Aspecte de criptografie utilizate în aplicaţiile Internet

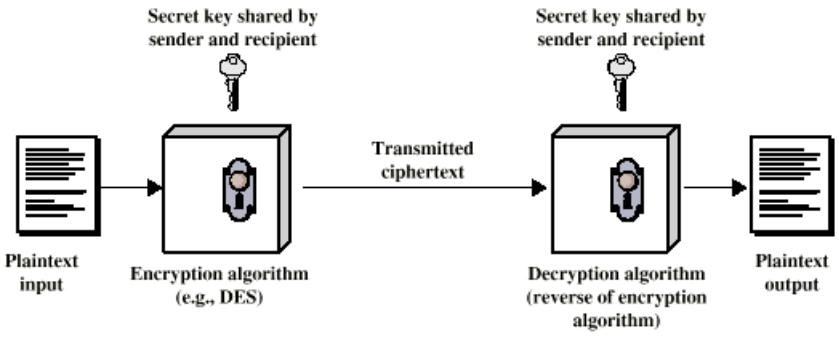
Concepte

- Hash, MD5, DES, AES, chei publice/private,...
- întîlnite în [aproape] toate protocoalele de nivel aplicaţie

Cerinţe de securitate

- Confidenţialitate
  - datele accesibile doar pt. cei autorizaţi
- Integritate
  - Datele nu pot fi modificate fără ca acest lucru să poată fi detectat

Criptare Convenţională



Ingrediente

- text clar (*plaintext*)
- algoritm de criptare
- cheie Secretă

- text cifrat (*ciphertext*)

- algoritm de decriptare

Cerinţe pt. securitate

- *Security thru obscurity* vs *strong security*
- al doilea caz: Algoritm de criptare puternic
  - chiar dacă este cunoscut să nu permită decriptarea sau deducerea cheii (doar cheia este secretă, nu şi algoritmul)
  - Chiar dacă sunt disponibile texte clare şi cifrate în număr limitat
- Emiţatorul şi receptorul trebuie să obţină cheia de criptare într-un mod sigur
- Cheia deconspirată - toată comunicaţia secretă este compromisă

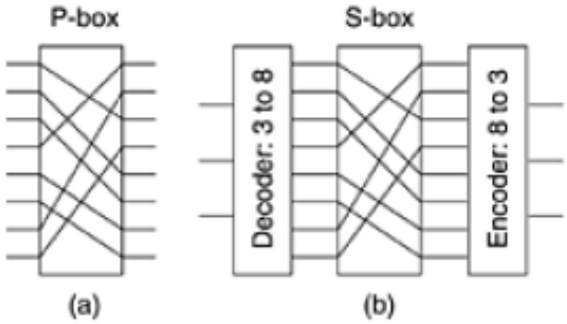
Atac asupra criptării

- Criptanaliză
  - Se bazează pe natura algoritmului şi cunoştinţe generale asupra caracteristicilor textului clar
  - Încercare de a deduce textul clar sau cheia
- Forţă brută (*brute force*)
  - Se încearcă toate cheile posibile pînă la obţinerea textului clar

Algoritmi

- Cifru *bloc*
  - Procesează text clar de dim. variabilă în blocuri de dim. fixă şi produce text cifrat de dim. fixă
  - Data encryption standard (DES)
  - Triple DES (3DES)

Blocuri componente pentru criptare



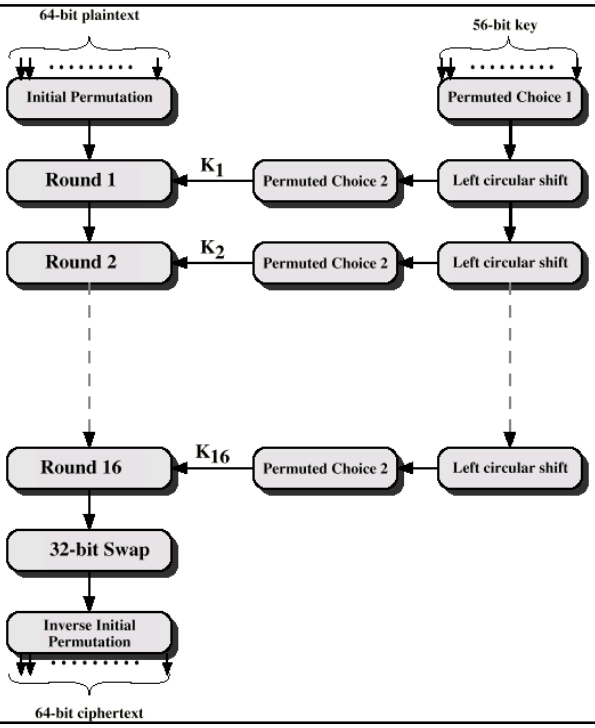
- P-box (*permutation*)
- S-box (*substitution*) = P-box cu 2^n intrări/ieşiri, aplicat unor cuvinte de n biţi

