaţia $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ se numeşte *generator pseudoaleatoriu* dacă pentru orice polinom p şi orice maşină probabilistă cu timp polinomial care face distincţia dintre ieşirea generatorului şi un şir cu adevărat aleatoriu inegalitatea:

$$|\Pr[A(y) = 1] - \Pr[A(G(s)) = 1]| < \frac{1}{p(n)},$$

are loc pentru orice n cu excepţia unui număr finit. Probabilităţile sunt luate peste $y \in \{0,1\}^{m(n)}$ şi $s \in \{0,1\}^n$ care sunt alese aleatoriu.

Transferul de bit. *Pseudoaleatorismul* are următoarea proprietate: dat fiind m biţi ai şirului pseudoaleatoriu, orice algoritm polinomial relativ la timp care încearcă să previzioneze următorul bit are probabilitate de succes mai mică ca $\frac{1}{2} + \frac{1}{p(n)}$ (nepredictibilitatea următorului bit). Fie n un parametru de securitate care se presupune că este ales astfel încât nu există nici o maşină care să spargă un generator

pseudoaleatoriu a cărui iniţializare este de lungime n. Vom nota $G_l(s)$ primii l biţi ai şirului pseudoaleatoriu iniţializat cu $s \in \{0,1\}^n$ şi cu $B_i(s)$ cel de-al i-lea bit al şirului cu iniţializarea s.

Într-o primă fază să considerăm următorul protocol:

a) *Faza de transfer:* A alege aleatoriu $s \in \{0,1\}^n$ şi transmite lui B $G_m(s)$ şi $B_{m+1}(s) \oplus b$.

b) *Faza de dezvăluire:* A comunică s, B verifică că $G_m(s)$ coincide cu ceea ce a primit anterior şi calculează:

$$b = B_{m+1}(s) \oplus (B_{m+1}(s) \oplus b).$$

Acest protocol are proprietatea că B nu poate indica, înaintea fazei de devăluire, ceea ce a primit de la A cu o probabilitate mai mică ca $1/2 + 1/poly(n)$ deoarece el nu poate realiza o predicţie a generatorului pseudoaleatoriu. Pe de altă parte A poate trişa dacă dispune de două iniţializări $s_1 \neq s_2$ pentru care $G_m(s_1) = G_m(s_2)$ şi $B_{m+1}(s_1) \neq B_{m+1}(s_2)$. Definiţia actuală a generatorului pseudoaleatoriu nu exclude asemenea posibilitate. Mai mult chiar, dat fiind un generator pseudoaleatoriu G se poate construi un generator $G^{'}$ care să satisfacă condiţiile amintite.

Nu este nici o cale de a forţa pe A să folosească o singură iniţializare deoarece pot fi două iniţializări care duc la acelaşi şir. Protocolul din secţiunea următoare forţează pe A să folosească un singur generator, sau dacă nu, el va putea fi detectat cu probabilitate mare.

Protocolul de transfer de bit. În această secţiune sunt prezentate un protocol de transfer de bit precum şi consideraţii asupra siguranţei acestuia.

Faza de transfer constă în următoarele:

a) B alege un vector aleatoriu $\mathbf{R} = (r_1, \ldots, r_{3n})$ şi îl comunică lui A.

b) A alege o iniţializare $s \in \{0,1\}^n$ şi comunică lui B vectorul $\mathbf{D} = = (d_1, \ldots, d_{3n})$ unde:

$$d = \begin{cases} B_i(s) \text{ dacă } r_i = 0 \\ B_i(s) \oplus b \text{ dacă } r_i = 1. \end{cases}$$

La *faza de transfer* A comunică lui B iniţializarea s, iar acesta verifică că dacă $r_i = 0$ atunci decide $d_i = B_i(s)$ şi dacă $r_i = 1$ decide $d_i = B_i(s) \oplus b$.

Acest protocol are proprietatea că B nu află nimic despre bitul b decât după faza de dezvăluire. În caz contrar, B ar trebui să facă distincţia dintre ieşirea unui generator şi un şir aleatoriu. Dacă A alege un şir aleatoriu în loc de un şir pseudoaleatoriu, atunci B nu află nimic depre b deoarece toţi vectorii $\mathbf{D}$ sunt echiprobabili indiferent de valoarea lui b. (Aceasta este adevărat chiar şi în cazul general în care B are o intrare suplimentară care îi permite să indice b cu probabilitatea $q > 1/2$.) Dacă nu există un algoritm probabilist $B^{'}$ care poate să afle ceva despre b dacă A foloseşte un algoritm pseudoaleatoriu, atunci $B^{'}$ poate construi un algoritm care să

facă distincţia dintre ieşirea lui G şi un şir cu adevărat aleatoriu. Dat fiind un şir x executăm protocolul cu A şi $B^{'}$, unde A alege un bit aleatoriu b în loc să creeze un şir pseudoaleatoriu folosind x. Să presupunem că $B^{'}$ indică b. Dacă el ghiceşte corect, atunci el decide x pseudoaleatoriu şi în caz contrar x aleatoriu. Diferenţa dintre probabilitatea de a decide între un şir pseudoaleatoriu şi un şir aleatoriu este egală cu avantajul pe care $B^{'}$ îl are de a indica corect b în cazul x este un şir pseudoaleatoriu.

Cum poate trişa A? Singura posibilitate a acestuia este de a exista două iniţializări s_1 şi s_2 astfel încât $G_{3n}(s_1)$ şi $G_{3n}(s_2)$ să fie egale în acele poziţii i în care $r_i = 0$ şi total diferite în acele poziţii i în care $r_i = 1$. Numim o astfel de pereche $\mathbf{R}-$falsificator.

Teorema 11.2.1. *Probabilitatea de a exista două iniţializări care să formeze un* $\mathbf{R}-$*falsificator este cel mult* 2^{-n} *(probabilitatea se ia peste mulţimea tuturor posibilităţilor lui* $\mathbf{R}$*).*

Rezultatele acestea se pot sintetiza în următoarea teoremă.

Teorema 11.2.2. *Dacă G este un generator pseudoaleator, atunci protocolul de transfer de bit prezentat satisface următoarele: pentru orice polinom p şi pentru orice parametru n suficient de mare:*

1. După faza de transfer, nu există nici un B polinomial în timp probabilist care să indice b cu probabilitatea mai mare de $1/2 + 1/p(n)$.

2. Probabilitatea ca A să indice alt bit este de cel mult 2^{-n}.

Protocolul de transfer indicat are costul computaţional de $O(n)$. Urmând rezultatele anterioare putem construi un protocol de transfer de mai mulţi biţi eficient. Să presupunem că A doreşte să transfere biţii $b_1, \ldots, b_m$ pe care îi face publici simultan. Ideea este de a forţa ca A să folosească un singur şir pseudoaleator şi să folosească şirul la transferul a cât mai mulţi biţi.

Protocol de tip electoral (solitaire). Ideea acestui tip de protocol a fost pusă în evidenţă de *Valteri Niemi* şi constă în calculul oricărei funcţii booleene fără ca intrările acesteia să fie făcute publice.

Vom exemplifica protocolul pentru calculul valorii funcţiei booleene $f : \{0,1\}^2 \to \{0,1\}$ unde $f(a,b) = a\&b$. În acest protocol, numit *solitaire,* dispunem de cinci cărţi de joc şi anume două negre notate cu N şi trei roşii notate cu R.

Iniţializare. Participanţii A şi B primesc fiecare câte două cărţi una N şi una R. Cealaltă carte R este pusă cu faţa în jos.

PAS 1. a) A generează un bit secret a şi foloseşte următoarea regulă de codificare a acestuia: dacă $a = 0$ el aranjează cărţile în forma $(\frac{N}{R})$, iar dacă $a = 1$ el aranjează cărţile în forma $(\frac{R}{N})$. Acesta pune cărţile cu faţa în jos *sub* cartea existentă.

b) B generează un bit secret b şi foloseşte următoarea regulă de codificare a acestuia: dacă $b = 1$ el aranjează cărţile în forma $(\frac{N}{R})$, iar dacă $b = 0$ el aranjează cărţile în forma $(\frac{R}{N})$. Acesta pune cărţile cu faţa în jos *peste* cărţile existente.

PAS 2. Teancul de cărţi este tăiat aleatoriu de către A şi B. Prin tăiere înţelegem operaţia de rotire circulară.

PAS 3. Cărţile sunt întoarse cu faţa în sus. Dacă avem $< rrrnn >$ (ordinea ciclică) atunci $f(a, b) = 1$, în caz contrar $f(a, b) = 0$.

Pentru calculul funcţiei *not* se taie pur şi simplu cărţile generate de A, iar pentru calculul funcţiei $f(a, b) = a \vee b$ se schimbă codificarea de la pasul 1 la punctele a) şi b). Deci putem construi un protocol pentru calculul oricărei funcţii booleene (evident cu un număr suficient de cărţi).

Această clasă de protocoale ne permite construcţia unor protocoale de tip cunoaştere zero a identităţii.

11.2.5. Alte protocoale avansate

Din categoria *protocolalelor avansate* mai fac parte următoarele:

-protocoale pentru semnături oarbe;

-protcoalele de dezvăluire a identităţii cu ajutorul cheilor publice;

-protocoalele de semnare simultană a contractelor;

-protocoalele de schimbare simultană a secretelor.

Descrierea acestora depăşeşte cadrul acestei cărţi (pentru detalii se poate consulta *Schneier* [69]).

11.3. Divizarea şi partajarea secretelor

Una dintre problemele ce se pot ridica în cazul aplicaţiilor criptografice este aceea de *partajare a secretului* (secret sharing). Acest secret poate fi, de exemplu, o cheie de criptare (serviciul de *key recovery*). Schema cea mai utilizată este schema majoritară m din n în care secretul K este distribuit la n participanţi ai schemei. Secretul trebuie reconstituit de *oricare* $m < n$ participanţi. Sunt prezentate în cele ce urmează o serie de scheme de partajare a secretului: schema vectorială, LaGrange, Asmuth-Bloom şi Karnin-Greene-Hellman. O schemă în care $m = n$ se va numi schemă de *divizare a secretului* (secret splitting). Alegerea parametrilor m (numărul minim de participanţi care pot iniţializa procedura de recuperare) şi n (numărul de participanţi din cadrul protocolului criptografic) trebuie făcută în mod judicios: o valoare prea mare a lui n implică costuri de distribuţie mari, o valoare mică a lui n implică riscuri de nerealizare a procedurii de recuperare a secretului, un raport prea

mic $\frac{m}{n}$ poate favoriza un atac de tip corupere a m participanţi. Pentru aprofundarea domeniului se poate consulta *Schneier* [69].

11.3.1. Protocol de divizare a secretului

Vom prezenta un protocol de $n-$divizare a secretului.

Iniţializare. n numărul participanţilor şi secretul K.

PAS 1. Se generează aleatoriu de către *autoritatea de distribuţie a secretului* TP un număr de $n-1$ valori k_i de aceeiaşi lungime ca secretul K.

PAS 2 (distribuţia secretului). Se distribuie participantului i valoarea k_i iar participantului n valoarea:

$$K \oplus \bigoplus_{i=1}^{n-1} k_i.$$

PAS 3 (recuperarea secretului). Valoarea secretă K se poate recupera numai cu informaţia oferită toţi cei n participanţi.

11.3.2. Protocolul de partajare LaGrange

Următoarea schemă majoritară este atribuită lui *Shamir* [68].

Iniţializare. n numărul participanţilor, m pragul minim de reconstrucţie al secretului K.

PAS 1. Se alege de către *autoritatea de distribuţie a secretului* TP un număr prim p suficient de mare (mai mare ca cel mai mare secret distribuit). Se generează de către autoritatea de distribuţie a secretului TP *aleatoriu* un polinom:

$$P(X) = \sum_{i=1}^{m-1} a_i X^i + K$$

de grad $m-1$, unde $p \geq \max_i a_i$.

PAS 2 (distribuţia secretului). Autoritatea TP distribuie participantului i valoarea $k_i = P(i),\ i = 1, ..., n$.

PAS 3 (recuperarea secretului). Cu informaţia oferită de m participanţi se poate recupera, prin rezolvarea unui sistem liniar de m ecuaţii, valoarea K. Dacă numărul participanţilor care pun la dispoziţie informaţia k_i este mai mic ca m, atunci *nu se poate* determina K.

Observaţia 11.3.1. În cazul în care coeficienţii polinomului P sunt aleşi aleatoriu, atunci tăria acestei scheme este echivalentă cu schema de cifrare OTP (one time pad).

Observaţia 11.3.2. Pentru evitarea atacului prin a treia persoană, schema se poate îmbunătăţii prin utilizarea unei valori aleatoare x_i în cadrul operaţiei de distribuţie a secretului: $k_i = P(x_i),\ i = 1, ..., n$.

11.3.3. Protocolul de partajare vectorial

Blakley [10] a propus o schemă de partajare a secretului bazată pe noţiunea de spaţiu $m-$dimensional, secretul K fiind un punct din acest spaţiu. Astfel, fiecărui participant (dintre cei n) i se distribuie (de către autoritatea de distribuţie a secretului TP) o ecuaţie de $m-1$ hiperplane care conţin K. Intersecţia oricăror $m-$hiperplane determină în mod unic punctul K.

11.3.4. Protocolul de partajare Asmuth-Bloom

Această schemă poate fi găsită în *Asmuth* [3] şi face apel la *CRT* (chinese remainder theorem).

Iniţializare. n numărul participanţilor, m pragul minim de reconstrucţie al secretului K.

PAS 1. Se alege de către *autoritatea de distribuţie a secretului* TP un număr prim $p > M$. Se generează de autoritatea de distribuţie a secretului n numere $d_i < p$ astfel ca:

i) d valori sunt în ordine crescătoare: $d_i < d_{i+1}$.

ii) $(d_i, d_j) = 1$ pentru orice $i \neq j$.

iii) $d_1 d_2 ... d_m > p d_{n-m+2} d_{n-m+3} ... d_n$.

Se alege de către TP un număr aleatoriu r şi se calculează: $K^{'} = K + rp$.

PAS 2 (distribuţia secretului). Autoritatea TP distribuie participantului i valoarea $k_i = K^{'} \bmod d_i,\ i = 1, ..., n$.

PAS 3 (recuperarea secretului). Cu informaţia oferită de m participanţi *se poate recupera*, prin aplicarea teoremei chinezeaşti a resturilor, valoarea K. Dacă numărul participanţilor care pun la dispoziţie informaţia k_i este mai mic ca m atunci *nu se poate determina* K.

11.3.5. Protocolul de partajare Karnin-Greene-Hellman

Această schemă poate fi găsită în *Karnin* [36] şi utilizează operaţia de multiplicare a matricilor.

Iniţializare. n numărul participanţilor, m pragul minim de reconstrucţie al secretului K.

PAS 1. Se aleg de către *autoritatea de distribuţie a secretului* TP un număr de $n+1$ vectori $m-$dimensional, $V_0, ..., V_n$, astfel ca oricare matrice formată din aceşti vectori să aibă rangul m. Se alege de autoritate vectorul U astfel ca $K = U^t V_0$.

PAS 2 (distribuţia secretului). Autoritatea TP distribuie participantului i valoarea $k_i = U^t V_i,\ i = 1, ..., n$.

PAS 3 (recuperarea secretului). Cu informaţia oferită de m participanţi *se poate recupera*, prin rezolvarea unui sistem de ecuaţii liniare, valoarea K. Dacă numărul participanţilor care pun la dispoziţie informaţia k_i este mai mic ca m atunci *nu se poate determina* K.

11.3.6. Atacuri asupra protocoalelor de partajare (divizare) a secretului

Schemele de partajare a secretului au avantajul, spre deosebire de schemele de divizare a secrtelui, că valoarea secretă K se poate afla dacă nu sunt disponibile informaţii oferite de un număr nu prea mare (mai mic decât $n - m$) participanţi ai protocolului. În cazul schemelor de divizare a secretului lipsa unui singur participant duce la nedisponibilitatea valorii secrete K şi pentru aflarea acesteia se face apel la autoritatea TP (*trusted*) care a distribuit valorile k_i.

11.4. Exemple de implementare

11.4.1. Scheme de autentificare

Vom prezenta o serie de exemple practice de implementare a schemelor de autentificare cu ar fi: *Feige-Fiat-Shamir, Guillou-Quiquater şi Schnorr.*

Feige-Fiat-Shamir

Este prezentat un protocol de autentificare a lui P (Prover) de către V (Verifier), bazat pe existenţa unei autorităţi de gestionare a cheilor. Pseudocodul schemei de identificare simplificate *Feige-Fiat-Shamir* este următorul:

PAS 0. Se va seta (de către autoritatea de gestionare a cheilor) numărul aleatoriu n (produs de două numere prime de ordin de mărime mare, de exemplu n are cel puţin 1024 biţi) care este distribuit utilizatorilor. Se va alege (de către autoritatea de gestionare a cheilor) v reziduu pătratic $\bmod n$: adică ecuaţia $x^2 \equiv v \bmod n$ are soluţie iar $v^{-1} \bmod n$ există. Acest v este cheia publică a lui P. Cheia privată a lui P este $s \equiv sqrt(v^{-1}) \bmod n$.

PAS 1. P alege $r < n$ aleatoriu. Se calculează $x = r^2 \bmod n$ număr pe care îl trimite lui V.

PAS 2. V trimite lui P un bit aleatoriu b.

PAS 3. Dacă $b = 0$ atunci P trimite lui V pe r. Dacă $b = 1$, atunci P trimite lui V: $y = r \cdot s \bmod n$.

PAS 4. Dacă $b = 0$ V va verifica dacă $x = r^2 \bmod n$, dovedind astfel că P cunoaşte $sqrt(x)$. Dacă $b = 1$, V va verifica dacă $x = y^2 \cdot v \bmod n$, dovedind astfel că P cunoaşte $sqrt(v^{-1})$.

Prin executarea paşiilor $1-4$ s-a realizat *acreditarea*. P şi V repetă acest protocol de t ori până când V este convins că P cunoaşte pe s. Acest protocol face parte din categoria protocoalelor de tip *taie şi alege*. Protocolul prezentat este de tip dezvăluire zero a identităţii.

Dacă P nu ştie pe s, atunci şansele ca acesta să trişeze pe Vsunt de $\frac{1}{2^t}$.

O metodă de atac pentru V este prin impersonalizarea acestuia cu V'. Sansele de reuşită ale astui atac sunt de $\frac{1}{2^t}$.

Pentru ca acest protocol să funcţioneze trebuie ca P să nu reutilizeze pe r.

Guillou-Quiquater

Protocolul Feige-Fiat-Scamir a fost primul protocol de tip cunoaştere zero a identităţii ce a fost implementat. Securitatea sa creşte direct proporţional cu numărul de iteraţii efectuate. Pentru o serie de implementări, cum ar fi smart cardurile, acest lucru este foarte bun: schimbul de date cu lumea exterioară nu se face instantaneu iar capacităţile de stocare pentru fiecare acreditare pot atinge limita de memorie din card.

Louis Guillou şi *Jean-Jaques Quisquater* au dezvoltat un protocol de dezvăluire zero a identităţii care este pretabil la aplicaţii mult mai complexe. Pentru fixarea ideilor să presupunem că P este un smart card care doreşte să-şi dovedească identitatea lui V. Identitatea lui P constă dintr-o mulţime de date cum ar fi: un şir de date care costă din numele cardului, perioada de valabilitate, numărul contului bancar, etc. Vom nota acest şir binar prin J (de fapt şirul de identificare fiind foarte lung este trecut print-o funcţie hash, acest lucru neafectând protocolul în nici un fel). Acesta este similar cheii publice. Altă informaţie publică, care este cunoscută de tate smard cardurile P este exponentul v şi modulul n produs de două numere prime secrete foarte mari. Cheia privată B este calculată după formula:

$$JB^v \equiv 1 \bmod n.$$

Smart cardul P trimite lui V secvenţa J. Acum P doreşte să dovedească lui V că secvenţa J este a sa. Pentru a realiza acest lucru trebuie să-i dovedească lui V că deţine secvenţa privată B. Pseudocodul schemei de identificare *Guillou-Quiquater* este următorul:

PAS 1. P va alege un număr aleatoriu $r \in [1, n-1]$. Se calculează secvenţa $T = r^v \bmod n$ care se transmite la V.

PAS 2. V alege un număr aleatoriu $d \in [0, v-1]$ pe care îl trimite lui P.

PAS 3. P calculează $D = rB^d \bmod n$ pe care îl trimite lui V.

PAS 4. V calculează $T' = D^v J^d \bmod n$. Dacă $T \equiv T'(\bmod n)$ atunci autentificarea este realizată.

De fapt calculele nu sunt complicate:

$$T' = D^v J^d = (rB^d)^v J^d = r^v B^{dv} J^d = r^v (JB^v)^d \bmod n$$

deoarece B a fost astfel construit astfel ca

$$JB^v \equiv 1 \bmod n.$$

Schnorr

Securitatea schemei de autentificate propusă de Claus Schnorr se bazează pe dificultatea calculului problemei logaritmului discret. Pentru a genera o pereche de chei vom proceda după cum urmează: alegem două numere prime p şi q astfel ca $q|p-1$, vom alege apoi un număr $a \neq 1$ astfel ca $a^q \equiv 1 \bmod p$. Aceste numere pot fi cunoscute de un grupt de utilizatori sau se pot face publice.

Pentru generarea unei perechi de chei publice/private se va alege un număr aleatoriu s (cheia privată) mai mic decât q. Se va calcula apoi cheia publică $v = a^{-s} \bmod p$. Pseudocodul schemei de identificare *Schnorr* este următorul:

PAS 1. P alege un număr aleatoriu $r < q$ şi calculează $x = a^r \bmod p$. Acesta este un pas de preprocesare şi se poate efectua înainte de prezenţa lui V.

PAS 2. P trimite x lui V.

PAS 3. V trimite lui P un număr aleatoriu $e \in [0, 2^{t-1}]$ (t este un parametru de securitate)

PAS 4. P calculează $y = (r + se) \bmod q$ pe care îl trimite lui V.

PAS 5. V verifică faptul că $x = a^y v^e \bmod p$.

Observaţia 11.4.1. Dificultatea spargerii algoritmului este de $O(2^t)$. Schnorr recomandă ca p să fie de minimum 512 biţi, q aproximativ 140 biţi iar t să fie 72.

Conversia schemelor de identificare în scheme de semnătură digitală

Schemele de identificare ce au fost prezentate se pot converti în scheme de semnătură digitală după următorul procedeu: se va înlocui V cu o funcţie hash, mesajul nu este trecut prin funcţia hash înainte de a fi semnat de fapt funcţia hash este încorporată algoritmului de semnătură. În principiu acest lucru se poate realiza cu orice schemă de identificare.

11.4.2. Algoritmi de schimb al cheilor

Vom prezenta în cele ce urmează o serie de algoritmi de schimb al cheilor.

Diffie-Hellman

Diffie-Hellman este primul algoritm de schimb al cheilor care are la bază conceptul de cheie publică. Acesta a fost formulat în anul 1976 de către Diffie şi Hellman. Securitatea acestui algoritm se bazează pe dificultatea calculului logaritmului discret într-un corp finit (această dificultate computaţională este similară cu dificultatea operaţiei de exponenţiere). Utilizatorii notaţi generic cu A şi B vor folosi algoritmul pentru generarea unei chei secrete de comunicaţie. Pseudocodul algoritmului Diffie-Hellman este următorul:

PAS 0. A şi B stabilesc, de comun acord pe un canal public, un număr mare n (astfel ca $\frac{n-1}{2}$ să fie număr prim) şi aleg g primitiv $\bmod n$ (n şi g pot fi folosiţi chiar de un grup de utilizatori).

PAS 1. Utilizatorul A alege aleatoriu numărul x suficient de mare şi va trimite lui B :

$$X = g^x \bmod n.$$

PAS 2. Utilizatorul B alege aleatoriu numărul y suficient de mare şi va trimite lui A :

$$Y = g^y \bmod n.$$

PAS 3. A va calcula:

$$k = Y^x \bmod n.$$

PAS 4. B va calcula:

$$k' = X^y \bmod n.$$

Ambele valori ale lui k şi $k^{'}$ sunt egale cu $g^{xy} \bmod n$, iar acestea nu se pot recupera din n, g, X şi Y fără a rezolva problema logaritmului discret. Deci k este secretă şi poate fi folosită ca cheie de comunicaţie.

Observaţia 11.4.2. Algoritmul Diffie-Helman se poate modifica pentru stabilirea unei chei secrete pentru trei sau mai mulţi utilizatori.

Observaţia 11.4.3. Generatorul g nu trebuie să fie neapărat primitiv ci trebuie să genereze un subgrup multiplicativ suficient mare.