DES - Data Encryption Standard

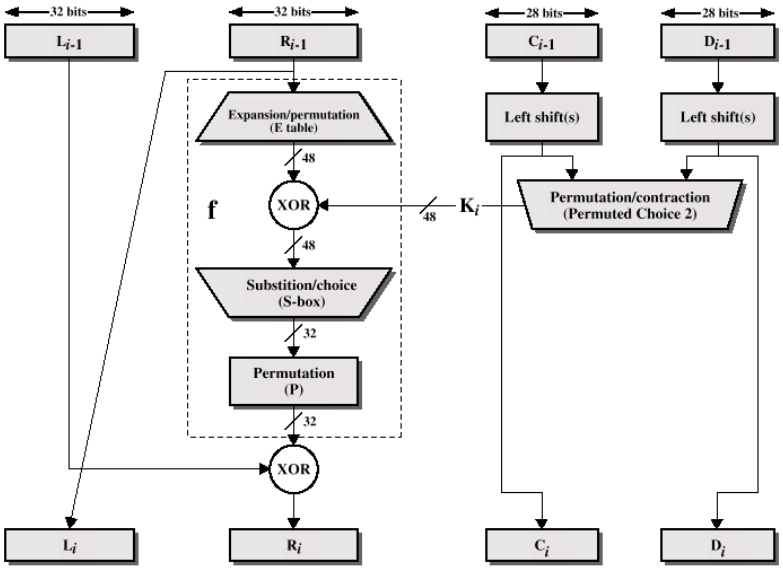
- standard US
- 1977: definit de *National Bureau of Standards*, acum *National Institute of Standards and Technology* (NIST)

- textul clar împărţit în blocuri de 64 biţi
- cheie de 56 biţi - *principala slăbiciune*

DES



detaliu: o iteraţie DES (din 16)



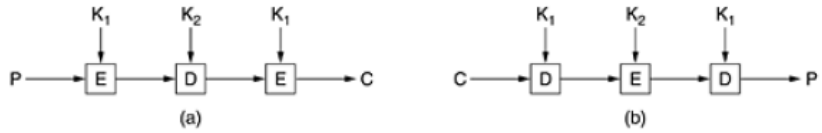
“Tăria” DES

- S-box ales: corespondenţă 48 biţi intrare → 32 biţi ieşire
- Structura aleasă pt. S-box nu a fost justificată matematic de către NIST (*take it as it is*) - suspiciuni de existenţă a unui “*backdoor”* - neconfirmate
- DES Declarat nesigur în 1998 (din cauza lungimii cheii - nu a vreunei vulnerabilităţi)
  - EFF: *Electronic Frontier Foundation*
  - DES Cracker machine, *Deep Crack*: cheie DES spartă în 22 ore.
- Alternativă directă: 3DES sau TDES

Triple DES

- ANSI X9.17 (1985)
- Incorporat în standardul DES, 1999
- Utilizează 3 chei şi 3 execuţii ale algoritmului DES
- Lungime cheie de 168 biţi

Triple DES



a) EDE (E=*Encryption,* D=*Decryption*)

avantaj: dacă cheile K1=K2, este echivalent cu DES → algoritm *backwards compatible* cu DES

dacă K1≠K2, cheia efectivă are 56+56=112b (considerat suficient la data propunerii)

b) EEE

cheia efectivă are 56+56+56=168b

avantaj: cheie mai puternică, considerat *sigur*

Algoritmi "succesori" ai DES

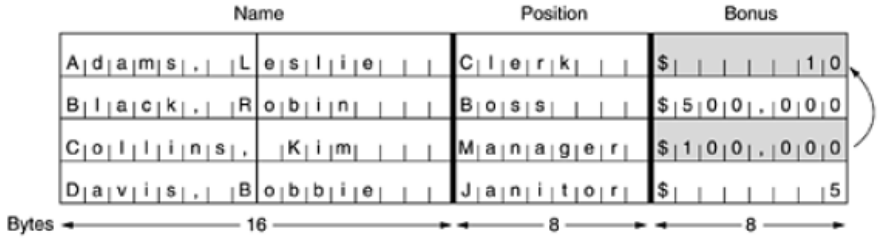
- AES: selectat de NIST prin concurs public: Joan Daemen şi Vincent Rijmen, *Rijndael*
- RC5
- IDEA
- BlowFish
- FEAL

- chei de 64-256 biţi

Dezavantaj *bock ciphers*

- DES, TDES, AES, etc: toate sînt *block ciphers*
- problemă: operează pe *blocuri* de n biţi
  - un mesaj M se divizează în blocuri B
  - M = BBBB....B
  - lungime M= oricît
  - lungime B =64b (DES, TDES) sau 128b (AES)
- Pentru un bloc B, E(B) =ct, indiferent de cîte ori este rulat şi de poziţia lui B în cadrul mesajului M

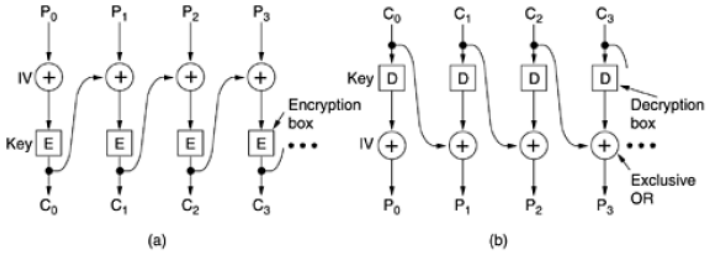
Exemplu de atac pe *block cipher*



- dimensiune B = 64b = 8 bytes dimensiune M = 16x8 bytes = 16xB
- E(M) = 16x E(B)
- E(M) nu poate fi decriptat abuziv, dar poate fi modificat, dacă e cunoscută structura lui M
- Adams ştie că nu primeşte bonus, dar că Collins primeşte
- Adams cunoaşte *structura* fişierului M (nu şi *conţinutul*) şi are acces la fişierul criptat E(M) înainte de a fi trimis la bancă
- Adams copiază al 12-lea bloc E(B) în poziţia 4 → va avea acelaşi bonus !
- Sursa: *Tanenbaum, Computer Networks, 4E*

Soluţia: CBC

- *Cipher Block Chaining*
- nu se mai “înlănţuie” E(M)=E(B)+E(B)+...+E(B)
- se foloseşte un *Initialization Vector* IV trimis ca atare (necriptat) împreună cu E(M)
  - Primul B este făcut XOR cu IV
  - Fiecare B ulterior este făcut XOR cu E(B) precedent
- rezultat: rezultatul fiecărui E(B) depinde de poziţia sa în M !
  - P = *Plaintext*, C = *Ciphertext*



Autentificarea Mesajelor

- Protecţie împotriva atacurilor active
  - Manipularea datelor
- Mesajul este autentic dacă este original şi provine de la sursa reală
- Autentificarea permite destinaţiei să verifice
  - Mesajul nu este alterat
  - Provine de la sursa reală
  - Secvenţierea mesajelor este corectă

Autentificare prin criptare

- Doar sursa şi destinaţia cunosc cheia
- Mesajul include:
  - cod detector de erori
  - număr de secvenţă
  - marcaj de timp

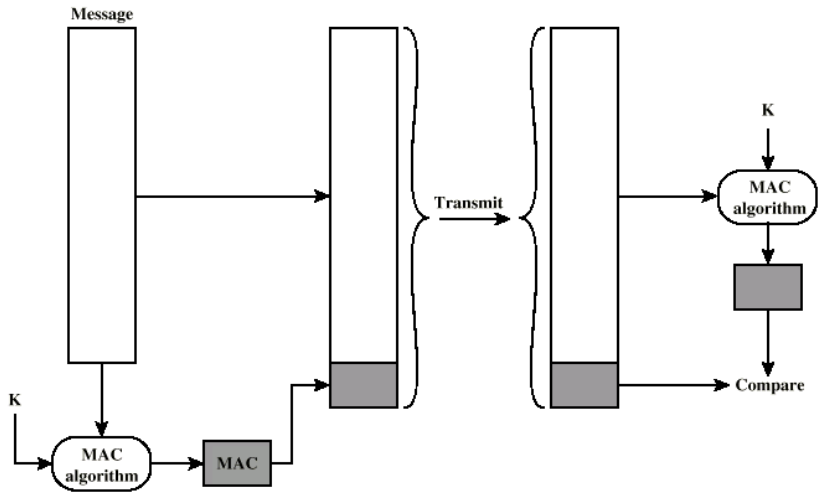
Autentificare fără criptare

- Etichetă de autentificare ataşată la fiecare mesaj
- Mesajul nu este criptat
- Util pentru:
  - Destinaţii multiple, numai una face autentificarea