Observaţia 11.4.4. Algoritmul Diffie-Hellman poate funcţiona în inele comutative, se poate extinde la curbele eliptice (vezi *N. Koblitz* [39]) şi se poate utiliza ca algoritm de semnătură digitală (vezi *ElGamal* [18]).

Protocolul staţie la staţie

Deoarece algoritmul Diffie-Hellman este vulnerabil la atacul prin a treia persoană, utilizatorii A şi B trebuie să semneze reciproc mesajele. Protocolul ce urmează presupune faptul că utilizatorul A are *un certificat al cheii publice* a lui B şi viceversa. Aceste certificate au fost semnate de către o terţă entitate considerată de încredere. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A generează un număr x pe care îl trimite lui B.

PAS 2. Utilizatorul B va genera un număr y. Folosind protocolul Diffie-Hellman utilizatorii calculează cheia secretă comună bazându-se pe valorile lui x şi y. Utilizatorul B va semna electronic valorile x şi y iar apoi va cifra semnătura cu ajutorul cheii k. Utilizatorul B va trimite lui A mesajul:

$$y, E_k[S_B(x, y)].$$

PAS 3. Utilizatorul A calculează deasemenea pe k, şi va descifra mesajul primit de la B verificând totodată semnătura acestuia. Utilizatorul A va trimite lui B un mesaj semnat:

$$E_k[S_A(x, y)].$$

PAS 4. Utilizatorul B va descifra mesajul şi va verifica semnătura lui A.

Protocolul celor trei iteraţii al lui Shamir

Acest protocol a fost inventat de *Adi Shamir* (nu a fost publicat niciodată) şi permite ca utilizatorii A şi B să comunice privat fără ca, apriori, să fi avut loc un schimb de chei private sau secrete. Protocolul presupune existenţa unui sistem de cifrare care este *comutativ:*

$$E_A[E_B[P]] = E_B[E_A[P]],$$

cheia secretă a lui A fiind notată cu A iar a lui B cu B. Să presupunem că utilizatorul A doreşte să trimită un mesaj M lui B. Protocolul decurge după cum urmează:

PAS 1. Utilizatorul A cifrează mesajul M cu cheia sa pe care îl trimite lui B :

$$C_1 = E_A[M].$$

PAS 2. Utilizatorul B cifrează mesajul primit cu cheia sa pe care apoi îl va trimite lui A :

$$C_2 = E_B[E_A[M]].$$

PAS 3. Utilizatorul A va descifra mesajul primit după care îl va trimite lui B:

$$D_A[E_B[E_A[M]] = D_A[E_A[E_B[M]] = E_B[M].$$

PAS 4. Utilizatorul B va descifra mesajul primit recuperând astfel pe M.

Observaţia 11.4.5. Operaţia de cifrare prin sumare modulo 2 este comutativă dar nu poate funcţiona în cadrul acestui protocol.

Observaţia 11.4.6. Protocolul celor trei iteraţii nu rezistă la un atac prin a treia persoană.

Un exemplu de cifru care este pretabil acestui protocol este următorul: fie p un număr prim suficient de mare astfel ca cel puţin un factor al lui $p-1$ să fie mare. Se alege un factor de cifrare e realtiv prim cu $p-1$. Se va calcula apoi d inversul multiplicativ al lui $e \bmod(p-1)$. Operaţia de cifrare va fi:

$$C = M^e \bmod p,$$

iar operaţia de descifrare va fi

$$M = C^d \bmod p.$$

Interceptorul pasiv E nu poate recupera mesajul M fără a rezolva problema logaritmului discret.

Transferul cifrat al cheii

Protocolul de transfer cifrat al cheii, cunoscut sub acronimul EKE, proiectat de *Steve Bellovin* şi *Michael Merritt*, asigură securitate şi autentificare în cadrul unei reţele de comunicaţii prin folosirea unei tehnici noi de utilizare a criptografiei cu chei secrete şi a criptografiei cu chei publice: o cheie secretă comună este folosită pentru a cifra o cheie publică generată aleatoriu.

Utilizatorii A şi B deţin o cheie comună (parolă) notată prin P. Folosirea protocolului permite *autentificarea reciprocă* şi *stabilirea unei chei secrete de sesiune* K :

PAS 1. Utilizatorul A generează aleatoriu o pereche de chei publică/privată. Acesta va cifra cheia publică $K^{'}$ cu ajutorul unui algoritm simetric şi a cheii P. Va trimite apoi rezultatul lui B :

$$A, E_P[K^{'}].$$

PAS 2. Utilizatorul B va recupera cheia $K^{'}$, va genera o cheie secretă de sesiune K pe care o va cifra cu ajutorul cheii publice primite anterior şi a cheii secrete P. Va trimite lui A mesajul:

$$E_P[E_{K'}[K]].$$

PAS 3. Utilizatorul A va recupera cheia K, va genera o secvenţă aleatoare R_A pe care o va cifra cu ajutorul cheii secrete de sesiune K. Va trimite lui B mesajul:

$$E_K[R_A].$$

PAS 4. Utilizatorul B recuperează pe R_A, va genera o secvenţă aleatoare R_B şi va trimite lui A mesajul:

$$E_K[R_A, R_B].$$

PAS 5. Utilizatorul A descifrează mesajul, confirmă valoarea lui R_A şi va trimite lui B mesajul cifrat:

$$E_K[R_B].$$

PAS 6. Utilizatorul B va confirma, după descifrarea mesajului, valoarea lui R_B. În acest moment protocolul se încheie iar ambele părţi comunica prin intermediul cheii secrete de sesiune K.

Observaţia 11.4.7. Protocolul EKE se poate implementa efectiv cu ajutorul algoritmilor de cifrare cu cheie publică RSA, ElGamal sau Diffie-Hellman.

Schemă îmbunătăţită de negociere a cheilor

Această schemă este folosită ca metodă de protecţie a schemelor de identificare pentru care parolele utilizate sunt slabe (cuvinte din dicţionar sau parole scurte, etc.) sau împotriva atacului prin cea de a treia persoană. Se foloseşte o funcţie hash de două variabile cu următoarea proprietate de coliziune pentru prima variabilă şi anticoliziune pentru cea de a doua variabilă:

$$H^{'}(x, y) = H(H(k, x) \bmod 2^m, x)$$

unde $H(k, x)$ este o funcţie arbitrară de variabile k şi x.

Se presupune că utilizatorii A şi B cunosc o parolă P şi au rulat un protocol (Diffie-Hellman) de schimb de chei secrete K. Utilizatorii folosesc parola P pentru a verifica dacă cele două chei de sesiune sunt identice (deci interceptorul nu poate practica un atac activ). Pseudocodul protocolului este următorul:

PAS 1. Utilizatorul A trimite lui B :

$$H^{'}[P, K].$$

PAS 2. Utilizatorul B calculează $H^{'}[P, K]$ şi compară cu ceea ce a primit de la A. Dacă totul este în regulă atunci acesta trimite lui A mesajul:

$$H^{'}[H[P, K]].$$

PAS 3. Utilizatorul A calculează $H^{'}[H[P, K]]$ şi compară cu ceea ce a primit de la utilizatorul B.

11.5. Aplicaţii

Exerciţiul 11.5.1. Elaboraţi o metodă de atac prin a treia persoană asupra protocolului Diffie-Hellman.

Exerciţiul 11.5.2. Studiaţi folosirea operaţia de cifrare prin $XOR-$are (care este comutativă) ca algoritm de cifrare în cadrul protocolului celor trei iteraţii al lui Shamir.

Exerciţiul 11.5.3. Studiaţi rezistenţa la atacul prin a treia persoană a protocolului celor trei iteraţii al lui Shamir.

Exerciţiul 11.5.4. Studiaţi rezistenţa la atacul prin a treia persoană a schemei îmbunătăţite de negociere a cheilor.

Exerciţiul 11.5.5. Studiaţi rezistenţa la atacul prin a treia persoană a protocoalelor $SKID2$ şi $SKID3$.

Exerciţiul 11.5.6. Care sunt punctele slabe ale protocolului Kerberos?

Capitolul 12

CRIPTANALIZA SISTEMELOR DE CIFRARE ANALOGICE

Protection of sensitive information is a desire reaching back to the beginnings of human culture.
Otto Horak, 1994.

12.1. Formularea problemei

Sistemele de comunicaţii se împart în sisteme de comunicaţii digitale în care informaţia este reprezentată sub formă de semnal discret (de exemplu semnalul binar) şi sisteme de comunicaţii analogice în care elementul purtător de informaţie este reprezentat de noţiunea de semnal continuu. Un exemplu de semnal analogic este semnalul vocal (cu banda de frecvenţă între 300-3000Hz).

12.2. Calcul Operaţional

Transformata Laplace realizează o trecere a semnalului din domeniul timp în *domeniul complex* iar transformata Fourier a unui semnal realizează o trecere din domeniul timp în *domeniul frecvenţă.* În cele ce urmează vom face o prezentare a celor două trasformate precum şi a principalelor proprietăţi ale acestora. Se vor prezenta de asemenea trasformatele discrete z, Fourier (TFD), cos (TCD) şi Walsh.

12.2.1. Transformata Laplace

Vom da, în cele ce urmează, o serie de definiţii referitoare la transformata Laplace şi principalele ei caracterizări.

Definiţia 12.2.1. Se numeşte funcţie *original Laplace* orice funcţie $f : \mathbf{R} \to \mathbf{C}$ având următoarele trei proprietăţi:
i) $f(t) = 0$ pentru orice $t < 0$;
ii) f este continuă pe porţiuni pe $\mathbf{R}$;
iii) există $M > 0$ şi $s_0 \geq 0$ (numit indice de creştere) astfel încât:

$$|f(t)| \leq Me^{s_0 t}, \text{ pentru orice } t \geq 0.$$

Mulţimea funcţiilor original Laplace se va nota prin $\mathcal{O}$.

Definiţia 12.2.2. Fie $f : \mathbf{R} \to \mathbf{C}$ o funcţie original Laplace cu indicele de creştere s_0, atunci funcţia complexă $F : S(s_0) \to \mathbf{C}$,

$$F(s) = \int_0^\infty f(t)e^{-st}dt$$

se numeşte *transformata Laplace* (sau imaginea) originalului f, $S(s_0) = \{s \in \mathbf{C} | \operatorname{Re}(s) > s_0\}$. Acest lucru se va nota prin $F(s) = \mathcal{L}\{f(t)\}$.

Definiţia 12.2.3. Fie $f, g : \mathbf{R} \to \mathbf{C}$ atunci vom defini convoluţia lui f cu g prin formula:

$$(f * g)(t) = \int_{-\infty}^\infty f(z)g(t-z)dz.$$

Vom da o serie proprietăţi ale trasformatei Laplace.

1. *Proprietatea asemănării*: Dacă $f \in \mathcal{O}$ şi $F(s) = \mathcal{L}\{f(t)\}$ atunci pentru orice $a > 0$ avem:

$$\mathcal{L}\{f(at)\} = \frac{1}{a}F(\frac{s}{a}).$$

2. *Proprietatea deplasării*: Dacă $f \in \mathcal{O}$ şi $F(s) = \mathcal{L}\{f(t)\}$ atunci pentru orice $\alpha \in \mathbf{C}$ avem:

$$\mathcal{L}\{f(t)e^{\alpha t}\} = F(s - \alpha).$$

3. *Proprietatea întârzierii*: Dacă $f \in \mathcal{O}$ şi $F(s) = \mathcal{L}\{f(t)\}$. Pentru orice $\tau > 0$ se defineşte funcţia $f_\tau : \mathbf{R} \to \mathbf{C}$ prin

$$f_\tau(t) = \begin{cases} 0 \text{ pentru } t < \tau \\ f(t-\tau) \text{ pentru } t \geq \tau \end{cases}$$

Atunci

$$\mathcal{L}\{f_\tau(t)\} = e^{-s\tau}F(s).$$

4. *Derivarea originalului*: Dacă $f \in \mathcal{O}$ are indicele de creştere s_0 şi dacă f este derivabilă a.p.t. cu $f^{'} \in \mathcal{O}$, atunci pentru $\mathrm{Re}(s) > s_0$

$$\mathcal{L}\{f^{'}(t)\} = sF(s) - f(0_+).$$

5. *Integrarea originalului*: Dacă $f \in \mathcal{O}$ şi $g(t) = \int\limits_0^t f(\tau)d\tau$ şi $g(0_+) = 0$, atunci

$$G(s) = \frac{1}{s}F(s).$$

6. *Derivarea imaginii*: Dacă $f \in \mathcal{O}$, atunci $t^n f(t) \in \mathcal{O}$ şi

$$\mathcal{L}\{t^n f(t)\} = (-1)^n F^{(n)}(s).$$

7. *Trasformata convoluţiei*: Dacă $f, g \in \mathcal{O}$, atunci

$$\mathcal{L}\{f * g\} = \mathcal{L}\{f(t)\} \cdot \mathcal{L}\{g(t)\}.$$

8. *Trasformata derivatei convoluţiei*: Dacă $f, g \in \mathcal{O}$ cu g derivabilă a.p.t. şi $g^{'} \in \mathcal{O}$ atunci

$$\mathcal{L}\{(f * g)'\} = s\mathcal{L}\{f(t)\} \cdot \mathcal{L}\{g(t)\}.$$

12.2.2. Transformata Fourier

Trasformata Fourier $\hat{f}$ se defineşte pentru funcţiile din $L^1(\mathbf{R}) = \{f$ măsurabilă şi $\int\limits_{-\infty}^{\infty} |f(t)|dt < \infty\}$ relativ la norma $||f||_1 = \int\limits_{-\infty}^{\infty} |f(t)|dt$ prin formula

$$\mathcal{F}\{f(t)\}(\omega) = \hat{f}(\omega) = \int\limits_{-\infty}^{\infty} f(t)e^{-i\omega t}dt.$$

Vom da o serie proprietăţi ale trasformatei Fourier:

1. Dacă $f \in L^1(\mathbf{R})$, atunci $\hat{f}: \mathbf{R} \to \mathbf{C}$ este o funcţie mărginită pe $\mathbf{R}$.

2. Operatorul Fourier $\mathcal{F} : L^1(\mathbf{R}) \to C^0(\mathbf{R}) \wedge M_{\mathbf{R}}$ este liniar şi continuu.

3. Dacă $f \in L^1(\mathbf{R})$ atunci $\lim_{|\omega|\to\infty} \hat{f}(\omega) = 0$.

4. Dacă $f, g \in L^1(\mathbf{R})$ atunci $f \hat{g}$ şi $\hat{f}\, g$ aparţin clasei $L^1(\mathbf{R})$ şi în plus

$$\int f(t) \hat{g}(t)dt = \int \hat{f}(t)g(t)dt.$$

5. *Derivarea originalului*: Dacă $f \in C^n(\mathbf{R}), n \geq 1$ şi $f^{(k)} \in L^1(\mathbf{R})$ cu $\lim_{|t|\to\infty} f^{(k)}(t) = 0$ pentru orice $0 \leq k \leq n$, atunci pentru orice $\omega \in \mathbf{R}$:

$$\mathcal{F}\{f^{(k)}(t)\}(\omega) = (\overset{\wedge}{f^{(k)}})(\omega) = (i\omega)^k \hat{f}(\omega),\ 0 \leq k \leq n.$$

6. *Derivarea imaginii*: Dacă $t^k f(t) \in L^1(\mathbf{R})$ atunci $\mathcal{F} = \hat{f} \in C^n(\mathbf{R})$ şi în plus pentru orice $\omega \in \mathbf{R}$:

$$(\hat{f})^{(k)}(\omega) = \mathcal{F}\{(-it)^k f(t)\}(\omega),\ 0 \leq k \leq n.$$

7. *Paritate şi imparitate*: i) $\mathcal{F}\{f(-t)\}(\omega) = \hat{f}(-\omega)$ pentru orice $\omega \in \mathbf{R}$.

ii) dacă f este pară/impară atunci $\hat{f}$ este pară/impară;

iii) dacă f este pară şi are avalori reale atunci $\hat{f}$ este la fel; dacă f este impară şi are valori reale, atunci $\hat{f}$ este impară cu valori pur imaginare.

8. *Proprietatea întârzierii*: Dacă $f \in L^1(\mathbf{R})$ şi $\tau \in \mathbf{R}$, fie $T_\tau f = f_\tau$ întârziata lui f cu τ deci pentru orice $t \in \mathbf{R}$, $f\tau(t) = f(t - \tau)$. Avem

$$\mathcal{F}\{f(t - \tau\} = (\overset{\wedge}{T_\tau}\, f)(\omega) = e^{-i\omega\tau} \hat{f}(\omega),$$

şi

$$T_\tau(\hat{f}) = \mathcal{F}\{f(t)e^{it\tau}\}.$$

9. *Proprietatea asemănării*: Dacă $f \in L^1(\mathbf{R})$ şi $a \in \mathbf{R}$, atunci

$$\hat{f}(at)(\omega) = \frac{1}{a} \hat{f}(\frac{\omega}{a}) \text{ pentru orice } \omega \in \mathbf{R}.$$

10. *Inversa transformatei Fourier*: Dacă $f : \mathbf{R} \to \mathbf{C}$ este continuă, derivabilă pe porţiuni şi f, $\hat{f}$aparţin clasei $L^1(\mathbf{R})$ atunci pentru orice $t \in \mathbf{R}$, are loc formula de inversare:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega)e^{it\omega}d\omega.$$

11. *Legătura dintre transformata Laplace şi transformata Fourier*: Dacă $f : \mathbf{R} \to \mathbf{C}$ este o funcţie din clasa $\mathcal{O}$ cu indicele de creştere $s_0 \geq 0$ atunci pentru orice $\sigma > s_0$ fixat avem:

$$\mathcal{L}\{f(t)\} = \mathcal{F}\{f(t)e^{-\sigma t}\}.$$

12. *Inversa transformatei Laplace*: Dacă $f : \mathbf{R} \to \mathbf{C}$ este o funcţie din clasa $\mathcal{O}$ cu indicele de creştere $s_0 \geq 0$ atunci pentru orice $\sigma > s_0$ fixat şi pentru orice $t \in \mathbf{R}$ avem:

$$f(t) = \frac{1}{2\pi i} \int_{\sigma - i\infty}^{\sigma + i\infty} F(s)e^{st} ds.$$

13. *Teorema lui Heaveside*: Dacă $f \in \mathcal{O}$ este continuă, cu indicele de creştere $s_0 \geq 0$ atunci pentru orice $\sigma > s_0$ fixat $\lim\limits_{\substack{R \to \infty \\ S \in \Gamma}} sF(s)e^{st} = 0$, atunci pentru orice $t \in \mathbf{R}$, avem

$$f(t) = \sum rez(F(s)e^{st}),$$

unde suma este luată după toate singularităţile lui $F(s) = \mathcal{L}\{f(t)\}$ la distanţă finită, presupuse în număr finit.

14. *Transformata convoluţiei*: Dacă $f, g : \mathbf{R} \to \mathbf{C}$ sunt din clasa $L^1(\mathbf{R})$, atunci:

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\}\mathcal{F}\{g\}.$$

12.2.3. Transformata Fourier Discretă

Multe tehnici de prelucrare a semnalelor discrete provin din adaptarea unor tehnici aplicate la semnale continuale, dar există şi multe abordări specifice.

Un semnal discret este un şir $x = (x_n), n \in \mathbf{Z}$ de numere reale sau complexe. Se mai scrie $x[n]$ în loc de x_n. Am notat cu S_d mulţimea acestor semnale. O subclasă importantă este $S_+ = \{x \in S_d | x_n = 0 \text{ pentru } n < 0\}$, numite semnale discrete cu suport pozitiv. Prelucrarea semnalelor discret are multe tangenţe comune cu prelucrarea datelor şi cu studiul complexităţii algoritmilor.

Fixăm $N \geq 2$ întreg. Vom numi *semnal finit de lungime* N orice şir de N numere (complexe) $x = (x_0, ..., x_{N-1}) \in \mathbf{C}^N$, identificat cu vectorul coloană $X = (x_0, ..., x_{N-1})^t$. Un astfel de şir poate fi prelungit prin periodicitate de perioadă N la un semnal din S_d.

Energia lui x este

$$E(x) = \sqrt{\sum_{n=0}^{N-1} |x_n|^2}.$$

Vom nota $v = \exp(-\frac{2\pi i}{N})$ şi vom construi matricea $W = (v^{kn})$, $0 \leq k, n \leq N-1$. Matricea W este inversabilă cu inversa $W^{-1} = \frac{1}{N}\ \bar{W}$. Evident W este o matrice simetrică, de tip Vandermonde, cu toate elementele de modul 1.

Definiţia 12.2.4. Dacă $x = (x_n)$, $0 \leq n \leq N-1$ este un semnal finit de lungime N, se numeşte *transformarea Fourier discretă* (TFD) a lui x şirul de numere complexe $f = (f_k)$ definit prin:

$$f_k = \sum_{n=0}^{N-1} x_n v^{kn}, 0 \leq k \leq N-1.$$

Se mai scrie $f = TFDx$.

Principalele proprietăţi ale TFD sunt următoarele:

1. *Scrierea matriceală a transformatei Fourier* este:

$$TFDx = W \cdot X.$$

2. *Formula de inversare a TFD* este:

$$X = \frac{1}{N}\ \bar{W} \cdot F$$

3. *Transformata Fourier este liniară şi bijectivă.*
4. Dacă x este par ($x_{-k} = x_k$), la fel este $f = TFDx$.
5. Dacă x are toate componentele reale, atunci $f_{-k} = \bar{f}_k$ pentru orice k.
6. *Relaţia lui Parseval*: Dacă $f = TFDx$, atunci

$$E(f) = \sqrt{N}E(x).$$

7. Dacă x şi y sunt semnale finite de lungime N, atunci:

$$TFD(x * y) = TFDx \bullet TFDy$$

şi

$$TFD(x \bullet y) = \frac{1}{N}(TFDx * TFDy)$$

unde $x \bullet y = (x_k \cdot y_k)$ este produsul pe componente şi $x * y = (\sum_{i=0}^{N-1} x_i y_{n-i})_{n=0,\ldots,N-1}$ este convoluţia discretă.

Observaţia 12.2.1. Calculul direct al TFD necesită $(N-1)^2$ înmulţiri complexe şi $N(N-1)$ adunări de numere complexe, presupunând că puterile lui v^{kn} sunt stocate.

Observaţia 12.2.2. Matematicienii americani Cooley şi Tukey au elaborat algoritmul TFR (transformata Fourier rapidă), care exploatează structura specială a matricei W şi reduce drastic numărul de operaţii ajungând la $N \log_2 N$ înmulţiri.

12.2.4. Transformata Cosinus Discretă

Definiţia 12.2.5. Dacă $x = (x_0, ..., x_{N-1})$ este un semnal finit dat de lungime N, se numeşte transformarea cos-discretă a lui x, şirul de numere reale $c = (c_k), 0 \leq k \leq N-1$ definit prin:

$$c_0 = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n,$$

$$c_k = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x_n \cos \frac{(2n+1)k\pi}{2N}.$$

pentru $1 \leq k \leq N-1$.

Inversa transformatei cos discretă:

$$x_n = \frac{1}{\sqrt{N}} c_0 + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} c_k \cos \frac{(2n+1)k\pi}{2N},\ 0 \leq n \leq N-1.$$

Observaţia 12.2.3. Transformata cos discretă se foloseşte în aplicaţiile de tip imagine (compresia imaginilor, de exemplu formatul *jpeg*).

12.2.5. Transformata Walsh

Presupunem $N = 2^n$ $(n \geq 1)$. Pentru orice $0 \leq t, k \leq N-1$, se consideră scrierile în baza 2, unice şi de lungime n :

$$t = (t_{n-1}...t_1t_0)_2 = t_{n-1} + 2^{n-1} + ... + t_1 2 + t_0$$

$$k = (k_{n-1}...k_1k_0)_2.$$

Apoi pentru $1 \leq p \leq n$, definim:

$$A_p = \begin{cases} k_p + k_{p-1} \bmod 2 \text{ pentru } 1 \leq p \leq n-1 \\ k_{n-1} \text{ pentru } p = n \end{cases}$$

şi vectorii coloană $N-$dimensionali $W_k = (w_{0k}, w_{1k}, ..., w_{N-1,k})^T$, unde $w_{tk} = (-1)^B$, $B = \sum_{p=1}^{n} A_p t_{n-p}$. Cu aceşti vectori se poate considera matricea

$$M = (W_0|W_1|...|W_{N-1}|)$$

de tip $N \times N$, simetrică, nesingulară, cu toate elementele egale cu ± 1. În plus, pentru orice $0 \leq r, s \leq N-1$ avem $W_r W_s = W_{r \oplus s}$ şi dacă $r \neq s < W_r, W_s >= 0$. Vectorii W_k, $0 \leq k \leq N-1$ formează o bază, numită baza Walsh, pentru spaţiul vectorial $\mathbf{R}^N$, ca şi pentru spaţiul vectorial complex $\mathbf{C}^N$. Pentru orice semnal finit $X \in M_{n,1}$ de lungime N, *transformata Walsh* este $W = M \cdot X$, cu inversa $X = M^{-1} \cdot W$. În acest caz se foloseşte termenul de *secvenţă* în loc de *frecvenţă.* Avem

$$\int_0^1 W_k(t) W_l(t) dt = \delta_{kl}$$

pentru orice k, l şi orice semnal $f = (f_0, ..., f_{N-1})$, finit de lungime N se scrie unic

$$f = \sum_{k=0}^{N-1} \alpha_k \cdot W_k \text{ cu } \alpha_k \in \mathbf{C}.$$

Transformata Walsh a lui f este $M \cdot f^T$.