Message Authentication Code (MAC)

- Generează codul de autentificare din mesaj şi cheie comună
- Cheia comună cunoscută doar de A şi B
- Dacă codul de autentificare corespunde:
  - Mesajul nu a fost alterat
  - Expeditorul este autentic
  - Dacă este şi nr de secvenţă, acesta este corect

Autentificarea mesajului folosind MAC



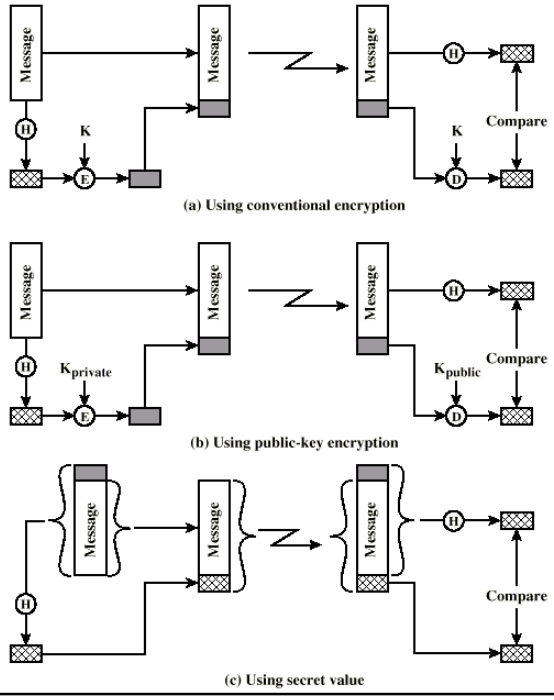
Funcţie *one way hash*

- Acceptă mesaje de lung. variabilă şi rezultă un *hash* (sumar) de lg. fixă
- Avantaje ale autentificării fără criptare
  - Criptarea este lentă
  - Hardware pt criptare este scump
  - Algoritmi sunt acoperiţi de patente
  - Controlul exportului (din USA)

Utilizarea *hash H* pentru autentificare

a,b) cu criptare

c) fără criptare



Autentificarea fără criptare (varianta c)

- Mesajul M, valoarea secretă S
- S e cunoscută de A şi B, nu se transmite prin reţea niciodată
- H = Hash(S+M)
- A trimite M+H către B

- B efectuează H' = Hash(S+M)
- B compară H' cu H
- H'==H ? mesajul e autentic

Funcţii *Hash* sigure

- Funcţia *hash* are următoarele proprietăţi:
  - Poate fi aplicată pt. orice lungime
  - Lungimea codului este fixă
  - Uşor şi rapid de calculat
  - Nu poate fi inversată
  - Este foarte dificil de a se găsi două mesaje diferite cu hash identic ***(singura vulnerabilitate reală)***
  - Hash foarte diferit chiar pt. 2 mesaje care diferă foarte puţin

Funcţii *hash*

- SHA-1: *Secure Hash Algorithm 1 (*NIST, 1993)
- Mesaj de intrare: max. 2^{64} biţi
  - procesat în blocuri de 512 biţi
- Ieşire: 160 bit *digest*

- MD5: *Message Digest 5* (Rivest, 1992)
- Mesaj de intrare: oricît
- Ieşire: 128 bit *digest*
- În loc de S-box foloseşte o tabelă a funcţiei *sin()*, pt. a elimina suspiciunile “de ce s-a ales o anumită structură a S-Box (vezi cazul DES)

Demo: md5sum
md5sum -c (check)