12.2.6. Transformata z

Pentru orice semnal discret x=(x_n), $n \in \mathbf{Z}$ se numeşte $z-$transformata lui x funcţia complexă

$$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n} \text{ (suma seriei Laurent).}$$

Domeniul de convergenţă este fie mulţimea vidă, fie o coroană circulară $K(0; r; R) = \{r < |z| < R\}$ centrată în origine. Se mai notează $X(z) = L_x(z)$.

Principalele proprietăţi ale transformatei Z sunt:

1. *Liniaritatea*: dacă $x, y \in S_d$ şi $\alpha \in \mathbf{R}$,atunci:

$$L_{x*y}(z) = L_x(z) + L_y(z), \; L_{\alpha x}(z) = \alpha L_x(z).$$

2. *Transformata convoluţiei*: dacă $x, y \in S_d$ au domeniu nevid de convergenţă pentru $z-$transformatele lor şi dacă există $x * y$, atunci

$$L_{x*y} = L_x \cdot L_y.$$

3. *Transformata treptei unitate discrete* u este

$$U(z) = \frac{z}{z-1}.$$

4. *Transformata* lui δ_k este z^{-k}.

5. Dacă $x \in S_d$, atunci $x * \delta_k$ ($k \in \mathbf{Z}$) este întârziatul lui x cu k unităţi şi

$$L_{x*\delta_k} = z^{-k}X(z).$$

6. *Inversa* $z-$*transformării* este

$$x_n = \sum \text{res}(z^{n-1}X(z)),$$

suma fiind luată după toate singularităţile lui $z^{n-1}X(z)$ la distanţă finită.

7. Dacă $X(z) = L_x(z)$, $Y(z) = L_y(z)$ şi $x \bullet y = (x_n y_n)$, atunci $L_{x\bullet y}(z)$ este suma reziduriloor funcţiei complexe

$$\frac{1}{u}X(\frac{z}{u})Y(u).$$

(în singularităţile aflate la distanţă finită).

12.3. Caracterizarea variabilelor aleatoare

Vom defini în cele ce urmează o serie de caracteristici numerice ale unei variabile aleatoare $\xi(t)$.

Funcţia de repartiţie a variabilei $\xi(t)$:

$$F(x;t) = \Pr(\xi(t) \leq x).$$

Densitatea de probabilitate:

$$\rho(x;t) = \frac{\partial F(x;t)}{\partial x} \text{ (dacă există)}.$$

Valoarea medie:

$$M(\xi(t)) = \int_{-\infty}^{\infty} x\rho(x;t)dx.$$

Valoarea medie pătratică:

$$M(\xi^2(t)) = \int_{-\infty}^{\infty} x^2\rho(x;t)dx.$$

Dispersia:

$$\begin{aligned} D^2(\xi(t)) &= M((M(\xi(t)) - \xi(t))^2) \\ &= M(\xi^2(t)) - M^2(\xi(t)). \end{aligned}$$

Funcţia de autocovarianţă:

$$B_\xi(t_1, t_2) = E((M(\xi(t_1)) - \xi(t_1))(M(\xi(t_2)) - \xi(t_2))).$$

Funcţia de autocorelaţie:

$$C_\xi(t_2 - t_1) = \int\int x_1 x_2 \rho(x_1, x_2; t_1, t_2) dx_1 dx_2.$$

Densitatea spectrală de putere:

$$\begin{aligned} q(\omega) &= \int_{-\infty}^{\infty} C_\xi(\tau) e^{-i\omega\tau} d\tau \\ &= \mathcal{F}(C_\xi(\tau)). \end{aligned}$$

12.4. Conversia Analogic/Digital

Vom prezenta o serie de tehnici de conversie analogic/digital: modulaţia în puls (pulse code modulation), modulaţia liniară şi vocoderul.

12.4.1. Modulaţia în puls

În cazul modulaţiei puls semnalul este convertit într-o serie de pulsuri. Semnalul este eşantionat cu o anume frecvenţă iar valoarea fiecărui eşantion este folosită pentru a determina o proprietate (de exemplu amplitudinea sau durata) a pulsului corespunzător. Problema ce se pune este aceea a frecvenţei minime de eşantionare necesare pentru recuperea din semnalul eşantionat a semnalul original. Numărul de eşantioane pe secundă se numeşte *rata de eşantionare*. Teorema următoare dă informaţii despre frecvenţa minimă de eşantionare.

Teorema 12.4.1. Fie $f(t)$ un semnal cu lungimea de bandă L determinat de frecvenţele f_1 şi f_2 $(f_2 > f_1)$. Rata minimă de eşantionare pentru caracterizarea semnalului este de $\dfrac{2f_2}{m}$ eşantioane pe secundă, unde $m = \left[\dfrac{f_2}{L}\right]$.

12.5. Cifrarea Analogică

Cifrarea analogică a unui semnal presupune modificarea frecvenţei semnalului respectiv. Vom reduce studiul sistemelor de cifrare analogică la: inversorul de spectru, rotirea benzilor inversorului, amestecarea benzilor şi multiplexarea în timp. În cele ce urmează vom presupune, dacă nu se specifică altfel, că semnalul de intrare este un semnal vocal cu lungimea de bandă $300 - 3000Hz$.

12.5.1. Inversarea spectrului

Una dintre cele mai simple metode de securizare a semnalului vocal este tehnica inversării spectrului care constă în interschimbarea frecvenţelor înalte cu frecvenţele joase. Să considerăm un semnal cu banda de frecvenţe între $300Hz - 3000Hz$ (vezi figura 12.1). Pentru fixarea ideilor să considerăm o singură componentă a semnalului sinusoidal: $V_m \cos(\omega_m t)$. Realizarea inversării spectrului se realizează prin intermediul unui *semnal purtător* $V_c \cos(\omega_c t)$ cu ajutorul unui *modulator echilibrat*:

$$V_m V_c \cos(\omega_m t)\cos(\omega_c t) = \frac{1}{2} V_m V_c \cos(\omega_m + \omega_c)t + \frac{1}{2} V_m V_c \cos(\omega_m - \omega_c)t.$$

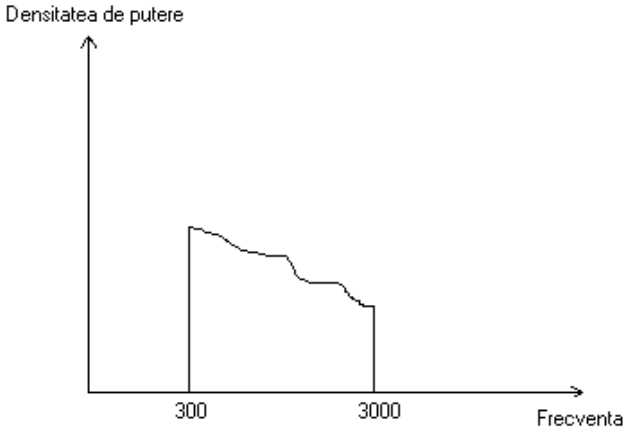

Figura 12.1: Semnal vocal limitat la banda $300 - 3000Hz$.

Vom alege $V_c = 1$ şi $\omega_m < \omega_c$. Prin aplicarea modulatorului echilibrat fiecărei componente sinusoidale vom obţine desitatea de putere din figura 12.2.

Cele două benzi ale semnalului se numesc partea superioară şi partea inferioară a semnalului. Valoarea $f_c = \dfrac{\omega_c}{2\pi}$ se numeşte *frecvenţa purtătoarei*. Remarcăm faptul

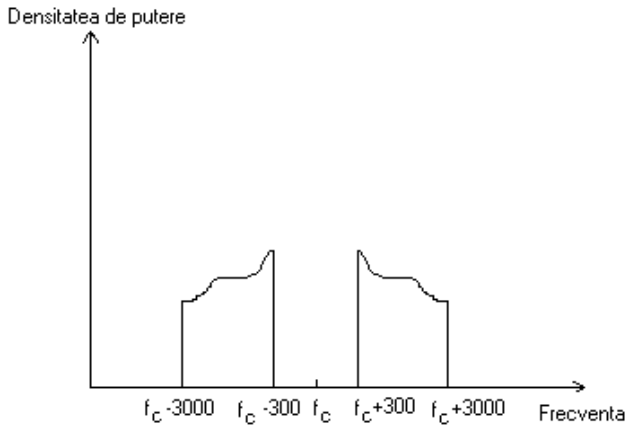

Figura 12.2: Spectrul densităţii de putere a ieşirii din modulatorul echilibrat.

că partea superioară este identică cu semnalul original dar translatat în frecvenţă iar partea inferioară este imaginea în oglindă a semnalului original. Deci printr-o alegere judicioasă a parametrului f_c al purtătoarei vom obţine semnalul inversat (vezi figura 12.3). Observăm că în descrierea sistemului de inversare spectrală nu avem nici o cheie ceea ce face ca tehnica de securizare să fie de fapt o codificare a informaţiei. O tehnică ce se bazează pe inversarea spectrului şi care face apel la cheia de cifrare este tehnica rotirii ciclice a spectrului.

12.5.2. Rotirea spectrului inversat

În paragraful anterior ne-am limita la un semnal vocal în lungimea de bandă $300-3000Hz$. Pentru ca semnalul inversat să fie în aceeiaşi lungime de bandă trebuie ca frecvenţa semnalului purtător să fie de $3300Hz$. Dacă frecvenţa purtătoarei este de $4000Hz$ atunci vom obţine semnalul din figura 12.4

Acest semnal nu este în aceeiaşi lungime de bandă cu semnalul original, dar se poate rearanja ca acesta să fie în aceeiaşi lungime de bandă ca şi originalul: se iau frecvenţele de peste $3000Hz$ şi se pun ca frecveţe joase, obţinându-se semnalul din figura 12.5 Un inversor spectral poate avea mai multe frecvenţe purtătoare, selecţia acestora realizându-se cu ajutorul unui generator pseudoaleatoriu iniţializat cu cheia de cifrare şi cheia de mesaj. Schimbarea frecvenţei purtătoarei se face de regulă la un interval de timp stabilit apriori (de exemplu $10-20ms$). Semnalul de ieşire poate fi recuperat prin căutare exhaustivă. Mai mult semnalul de ieşire are o componentă inteligibilă reziduală mare în cazul semnalului vocal.

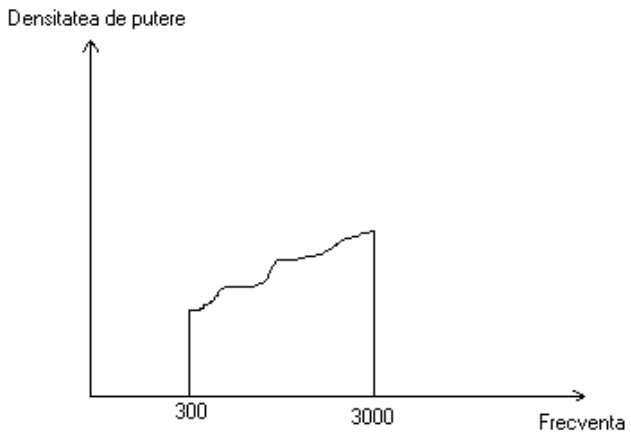

Figura 12.3: Spectrul densităţii de putere a semnalului inversat.

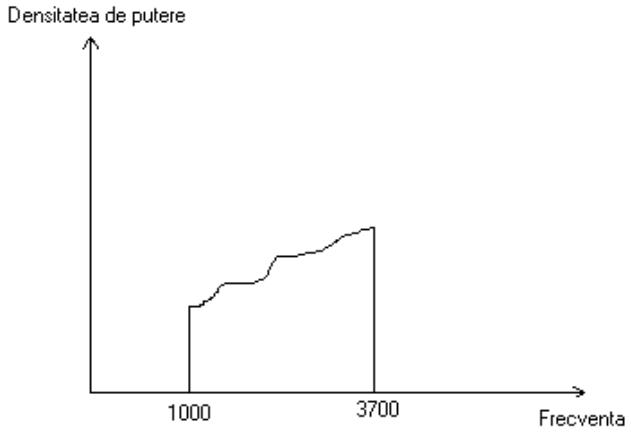

Figura 12.4: Semnal inversat (frecvenţa purtătoarei 4000Hz).

12.5.3. Amestecarea spectrului

Amestecarea spectrului este o tehnică prin care domeniul spectrului este împărţit în n sub-benzi de lungimi egale semnalul cifrat fiind realizat prin permutarea (obţinută din cheia de cifrare) acestor lungimi de bandă, urmată de o eventuală inversare a acestora. La fel ca la rotirea inversorului spectral permutarea se poate schimba la o perioadă de timp stabilită apriori. Numărul total de posibile decriptări este în acest

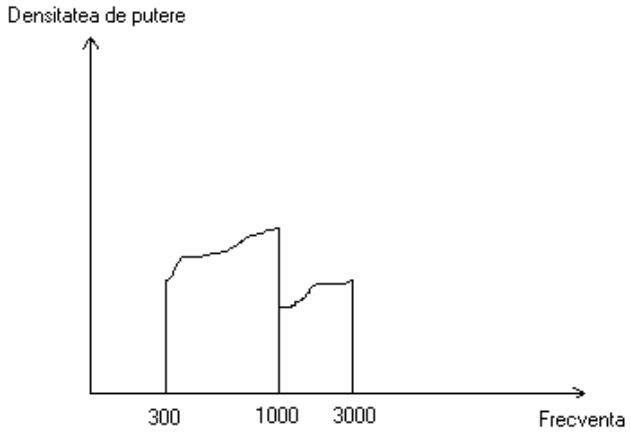

Figura 12.5: Semnal inversat şi rotit în bandă (frecvenţa purtătoarei 4000Hz).

caz $2^n \times n!$. În figura 12.6 este prezentat un exemplu pentru $n = 4$ permutarea fiind $(4, 1, 3, 2)$ secvenţa de inversare fiind 1001 (secvenţele 4 şi 2 fiind inversate iar 1 şi 3 neinversate).

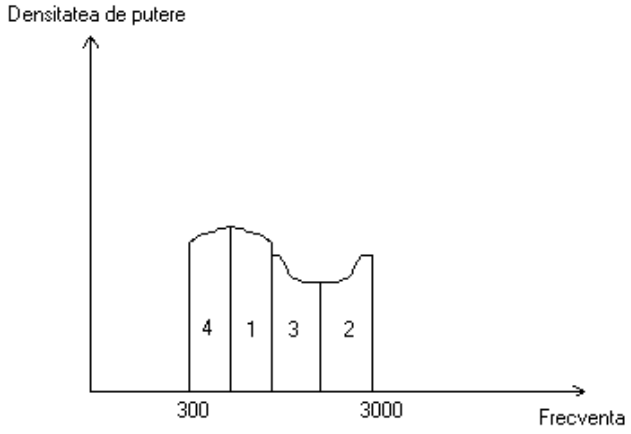

Figura 12.6: Exemplu de sistem de cifrare prin amestecarea spectrului n=4.

12.5.4. Multiplexarea în timp

Vom investiga o metodă prin care se intervine asupra elementului timp al semnalului. Metoda presupune divizarea semnalului analogic în *secvenţe* (frame) de lungimi egale. Fiecare secvenţă este divizată în elemente periodice în timp numite *segmente* acestea fiind apoi permutate acestea constituind semnalul de ieşire. Această metodă se numeşte *multiplexarea în timp* (time division multiplexing scris prescurtat **t.d.m**). Setarea unui astfel de sistem de cifrare presupune stabilirea unor valori ale segmentelor şi a secvenţelor. Alegerea acestor valori numerice se face experimental, neexistând o metodă matematică. Descrierea acestui tip de sistem de cifrare precum şi studii asupra securităţii sistemelor **t.d.m.** se poate găsi în *Beker* [7].

12.6. Aplicaţii

Exerciţiul 12.6.1. Care este rata minimă de eşantionare pentru recuperarea unui semnal vocal limitat între
i) $0 - 4KHz$;
ii) $300 - 3400Hz$.

Exerciţiul 12.6.2. Descrieţi o procedură de atac (bazată pe analiza transformatei Fourier) asupra metodei de cifrare prin rotire a spectrului inversat pentru un semnal vocal.

Exerciţiul 12.6.3. Descrieţi o procedură de atac (bazată pe analiza transformatei Fourier) asupra metodei de cifrare prin amestecare a spectrului pentru un semnal vocal.

Exerciţiul 12.6.4. Procedurile de mai sus sunt eficiente în cazul cifrării unui zgomot alb?

Capitolul 13

MANAGEMENTUL CHEILOR CRIPTOGRAFICE

No message is safe in cipher unless
the key phrase is comparable in
length with the message itself.
Paker Hitt, 1914.

13.1. Managementul cheilor

Managementul adecvat al cheilor de criptare este o componentă deosebit de importantă în utilizarea criptografiei în stabilirea obiectivelor de securitate (vezi *Schneier* [69]). Securitatea informaţiei protejate prin metode criptografice depinde direct de protecţia realizată asupra cheilor criptografice. Toate cheile (private şi secrete) trebuie protejate împotriva modificărilor precum şi împotriva accesului neautorizat. În cele ce urmează prezentăm o serie de principii, formulate sub formă de recomandări, pentru managementul efectiv al cheilor criptografice.

1. *Fiţi siguri că utilizatorii sunt informaţi de obligaţiile şi responsabilităţile lor, şi că au înţeles importanţa păstrării cheilor în loc sigur.*

Securitatea cheilor criptografice dintr-un sistem cu semnătură electronică sau digitală este fundamentul sistemului de securitate; de aceea, utilizatorii trebuie să păstreze controlul cheilor corespunzătoare. Înainte de a primi o cheie (dacă aceasta este o cheie controlată de utilizator pe termen lung) utilizatorii trebuie să cunoască responsabilităţile şi obligaţii ce le revin în special privitor la vulnerabilităţile rezultate din compromiterea sau pierderea cheii. Administratorul de securitate trebuie să fie pregătit pentru posibilitatea compromiterii cheii. Este obligatoriu ca acesta să dispună de un plan de acţiune în cazul compromiterii sau pierderii cheilor de sistem

sau a componentelor cheilor la un server central; aceast plan trebuie realizat înainte ca sistemul să devină operaţional. Planul de acţiune, în acest caz, cuprinde măsuri ce trebuie luate, soft şi hard, asupra cheilor de sistem, cheilor utilizator, semnăturilor generate, datelor criptate, etc. Dacă cheia privată a unui utilizator este pierdută sau compromisă atunci ceilalţi utilizatori trebuie înştiinţaţi de aceast lucru, aşa că ei nu vor mai cripta mesaje folosind cheia publică compromisă, nici să accepte mesaje semnate cu cheia privată compromisă. Utilizatorii trebuie să fie capabili să stocheze în condiţii de siguranţă cheile private, astfel încât nici un intrus să nu o poată găsi, iar acestea trebuie să fie disponibile pentru o folosire legitimă. Cheile criptografice trebuie să fie valide pâna la expirarea datei specificate.

2. *Semnaţi şi verificaţi codul care implementează funcţiile criptografice.*

Softul serverului pentru managementul cheii centrale trebuie semnat electronic şi verificat periodic cu scopul verificării integrităţii acestuia. Aceast lucru furnizează un mijloc de detecţie a modificărilor neautorizate ale sistemului software. În interiorul unui modul de criptografic, această caracteristică de generare şi verificare a autenticităţii codului este cerută de standardul american FIPS PUB 140-1.

3. *Un sistem implementat pentru o agenţie guvernamentală trebuie să aibă cheile proprii păstrate central şi sistemul software controlat de angajaţii abilitaţi.*

Pentru securitatea sistemului este esenţial un controlul asupra cheilor de bază şi asupra softul şi hardul destinat administrării cheilor. În situaţia în care o agenţie guvernamentală lucrează cu un sistem care a fost dezvoltat de un contractor, angajaţii guvernamentali trebuie să controleze în totalitate documentaţia şi implementarea funcţiilor criptografice. Acest control se aplică de asemenea configuraţiei softului precum şi managementului cheilor. Odată ce sistemul devine operaţional, angajaţilor neguvernamentali li se va da un acces restricţionat la datele centrale, codul şi modulele de criptare, inclusiv şi la aceia care au fost angajaţi să dezvolte şi să menţină sistemul.

4. *Asigurarea managementului cheilor.*

Managementul cheilor este esenţial pentru generarea sigură, păstrarea, distribuţia şi transferul cheilor criptografice. Unul din principiile fundamentale pentru protecţia cheilor criptografice este *partajarea informaţiei* şi *controlul dual*. Partajarea informaţiei precum şi controlul dual pot fi folosite pentru protecţia cheilor utilizatorilor care sunt stocare central, a cheilor private de bază, distribuţiei codurilor (conţinute în casete) de utilizator, iniţializarii tuturor modulelor de criptare din sistem, autorizării folosirii lor în funcţiile criptografice efectuate de sistem. Alt rol al managementului cheilor este acela de întreţinere a cheilor, în special de înlocuire a cheilor la sfârşitul perioadei de valabilitate. Perioada de valabilitate este determinată de sensibilitatea informaţiei şi de riscul compromiterii cheilor.

Un rol important în mangementul cheilor este jucat de serverele centrale. În

sistemele cu chei publice serverele centrale includ un CA (Autoritate de Certificare), ce este o entitate care generează şi revocă certificatele cheilor publice şi poate genera perechi de chei (publice şi private). Cheia privată a CA trebuie protejată prin partajea informaţiei şi aplicarea controlului dual. În sistemele cu chei secrete sau publice, securitatea serverului central este obigatorie pentru securitatea criptografică a sistemului.

13.2. Generarea cheilor

Generarea cheilor este cea mai sensibilă dintre toate funcţiile criptografice. Orice eroare în implementarea funcţiilor de generare a cheilor sau a măsurilor de securitate fizică a acestor funcţii va afecta serios integritatea altor mecanisme criptografice. Măsurile de securitate fizică sunt necesare pentru prevenirea dezvăluirilor neautorizate, inserărilor şi ştergerii cheilor produse de sistem. În mod special, toate resursele vor genera în mod automat chei şi vor iniţializa vectori care trebuie protejaţi fizic în scopul prevenirii:

- submінării, modificării şi înlocuirii cheilor;
- modificării şi înlocuirii vectorilor de iniţializare;
- modifcării şi înlocuirii algoritmilor de generare.

Depinzând de structura managementului cerut, există aplicaţii unde este necesară generarea cheilor precum şi aplicaţii unde poate fi mai mult dorită distribuţia cheilor de la altă sursă, cum ar fi autoritatea centrală.

1. *Cheile centrale sau de rutare trebuie controlate din timpul generării.*

Cheile centrale sau de rutare sunt cele mai indicate să fie folosite în aplicaţii sensibile cum ar fi criptarea cheilor utilizator, semnarea unei baze de date cu o cheie centrală pentru integritate, legătura unei chei pereche cu un utilizator sau generarea cheiilor utilizatorilor. Dacă aceste chei sunt compromise, o compromitere completă a sistemului devine în mod real foarte ameninţătoare. Este esenţial să menţinem securitatea acestor chei centrale chiar de la începutul procesului de generare. Nici un proprietar al cheilor sau al componentelor cheilor nu va putea vreodată să folosească cheile sau componentele acestora. Dacă partajarea informaţiei şi controlul dual sunt o cerinţă pentru cheile centrale sau de rutare, atunci un eşec în obţinerea partajării informaţiei sau a controlului dual al acestor chei în orice moment din ciclul de viaţă al acestora ar putea prezenta o problemă de securitate şi o potenţială compromitere a sistemului.

2. Dacă o cheie este stocată într-o casetă de cod ce se accesează prin intermediul unui cod PIN, atunci numai proprietarul acestei casete va putea fi în posesia PIN-ului corespunzatoar.

Aceast lucru trebuie aplicat şi celor care genereză casete de cod precum şi numere

de identificare personale (PIN), la fel ca şi oricăror alţi utilizatori. Pentru a preveni ca un curier să aibă un control exclusiv asupra celor două elemente, administratorul de securitate trebuie să distribuie caseta şi PIN-ul în pachete separate distribuite în zile diferite. Recepţia fiecărei informaţii va fi întotdeauna confirmată emiţătorului original. Un neajuns în menţinerea controlului asupra casetei şi a PIN-ului sau corespunzator poate conduce la o cheie compromisă şi la folosirea greşită a unei funcţii criptografice din sistem.

13.3. Protecţia cheilor criptografice

13.3.1. Cheie utilizator

Cheile criptografice necesită o protecţie fizică specială. Dacă cheile sau componentele ale acestora sunt memorate într-o casetă (ex: floppy disk, PC, smart card, etc.), atunci această casetă trebuie să fie stocată într-o manieră specială în scopul de a preveni persoanele neautorizate de a accesa cheile sau componentele ale acestora. De exemplu, dacă componentele cheilor pentru iniţoalizarea unui CA sau a unei facilităţi a managementului cheilor sunt stocate într-o casetă păstrată într-un seif atunci mai multe persoane pot avea acces la caseta respectivă. De aceea, o protecţie în plus este necesară pentru fiecare casetă, de exemplu prin folosirea unei încuietori fizice, astfel încât să se permită proprietarului unei casete să determine dacă o altă persoană a folosit sau nu caseta.

1. *Pentru protecţia cheilor la compromitere şi folosire greşită se folosesc pauzele de autentificare.*

Rolul pauzei de autentificare pentru un modul criptografic sau pentru o casetă este acela de minimizare a posbilităţii ca un individ neautorizat să acceseze un modul criptografic activ şi folosirea cheilor sale de criptare. Aceasta s-ar putea întâmpla dacă modulul criptografic este lăsat activ (din neatenţie) de un utilizator care i-a stabilit autenticitatea şi i-a încărcat cheia criptografică. O alternativă este să forţăm utilizatorul să se reautentifice periodic la modulul criptografic, şi să-i permitem să stea logat numai pentru o anumită perioadă de timp. Pentru aplicaţii sensibile poate fi necesar să se restricţioneze timpul în care aceasta poate avea loc.

2. *Semnaţi toate datele înmagazinate central şi criptaţi datele sensibile cum ar fi cheile secrete care sunt folosite în scopul realizării confidenţialităţii.*

Toate datele memorate central care se referă la cheile utilizatorilor trebuie semnate pentru integritate şi eventual criptate pentru confidenţialtate (toate cheile secrete ale utilizatorilor şi toate cheile secrete ale CA trebuie criptate). Înregistrările cheilor individuale din baza de date -precum şi toată baza de date - trebuie semnate. Pentru a permite detecţia încercărilor de fraudare, fiecare înregistrare a cheilor individuale trebuie semnată, în aşa fel încât integritatea să poată fi controlată înainte ca

această cheie să fie folosită într-o funcţie criptografică. Dacă semnăm întreaga bază de date, atunci trebuie semnat cel mai puţin important câmp care nu se schimbă regulat (aceasta permite o verificare rapidă).

3. *Faceţi ce este necesar pentru restabilirea proprietăţilor cheilor.*

Sistemele IT trebuie să protejeze confidenţialitatea informaţiei. Trebuie să existe dispozitive de siguranţă care să asigure că înregistrarile sensibile să nu fie nici pierdute de proprietarii în drept, nici accesate de către indivizi neautorizati. Posibilitatea recuperării cheilor generează aceste controale. Toate componentele cheilor trebuie să fie disponibile unei organizaţii, indiferent dacă utilizatorul asociat lucrează în mod curent în organizaţie. Când salariaţii părăsesc organizaţiile voluntar sau când sunt mutaţi din alte motive, organizaţia trebuie să acceseze componentele cheilor pentru a iniţializa procedura de recuperare a datelor criptate. Posibilitatea recuperării cheilor permite organizaţiilor să recupereze componentele cheilor. Este foarte important să existe copii de siguranţă ale cheilor de bază (root), atât timp cât compromiterea sau pierderea acestor componente poate impiedica accesul la cheile bazei centrale de date, şi posibilitatea de a nu permite unor utilizatori să decripteze date sau să execute verificarea semnăturii.

13.3.2. Arhivarea cheilor

1. *Arhivaţi cheile de utilizator pentru o perioadă de timp de criptare suficient de lungă.*

O perioadă de *timp de criptare* este timpul în care o cheie poate fi folosită pentru verificarea semnăturii sau pentru decriptare; trebuie să dureze mai mult decât timpul de viaţă a unei chei (unde timpul de viaţă este perioada în care o cheie poate fi folosită să genereze o semnătură şi/sau să realizeze o criptare). Cheile trebuie arhivate pentru o lungă perioadă de criptare (de ordinul decadelor), astfel încât ele pot fi folosite pentru verificarea semnăturilor şi decriptarea textelor cifrate în timpul perioadei de criptare.

13.3.3. Distrugerea cheilor

1. *Determinaţi un timp de viaţă rezonabil pentru cheile asociate cu diferite tipuri de utilizatori.*

Utilizatorii cu diferite roluri în sistem trebuie să aibă chei cu un timp de viaţă care să ţină cont de rolul şi responsabilităţile acestora, aplicaţiile pentru care aceste chei sunt folosite şi securitatea serviciilor care sunt date de aceste chei (autentificarea utilizator/dată, confidenţialitatea, integritatea datelor,etc). Generarea unei noi chei nu trebuie făcută prea des deoarece aceasta ar deveni obositoare; totuşi trebuie

făcută destul de des pentru a minimiza pierderile cauzate de o posibilă compromitere a cheii.

2. *Folosiţi funcţia de dezactivare sau anulare a cheilor astfel încât informaţia semnată anterior cu date compromise sau pierdute să poată fi verificată.*

Este posibil să desemnăm o cheie pentru semnătura ca fiind pierdută sau compromisă, astfel încât semnatura generată anterior pentru o dată specificată să poată fi verificată. Altfel, toate datele semnate anterior cu o cheie pierdută sau compromisă vor trebui să fie revăzute şi semnate încă o dată.

13.4. Lungimea cheilor criptografice

Problema lungimii cheilor trebuie abordată în funcţie de *tipul de informaţie* (categoria de secret) ce se doreşte protejată, de proprietăţile celei mai *eficiente metode de atac* (care în mod ideal este atacul brut) precum şi de *capacităţile tehnice* (echipamente hardware şi software dedicate) şi *umane* (gradul de pregătire, diversitatea şi numărul specialiştilor) ale adversarului. Trebuie să reţinem faptul că, în anumite condiţii, autorităţile trebuie să aibă acces la mesajele cifrate (pentru asigurarea disponibilităţii informaţiei). Acest lucru se poate face prin partajarea cheilor de cifrare (pentru produsele profesionale) sau prin restrângerea acesteia la o dimensiune rezonabilă (pentru produsele comerciale). Lungimea cheilor criptografice precum şi a parametrilor secreţi (utilizaţi în autentificare) trebuie să fie direct proporţională cu valoarea şi perioada de valabilitate a informaţiei protejate. Reamintim faptul că utilizarea cheilor asimetrice este indicată pentru realizarea autenficării şi semnării datelor pe când a cheilor simetrice pentru asigurarea confidenţialităţii acestora. În tabelul 13.4.1 se prezintă echivalenţa dintre lungimile cheilor publice (asimetrice) şi secrete (simetrice) raportate la metoda de atac brut. Tabelul 13.4.2 prezintă lungimea minimă a cheilor secrete relativ la tipul de informaţie protejată precum şi durata de viaţă a acestora (vezi *Schneier* [69]).

Lungimea cheii publice	Lungimea cheii secrete
384 biţi	56 biţi
512 biţi	64 biţi
768 biţi	80 biţi
1792 biţi	112 biţi
2304 biţi	128 biţi

Tabelul 13.4.1: Lungimile comparative ale cheilor asimetrice şi a cheilor simetrice.

Tipul informaţiei	Timpul de viaţă al cheii	Lungimea minimă a cheii
Informaţii tactice militare	minute/ore	56-64 biţi
Lansare de produse	zile/săptamâni	64 biţi
Plan de afaceri pe termen lung	ani	64 biţi
Secrete de producţie comerciale	decade	112 biţi
Documentaţia bombei H	cel puţin 40 ani	128 biţi
Identitatea spionilor	cel puţin 50 ani	128 biţi
Afaceri interne	cel puţin 50 ani	128 biţi
Afaceri diplomatice	cel puţin 65 ani	256 biţi
Resurse strategice	100 ani	256 biţi

Tabelul 13.4.2: Lungimea minimă a cheilor simetrice raportată la tipul de informaţie.

13.5. Aplicaţii

Exerciţiul 13.5.1. Propuneţi o procedură de generare a cheilor simetrice.

Exerciţiul 13.5.2. Propuneţi o procedură de generare a cheilor asimetrice de tip RSA.

Exerciţiul 13.5.3. Propuneţi o procedură de asigurare a integrităţii cheilor.

Exerciţiul 13.5.4. Propuneţi o procedură de asigurare a autenticităţii cheilor.

Exerciţiul 13.5.5. Propuneţi o procedură de asigurare a nerepudierii cheilor.

Exerciţiul 13.5.6. Propuneţi o procedură de recuperare a cheilor secrete prin metode de *key sharing* şi/sau *key splitting*.

Exerciţiul 13.5.7. Propuneţi o procedură de transfer a cheilor.

Exerciţiul 13.5.8. Care este rolul perioadei de valabilitate a cheilor criptografice?

Exerciţiul 13.5.9. Care este diferenţa dintre *key sharing* şi *key splitting*?

Anexa A

METODE ŞI TEHNICI DE PROGRAMARE

A.1. Structuri de date

După cum s-a remarcat în acestă lucrare nu s-au prezentat excesiv demonstraţii matematice, ci s-a pus accent pe formularea (codificarea) algoritmului. Pentru ca problema astfel formulată să se poată rezolva eficient se va face apel la o serie de tehnici de programare care vor fi prezentate pe scurt în cele ce urmează. Descrierea completă a tehnicilor de programare amintite în cadrul acestui capitol face obiectul cursurilor de tehnici de programare (vezi *Basse* [5], *Cormen* [14] şi *Knuth* [38]).

Structurile de date cu ajutorul cărora se implementează tehnicile de programare prezentate în cele ce urmează pot fi statice (de exemplu, vectorii ori matricile: se alocă memorie în faza iniţială) sau dinamice (se alocă memorie în funcţie de necesităţi).

Dintre cele mai utilizate structuri de date dinamice amintim:

-lista: aceasta poate fi simplu sau dublu înlănţuită;

-stiva (este un caz particular de listă dublu înlănţuită): structură de date tip **LIFO** (*last in first out*);

-coada (este un caz particular de listă): structură de date tip **FIFO** (*first in first out*);

-arborii: structură de date ce poate fi definită şi ca un caz particular de graf;

Principalele operaţii ce se efectuează asupra listelor sau arborilor sunt următoarele:

-iniţializare;

-inserţia unui nod după sau înainte de nodul k;

-ştergerea nodului k;
-combinarea a două sau mai multe liste într-una singură;
-partiţionarea unei liste;
-copierea unei liste;
-determinarea numărului de noduri dintr-o listă;
-sortarea nodurilor unei liste (după un anume criteriu).
-căutarea în listă a nodului cu o valoare particulară într-un anumit câmp.

A.2. Alocarea memoriei

Alocarea optimă a memoriei este un element necesar scrierii unui cod eficient şi se poate face *static*, situaţie în care variabilele sunt vectori unu sau multi dimensionali, sau *dinamic* (pointer) pe măsură ce avem nevoie de acestea. Următoarele operaţii trebuiesc efectuate în cazul utilizării alocării dinamice a memoriei: *iniţializarea* pointerului, *utilizarea* pointerului şi *eliberarea* memoriei.

A.3. Recursivitate

Tehnica recursivităţii se aplică în problemele în care dispunem de o formulă recursivă de calcul. Tehnica fiind des utilizată, nu vom insista asupra ei amintind doar că există variante recursive ale metodelor backtracking, divide et impera, greedy (acestea sunt prezentate în cele ce urmează). Întreaga tehnică a recursivităţii se bazează pe noţiunea de stivă.

Exemplu de probleme ce se rezolvă cu ajutorul tehnicii recursive:
-calculul funcţiei factorial, al celui mai mare divizor comun, al şirului Fibonacci etc;
-generarea permutărilor;
-ieşirea dintr-un labirint;
-analiza sintactică.

A.4. Metoda backtracking

Această metodă evită, în general, generarea tuturor soluţiilor posibile. Elementele vectorului n dimensional $\mathbf{x}$ (soluţia problemei) primesc pe rând valori. Astfel lui $x_k \in S_k$ i se atribuie o valoare numai dacă au fost deja atribuite valori lui $x_1, \ldots, x_{k-1}$. Mai mult, odată stabilită o valoare pentru x_k, nu se trece direct la atribuirea de valori lui x_{k+1}, ci se verifică nişte *condiţii de continuare* referitoare la $x_1, \ldots, x_k$. Aceste condiţii stabilesc situaţiile în care are sens să trecem la determinarea lui x_{k+1}. Neîndeplinirea lor exprimă faptul că, oricum am alege $x_{k+1}, \ldots, x_n$

nu vom putea ajunge la o soluţie rezultat, adică la o soluţie pentru care *condiţiile interne* (relaţii între componentele lui $\mathbf{x}$) sunt satisfăcute. Dacă condiţiile de continuare nu sunt îndeplinite, se va alege un alt x_k din S_k, iar dacă S_k s-a epuizat ne întoarcem la S_{k-1}. Există şi o variantă recursivă a acestei metode.

Exemplu de probleme ce se rezolvă cu ajutorul tehnicii backtracking:

-problema celor opt dame;

-generarea permutărilor, aranjametelor sau combinărilor;

-produsul cartezian a n mulţimi;

-problema comis-voiajorului;

-problema colorării hărţilor;

-problema plăţii unei sume s utilizând n tipuri de monede.

Tehnica backtracking are o complexitate exponenţială şi este de folos atunci când nu avem la dispoziţie un algoritm mai performant.

A.5. Tehnica Divide et Impera

Este o tehnică (care se aplică în principal în forma recursivă) ce constă în următoarele:

-dacă problema este rezolvabilă direct atunci ea se rezolvă;

-altfel, se descompune în două sau mai multe probleme mai simple, de aceeaşi natură cu problema iniţială (numite subprobleme), care se rezolvă prin aceeaşi metodă; soluţia problemei iniţiale se obţine prin combinarea soluţiilor subproblemei.

Exemplu de probleme ce se rezolvă cu ajutorul tehnicii divide et impera:

-problema tăieturilor;

-sortarea rapidă (quick sort);

-sortarea prin interclasare.

Metoda divide et impera reduce o complexitate de ordinul $O(n^2)$ la un ordin de complexitate de $O(n^{lg3})$.

A.6. Tehnica branch and bound

Tehnică de optimizare care face parte din domeniul Cercetării Operaţionale şi constă în esenţă în rezolvarea unei probleme de optimizare cu condiţii relaxate (problemă continuă) relativ la problema iniţială (problemă discretă) şi apoi spargerea problemei în două probleme care se rezolvă cu ajutorul algoritmului continuu. Procedura de căutare optimă a soluţiei se aplică iterativ prin generarea de alte două noi probleme obţinute din problema curentă prin introducerea unei restricţii liniare. Algoritmul se încheie când procesul de ramificare a fost stopat (problema nu are soluţii, problema are soluţie optimă în mulţimea iniţială, problema nu are soluţie

în mulţimea iniţială, dar valoarea optimă a funcţiei obiectiv este superioară celei atinse într-o soluţie din mulţimea iniţială generată la un alt nod). Soluţia optimă este atinsă în unul dintre nodurile stopate (după cele trei situaţii). În acest caz o soluţie optimă a problemei iniţiale este dată de orice nod stopat după regula 2 care realizează minimul în clasa acestor noduri.

A.7. Programarea dinamică

Tehnica programării dinamice (progresive, regresive sau mixte) a fost expusă pe larg în lucrare. În esenţă metoda este aplicabilă problemelor în care rezultatul se obţine ca urmare a unui şir de decizii $D_1, \ldots, D_n$. În urma deciziei D_1, sistemul evoluează din starea S_0 în starea S_1, în urma deciziei D_2 sistemul evoluează din starea S_1 în starea $S_2, \ldots$, în urma deciziei D_n sistemul evoluează din starea S_{n-1} în starea S_n. În afara condiţiilor arătate până acum este necesar să fie satisfăcut principiul programării dinamice pe care îl enunţăm în continuare.

Dacă $D_1, \ldots, D_n$ este un şir optim de decizii care transformă sistemul din starea iniţială S_0 în starea finală S_n atunci trebuie îndeplinită una dintre condiţiile următoare:

1) $D_k, \ldots, D_n$ este un şir optim de decizii care duce sistemul din starea S_{k-1} în starea S_n pentru orice $k = \overline{1,n}$ (metoda progresivă);

2) $D_1, \ldots, D_k$ este un şir optim de decizii care duce sistemul din starea S_0 în starea S_k pentru orice $k = \overline{1,n}$ (metoda regresivă);

3) $D_{k+1}, \ldots, D_n$ şi $D_1, \ldots, D_k$ sunt şiruri de decizii optime ce duc sistemul din starea S_k în starea S_n, respectiv din starea S_0 în starea S_k pentru orice $k = \overline{1,n}$ (metoda mixtă).

Exemplu de probleme ce se rezolvă cu ajutorul acestei tehnici:

-determinarea subşirului crescător de lungime maximală;

-determinarea drumurilor de cost minim într-un graf;

-înmulţirea optimă a unui şir de matrice;

-problemele de investiţii.

A.8. Tehnica greedy

Această metodă se foloseşte în problemele în care, dată fiind o mulţime A cu n elemente se cere să se determine o submulţime B a sa, care să îndeplinească anumite condiţii (eventual un anumit criteriu de optim).

Metoda Greedy de rezolvare a unor probleme constă în următorii paşi:

-se iniţializează mulţimea B la mulţimea vidă;

-se alege un anumit element din A;

-se verifică dacă elementul ales poate fi adăugat mulţimii B;

-procedeul continua astfel, repetitiv, până ce au fost determinate toate elementele din B.

Exemplu de probleme ce se rezolvă cu ajutorul tehnicii Greedy:

-problema rucsacului;

-determinarea drumurilor de cost minim într-un graf (algoritmul lui Dijkstra);

-problema comis-voiajorului.

Complexitatea algoritmilor Greedy este de ordinul de complexitate $O(n^2)$.

A.9. Aplicaţii

Exerciţiul A.9.1. Implementaţi în pseudocod următoarele structuri de date: lista, stiva, coada şi arborii.

Exerciţiul A.9.2. Implementaţi în pseudocod următoarele operaţii asupra listelor: iniţializare, inserţie, ştergere, combinare, partiţionare şi copiere.

Exerciţiul A.9.3. Studiaţi din *Knuth* [38] algoritmii de sortare: Quick Sort şi Divide et Impera. Realizaţi operaţia de implementare a acestora cu tehnicile de programare prezentate.

Exerciţiul A.9.4. Studiaţi din *Knuth* [38] algoritmii de căutare. Realizaţi operaţia de implementare a acestora cu tehnicile de programare prezentate.

Exerciţiul A.9.5. Proiectaţi în pseudocod un algoritm de generare a permutărilor de n elemente.

Exerciţiul A.9.6. Implementaţi în pseudocod un algoritm de de terminare a drumurilor de cost minim (de exemplu algoritmul lui Dijkstra).

Exerciţiul A.9.7. Studiaţi tehnica alocării dinamice a memoriei în limbajele C/C++ şi Pascal.

Anexa B

ELEMENTE DE TEORIA PROBABILITĂŢILOR

Contextul general de lucru este acela al unei variabile aleatoare X care poate lua valori din $\mathbf{R}^n$.

B.1. Caracteristici ale variabilelor aleatoare

Definiţia B.1.1. Funcţia de repartiţie $F(\cdot)$ a unei variabile aleatoare X este dată de:

$$F(x) = \Pr(X \leq x).$$

Definiţia B.1.2. Densitatea unei variabile aleatoare X este o funcţie $\rho(\cdot) : \mathbf{R} \to [0, \infty)$ cu proprietatea că:

$$F(x) = \int_{-\infty}^{x} \rho(t)dt.$$

Observaţia B.1.1. Evident că $F(\infty) = 1$ şi deci $\int_{-\infty}^{\infty} \rho(t)dt = 1$.

Definiţia B.1.3. Două variabile aleatoare X şi Y, având densităţile ρ_X respectiv ρ_Y se numesc independente dacă densitatea comună ρ_{XY} este :

$$\rho_{XY}(x, y) = \rho_X(x)\rho_Y(y).$$

Definiţia B.1.4. Momentul de ordinul k al unei variabile aleatoare X este dat de formula:

$$M(X^k) = \int_{-\infty}^{x} t^k \rho(t) dt.$$

Definiţia B.1.5. Momentul centrat de ordinul k al unei variabile aleatoare X este dat de formula:

$$M((X - M(x))^k) = \int_{-\infty}^{x} (t - m)^k \rho(t) dt$$

unde m este momentul de ordinul 1 al variabilei aleatoare X.

Observaţia B.1.2. Momentul de ordinul 1 este media variabilei aleatoare.

Observaţia B.1.3. Dispersia unei variabile aleatoare este momentul centrat de ordinul 2.

Definiţia B.1.6. Funcţia caracteristică (numită şi transformata Fourier) a unei variabile aleatoare X este definită prin formula:

$$\varphi(t) = M(e^{itx}).$$

Observaţia B.1.4. Funcţia caracteristică a unei variabile aleatoare este unică.

Observaţia B.1.5. Funcţia caracteristică a sumei de variabile aleatoare independente este egală cu produsul funcţiilor caracteristice.

Definiţia B.1.7. Funcţia generatoare de momente (numită şi transformata Laplace) a unei variabile aleatoare X este definită prin formula:

$$G(t) = M(e^{tx}).$$

Observaţia B.1.6. Funcţia generatoare de momente caracterizează în mod unic o variabilă aleatoare.

Anexa C

STATISTICĂ DESCRIPTIVĂ

C.1. Momentele unei variabile aleatoare

Valoarea medie a variabilei aleatoare continue X cu densitatea $p(x)$ este definită prin:

$$M(X) = \int xp(x)dx.$$

În cazul discret în care variabila X ia valoarea i cu probabilitatea p_i avem formula:

$$M(X) = \sum_{i=1}^{n} ip_i.$$

Dispersia variabilei aleatoare $D^2(X)$ este definita prin formula:

$$\begin{aligned} D^2(X) &= M([M(X) - X]^2) \\ &= M(X^2) - M^2(X). \end{aligned}$$

Deviaţia standard (termen introdus de *Pearson* în anul 1894) a variabilei aleatoare este $D(X)$.

Modulul unei variabile aleatoare X este valoarea cea mai mare (în valoare absolută) a variabilei.

Mediana (termen introdus de Galton în anul 1882) unei variabile aleatoare X este acea valoare med pentru care are loc:

$$\Pr(X < med) = \Pr(X > med).$$

Mediana poate fi diferită de valoarea medie. Generalizarea medianei este $\alpha-$*cuantila* u_α ce este definită prin formula:

$$\Pr(X < u_\alpha) = \alpha.$$

Cuantila de ordinul $\frac{1}{2}$ este mediana.

Avem următoarea inegalitate (Cebâşev) care exprima o constrângere asupra abaterii unei variabile aleatoare de la valoarea sa medie:

$$\Pr(|X - M(X)| \geq kD(X)) \leq \frac{1}{k^2}.$$

Covarianţa dintre două variabile aleatoare X şi Y este

$$Cov(X, Y) = E((X - E(X)(Y - E(Y)).$$

Corelaţia dintre două variabile aleatoare X şi Y este

$$Cor(X, Y) = \frac{Cov(X, Y)}{\sqrt{D^2(X)D^2(Y)}}.$$

Observaţia C.1.1. Dacă variabilele aleatoare X şi Y sunt independente atunci $Cor(X, Y) = 0$. Reciproc dacă variabilele aleatoare X şi Y sunt necorelate şi vectorul bidimensional (X, Y) are o repartiţie gaussiană atunci acestea sunt independente.

C.2. Teoria selecţiei

Fie $X_1, ..., X_n$ o selecţie de volum n asupra variabilei aleatoare X de medie m şi dispersie σ^2.

Media de selecţie este definită prin formula:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}.$$

Media de selecţie este un *estimator nedeplasat* al valorii medii a variabilei aleatoare X :

$$M(\bar{X}) = m.$$

Dispersia mediei de selecţie $\bar{X}$ este dată de:

$$D^2(\bar{X}) = \frac{\sigma^2}{n}.$$

Dispersia de selecţie este definită prin formula:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2.$$

Valoarea s se numeşte *deviaţia standard de selecţie.*

Dispersia de selecţie este un *estimator nedeplasat* al dispesiei variabilei aleatoare X :

$$M(s^2) = \sigma^2.$$

Eroarea standard a mediei este:

$$S_X = \frac{s}{\sqrt{n}}.$$

Modulul de selecţie este $\max_i |X_i|$.

Mediana de selecţie este acea valoare pentru care jumatate din valorile ordonate ale selecţiei ordonate $X_{\sigma(1)}, ..., X_{\sigma(n)}$ sunt mai mici şi jumătate sunt mai mari decât valoarea medianei.

Fie $X_1, ..., X_n$ şi $Y_1, ..., Y_n$ selecţii de volum n asupra variabilei aleatoare X respectiv Y. Atunci *covarianţa de selecţie* este

$$s^2_{XY} = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y}).$$

corelaţia rezultată în urma selecţiei fiind dată de formula:

$$\rho_{XY} = \frac{s^2_{XY}}{s_X s_Y}.$$

C.3. Aplicaţii

Exerciţiul C.3.1. Să se calculeze valoarea medie, dispersia, deviaţia standard, modulul şi mediana pentru repartiţia discretă data de:

$$\begin{pmatrix} 0 & 1 & 2 & ... & n-1 \\ p_0 & p_1 & p_2 & ... & p_{n-1} \end{pmatrix}.$$

Caz particular $n = 4,\ p_i = 0,25$ pentru orice i.

Exerciţiul C.3.2. Să se calculeze valoarea medie, dispersia, deviaţia standard, modulul şi mediana pentru repartiţia normală $N(m, \sigma^2)$ data de funcţia de densitate:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-m)^2}{2\sigma^2}}, x \in \mathbf{R}.$$

Exerciţiul C.3.3. Să se calculeze media de selecţie, dispersia de selecţie, deviaţia standard de selecţie, eroarea standard de selecţie, modulul de selecţie şi mediana de selecţie pentru eşantionul $\{3, 5, 2, 7, 0, 3, 3, 1, 2, 4\}$.

Anexa D

TEORIA ESTIMAŢIEI

D.1. Tipuri de estimatori

Fie $X_1, ..., X_n$ o selecţie asupra variabilei aleatoare X care are funcţia de repartiţie $F(\mathbf{x}; \theta)$, unde $\theta \in \Theta \subset \mathbf{R}$ este un parametru necunoscut cu valoarea adevărată θ_0.

Vom nota prin:

$$P(\mathbf{x}; \theta) = f(x_1, ..., x_n; \theta) = \prod_{j=1}^{n} f(x_j; \theta)$$

densitatea comună a variabilei de selecţie.

Definiţia D.1.1. Orice aplicaţie $t_n : \mathbf{R}^n \to \Theta$ se numeşte estimator al lui θ.

Definiţia D.1.2. Şirul de estimatori (t_n) este consistent simplu dacă acesta converge în probabilitate la θ adică

$$\lim_{n\to\infty} P(|t_n - \theta| > \varepsilon) = 0 \text{ pentru orice } \theta \in \Theta.$$

Definiţia D.1.3. Şirul de estimatori (t_n) este consistent tare dacă acesta converge aproape sigur la θ.

Definiţia D.1.4. Şirul de estimatori (t_n) este consistent în eroare medie pătratică (EMP) dacă:

$$\lim_{n\to\infty} M((t_n - \theta)^2) = 0 \text{ pentru orice } \theta \in \Theta.$$

Teorema D.1.1. *Dacă (t_n) este convergent tare atunci acesta este simplu convergent.*

Teorema D.1.2. *Dacă* $\lim_{n\to\infty} M(t_n) = 0$ *şi* $D^2(t_n) = 0$ *pentru orice* $\theta \in \Theta$ *atunci şirul de estimatori este consistent în EMP.*

Teorema D.1.3. *Dacă* (t_n) *este consistent în eroare medie pătratică atunci acesta este consistent simplu. Reciproc dacă are loc uniform marginirea lui* (t_n) *adică există* $K \geq 0$ *astfel ca* $|t_n - \theta| \leq K$ *pentru orice* $\theta \in \Theta$ *şi orice* $\mathbf{x} \in \mathbf{R}^n$ *atunci dacă* (t_n) *este consistent simplu el este şi consistent în EMP.*

Definiţia D.1.5. Estimatorul (t_n) este nedeplasat dacă

$$M_\theta(t_n) = 0 \text{ pentru orice } \theta \in \Theta.$$

Observaţia D.1.1. Sunt de interes practic estimatorii nedeplasaţi de disperise minimă (vezi inegalitatea lui Cebâşev).

D.2. Margini inferioare ale estimatorilor

Următoarea teoremă (Rao-Cramer) furnizează o margine inferioară a estimatorului.

Teorema D.2.1. *Dacă sunt îndeplinite condiţiile următoare:*
parametrul θ *aparţine unui interval al numerelor reale:*

$$\theta \in (a, b);$$

derivata funcţiei de verosimilitate este mărginită

$$\frac{\partial P(\mathbf{x};\theta)}{\partial \theta} < \infty \text{ pentru orice } \theta \in (a,b) \text{şi orice } \mathbf{x} \in \mathbf{R}^n;$$

există şi este finit momentul de ordinul 2 al derivatei logaritmului funcţiei de verosimilitate

$$M\left(\left(\frac{\partial \ln P(\mathbf{x};\theta)}{\partial \theta}\right)^2\right) < \infty \text{ pentru orice } \theta \in (a,b)$$

se poate deriva, în raport cu θ*, sub integrală*

$$\frac{\partial}{\partial \theta} \int\limits_{\mathbf{R}^n} P(\mathbf{x};\theta) dx_1 ... dx_n = \int\limits_{\mathbf{R}^n} \frac{\partial P(\mathbf{x};\theta)}{\partial \theta} dx_1 ... dx_n;$$

pentru orice t_n *estimator al lui* $\tau(\theta)$:

$$\frac{\partial}{\partial \theta} \int\limits_{\mathbf{R}^n} t_n(\mathbf{x}) P(\mathbf{x};\theta) dx_1 ... dx_n = \int\limits_{\mathbf{R}^n} t_n(\mathbf{x}) \frac{\partial P(\mathbf{x};\theta)}{\partial \theta} dx_1 ... dx_n;$$

Atunci are loc inegalitatea:

$$D^2(t_n\ ) \geq \frac{(\tau^{'}(\theta))^2}{M\left(\left(\frac{\partial \ln P(\mathbf{x};\theta)}{\partial\theta}\right)^2\right)},$$

egalitatea obţinându-se dacă şi numai dacă există o constantă A astfel ca:

$$A(t_n(\mathbf{x}) - \tau(\theta)) = \frac{\partial \ln P(\mathbf{x};\theta)}{\partial\theta}.$$

Demonstraţie. Din faptul că $P(\mathbf{x};\theta)$ este densitate şi folosind propietatea de derivare sub integrală obţinem:

$$\int\limits_{\mathbf{R}^n} \frac{\partial \ln P(\mathbf{x};\theta)}{\partial\theta} P(\mathbf{x};\theta)dx_1...dx_n = 0.$$

Dar t_n este un estimator al lui $\tau(\theta)$:

$$M(t_n) = \int\limits_{\mathbf{R}^n} t_n P(\mathbf{x};\theta)dx_1...dx_n = \tau(\theta)$$

de unde prin derivarea realţiei şi utilizatea ultimei ipoteze din enunţul teoremei obţinem:

$$\int\limits_{\mathbf{R}^n} t_n \frac{\partial P(\mathbf{x};\theta)}{\partial\theta} dx_1...dx_n = \tau^{'}(\theta).$$

Deci vom obţine

$$\tau^{'}(\theta) = \int\limits_{\mathbf{R}^n} (t_n - \tau(\theta)) \frac{\partial \ln P(\mathbf{x};\theta)}{\partial\theta} P(\mathbf{x};\theta)dx_1...dx_n = 0.$$

Ridicând ultima relaţie la pătrat şi aplicând inegalitatea lui Schwartz obţinem concluzia teoremei.

Observaţia D.2.1. Valoarea $i(\theta) = M\left(\left(\frac{\partial \ln P(\mathbf{x};\theta)}{\partial\theta}\right)^2\right)$ se numeşte *informaţia Fisher.*

Observaţia D.2.2. O estimaţie nedeplasată a parametrului θ pentru care dispersia este egală cu marginea inferioară, dată de inegalitatea Rao-Cramer, se va numi *estimaţie eficientă.*

D.3. Estimaţia de verosimilitate maximă

Fie $X_1, ..., X_n$ o selecţie asupra variabilei aleatoare X care are funcţia de repartiţie $F(\mathbf{x}; \theta)$, unde $\theta \in \Theta \subset \mathbf{R}$ este un parametru necunoscut cu valoarea adevărată θ_0. *Funcţia de verosimilitate* este definită prin formula:

$$L_n(x_1, x_2, ..., x_n; \theta) = \ln p(x_1, x_2, ..., x_n; \theta)$$

Definiţia D.3.1. Vom defini *estimaţia de verosimilitate maximă* acea estimaţie $\hat{\theta}_n$ care realizează maximul funcţiei de verosimilitate:

$$\hat{\theta}_n = \sup_{\theta} L_n(x_1, x_2, ..., x_n; \theta).$$

Estimaţia de verosimilitate maximă se găseşte ca soluţie a *ecuaţiei de verosimilitate maximă*:

$$\frac{dL_n(x_1, x_2, ..., x_n; \theta)}{d\theta} = 0.$$

Ecuaţia de verosimilitate maximă are, în anumite condiţii de regularitate ale funcţiei de verosimilitate (vezi *Dumitrescu* [17]), cu o probabilitate tinzând la unu, o soluţie care este estimaţie consistentă (în cazul unui proces stocastic neergodic estimaţia este consistentă tare). Mai mult are loc următoarea convergenţă în repartiţie:

$$k\sqrt{n}(\hat{\theta}_n - \theta_0) \xrightarrow[n\to\infty]{rep} N(0, 1),$$

unde

$$M\left(\frac{\partial^2 \ln p}{\partial \theta^2}\right)_{\theta=\theta_0} = -k^2.$$

D.4. Estimaţia Bayesiană

Metoda Bayesiană de estimare a unui parametru porneşte de la premiza că, la rândul său, parametrul este o variabilă aleatoare pentru care se cunoaşte o repartiţie a priori ce poate fi ajustată pe baza observaţiilor făcute asupra populaţiei.

Alegerea repartiţiei a priori se poate face pe baza interpretării obiective sau pe baza unor supoziţii subiective asupra parametrului.

Pentru fixarea ideiilor să considerăm populaţia statistică $(E, \mathcal{K}, \mathcal{P}_\theta)$, cu $\theta \in \Theta \subset \mathbf{R}^k, k \geq 1$ şi probabilitatea a priori λ pe spaţiul parametrilor. Vom considera o *funcţie de pierdere* $\omega : \Theta \times \Theta \to \mathbf{R}_+$ măsurabilă în ambele argumente. Fie $\{f_t, t \in \mathbf{N}\}$ un proces stocastic cu valori reale şi fie $(S, \mathcal{S}_n, F_{n,\theta})$ spaţiul selecţiilor $n-$dimensionale.

Definiţia D.4.1. Pentru o estimaţie $g_n : S \to \Theta$ a parametrului θ definim *funcţia de risc* asociată lui g_n prin:

$$R_{g_n}(\theta) = \int_S \omega(\theta, g_n(x))dF_{n,\theta}.$$

Definiţia D.4.2. *Riscul maxim* asociat estimaţiei g_n este:

$$R(g_n) = \sup_{\theta\in\Theta} R_{g_n}(\theta).$$

Definiţia D.4.3. *Estimaţia minimax* este acea estimaţie care minimizează riscul maxim.

Definiţia D.4.4. *Riscul mediu* asociat estimaţiei g_n şi probabilităţii a priori λ se defineşte prin:

$$\overline{R}(g_n) = \int_\Theta R_{g_n}(\theta)d\lambda(\theta).$$

Definiţia D.4.5. Estimaţia g_n^* se numeşte *estimaţie bayesiană* asociată funcţiei de pierdere $\omega(\theta, \theta')$ şi repartiţiei a priori λ dacă minimizează riscul mediu.

Dacă $p(\cdot)$ desitatea pe spaţiul parametrilor, atunci utilizând formula lui Bayes se poate construi o *densitate de trecere a posteriori*:

$$p_n(\theta|x_1, x_2, ..., x_n) = \frac{p(\theta)p(x_1, ..., x_n; \theta)}{\int_\Theta p(\theta)p(x_1, ..., x_n; \theta)}.$$

Definiţia D.4.6. *Riscul a posteriori* asociat unei estimaţii g_n, unei funcţii de pierdere ω şi unei probabilităţi a priori λ este definit prin:

$$\rho(g_n|x_1, x_2, ..., x_n) = \int_\Theta \omega(\theta, g_n(x_1, x_2, ..., x_n))p_n(\theta|x_1, x_2, ..., x_n)d\theta.$$

Avem următoarea teoremă (vezi *Dumitrescu* [17]):

Teorema D.4.1. *g_n^* este o estimaţie bayesiană asociată funcţiei de pierdere ω şi probabilităţii a priori λ (cu densitatea $p(\cdot)$) dacă şi numai dacă g_n^* minimizează aproape sigur riscul a posteriori corespunzător.*

Anexa E

REPATIŢII STATISTICE

E.1. Repartiţii continue

Vom prezenta o serie de distribuţii continue elementare precum şi principalele lor proprietăţi.

E.1.1. Repartiţia normală

Definiţia E.1.1. (Repartiţia normală). Vom spune că variabila aleatoare X urmează o repartiţie normală de parametrii μ şi σ^2 (vom nota acest lucru prin $X \sim N(\mu, \sigma^2)$) dacă are densitatea de repartiţie:

$$n(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbf{R}.$$

Teorema E.1.1. *Media şi dispersia unei variabile aleatoare X repartizată normal de parametrii μ şi σ^2 este:*

$$M(X) = \mu,$$

respectiv

$$D^2(X) = \sigma^2.$$

Teorema E.1.2. *Funcţia caracteristică a unei variabile aleatoare X reparizată $N(0,1)$ este:*

$$\varphi(t) = e^{-\frac{1}{2}t^2}.$$

Teorema E.1.3. *Dacă X_1 şi X_2 au o repartiţie $N(\mu_1, \sigma_1^2)$ respectiv $N(\mu_2, \sigma_2^2)$ atunci*

$$\alpha X_1 + \beta X_2 \sim N(\alpha\mu_1 + \beta\mu_2, \alpha^2\sigma_1^2 + \beta^2\sigma_2^2)$$

unde $\alpha, \beta \in \mathbf{R}$.

E.1.2. Repartiţia lognormală

Definiţia E.1.2. (Repartiţia lognormală). Vom spune că variabila aleatoare $X(X > 0)$ urmează o repartiţie lognormală (vom nota acest lucru prin $X \sim LN(\mu, \sigma^2)$) dacă logaritmul ei are densitatea de repartiţie $N(\mu, \sigma^2)$.

Teorema E.1.4. *Media şi dispersia unei variabile aleatoare X repartizată lognormal de parametrii μ şi σ^2 este:*

$$M(X) = e^{\mu + \frac{1}{2}\sigma^2},$$

respectiv

$$D^2(X) = e^{2\mu+\sigma^2}(e^{\sigma^2} - 1).$$

Teorema E.1.5. *Dacă variabila aleatoare X urmează o repartiţie lognormală de parametrii μ şi $\sigma^2 \to 0$ atunci repartiţia lognormală standardizată tinde către repartiţia $N(0, 1)$.*

Teorema E.1.6. *Dacă X_1 şi X_2 au o repartiţie $LN(\mu_1, \sigma_1^2)$ respectiv $LN(\mu_2, \sigma_2^2)$ atunci:*

$$e^{\beta} X_1^{\alpha_1} X_2^{\alpha_2} \sim N(\beta + \alpha_1\mu_1 + \alpha_2\mu_2, \alpha_1^2\sigma_1^2 + \alpha_2^2\sigma_2^2)$$

unde $\beta > 0$ şi $\alpha_1, \alpha_2 \in \mathbf{R}$.

E.1.3. Repartiţia uniformă

Definiţia E.1.3. Vom spune că variabila aleatoare X are o repartiţie uniformă în intervalul $[a, b]$ (vom scrie acest lucru prin $X \sim U(a, b)$) dacă densitatea sa de repartiţie este:

$$u(x) = \frac{1}{b-a}, \quad x \in [a, b] \text{şi} \quad u(x) = 0, \quad x \notin [a, b].$$

Teorema E.1.7. *Media şi dispersia unei variabile aleatoare X repartizată $U(a,b)$:*

$$M(X) = \frac{a+b}{2},$$

respectiv

$$D^2(X) = \frac{(b-a)^2}{12}.$$

Teorema E.1.8. *(Teorema limită centrală) Dacă $X_1, ..., X_n$ sunt variabile aleatoare independente identic repartizate $U(0,1)$ atunci pentru valori mari ale lui n :*

$$\frac{\sum_{i=1}^{n} X_i - M(\sum_{i=1}^{n} X_i)}{D(X)} \sim N(0,1).$$

Teorema E.1.9. *Funcţia caracteristică a unei variabile aleatoare $X \sim U(0,1)$ este:*

$$\varphi(t) = \frac{e^{it} - 1}{it}.$$

Exerciţiul E.1.1. Să se găsească densitatea de repartiţie a unei variabilei aleatoare X^2 unde $X \sim U(0,1)$.

Exerciţiul E.1.2. Să se găsească densitatea de repartiţie a unei variabilei aleatoare $X + Y$ unde $X, Y \sim U(0,1)$.

E.1.4. Repartiţia exponenţială

Definiţia E.1.4. Vom spune despre variabila aleatoare X că urmează o repartiţie exponenţială de parametru $\lambda > 0$ (vom scrie acest lucru $X \sim Exp(\lambda)$) dacă densitatea de repartiţie este $f(t) = \lambda e^{-\lambda t}$ pentru $t \geq 0$.

Teorema E.1.10. *Dacă $X \sim Exp(\lambda)$ atunci*

$$M(X) = \frac{1}{\lambda},$$

$$D^2(X) = \frac{1}{\lambda^2}$$

şi

$$M(X^r) = \frac{r!}{\lambda^r}.$$

Teorema E.1.11. *Funcţia caracteristică a unei variabile aleatoare* $X \sim Exp(\lambda)$ *este:*

$$\varphi(t) = \frac{\lambda}{\lambda - i\theta}.$$

Exerciţiul E.1.3. Dacă X şi Y sunt variabile aleatoare independente repartizate $Exp(\lambda)$ atunci

$$\frac{X}{X+Y} \sim Exp(\lambda).$$

E.1.5. Repartiţia gama

Definiţia E.1.5. Vom spune despre o variabilă aleatoare că urmează o repartiţie gama de parametrii α şi β (vom scrie acest lucru $X \sim \gamma(\alpha, \beta)$) dacă are densitatea de repartiţie:

$$f(x) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}}, \quad 0 \leq x < \infty.$$

unde Γ este funcţia lui Euler definită prin formula:

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx.$$

Teorema E.1.12. *Funcţia generatoare de momente a variabilei aleatoare* $X \sim \gamma(\alpha, \beta)$ *este:*

$$G(t) = M(e^{tX}) = \frac{1}{(1-t\beta)^\alpha}.$$

Teorema E.1.13. *Dacă* $X \sim \gamma(\alpha_1, \beta)$ *şi* $Y \sim \gamma(\alpha_2, \beta)$ *sunt variabile aleatoare independente atunci:*

a) $X + Y \sim \gamma(\alpha_1 + \alpha_2, \beta)$;

b) $\frac{X}{Y}$ *are densitatea de repartiţie:*

$$h(x) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} x^{\alpha_1 - 1} (1+x)^{-(\alpha_1+\alpha_2)}, \quad x \geq 0.$$

Generalizând punctul c) din teorema anterioară, la n variabile aleatoare, obţinem repartiţia Dirichlet.

Repartiţia Dirichlet

Teorema E.1.14. *Dacă $X_i \sim \gamma(\alpha_i, 1)$ sunt $k+1$ variabile aleatoare independente atunci repartiţia variabilelor aleatoare $Y_1, ..., Y_{k+1}$ date de:*

$$\begin{cases} Y_j = \frac{X_j}{X_1+....+X_{k+1}}, \quad j = 1, ..., k \\ Y_{k+1} = X_1 + ... + X_{k+1} \end{cases}$$

are forma (repartiţia Dirichlet):

$$\frac{\Gamma(\sum\limits_{i=1}^{k+1} \alpha_i) \prod\limits_{i=1}^{k} y_i^{\alpha_i - 1}(1 - \sum\limits_{i=1}^{k} y_i)^{\alpha_{k+1}-1}}{\prod\limits_{i=1}^{k+1} \Gamma(\alpha_i)},$$

unde $0 < y_j$ şi $y_1 + ... + y_k < 1$.

Repartiţia χ^2

Un alt caz particular important al repartiţiei gama se obţine pentru $\alpha = \frac{n}{2}$ cu orice n întreg pozitiv şi $\beta = 2\sigma^2$. Repartiţia astfel obţinută se numeşte repartiţia χ^2 cu n grade de libertate.

Observaţia E.1.1. Pentru $n = 2$ repartiţia χ^2 devine repartiţia exponenţială.

Teorema E.1.15. *Dacă $X \sim \chi^2(n)$, atunci*

$$M(X^r) = 2^r \sigma^{2r} \frac{\Gamma(r + \frac{n}{2})}{\Gamma(\frac{n}{2})}$$

iar funcţia generatoare de momente este:

$$G(t) = M(e^{tX}) = \frac{1}{(1 - 2t\sigma^2)^{\frac{n}{2}}}.$$

Teorema E.1.16. *Dacă $X_i \sim N(0, \sigma^2)$ sunt n variabile independente atunci variabila aleatoare:*

$$Y = \sum_{i=1}^{n} X_j^2 \sim \chi^2(n).$$

Teorema E.1.17. *Dacă* $X \sim \chi^2(n)$, *atunci asimptotic*

$$\frac{X - n\sigma^2}{\sigma^2\sqrt{2n}} \sim N(0,1).$$

De obicei, cunatilele repartiţiei $\chi^2(n)$ sunt tabelate până la $n = 30$, deoarce pentru $n > 30$ este posibil să se folosească cunatilele repartiţiei $N(0,1)$.

E.1.6. Repartiţia beta

Definiţia E.1.6. Vom spune că variabila aleatoare X urmează o repartiţiebeta de parametrii α şi β (vom scrie acest lucru ca $X \sim Be(\alpha, \beta)$), dacă are densitatea de repartiţie:

$$f(x) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, & 0 < x < 1; \alpha, \beta > 0 \\ 0 & \text{în rest.} \end{cases}$$

Repartiţia F

Definiţia E.1.7. Repartiţia $F(m,n)$ este obţinută ca o transformare a repartiţie beta:

$$Y = \frac{\beta X}{\alpha(1-X)} \text{ unde } X \sim Be(\alpha, \beta)$$

cu $\alpha = \frac{m}{2}$ şi $\beta = \frac{n}{2}$. Distribuţia corespunzătoare variabilei $Y \sim F(m,n)$ este:

$$f(x) = \begin{cases} \frac{\Gamma(\frac{m+n}{2})}{\Gamma(\frac{m}{2})\Gamma(\frac{n}{2})} \left(\frac{\alpha}{\beta}\right)^{\alpha} y^{\alpha-1}(1+\frac{\alpha}{\beta}y)^{-(\alpha+\beta)}, & 0 < x < 1; \alpha, \beta > 0 \\ 0 & \text{în rest.} \end{cases}$$

Teorema E.1.18. *Dacă* $X \sim \chi^2(m)$ *şi* $Y \sim \chi^2(n)$ *atunci:*

$$F = \frac{X/m}{Y/n} \sim Be(\alpha, \beta).$$

Repartiţia t

Definiţia E.1.8. Repartiţia $t(n)$ (t cu n grade de liberatate) este obţinută ca o transformare a repartiţie $F(1;n)$ şi anume:

$$t = \sqrt{X}$$

iar densitatea de repartiţie corespunzătoare este:

$$f(x) = \begin{cases} \frac{\Gamma(\frac{n+1}{2})}{\sqrt{\pi n}\Gamma(\frac{n}{2})}(1+\frac{x^2}{n})^{-\frac{n+1}{2}}, & -\infty < x < \infty; n = 1, 2, 3, ... \\ 0 & \text{în rest.} \end{cases}$$

Teorema E.1.19. *Dacă* $X \sim N(0,1)$ *şi* $Y \sim \chi^2(n)$ *atunci*

$$t = \frac{X}{\sqrt{\frac{Y}{n}}} \sim t(n).$$

E.1.7. Repartiţia Cauchy

Definiţia E.1.9. Repartiţia Cauchy $\mathcal{C}(\mu;\theta)$ este definită de densitatea de repartiţie:

$$f(x) = \frac{\mu}{\pi}\frac{1}{\mu^2 + (x-\theta)^2}, \quad x \in \mathbf{R},\ \mu > 0.$$

Observaţia E.1.2. Repartiţia Cauchy este un exemplu de repartiţie care nu are momente.

E.2. Distribuţii discrete

Vom prezenta principalele distribuţii discrete: Bernoulli, Binomială, Poisson, hipergeometrică şi geometrică.

E.2.1. Distribuţia Bernoulli

Definiţia E.2.1. Distribuţia Bernoulli de parametru p este dată de $\Pr(X = 1) = p$ şi $\Pr(X = 0) = 1 - p$, cu $p \in [0, 1]$.

Teorema E.2.1. *Dacă variabila* X *urmează o distribuţie Bernoulli de parametru* p *atunci media este* $M(X) = p$ *iar dispersia* $D^2(X) = p(1-p)$.

E.2.2. Distribuţia binomială

Definiţia E.2.2. Distribuţia binomială $Bi(n,p)$ de parametrii n şi p este dată de $Pr(X = i) = C_n^i p^{n-i}(1-p)^i$ pentru $i = 0, ..., n$.

Teorema E.2.2. *Dacă variabila X urmează o distribuţie binomială $Bi(n,p)$ atunci media este $M(X) = np$ iar dispersia $D^2(X) = np(1-p)$.*

Teorema E.2.3. *Dacă $X_1, ..., X_n$ sunt n variabile aleatoare independente repartizate Bernoulli de parametru p atunci variabila $\sum_{i=1}^{n} X_i$ urmează o repartiţie binomială $Bi(n,p)$.*

Observaţia E.2.1. Distribuţia binomială $Bi(n,p)$ de parametrii n şi p este modelată de *selecţia cu revenire* dintr-o urnă $U(a,b)$ care conţine a bile albe şi b bile negre şi este probabilitatea ca din n extrageri să se obţină k bile albe:

$$\Pr(X = k) = C_n^k \left(\frac{a}{a+b}\right)^k \left(1 - \frac{n}{a+b}\right)^{n-k}.$$

E.2.3. Distribuţia Poisson

Distribuţia Poisson se obţine din distribuţia $Bi(n,p)$ când $n \to \infty$, $p \to 0$ şi produsul $n \cdot p = \lambda$ este constant. Funcţia densitate de probabilitate este dată de:

$$\Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, 2, ...$$

E.2.4. Distribuţia hipergeometică

Distribuţia hipergeometrică este modelată de o *selecţie fără revenire* dintr-o urnă $U(a,b)$ care conţine a bile albe şi b bile negre şi este probabilitatea ca din n extrageri să se obţină k bile albe:

$$\Pr(X = k) = \frac{C_a^k C_b^{n-k}}{C_{a+b}^n} \text{ unde } n \leq a + b \text{şi } k \leq a.$$

E.2.5. Distribuţia geometrică

Distribuţia geometrică are funcţia de densitate de probabilitate dată de:

$$\Pr(X = n) = p(1-p)^n, \text{ pentru } n = 0, 1, ...$$

E.3. Calculul numeric al cuantilelor

Vom prezenta două metode numerice de calcul a cuantilei de ordinul α pentru repartiţia $N(0,1)$ respectiv repartiţia $\chi^2(n)$.

E.3.1. Cuantila repartiţiei normale

Fără restrângerea generalităţii putem presupune că $\alpha \in [0, 0, 5]$. Atunci, utilizând tehnica dezvoltării în fracţii continue, vom obţine:

$$u_\alpha = \omega - \frac{\sum_{i=0}^{2} a_i \omega^i}{\sum_{i=0}^{3} b_i \omega^i}, \ \omega = \sqrt{\ln \frac{1}{\alpha^2}},$$

unde constantele a_i şi b_i sunt:

$$\begin{cases} a_0 = 2,515517 \\ a_1 = 0,802853 \\ a_2 = 0,010328 \\ b_0 = 1 \\ b_1 = 1,432877 \\ b_2 = 0,189269 \\ b_3 = 0,001308. \end{cases}$$

E.3.2. Cuantilele repartiţiei chi-pătrat

Cuantila de ordinul α a repartiţiei $\chi^2(n)$, pentru $n > 30$, este dată de:

$$h_\alpha = n + \sqrt{2n} u_\alpha + \frac{2}{3} u_\alpha^2 - \frac{2}{3} + O(\frac{1}{\sqrt{n}}).$$

Pentru detalii se poate consulta *Abramowitz* şi *Stegun* [1].

Anexa F

SERII DINAMICE STAŢIONARE

F.1. Exemple de serii dinamice

Prezentăm conceptul de staţionaritate, strict staţionaritate, funcţiile de autocorelaţie şi de autocovarianţă precum şi tehnici şi metode de eliminare a componentei tendinţă şi a componentei sezoniere. O *serie dinamică* este o mulţime de observaţii x_t înregistrate la un moment specificat de timp t. Vom nota o serie dinamică cu mulţimea de timp continuă T cu $x(t), t \in T$.

Exemple de serii dinamice sunt: curentul printr-un rezistor, populaţia unei colectivităţi, numărul pasagerilor unei companii aviatice, etc.

F.2. Procese stochastice

Primul pas în analiza unei serii dinamice constă în alegerea unui model (sau clase de modele) pentru date. Pentru a specifica natura probabilistă a observaţiilor este natural să presupunem că fiecare observaţie x_t este o realizare a unei variabile aleatoare X_t.

Definiţia F.2.1. Un proces stochastic este o familie de variabile aleatoare $\{X_t, t \in T\}$ definite pe un spaţiu de probabilitate $(\Omega, \mathcal{F}, P)$.

Definiţia F.2.2. Funcţiile $\{X_\bullet(\omega), \omega \in \Omega\}$ definite pe T se numesc realizările procesului $\{X_t, t \in T\}$.

Exemplul F.2.1. (Sinusoida cu şi fără amplitudine) Fie A şi Θ variabile aleatoare independente cu $A \geq 0$ şi $\Theta \in [0, 2\pi)$ uniform distribuite.

Un proces stochastic $\{X(t), t \in \mathbf{R}\}$ se poate defini în termenii lui A şi Θ pentru orice $\nu \geq 0$ şi $r > 0$ specificaţi:

$X_t = \frac{A}{r}\cos(\nu t + \Theta)$

sau

$$X_t(\omega) = \frac{1}{r}A(\omega)\cos(\nu t + \Theta(\omega))$$

unde ω este un element al spaţiului de probabilitate Ω pe care sunt definiţi A şi Θ.

Realizările procesului sunt funcţii de t şi se obţin fixându-l pe ω adică sunt de forma:

$$x(t) = \frac{1}{r}a\cos(\nu t + \theta).$$

Exemplul F.2.2. Procesul binar $\{X_t; t = 1, 2, ...\}$ unde X_t sunt variabile aleatoare independente astfel încât:

$$P(X_t = 1) = P(X_t = 0) = 0,5$$

Conform teoremei Kolmogorov există un spaţiu de probabilitate $(\Omega, \mathcal{F}, P)$ cu variabilele $X_1, X_2, ...$ definite pe Ω astfel încât:

$$P(X_1 = i_1, ..., X_n = i_n) = \frac{1}{2^n} \quad \forall (i_1, ..., i_n) \in \{0, 1\}^n$$

Exemplul F.2.3. Mersul aleatoriu simetric (symetric random walk) este procesul $\{S_t, t = 0, 1, ...\}$ definit de starea iniţială S_0 şi de

$$S_t = \sum_{i=1}^{t} X_i, \ t \geq 1$$

Forma generală a mersului aleatoriu este definită analog înlocuind şirul de variabile $X_1, X_2, ...$ cu un şir de variabile aleatoare independente şi identic distribuite care nu satisfac restricţia din exemplul precedent. Existenţa unui astfel de şir este asigurată de teorema Kolmogorov.

Exemplul F.2.4. Procesul de ramificaţie Galton-Watson, care modelează creşterea unei populaţiei, este definit de ecuaţiile $X_0 = x$ (mărimea populaţiei la momentul zero) şi

$$X_{t+1} = \sum_{j=1}^{X_t} Z_{t,j},$$

$t = 0, 1, 2, ...$ unde $Z_{t,j}$, $t = 0, 1, ..., j = 1, 2, ...$ sunt variabile aleatoare nenegative independente şi identic distribuite. Vom folosi procesul Galton-Watson la atacul algoritmului de cifrare **A5** utilizat de sistemul de telefonie mobilă GSM.

Definiţia F.2.3. (Funcţia de distribuţie a unui proces stochastic $\{X_t, t \in T \subset \mathbf{R}\}$). Fie $\mathcal{T}$ mulţimea tuturor vectorilor de forma $\{\mathbf{t} = (t_1, ..., t_n) \in T^n : t_1 < t_2 < ... < t_n, n = 1, 2, ..\}$. Atunci funcţiile de distribuţie finit dimensionale ale lui $\{X_t, t \in T\}$ sunt funcţiile $\{F_t(.), \mathbf{t} \in \mathcal{T}\}$ definite pentru $\mathbf{t} = (t_1, ..., t_n)$ de

$$F_{\mathbf{t}}(\mathbf{x}) = P(X_{t_1} \leq x_1, ..., X_{t_n} \leq x_n), \qquad \mathbf{x} = (x_1, ..., x_n)' \in \mathbf{R}^n$$

Teorema F.2.1. *(Kolmogorov) Funcţiile distribuţie de probabilitate* $\{F_t(.), \mathbf{t} \in \mathcal{T}\}$ *sunt funcţii de distribuţie ale unui proces stochastic dacă şi numai dacă*

$$\underset{n\in\mathbf{N}}{(\forall n)}\underset{t\in\mathcal{T}}{(\forall t)}\underset{i=\overline{1,n}}{(\forall i)} \ (\lim_{x_i \to \infty} F_{\mathbf{t}}(\mathbf{x}) = F_{\mathbf{t}(i)}(\mathbf{x}(i))) \qquad \mathbf{(1)}$$

unde $\mathbf{t}(i)$ *şi* $\mathbf{x}(i)$ *sunt vectorii* $(n-1)$*- dimensionali obţinuţi din* $\mathbf{t}$ *respectiv din* $\mathbf{x}$ *prin eliminarea componentei* i *din aceşti vectori.*

Dacă $\phi_{\mathbf{t}}(.)$ este *funcţia caracteristică* corespunzătoare lui $F_t(.)$ adică

$$\phi_{\mathbf{t}}(x) = \int_{\mathbf{R}^n} e^{i\mathbf{u}'\mathbf{x}} F_{\mathbf{t}}(dx_1, ...dx_n), \qquad \mathbf{u} = (u_1, ..., u_n)' \in \mathbf{R}^n$$

atunci (1) este echivalentă cu

$$\lim_{u_i \to \infty} \phi_{\mathbf{t}}(\mathbf{u}) = \phi_{\mathbf{t}(i)}(\mathbf{u}(i))$$

unde $\mathbf{u}(i)$ este vectorul $(n-1)$- dimensional obţinut din $\mathbf{u}$ prin eliminarea componentei i.

F.3. Staţionaritate şi strict staţionaritate

Când avem un număr finit de variabile aleatoare pentru a vedea dependenţa dintre acestea trebuie să calculăm *matricea de covarianţă*. Vom extinde acest concept la un număr infinit de variabile aleatoare.

Definiţia F.3.1. (Funcţia de autocovarianţă). Dacă $\{X_t, t \in T\}$ este un proces astfel încât $Var(X_t) < \infty$ pentru orice $t \in T$, funcţia de autocovarianţă $\gamma_X(.,.)$ a procesului $\{X_t, t \in T\}$ este definită ca:

$$\gamma_X(r, s) = Cov(X_r, X_s) = E[(X_r - EX_r)(X_s - EX_s)], \qquad r, s \in T$$

Definiţia F.3.2. (Staţionaritate) O serie dinamică $\{X_t, t \in \mathbf{Z}\}$ cu mulţimea de timp $\mathbf{Z}$ se numeşte staţionară dacă

i) $E|X_t|^2 < \infty, \quad \forall t \in \mathbf{Z}$

ii)$EX_t = m, \quad \forall t \in \mathbf{Z}$

iii)$\gamma_X(r,s) = \gamma_X(r+t, s+t), \quad \forall r,s,t \in \mathbf{Z}$

Observaţia F.3.1. Conceptul de staţionaritate definit mai sus se numeşte şi *staţionaritate slabă*.

Observaţia F.3.2. Dacă $\{X_t, t \in \mathbf{Z}\}$ este staţionară atunci $\gamma_X(r,s) = \gamma_X(r-s,0)$ $\forall r,s \in \mathbf{Z}$. Vom redefini funcţia de autocovarianţă a unui proces stochastic staţionar ca funcţie de o variabilă

$$\gamma_X(h) \equiv \gamma_X(h,0) = Cov(X_{t+h}, X_t) \qquad \forall t,h \in \mathbf{Z}$$

Ne vom referi la funcţia $\gamma_X(.)$ ca la funcţia de autocovarianţă a procesului $\{X_t, t \in \mathbf{Z}\}$ şi la $\gamma_X(h)$ ca la valoarea acesteia la *lag* h. Funcţia de autocorelaţie (f.a.c) a procesului $\{X_t, t \in \mathbf{Z}\}$ este definită ca:

$$\rho_X(h) \equiv \gamma_X(h)/\gamma_X(0) = Corr(X_{t+h}, X_t) \qquad \forall t,h \in \mathbf{Z}$$

Definiţia F.3.3. (Strict staţionaritate) O serie dinamică $\{X_t, t \in \mathbf{Z}\}$ se numeşte strict staţionară dacă distribuţiile vectorilor $(X_{t_1}, ..., X_{t_k})'$ şi $(X_{t_1+h}, ..., X_{t_k+h})'$ sunt aceleaşi pentru $\forall k$ şi pentru $\forall t_1, ..., t_k, h \in \mathbf{Z}$

F.3.1. Relaţia dintre Staţionaritate şi Strict Staţionaritate

Dacă $\{X_t, t \in \mathbf{Z}\}$ este strict staţionară luând $k = 1$ în definiţia anterioară rezultă imediat că X_t are aceeaşi distribuţie pentru orice $t \in \mathbf{Z}$. Dacă $E|X_t|^2 < \infty$ rezultă că EX_t şi $Var(X_t)$ sunt constante. Dacă luăm $k = 2$ în definiţia strict staţionarităţii găsim că X_{t+h} şi X_t au aceeaşi distribuţie deci aceeaşi covarianţă pentru orice h $\in \mathbf{Z}$. Deci un proces strict staţionar cu momente de ordinul doi finite este staţionar.

Reciproca afirmaţiei precedente nu este adevărată. De exemplu dacă $\{X_t, t \in \mathbf{Z}\}$ este un şir de variabile aleatoare independente astfel încât pentru n impar X_t este distribuit exponenţial cu media 1 şi pentru n par X_t este distribuit normal cu medie 1 şi dispersie 1 atunci X_t este staţionar cu $\gamma_X(0) = 1$ şi $\gamma_X(h) = 0$ pentru $h \neq 0$. Dar X_1 şi X_2 au distribuţii diferite deci $\{X_t, t \in \mathbf{Z}\}$ nu poate fi strict staţionar.

Vom da un exemplu de proces stochastic în care strict staţionaritatea implică slab staţionariatatea.

Definiţia F.3.4. (Serii dinamice Gaussiene) Procesul $\{X_t, t \in \mathbf{Z}\}$ este o serie dinamică Gaussiană dacă şi numai dacă toate funcţiile de distribuţie ale lui $\{X_t, t \in \mathbf{Z}\}$ sunt normale multidimensionale.

Dacă $\{X_t, t \in \mathbf{Z}\}$ este un proces staţionar Gaussian atunci $\{X_t, t \in \mathbf{Z}\}$ este strict staţionar deoarece $\forall n \in \{1, 2, ...\}$ şi oricare $h, t_1, t_2... \in \mathbf{Z}$ vectorii aleatori $(X_{t_1}, ..., X_{t_k})'$ şi $(X_{t_1+h}, ..., X_{t_k+h})'$ au aceeaşi medie şi aceeaşi matrice de covarianţă deci aceeaşi distribuţie.

Exemplul F.3.1. Fie $X_t = A\cos\theta t + B\sin\theta t$ unde A şi B sunt două variabile aleatoare necorelate de medie zero şi dispersie unu iar $\theta \in [-\pi, \pi]$. Această serie este staţionară deoarece:

$$Cov(X_{t+h}, X_t) = Cov(A\cos(\theta(t+h)) + B\sin(\theta(t+h)), A\cos(\theta t) + B\sin(\theta t)) =$$

$$= \cos(\theta t)\cos(\theta(t+h)) + \sin(\theta t)\sin(\theta(t+h)) = \cos(\theta h)$$

care nu depinde de t.

Exemplul F.3.2. Definim procesul $X_t = Z_t + \theta Z_{t-1}$ unde Z_t este un şir de variabile aleatoare independente identic distribuite de medie zero şi dispersie σ_z^2. Funcţia de autocovarianţă este dată de:

$$Cov(X_{t+h}, X_t) = Cov(Z_{t+h} + \theta Z_{t+h-1}, Z_t + \theta Z_{t-1}) = \begin{cases} (1+\theta^2)\sigma_z^2 \text{ dacă } h = 0 \\ \theta\sigma_z^2 \text{ dacă } h = \pm 1 \\ 0 \text{ dacă } |h| > 1 \end{cases}$$

şi deci $\{X_t\}$ este staţionar. Se poate arăta că $\{X_t\}$ este strict staţionar.

Exemplul F.3.3. Fie Y_t o serie dinamică staţionară. Definim seria dinamică X_t prin: $X_t = Y_t$ dacă t este par şi $X_t = Y_t + 1$ dacă t este impar. Chiar dacă $Cov(X_{t+h}, X_t) = \gamma_Y(h)$, seria $\{X_t\}$ nu este staţionară deoarece nu are media constantă.

Exemplul F.3.4. (Referitor la exemplul 3). Fie $S_t = X_1 + X_2 + ... + X_t$ mersul aleatoriu unde (X_i) sunt variabile aleatoare independente identic repartizate de medie zero şi dispersie σ^2. Pentru $h > 0$

$$Cov(S_{t+h}, S_t) = Cov(\sum_{i=1}^{t+h} X_i, \sum_{j=1}^{t} X_j) = Cov(\sum_{i=1}^{t} X_i, \sum_{j=1}^{t} X_j) = \sigma^2 t$$

care depinde de t deci, seria S_t este nestaţionară.

Procesele staţionare joacă un rol imporant în analiza seriilor dinamice. Multe dintre serii dinamice observate sunt nestaţionare. Acestea sunt transformate în serii dinamice staţionare folosind tehniciile prezentate în paragraful următor.

Teoria seriilor dinamice staţionare (dezvoltată în ultima parte a lucrării) este folosită la analiza şi predicţia seriilor rezultate. În toate acesta funcţia de autocorelaţie joacă un rol improtant. Proprietăţile ei vor fi prezentate în unul din paragrafele următoare.

Exerciţiul F.3.1. Fie $a, b, c \in \mathbf{R}$ şi Z_t variabile aleatoare independente repartizate $N(0, \sigma^2)$. Să se indice care dintre procesele de mai jos este staţionar. Pentru fiecare proces staţionar să se indice media şi să construiască funcţia de autocovarianţă şi funcţia de autocorelaţie.

1.$X_t = a + bZ_t + cZ_{t-1}$

2.$X_t = Z_t Z_{t-1}$

3.$X_t = a + bZ_t$

Exerciţiul F.3.2. Fie $\{X_t\}$ şi $\{Y_t\}$ două procese staţionare necorelate. Arătaţi că $\{X_t + Y_t\}$ este staţioanar şi că funcţia de autocovarianţă este suma funcţiilor de autocovarianţă a proceselor $\{X_t\}$ şi $\{Y_t\}$.

F.4. Estimarea şi eliminarea componentelor tendinţă şi sezoniere

Primul pas ce îl facem în analiza unei serii dinamice este să îi facem graficul. Dacă sunt discontinuităţi vizibile în această serie, ca de exemplu schimbarea bruscă a nivelului, se recomandă să se analizeze seria rupând-o în segmente omogene. Dacă sunt date neuzuale ele trebuie analizate cu atenţie pentru a vedea dacă este justificat să le îndepărtăm (poate au fost înregistrate din greşeală).

În continuare trebuie să stabilizăm dispersia seriei dinamice $\{X_t, t \in T\}$. Pentru a realiza acest lucru putem proceda în felul următor:

i) dacă deviaţia standard a unei serii dinamice este proporţională cu nivelul acesteia atunci transformăm seria X_t în $Y_t = logX_t$. Seria Y_t are dispersia constantă în timp.

ii) dacă varianţa seriei dinamice este proporţională cu nivelul acesteia atunci transformăm seria X_t în $Y_t = X_t^{\frac{1}{2}}$. Seria Y_t are dispersia constantă în timp.

Cele două transformări de mai sus sunt cazuri particulare ale transformărilor *Box-Cox* definite astfel

$$Y_t = \frac{X_t^{\lambda} - 1}{\lambda}.$$

De exemplu pentru $\lambda \to 0$ obţinem transformarea i) iar pentru $\lambda = \frac{1}{2}$ obţinem o transformare echivalentă cu ii).

În continuare realizăm descompunerea

$$X_t = m_t + s_t + Y_t$$

unde m_t este o funcţie care variază lent cunoscută ca *componenta de tendinţă* (direcţie, trend), s_t este o funcţie periodică numită *componenta sezonieră* iar Y_t este *componenta zgomot alb* care este staţionară în sensul definiţiei 5.

Scopul acestui paragraf este de a estima şi extrage *componentele deterministe* m_t şi s_t în speranţa că *componenta reziduală* Y_t va fi un proces staţionar. Vom putea apoi să găsim un model probabilist pentru $\{Y_t\}$, să-i analizăm proprietăţile şi să-l folosim împreună cu m_t şi s_t pentru predicţia lui X_t.

O abordare alternativă (Box şi Jenkins) este de a aplica operatorul diferenţiere, eventual succesiv, datelor $\{X_t\}$ până când rezultatele sunt realizările unui proces staţionar $\{W_t\}$. Vom folosi apoi teoria proceselor staţionare pentru modelarea, analiza şi predicţia lui $\{W_t\}$ şi deci a seriei originale.

F.4.1. Eliminarea tendinţei în absenţa componenetei sezoniere

În absenţa componentei sezoniere modelul devine:

$$X_t = m_t + Y_t, \qquad t = 1, 2, ..., n$$

Fără restrângerea generalităţii putem presupune că $EY_t = 0$

Metoda 1 (Metoda celor mai mici pătrate).

Luăm m_t de forma $a_0 + a_1 t + a_2 t^2$. Numerele a_i se determină atfel încât $\sum_{t=1}^{n} (x_t - m_t)^2$ să fie minim.

Metoda 2 (Netezire prin proces MA).

Fie $q \in \mathbf{N}$. Să considerăm procesul dublu-MA

$$W_t = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t+j}$$

al procesului $\{X_t\}$. Pentru $q + 1 \leq t \leq n - q$ avem

$$W_t = \frac{1}{2q+1} \sum_{j=-q}^{q} m_{t+j} + \frac{1}{2q+1} \sum_{j=-q}^{q} Y_{t+j} \simeq m_t$$

dacă presupunem că m_t este liniar în intervalul $[t - q, t + q]$ şi că media termenilor eroare din acest interval este aproape de zero. Avem deci estimarea

$$\hat{m}_t = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t+j} \qquad q+1 \leq t \leq n-q \qquad \textbf{(2)}$$

Deoarece X_t nu este observat pentru $t \leq 0$ sau $t > n$ nu putem folosi relaţia de mai sus pentru $t \leq q$ sau $t > n-q$. Putem însă să folosim procesul simplu -MA ca de exemplu:

$$\hat{m}_t = \sum_{j=0}^{n-t} \alpha(1-\alpha)^j X_{t+j} \qquad t = 1, ..., q$$

şi

$$\hat{m}_t = \sum_{j=0}^{n-1} \alpha(1-\alpha)^j X_{t-j} \qquad t = n-q+1, ..., n$$

relaţiile de mai sus nu depind prea mult de α. Chatfield(1974) a găsit în mod empiric că pentru $\alpha \in (0.1, 0.3)$ estimaţiile sunt satistfăcătoare.

Estimaţia $\{\hat{m}_t\}$dată de (2) poate fi gândită ca un proces obţinut din $\{X_t\}$ prin aplicarea operatorului filtru liniar $\hat{m}_t = \sum\limits_{j=-\infty}^{\infty} X_{t+j}$ cu ponderile $a_j = \frac{1}{2q+1}$, $-q \leq j \leq q$ şi $a_j = 0$, $|j| > q$. Acest filtru particular este un *filtru trece jos* deoarece din datele $\{x_t\}$ îndepărtează fluctuaţiile rapide (sau frecvenţele înalte) ale componenetei $\{Y_t\}$şi scoate estimarea componentei tendinţă care variază lent $\{\hat{m}_t\}$.

Metoda 3 (Diferenţiere pentru a genera date staţionare).

În loc să îndepărtăm din $\{X_t\}$componenta zgomot ca în Metoda 2, eliminăm componeneta tendinţă prin diferenţiere. Vom defini mai întâi *operatorul de diferenţiere* ∇ prin

$$\nabla X_t = X_t - X_{t-1} = (1-B)X_t$$

unde B este *operatorul tranzlaţie temporală* definit de:

$$BX_t = X_{t-1}$$

Puterile operatorilor B şi ∇ se definesc în mod natural adică $B^j(X_t) = X_{t-j}$ şi $\nabla^j(X_t) = \nabla(\nabla^{j-1}(X_t))$, $j \geq 1$ cu $\nabla^0(X_t) = X_t$. Dacă operatorul ∇ se aplică *tendinţei liniare* $m_t = a + bt$ atunci $\nabla m_t = a$. În mod analog orice tendinţă polinomială de grad k se reduce la o constantă prin aplicarea operatorului ∇^k.

Având deci modelul $X_t = m_t + Y_t$ unde $m_t = \sum_{j=0}^{k} a_j t^j$ şi Y_t este staţionară cu media zero obţinem:

$$\nabla^k X_t = k! a_k + \nabla^k Y_t$$

un proces staţionar cu media k!a_k. În practică operatorul de diferenţiere are gradul k mic (1 sau 2).

F.4.2. Eliminarea simultană a componentelor tendinţă şi sezoniere

Metodele decrise în secţiunea precedentă pot fi adaptate în mod natural pentru a elimina simultan componenta tendinţă şi componenta sezonieră.

Să considerăm modelul

$$X_t = m_t + s_t + Y_t$$

unde $EY_t = 0$ şi $s_{t+d} = s_t$ şi $\sum_{j=1}^{d} s_j = 0$.

Notăm cu $x_{j,k}, \quad j = 1, ..., n_u; \quad k = 1, ..., n_s$ al k-lea element din unitatea j(ex.: luna k din anul j şi $d = 12$). Deci $x_{j,k} = x_{k+n_s(j-1)}$, $j = 1, ..., n_u; k = 1, ..., n_s$.

Metoda S1(metoda tendinţei mici). Dacă tendinţa este mică putem presupune că termenul tendinţă este constant să presupunem m_j pentru a -j-a unitate. Deoarece $\sum_{k=1}^{n_s} s_k = 0$ un estimator natural nedelasat pentru m_j este

$$\hat{m_j} = \frac{1}{n_s} \sum_{k=1}^{n_s} x_{j,k}$$

iar pentru $s_k, k = 1, ..., n_s$

$$\hat{s_k} = \frac{1}{n_u} \sum_{k=1}^{n_u} (x_{j,k} - \hat{m_j})$$

care satisface condiţia $\sum_{k=1}^{n_s} s_k = 0$. Estimaţia termenului eroare va fi:

$$\hat{Y}_{j,k} = x_{j,k} - \hat{m_j} - \hat{s_k}, \quad j = 1, ..., n_u; k = 1, ..., n_s$$

Metoda S2(Estimaţia MA).

Această tehnică este de preferat deoarece nu se bazează pe presupunearea că m_t este aproximativ constantă în timpul unui ciclu. Să presupunem că avem observaţiile $\{x_1, ..., x_n\}$. Tendinţa este estimată prin aplicarea unui filtru MA care elimină componenta sezonieră. Dacă perioada d este pară ($d = 2q$) atunci folosim:

$$\hat{m}_j = \frac{1}{d}(0.5x_{t-q} + x_{t-q+1} + ... + x_{t+q-1} + 0.5x_{t+q}) \quad q < t \leq n-q$$

Dacă perioada este impară folosim filtrul definit de **(2)**.

Al doilea pas este să estimăm componenta sezonieră. Pentru fiecare $k = 1, ..., d$ calculăm media deviaţiilor $\{(x_{k+jd} - \hat{m}_{k+jd}: q < k + jd \leq n - q\}$. Estimaţia componentei s_k va fi

$$\hat{s}_k = w_k - \frac{1}{d}\sum_{i=1}^{d} w_i, \quad k = 1, ..., d$$

şi $\hat{s}_k = \hat{s}_{k-d}, \quad k > d.$

Seria *desezonalizată* va fi $d_t = x_t - \hat{s}_t, \quad t = 1, ..., n.$

Estimaţia termenului eroare va fi:

$$\hat{Y}_t = x_t - \hat{m}_t - \hat{s}_t, \quad t = 1, ..., n.$$

Metoda S3(Diferenţierea la lag d).

Tehnica diferenţierii prezentată în secţiunea precedentă poate fi adaptată pentru cazul sezonier de perioadă d prin introducerea *operatorului diferenţiere la lag-d* ∇_d definit de

$$\nabla_d X_t = X_t - X_{t-d} = (1 - B^d)X_t$$

(Acest operator nu trebuie confundat cu operatorul $\nabla^d = (1 - B)^d$ definit anterior.)

Aplicând operatorul ∇_d modelului:

$$X_t = m_t + s_t + Y_t,$$

unde $\{s_t\}$ are perioada d obţinem:

$$\nabla_d X_t = m_t - m_{t-d} + Y_t - Y_{t-d}$$

care dă descompunerea seriei deferenţă $\nabla_d X_t$ în componenta tendinţă $m_t - m_{t-d}$ şi termenul zgomot $Y_t - Y_{t-d}$. Componenta tendinţă, $m_t - m_{t-d}$, poate fi estimată folosind metodele descrise anterior prin aplicarea unor puteri ale operatorului ∇.

F.5. Funcţia de autocovarianţă a unui proces staţionar

În acest paragraf vom prezenta o serie de proprietăţi ale funcţiei de autocovarianţă a unui proces stochastic staţionar.

Teorema F.5.1. *(Proprietate elementară). Dacă $\gamma(.)$ este funcţia de autocovarianţă a unui proces staţionar $\{X_t, t \in \mathbf{Z}\}$. Atunci:*
$\gamma(0) \geq 0$,
$|\gamma(h)| \leq \gamma(0) \quad \forall h \in \mathbf{Z}$,
$\gamma(h) = \gamma(-h) \quad \forall h \in \mathbf{Z}$.

Demonstraţie.
Prima proprietate este evidentă din faptul că $Var(X_t) \geq 0$, iar cea de-a doua din inegalitatea *Cauchy-Schwarz:*

$$|\gamma(h)| = |Cov(X_{t+h}, X_t)| \leq \sqrt{Var(X_{t+h})Var(X_t)} = \sqrt{\gamma(0)\gamma(0)} = \gamma(0)$$

şi cea de-a treia se stabileşte observând că:

$$\gamma(-h) = Cov(X_{t-h}, X_t) = Cov(X_t, X_{t+h}) = \gamma(h). \qquad \square$$

Funcţia de autocovarianţă este *nenegativ definită.*

Definiţia F.5.1. (Nenegativ definirea) O funcţie reală $k : \mathbf{Z} \to \mathbf{R}$ se numeşte nenegativ definită dacă şi numai dacă

$$\sum_{i,j}^{n} a_i k(t_i - t_j) a_j \geq 0$$

pentru orice $n \in \mathbf{Z}$ şi orice $\mathbf{a} = (a_1, ..., a_n)' \in \mathbf{R}^n$ şi $\mathbf{t} = (t_1, ..., t_n)' \in \mathbf{Z}^n$.

Teorema F.5.2. *(Caracterizarea funcţiei de autocovarianţă) O funcţie reală pară definită pe $\mathbf{Z}$ este nenegativ definită dacă şi numai dacă este funcţia de autocovarianţă a unei serii dinamice staţionare.*

Demonstraţie.
Pentru a arăta că funcţia de autocovarianţă $\gamma(.)$ a unei serii dinamice staţionare $\{X_t\}$ este nenegativ definită observăm că dacă $\mathbf{a} = (a_1, ..., a_n)' \in \mathbf{R}^n$ şi $\mathbf{t} = (t_1, ..., t_n)' \in \mathbf{Z}^n$ şi $\mathbf{Z}_t = (X_{t_1} - EX_{t_1}, ..., X_{t_n} - EX_{t_n})'$ atunci:

$$0 \leq Var(\mathbf{a}'\mathbf{Z}_t) = \mathbf{a}' E\mathbf{Z}_t\mathbf{Z}_t'\mathbf{a} = \mathbf{a}'\Gamma_n\mathbf{a} = \sum_{i,j=1}^{n} a_i\gamma(t_i - t_j)a_j$$

unde $\Gamma_n = [\gamma(t_i - t_j)]_{i,j=1}^n$ este matricea de covarianţă a lui $(X_{t_1}, ..., X_{t_n})'$.

Reciproc, fie $k : \mathbf{Z} \to \mathbf{R}$ o funcţie pară nenegativ defintă. Trebuie să arătăm că există un proces staţionar cu $k(.)$ funcţia de autocovarianţă. Pentru aceasta vom folosi teorema lui Kolmogorov. Pentru orice $n \in \mathbf{N}$ şi orice $\mathbf{t} = (t_1, ..., t_n) \in \mathbf{Z}^n$: $t_1 < t_2 < ... < t_n$ fie $F_{\mathbf{t}}$ funcţia de distribuţie pe $\mathbf{R}^n$ cu funcţia caracteristică

$$\phi_{\mathbf{t}}(\mathbf{u}) = \exp(-\frac{\mathbf{u}' K \mathbf{u}}{2}),$$

unde $\mathbf{u} = (u_1, ..., u_n)' \in \mathbf{R}^n$ şi $K = [k(t_i - t_j)]_{i,j=1}^n$. Deoarece k este nenegativ definită deci matricea K este nenegativ definită rezultă că $\phi_{\mathbf{t}}$ este funcţia caracteristică a distribuţiei normale de medie zero şi matrice de covarianţă K. Deci

$$\phi_{\mathbf{t}(i)}(\mathbf{u}(i)) = \lim_{u_i \to \infty} \phi_{\mathbf{t}}(\mathbf{u}) \qquad \forall \mathbf{t} \in \mathcal{T}$$

adică funcţiile de distribuţie $F_{\mathbf{t}}$ sunt consistente şi deci conform teoremei Kolmogorov există o serie dinamică cu funcţia de distribuţie $F_{\mathbf{t}}$ şi funcţia caracteristică $\phi_{\mathbf{t}}$. În particular distribuţia comună a variabilelor X_i şi X_j este o normală bivariată de medie zero şi matrice de covarianţă

$$\begin{pmatrix} k(0) & k(i-j) \\ k(i-j) & k(0) \end{pmatrix}$$

care arată faptul că $Cov(X_i, X_j) = k(i - j)$. □

Observaţia F.5.1. Aşa cum rezultă din demonstraţia teoremei anterioare pentru orice funcţie de autocovarianta $\gamma(.)$ există o serie dinamică Gaussiană a cărei funcţie de autocovarianţă este $\gamma(.)$.

Observaţia F.5.2. Pentru a verifica că o anumită funcţie este nenegativ definită uneori este mai uşor să indicăm un proces staţionar cu funcţia de autocovarianţă identică cu funcţia dată decât să aplicăm definiţia. De exemplu funcţia $k(h) = cos(\theta h)$ este funcţia de autocovarianţă a procesului din exemplul 4 din paragraful 3 şi deci $k(h)$ nenegativ definită. O altă modalitate de a verifica nenegativ definirea este cu teorema lui Herglotz.

Observaţia F.5.3. Funcţia de autocorelaţie $\rho(.)$ are toate proprietăţile funcţiei de autocovarianţă şi satisface proprietatea adiţională $\rho(0) = 1$.

Exerciţiul F.5.1. Care din următoarele funcţii este funcţia de autocovarianţă a unui proces stochastic?

1. $f(h) = 1$ dacă $h = 0$ şi $f(h) = 1/h$ dacă $h \neq 0$.
2. $f(h) = (-1)^{|h|}$

Exerciţiul F.5.2. Fie $\{S_t, t = 0, 1, ...\}$ procesul definit de $S_0 = 0$ şi

$$S_t = \mu + S_{t-1} + X_t,\ t = 1, 2, ...$$

unde $X_1, X_2, ...$ sunt variabile aleatoare independente, identic distribuite de medie zero şi dispersie σ^2. Calculaţi media lui S_t şi funcţia de autocovarianţă a procesului $\{S_t\}$. Arătaţi că procesul $\{\nabla S_t\}$ este staţionar şi calculaţi media şi funcţia de autocovarianţă.

F.5.1. Funcţia de autocovarianţă de selecţie

Din observaţiile $\{x_1, ..., x_n\}$ efectuate asupra unei serii dinamice staţionare $\{X_t\}$ vrem să estimăm funcţia de autocovarianţă $\gamma(.)$. care va furniza informaţii asupra structurii de dependenţă.

Definiţia F.5.2. Funcţia de autocovarianţă de selecţie a datelor $\{x_1, ..., x_n\}$ este definită prin

$$\overset{\wedge}{\gamma}(h) = \frac{1}{n}\sum_{j=1}^{n-h}(x_{j+h} - \bar{x})(x_j - \bar{x}), \qquad 0 \leq h < n$$

şi $\overset{\wedge}{\gamma}(h) = \overset{\wedge}{\gamma}(-h), -n < h \leq 0$, unde $\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j$

Definiţia F.5.3. Funţia de autocorelaţie de selecţie a datelor $\{x_1, ..., x_n\}$ este definită prin

$$\overset{\wedge}{\rho}(h) = \frac{\overset{\wedge}{\gamma}(h)}{\overset{\wedge}{\gamma}(0)}, \qquad |h| < n$$

Exerciţiul F.5.3. Dacă $X_t = a + bt, t = 1, 2, ..., n$, unde $a, b \in \mathbf{R}$ funcţia de autocorelaţie are proprietatea: $\overset{\wedge}{\rho}(k) \underset{n\to\infty}{\longrightarrow} 1$ pentru k fixat.

Anexa G

MODELUL AUTOREGRESIV-MEDIE MOBILĂ

G.1. Modelul autoregresiv AR(p)

Vom defini şi vom da principalele proprietăţi ale proceselor autoregresive $AR(p)$.

Definiţia G.1.1. Un proces $\{X_t, t = 0, \pm 1, \pm 2, ...\}$ se numeşte proces $AR(p)$ dacă

$$X_t - \phi_1 X_{t-1} - ... - \phi_p X_{t-p} = Z_t \text{ pentru orice } t$$

unde $\{Z_t\} \sim WN(0; \sigma^2)$ (zgomot alb).

Spunem că $\{X_t\}$ este un proces $AR(p)$ de medie μ dacă şi numai dacă $\{X_t - \mu\}$ este proces $AR(p)$. Ecuaţia de mai sus se scrie în forma simbolică:

$$\Phi(B)X_t = Z_t, \quad t = 0, \pm 1, \pm 2, ...$$

unde $\Phi(B) = 1 - \phi_1 B - ... - \phi_p B^p$ este polinomul autoregresiv.

Teorema G.1.1. Procesul $AR(p)$ este un proces inversabil.

Teorema G.1.2. Un proces $AR(p)$ este staţionar dacă şi numai dacă rădăcinile ecuaţiei polinomiale autoregresive $\Phi(z) = 0$ sunt de modul mai mare ca 1.

G.2. Modelul medie mobilă MA(q)

Vom defini şi vom da principalele proprietăţi ale proceselor autoregresive $MA(q)$.

Definiţia G.2.1. Un proces $\{X_t, t = 0, \pm 1, \pm 2, ...\}$ se numeşte proces $MA(q)$ dacă

$$X_t = Z_t - \theta_1 Z_{t-1} - ... - \theta_q Z_{t-q} \text{ pentru orice } t$$

unde $\{Z_t\} \sim WN(0; \sigma^2)$ (zgomot alb).

Spunem că $\{X_t\}$ este un proces $MA(q)$ de medie μ dacă şi numai dacă $\{X_t - \mu\}$ este proces $MA(q)$. Ecuaţia de mai sus se scrie în forma simbolică:

$$X_t = \Theta(B)Z_t, \quad t = 0, \pm 1, \pm 2, ...$$

unde: $\Theta(B) = 1 - \theta_1 B - ... - \theta_p B^p$ este polinomul medie mobilă.

Teorema G.2.1. Procesul $MA(q)$ este un proces staţionar.

Teorema G.2.2. Un proces $MA(q)$ este inversabil dacă şi numai dacă rădăcinile ecuaţiei polinomiale medie mobilă $\Theta(z) = 0$ sunt de modul mai mare ca 1.

G.3. Modelul ARMA(p,q)

Procesul $ARMA(p, q)$ este o combinaţie a celor două clase de procese menţionate anterior.

Definiţia G.3.1. Un proces $\{X_t, t = 0, \pm 1, \pm 2, ...\}$ se numeşte proces auto regresiv -medie mobilă $ARMA(p, q)$ dacă:

$$X_t - \phi_1 X_{t-1} - ... - \phi_p X_{t-p} = Z_t - \theta_1 Z_{t-1} - ... - \theta_q Z_{t-q} \text{ pentru orice } t$$

unde $\{Z_t\} \sim WN(0; \sigma^2)$ (zgomot alb).

Spunem că $\{X_t\}$ este un proces $ARMA(p, q)$ de medie μ dacă şi numai dacă $\{X_t - \mu\}$ este proces $ARMA(p, q)$. Ecuaţia de mai sus se scrie în forma simbolică:

$$\Phi(B)X_t = \Theta(B)Z_t, \quad t = 0, \pm 1, \pm 2, ...$$

unde: $\Phi(B)$ şi $\Theta(B)$ este polinomul autoregresiv respectiv polinomul medie mobilă.

G.4. Modelul ARIMA(p,d,q)

De foarte multe ori un proces ARMA (p,q) nu este staţionar şi pentru al transforma într-unul staţionar este necesară diferenţierea datelor.

Definiţia G.4.1. Un proces $\{X_t, t = 0, \pm 1, \pm 2, ...\}$ se numeşte proces auto regresiv integrat -medie mobilă $ARIMA(p, d, q)$ dacă:

$$\Phi(B)\nabla^d X_t = \Theta(B)Z_t, \quad t = 0, \pm 1, \pm 2, ... \text{ pentru orice } t$$

unde $\{Z_t\} \sim WN(0; \sigma^2)$ (zgomot alb) iar ∇^d este operatorul de diferenţiere de ordinul d (vezi ANEXA F).

Observaţia G.4.1. Modelele un proces stochastic poate avea şi o componentă sezonieră. Ecuaţia unui astfel de proces $ARIMA(p, d, q) \times (P, D, Q)$ este:

$$\Phi_s(B^s)\Phi(B)\nabla_s^D\nabla^d X_t = \Theta_s(B^s)\Theta(B)Z_t, \quad t = 0, \pm 1, \pm 2, ... \text{ pentru orice } t$$

unde indicele s are seminificaţia componentei sezoniere, $\Phi_s(B^s)$ este polinomul autoregresiv sezonier, $\Theta_s(B^s)$ este polinomul medie mobilă sezonier, ∇_s^D este operatorul de diferenţiere sezonieră de ordinul D iar restul parametrilor sunt cu aceeiaşi specificaţie ca la un model $ARIMA(p, d, q)$.

G.5. Probleme puse proceselor ARIMA(p,d,q)

Asupra proceselor $ARIMA(p, d, q)$ se pun următoarele probleme:

i) *identificarea modelului* (de regulă se face cu ajutorul funcţiei de autocorelaţie şi de autocorelaţie parţială) care presupune elaborarea unor tehnici şi metode de estimare, pe baza observaţiilor, a numerelor întregi p, q şi d;

ii) *estimarea mediei* procesului precum şi a *parametrilor modelului* adică a valorilor ϕ_i şi θ_i;

iii) *diagnoza* şi *testarea modelului*, cu ajutorul testelor statistice, şi în funcţie de rezultatele acestor teste reajustarea numărului de parametrii şi a valorii acestora;

iv) *prognoza* sau *previzionarea* (evident cu o anumită probabilitate) a următoarelor date.

Pentru clarificarea tehnicilor utilizate cititorul poate consulta *Box* [12] şi *Brockwell* [13].

Anexa H

SIMULAREA VARIABILELOR ALEATOARE

În scopul realizării testării produselor software şi/sau hardware sunt necesare de cele mai multe ori efectuarea unor simulări ale variabilelor aleatoare cu o distribuţie de probabilitate bine definită care pote fi discretă sau continuă. Tehnicile de simulare presupun realizarea simulării unei variabile aleatoare uniforme discrete (care se poate realiza cu ajutorul unui generator pseudoaleator ca de exemplu un generator congruenţial liniar) sau continue. Generare unei variabile aleatoare cu o distribuţie neuniformă se poate face prin metodele invesiei, rejecţiei, etc. În *Devroye* [15] sunt prezentate metode şi tehnici exhaustive de generare a variabilelor aleatoare neuniforme.

H.1. Tehnici de simulare

Teorema H.1.1. *(Metoda inversiei) Dacă F este o distribuţie continuă pe* $\mathbf{R}$ *cu inversa F^{-1} definită de:*

$$F^{-1}(u) = \inf\{x|F(x) = u, 0 < u < 1\}.$$

Dacă U este o variabilă uniformă pe $[0,1]$ *atunci $F^{-1}(U)$ are funcţia de distribuţie F. Deasemenea dacă X are funcţia de distribuţie F, atunci $F(X)$ este uniform distribuită pe* $[0,1]$.

Demonstraţie. Avem:

$$\begin{aligned} \Pr(F^{-1}(u) &\leq x) = \Pr(\inf\{y|F(y) = U\} \leq x) \\ &= \Pr(U \leq F(x)) = F(x). \end{aligned}$$

Pentru cea de a doua afirmaţie avem pentru orice $0 < u < 1$:

$$\Pr(F(X) \leq u) = \Pr(X \leq F^{-1}(u)) = F(F^{-1}(u)) = u.$$

Exemplul H.1.1. Distribuţia exponenţială dată de densitatea $\rho(x) = \lambda e^{-\lambda t}$ (pentru $x \geq 0$) şi funcţia de repartiţie $F(x) = 1 - e^{-\lambda x}$ se poate obţine prin aplicarea metodei inversei: dacă variabila U este repartizată uniform pe $[0, 1]$ atunci

$$-\frac{1}{\lambda} \ln(1 - U)$$

are o distribuţie exponenţială.

Teorema H.1.2. *(Metoda rejecţiei) Fie X un vector aleator cu densitatea f pe $\mathbf{R}^d$ şi U o variabilă aleatoare unifrom distribuită pe $[0, 1]$. Atunci $(X, cUf(X))$ este uniform distribuită pe $A = \{(x, u) | x \in \mathbf{R}^d, 0 \leq u \leq cf(x)\}$, unde $c > 0$ este o constantă arbitrară. Viceversa, dacă (X, U) este vector aleatoriu în $\mathbf{R}^{d+1}$ uniform distribuit pe A, atunci X are densitatea f pe $\mathbf{R}^d$.*

Metoda inversiei poate fi destul de dificil de aplicat în anumite cazuri practice deoarece calculul inversei efectiv al inversei se poate dovedi nefezabil. Metoda lui *Von Neumann* (vezi spre exemplu *Sobol* [86]) rezolvă această problemă. Pentru a genera o variabilă aleatoare cu densitatea $f(x)$ defintă spre exemplu în intervalul (a, b) vom genera două puncte γ' şi γ'' din $[0, 1]$. Se calculează valorile $\eta' = a + \gamma'(b - a)$ şi $\eta'' = \gamma'' \max\limits_{x \in (a,b)} f(x)$. Dacă punctul $\Upsilon(\eta', \eta'')$ este în interiorul graficului lui $f(x)$ atunci procedura lui Von Neumann returnează valoarea η'. Situaţia este exemplificată în figura de mai jos.

H.2. Legea tare a numerelor mari

Generarea unei variabile aleatoare continue se poate face şi cu ajutorul legii tari a numerelor mari.

Teorema H.2.1. *(Legea tare a numerelor mari) Fie $X_1, ..., X_n$ un şir de n variabile aleatoare independente identic repartizate. Atunci pentru n suficient de mare variabila aleatoare:*

$$\frac{\sum\limits_{i=1}^{n} X_i}{n} \sim N(0; 1)$$

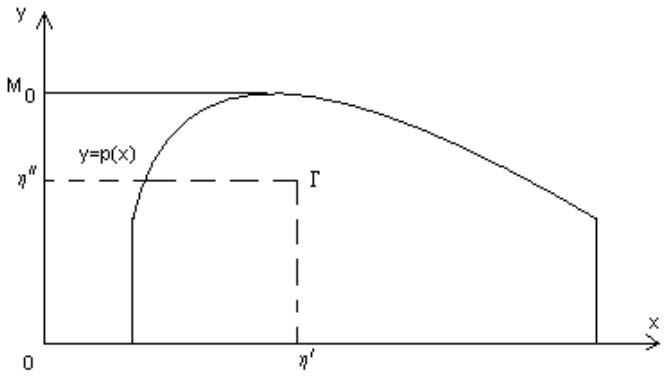

Figura 8.1: Metoda lui von Neumann de generare a variabilelor aleatoare continue.

(de fapt nu este necesar ca variabilele aleatoare $X_1, ..., X_n$ să fie independente identic repartizate ci doar ca ponderea acestora în sumă să fie uniformă). Legea slabă a numerelor mari spune că în aceleeaşi condiţii ca mai sus avem:

$$\lim_n \frac{\sum_{i=1}^{n} X_i}{n} = 0.$$

Din variabile aleatoare normale şi independente se pot genera toate celelalte distribuţii continue. De exemplu distribuţia χ^2 se poate genera utilizând următoarea teoremă.

Teorema H.2.2. *(Distribuţia χ^2) Dacă $X_1, ..., X_n$ sunt independente repartizate $N(0,1)$ atunci:*

$$\sum_{i=1}^{n} X_i^2 \sim \chi^2(n-1).$$

Pentru cazul discret sunt rezultate similare sau se poate opta pentru discretizarea variabilei continue. În acest sens vom prezenta o metodă de a genera variabile aleatoare (discrete) de tip Bernoulli ($\Pr(X = 1) = p$) pornind de la o variabilă aleatoare (continuă) uniformă pe $[0, 1]$:

1) se generează y o variabilă aleatoare uniformă pe $[0, 1]$;

2) dacă $y > p$ atunci $X = 0$, în caz contrar $X = 1$.

Anexa I

ELEMENTE DE TEORIA NUMERELOR

I.1. Teste de primalitate

Vom prezenta o serie de teoreme fundamentale din teoria numerelor cunoscute şi sub denumirea de *teste de primalitate*.

Teorema I.1.1. *(Teorema lui Fermat) Dacă p este număr prim şi a este un număr întreg care nu se divide cu p atunci:*

$$a^{p-1} \equiv 1 \bmod p.$$

Reciproca teoremei lui Fermat nu este adevărată, numerele care *îndeplinesc concluzia teoremei lui Fermat* şi care *nu sunt prime* se numesc *numere pseudoprime* în baza a.

Definiţia I.1.1. (Funcţia lui Euler) Vom defini funcţia lui Euler $\varphi(n)$ prin formula:

$$\varphi(n) = card\{p|\ (p,n) = 1,\ p < n\}.$$

Dacă $(m,n) = 1$ atunci $\varphi(mn) = \varphi(m)\varphi(n)$. Avem următoarea formulă pentru calculul lui $\varphi(n)$ în cazul în care $n = \prod_{i=1}^{k} p_i^{\alpha_i}$:

$$\varphi(n) = \prod_{i=1}^{k}(p_i - 1)p_i^{\alpha_i - 1} = n\prod_{i=1}^{k}(1 - \frac{1}{p_i}).$$

Observaţia I.1.1. Avem următoarea limită inferioară pentru $\varphi(n)$:

$$\varphi(n) > \frac{n}{6 \ln \ln n} \text{ pentru orice } n \geq 5.$$

Teorema I.1.2. *(Teorema lui Euler) Dacă a şi n sunt numere întregi relativ prime atunci:*

$$a^{\varphi(n)} \equiv 1 \bmod n.$$

Teorema I.1.3. *(Teorema lui Wilson) Numărul p este prim dacă şi numai dacă:*

$$(p-1)! \equiv 1 \bmod p.$$

I.2. Lema chinezească a resturilor

Teorema I.2.1. *(Lema chinezească a resturilor- CRT) Fie $m_1, ..., m_k$ numere întregi cu $(m_i, m_j) = 1$ pentru orice $i \neq j$. Atunci sistemul*

$$x \equiv a_i \bmod m_i$$

are o soluţie unică modulo $\prod_{i=1}^{k} m_i$.

Demonstraţie. Existenţa soluţiei. Vom nota

$$M = \prod_{i=1}^{k} m_i$$

şi

$$M_i = \frac{M}{m_i} \text{ pentru orice } i = 1, ..k.$$

Deoarece $(m_i, m_j) = 1$ pentru orice $i \neq j$ avem $(M_j, m_j) = 1$ pentru orice j adică există N_j astfel ca $M_j N_j = 1 \bmod m_j$. Atunci dacă notăm

$$x = \sum_{i=0}^{k} a_i M_i N_i$$

şi reducem modulo m_i avem:

$$x = \sum_{j=0}^{k} a_j M_j N_j \bmod m_i \text{ pentru orice } i.$$

Folosind faptul că $(M_i, m_j) \neq 1$ pentru $i \neq j$ obţinem:

$$\begin{aligned} x &= a_i M_i N_i \bmod m_i \\ &= a_i \bmod m_i \text{ pentru orice } i. \end{aligned}$$

Unicitatea soluţiei. Fie $x^{'}$ şi $x^{''}$ două soluţii atunci

$$x = x^{'} - x^{''} = 0 \bmod m_i \text{ pentru orice } i$$

deci

$$x = 0 \bmod M.$$

I.3. Numărul de numere prime

Teorema I.3.1. *(Teorema numerelor prime) Vom nota prin $\pi(x)$ numărul de numere prime mai mici sau egale cu x. Avem:*

$$\lim_{x \to \infty} \frac{\pi(x) \ln x}{x} = 1.$$

Observaţia I.3.1. Avem inegalităţile:

$$\pi(x) < 1,25506 \frac{x}{\ln x} \text{ pentru orice } x > 1$$

respectiv

$$\pi(x) > \frac{x}{\ln x} \text{ pentru } x \geq 17.$$

I.4. Simbolurile lui Legendre şi Jacobi

Următoarele definiţii (simbolul lui Legendre şi simbolul lui Jacobi) sunt utilizate frecvent în probleme de teoria numerelor (de exemplu algoritmul lui Rabin sau algoritmul de cifrare cu ajutorul curbelor eliptice).

Definiţia I.4.1. Fie p un număr prim impar şi a un număr întreg. *Simbolul lui Legendre* $\left(\frac{a}{p}\right)$este definit ca:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 \text{ dacă } p|a, \\ 1 \text{ dacă } a \text{ este reziduu pătratic modulo } p, \\ -1 \text{ dacă } a \text{ nu este reziduu pătratic modulo } p. \end{cases}$$

Teorema I.4.1. *Fie p un număr prim impar şi a, b un numere întregi. Atunci:*
i) $\left(\frac{a}{p}\right) = a^{(p-1)/2} \bmod p$.
ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. *Deci dacă* $a \in \mathbf{Z}_p^*$ *atunci* $\left(\frac{a^2}{p}\right) = 1$.
iii) Dacă $a \equiv b \bmod p$ *atunci* $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
iv) $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$.
v) (legea reciprocităţii pătratice) Dacă q este un număr prim impar diferit de p atunci

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{\frac{(p-1)(q-1)}{4}}$$

Simbolul lui Jacobi este o generalizare a simbolului lui Legendre.

Definiţia I.4.2. Fie $n \geq 3$ un număr impar cu factorizarea în numere prime $n = p_1^{e_1} \ldots p_k^{e_k}$. Atunci *simbolul lui Jacobi* $\left(\frac{a}{n}\right)$ este definit prin formula:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \ldots \left(\frac{a}{p_k}\right)^{e_k}.$$

Observaţia I.4.1. Dacă n este număr prim atunci simbolul lui Jacobi este chiar simbolul lui Legendre.

Teorema I.4.2. *Fie $m, n \geq 3$ numere întregi impare şi a, b un numere întregi. Atunci:*
i) $\left(\frac{a}{n}\right) = 0, 1,$*sau* -1. *Mai mult* $\left(\frac{a}{n}\right) = 0$ *dacă şi numai dacă* $gcd(a, n) \neq 1$.
ii) $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$. *Deci dacă* $a \in \mathbf{Z}_n^*$ *atunci* $\left(\frac{a^2}{n}\right) = 1$.
iii) Dacă $a \equiv b \bmod n$ *atunci* $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.
iv) $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right)$.
v) $\left(\frac{1}{n}\right) = 1$.
vi) $\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$.
vi) $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$.
vii) $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)(-1)^{\frac{(m-1)(n-1)}{4}}$.

Bibliografie

[1] **Abramowitz M., Stegun I.A.,** *Handbook of Mathematical Functions,* Washington, D.C., U.S. Government Printing Office, 1964.

[2] **Al-Kindi,** *Manual for Derypting Secret Messages,* 850 AD.

[3] **Asmuth C., Bloom J.,** *A Modular Approach to Key Safeguarding*, IEEE Transactions on Information Theory, vol. IT29, n. 2, Mar 1983, pag. 208-210.

[4] **Azencott R.,Dacuhna-Castelle D.,** *Series of Irregular Observations. Forecasting and model building,* Springer-Verlag, 1986.

[5] **Basse S., Gelder A.,** *Computer Algorithms, Introduction to Design and Analysis,* Addison Wesley Longman, 2000.

[6] **Bauer F.,** *Decrypted Secrets,* Springer Verlag, 2000.

[7] **Beker H., Piper F.,** *Cipher Systems,* Northwood Books London, 1982.

[8] **Biham E., Shamir A.,** *Differential Fault Analysis of Secret Key Cryptosystem.*

[9] **Biham E., Kocher P.,** *A Known Plaintext Attack on the PKZIP Stream Cipher,* Technion Technical Report, 1997.

[10] **Blakley G.R.,** *Safeguarding Cryptographic Keys*, Proceedings of the National Computer Conference, 1979, American Federation of Information Processing Society, vol. 48, 1979, pag. 313-317.

[11] **Boneh D., ş.a.,** *On the importance of Checking Cryptographic Protocols for Faults.*

[12] **Box E.P, Jenkins G.M.,** *Time series analysis, forecasting and control,* Holden-Day, 1970.

[13] **Brockwell P., Davis R.,** *Time Series Theory and Methodes,* Springer-Verlag, 1987.

[14] **Cormen T., Leiserson C., Rivest R.,** *Introduction to Algorithms,* MIT Press, 1990.

[15] **Devroye L.,** *Non-Uniform Random Variate Generation,* Springer-Verlag, 1986.

[16] **Diffie W., Hellman M.E.,** *New Directions in Cryptography,* IEEE Transactions on Information Theory, vol. IT.-22, no.6, nov 1976, pag. 644-654.

[17] **Dumitrescu M.,** *Curs de Statistică Matematică,* Editura Universităţii Bucureşti, 1989.

[18] **ElGamal T.,** *A Public-Key Cryptosystem and a Signature Based on Discrete Logarithms,* Adv. in Cryptology Crypto84, Springer Verlag, 1985, pag. 10-18.

[19] **Forré R.,** *The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition,* Adv. in Cryptology Crypto88, Springer Verlag, 1988, pag. 450-468.

[20] **Friedman W.,** *Military Cryptanalysis,* Aegean Park Press, 1980.

[21] **Golic J.,** *The number of output sequences of a binary sequence generator,* Adv. in Cryptology Eurocrypt91, 1991.

[22] **Golic J.,** *Correlation Via Linear Sequential Circuit Approximation of Combiners with Memory,* Adv. in Cryptology Eurocrypt92, 1992.

[23] **Golic J.,** *Embedding a Probabilistic Correlation Attacks on Clocked- Controlled Shift Registers,* Adv. in Cryptology Eurocrypt94, 1994.

[24] **Golic J.,** *Towards Fast Correlation Attacks on Irregularly Clocked Shift Registers,* Adv. in Cryptology Eurocrypt95, 1995.

[25] **Golic J.,** *Cryptanalysis of the Alleged A5 Stream Cipher,* Adv. in Cryptology EuroCrypt97, 1997.

[26] **Golic J., Petrovic S.,** *A Generalized Correlation Attack with a Probabilistic Constrained Distance*, Adv. in Cryptology Eurocrypt92, 1992.

[27] **Golic J., Menicocci R.,** *Edit Distance Correlation Attack on the Alternating Step Generator,* Adv. in Cryptology Crypto97, 1997.

[28] **Golic J., Mihaljevic M.,** *A generalized Correlation Attack on a Class of Stream Chiper Based on the Levenshtein Distance,* Journal of Cryptology vol. 3, 1991.

[29] **Golic J., Mihaljevic M.,** *A comparation of Cryptanalitic Principles Based on Iterative Error-Correction,* Adv. in Cryptology Eurocrypt91 1991.

[30] **Golic J., Mihaljevic M.,** *Convergence of Bayesian iterative error-correction procedure on a noisy shift register sequence,* Adv. in Cryptology Eurocrypt92, 1992.

[31] **Golomb S.W.**, *Shift-register sequences*, Aegean Park Press, Laguna Hills, California, 1982.

[32] **Guiaşu S.,** *Information Theory with Applications,* McGrow-Hill, 1977.

[33] **IEEE P1363/D13,** *Standard Specification for Public Key Cryptography,* Draft Version 13, November 1999.

[34] **Jamsa K., Cope K.,** *Internet Programming,* Jamsa Press, 1995.

[35] **Juang W.N,** *Efficient password authenticated key agreement using smart cards,* Computer and Security, (2004) 23, pag. 167-173.

[36] **Karnin E.D, Greene J.W., Hellman M.E.,** *On Sharing Secret Systems*, IEEE Transactions on Information Theory, vol. IT29, 1983, pag. 35-41.

[37] **Kelly J**., *Terror groups hide behind Wed encryption.* USA Today, February 2001.

[38] **Knuth D.,** *Arta programării calculatoarelor,* Ed. Teora, 1999.

[39] **Koblitz N.,** *Elliptic Curve Cryptosystems,* Mathematics of Computations, vol. 48, nr.177, 1987, pag. 203-209.

[40] **Koblitz N.,** *Introduction to Elliptic Curves and Modular Forms,* Springer-Verlag, 1984.

[41] **Koblitz N.,** *A Course in Number Theory and Cryptography,* Springer-Verlag, 1988.

[42] **Koblitz N.,** *Algebric aspects of Cryptography,* Springer-Verlag, 1999.

[43] **Kocher P.,** *Timing attacks on Implementation of Diffie-Hellman, RSA, DSS and other Systems.*

[44] **Massey J.**, *Shift-register synthesis and BCH decoding*, IEEE Trans. Information Theory, vol.IT-15, nr.1(ian. 1969), pag.122-127.

[45] **Massey J., Rueppel R.A.,** *Linear ciphers and random sequence generators with multiple clocks*, Proceedings Eurocrypt'84, Springer-Verlag Lecture Notes in Computer Science nr.209, Springer-Verlag, New York, 1984, pag. 74-87.

[46] **Maurer U.,** *Universal randomness test*, Journal of Cryptology, 1992.

[47] **Maurer U.,** *Cascade Chipers: The importance of being first,* J. of Cryptology vol. 6., 1993.

[48] **Maurer U.,** *The role of the Information Theory in Cryptography,* Cryptography and Coding IV 1993.

[49] **Maurer U.,** *Information -Theoretically Secure Secret-Key Agreement by NOT Authenticated Public Discussion,* Adv. in Cryptology EuroCrypt97, 1997.

[50] **Menenzes A.J., et. al.,** *Handbook of Applied Cryptography*, CRC Press, 1997.

[51] **Naccache D., Simion E., Mihăiţă, Olimid R.F., Oprina A.G.**, *Criptografie şi Securitatea informaţiei. Aplicaţii*, MATRIXROM 2011.

[52] **Nyberg K.**, *Differentially uniform mappings for cryptography*, Adv. in Cryptology Eurocrypt93, 1993.

[53] **Pankratz A.,** *Forecasting with Dynamic Regression Models,* Wiley and Sons, 1991.

[54] **Peyravian M.,** *Non-PKI methods for public key distribution,* Computer and Security, (2004) 23, pag. 97-103.

[55] **Pieprzyk J., Sadeghiyan B.,** *Design of Hashing Algorithms,* Springer Verlag, 1993.

[56] **Popescu A.**, *Transformata Walsh-Hadamard în Criptografie*, Sesiunea de Comunicări Ştiinţifice a Universităţii Hyperon, 2004.

[57] **Preda V.,** *Teoria Deciziilor Statistice*, Ed. Academiei, 1991.

[58] **Preda V., Simion E.,** *Statistical Cryptanalytics Techniques*, Annals of University of Bucharest, 2002, pag. 73-82.

[59] **Preda V., Simion E.,** *Confirmatory Test used in Cipher System Evaluations*, Annals of University of Bucharest, 2001, pag. 105-126.

[60] **Preda V., Simion E.,** *Correlated Stochastic Time Series and Cryptographic Applications*, Annals of University of Bucharest, 2000, pag. 105-122.

[61] **Povros N**., *Defeding Against Statistical Steganalysis, Proceedings of the 10th USENIX Security Symposium*, 323-335, August 2001.

[62] **Povros N., Honeyman P.,** *Detecting Steganographic Content on the Internet,* Technical Report, Center for Information Technology Integration, University of Michigan, 2002.

[63] **Preneel B., et. al,** *Propagtion Characteristics of Boolean Functions,* Adv. in Cryptology Eurocrypt90, 1990, pag. 161-173.

[64] **Preneel B., et. al,** *Boolean Functions Satisfying Higher Order Propagation Criteria,* Adv. in Cryptology Eurocrypt91, 1991, pag. 141-152.

[65] **Rueppel R. A.,** *Analysis and Design of Stream Ciphers*, Springer-Verlag, 1986.

[66] **Revesz P.,** *Random Walk in Random and Non-Random Environments*, Singapore, World Scientific, 1990.

[67] **Rukhin A.**, *Approximate entropy for testing randomness*, Journal of Applied Probability, vol.37, 2000.

[68] **Shamir A.,** *How to Share a Secret*, Communications of the ACM, vol. 24, n. 11, Nov 1979, pag. 612-613.

[69] **Schneier B.,** *Applied Cryptography*, Adison-Wesley, 1998.

[70] **Schneier B.,** *A self-study course in block-cipher cryptanalysis*, Criptologia, 2001.

[71] **Salomaa A.,** *Public Key Cryptography*, Springer-Verlag, 1989.

[72] **Schroeder M.R.,** *Number Theory in Science and Communications*, Third Edition, Springer Verlag, 1997.

[73] **Shannon C. E.,** *Communication Theory of Secrecy Systems*, Bell Systems Technology Journal, vol.28, nr.4(oct. 1949), pag. 656-715.

[74] **Siegenthaler T.,** *Correlation-immunity of nonlinear combining functions for cryptographic application,* IEEE Trans. Information Theory, vol.IT-31, nr.5(ian. 1984), pag. 776-780.

[75] **Simion E., Oprişan Ghe.,** *Elemente de Cercetări Operaţionale şi Criptologie*, Ed. Politehnica Press, 2003.

[76] **Simion E., Andraşiu M., Naccache D., Simion Gh.,** *Cercetări operaţionale, Probabilităţi şi Criptologie. Aplicaţii*, Academia Tehnică Militară, 2011.

[77] **Simion E.,** *Remarks on the Bayesian Estimation of Shannon Entropy Using Priori Truncated Dirichlet Distribution*, SCM tom 51, nr. 2, 1999, pag. 277-288.

[78] **Simion E.,** *Information Reconciliation for Almost Uniform Distributions*, SCM tom 51, nr. 4, 1999, pag. 621-630.

[79] **Simion E.,** *Asymptotic Behavior of the Linear Complexity Profile and Applications at Evaluations of Cryptographic Algorithms*, Proceedings of the 3rd annual meeting of Romanian Society of Mathematical Sciences, Craiova, May 26-29, 1999, pag. 222-230.

[80] **Simion E.,** *Truncated Bayesian Estimation of Renyi Entropies and Cryptographic Applications*, Mathematical Reports tom 2 (52), nr.2/2000, pag. 215-226.

[81] **Simion E.,** *Statistical Estimators for the Effective Key Size of a Cipher System*, Annals of University of Bucharest, Proceedings of the annual meeting of the Annual Meeting of the Faculty of Mathematics, 1999, pag. 73-84.

[82] **Simion E.,** *The Linear Complexity Profile and Applications at the Evaluation of Cryptographic Algorithms*, Annals of University of Bucharest, Proceedings of the annual meeting of the Annual Meeting of the Faculty of Mathematics, 1999, pag. 85-93.

[83] **Simion E.,** *Asymptotic Behavior of Some Statistical Estimators used in Cryptography*, Annals of University of Bucharest, no.1, 2000, pag. 85-89.

[84] **Simion E.,** *Linear Complexity Computations of Cryptographic Systems*, International Conference on Telecommunications, Bucharest, 4-7 June, 2001, pag. 85-92.

[85] **Simion E., Voicu I.,** *Game theory in cryptography*, Proceedings of the International Conference Communications 2002 organized by IEEE Romanian Section and Military Technical Academy, December 2002, Bucharest, pag. 576-578.

[86] **Sobol I. M.,** *The Monte Carlo Method*, Mir Publisher, Moskow, 1972.

[87] **Soto J.,** *Randomness testing of the Advanced Encryption Standard Candidate Algorithms*, http://crs.nist.gov/encryption/aes/aes_home.htm, 2000.

[88] **Stallings W.,** *Cryptography and Netwwork Security: Principles and Practice,* Prentice Hall, Second Edition, 1999.

[89] **Tilborg, Henk C.A. van,** *Fundamentals of Cryptology,* Kluwer Academic Publisher, Second Edition, 2001.

[90] **Voicu I., Simion E**., *Steganography in the digital age*, Proceedings of the International Conference Communications 2002 organized by IEEE Romanian Section and Military Technical Academy, December 2002, Bucharest, pag.571-575.

[91] **Voicu I.,** *Inserarea Mesajelor Secret Digitale în Imagini şi Secvenţe Video,* Ed. Medro, 2000.

[92] **Wang X.**, *How to break MD5 and other Hash functions*, Proceedings of Eurocrypt 2005, Aarhus, Danemark, May-June 2005.

[93] **Wang X.**, *Collision for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, Proceedings of Eurocrypt 2005, Danemark, 2005.

[94] **Wang X.**, *Collision for SHA0 and SHA-1*, Proceedings of Eurocrypt 2005, Danemark, 2005.

[95] **Wu H.**, *The missue of RC4 in MicroSoft Word and Excel*, http://www.citeseer.com.

[96] **Welschenbach M.,** *Cryptography in C and C++,* APress, 2001.

[97] **Zdenek F.,** *Information Theory of Continuous Random Variables,* IEEE Transactions on Information Theory, vol.43, No.3, May 1997.

[98] **Zeng K.C., Huang M.Q.,** *On the linear syndrome method in cryptanalysis*, Adv. in Cryptology Eurocrypt88, 1988.

[99] **Zeng K.C., Yang C.H., Rao T.R.N.,** *On the linear consistency test (LCT) in cryptanalysis with application*, Adv. in Cryptology Eurocrypt89, 1989.

[100] **Zeng K.C., Yang C.H., Rao T.R.N.,** *An improved linear syndrome algorithm in cryptanalysis with application*, Adv. in Cryptology Eurocrypt89, 1989.

[101] **Zeng K.C., J. Sebbery,** *Practical Approach to Attaining Security against Adaptively chosen Cipher Text Attack*, Adv. in Cryptology Eurocrypt92, 1992.

[102] **Zeng K.C., Yang C.H., Rao T.R.N.,** *Large primes in stream-cipher cryptography,* Adv. in Cryptology Asiacrypt90, 1990.

[103] *Basic Cryptanalysis*, Field Manual no. 34-40-2, Washington, DC.

[104] *Terrorist Manual bin Laden Jihad*, www.skfriends.com.

[105] *Security Requirements for Cryptographic Modules*, Federal Information Standards Publication 140-2, 1999.

[106] *Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408)*, August 1999, versiunea 2.1.

[107] *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, NIST Special Publication 800-22, 2000.

[108] *Procedures for handling security patches*, NIST Special Publication 800-40, 2002.