Exemplu MD5

•funcţie predefinită în PHP:

```
<?php
$string = 'Something';
$hash = md5($string);
echo $hash;
?>
```

rezultat: **73f9977556584a369800e775b48f3dbe**

funcţii *hash* în /etc/passwd, /etc/shadow

/etc/passwd

user1:x:1242:1200:Gigi:/home/gigi:/bin/bash

/etc/shadow

user1:6H0rJDfQL$q9oeEw6Q6BR7R6K6B9upfo1YYDt.k287RsrHMlffM9kCD.ziHLieR5VS3vcqp1PhB1YOUO8ji7qoC72QULsn.:14700:0:99999:7::

Caracterele n: tipul de *hash* utilizat

- nimic: DES (primele 8 ch. din parolă= cheia, text=00000000)
- 1: MD5
- 2: Blowfish
- 5, 6: SHA-256, SHA-512

funcţii *hash* în /etc/shadow

/etc/shadow

user1:6H0rJDfQL$q9oeEw6Q6BR7R6K6B9upfo1YYDt.k287RsrHMlffM9kCD.ziHLieR5VS3vcqp1PhB1YOUO8ji7qoC72QULsn.:14700:0:99999:7::

- 14700: nr de zile, începînd cu 1 ian 1970, de cînd parola a fost schimbată ultima oară (1 ian 1970 s.n. *epoch*)
- 0: nr. de zile minim în care parola nu poate fi schimbată (0= nu e cazul)
- 999999: nr. de zile maxim în care parola poate fi folosită înainte de a expira
- 7: nr. de zile în care utilizatorul e avertizat despre expirarea parolei
- :: cîmpuri rezervate

funcţii *hash* în /etc/shadow

/etc/shadow

user1:6H0rJDfQL$q9oeEw6Q6BR7R6K6B9upfo1YYDt.k287RsrHMlffM9kC
D.ziHLieR5VS3vcqp1PhB1YOUO8ji7qoC72QULsn.:14700:0:99999:7::

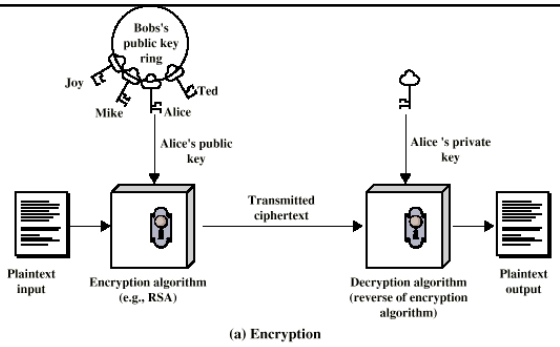
- grupul colorat între $$: *salt* (engl)
- un *salt* este un grup de caractere aleatoare
- se adaugă la parola în clar înainte de *hash*
- rol: aceeaşi parolă introdusă de 2 ori nu generează acelaşi *salt*

Criptare cu cheie publică (PK)

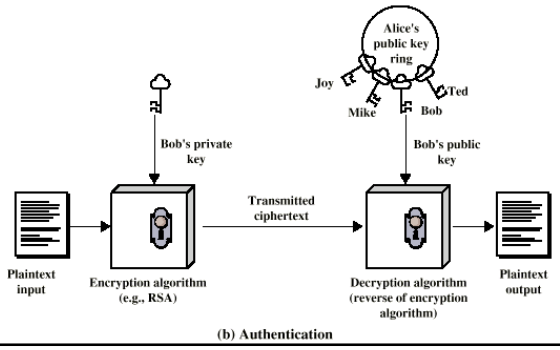
- Origine: Diffie-Helman, 1976
- Se bazează pe alg. matematici (uzual: RSA)
- Asimetric: două chei, publică + privată
- Ingrediente
  - Text clar
  - Algoritm criptare
  - Cheie publică şi privată
  - Text cifrat
  - Algoritm de decriptare
- Avantaj: nu este necesar schimbul prealabil de chei; oricine poate comunica în secret cu oricine fără să se întîlnească în prealabil
- Dezavantaj: chei mai lungi (min. 1024b) decît la algoritmii simetrici, pt. acelaşi nivel de securitate

Criptare cu PK

(a) Bob trimite un mesaj criptat lui Alice, folosind cheia publică a lui Alice; doar Alice poate decripta



(b) Bob semnează un mesaj pt. Alice; oricine poate autentifica semnătura, doar Bob îl poate modifica



Cheie Publică - Operare

- Una dintre chei este publică
  - cea pentru criptare
- A doua cheie este privată
  - cu ea se face decriptarea
- Nu se poate determina cheia de decriptare cunoscînd cheia de criptare şi algoritmul
- Din perechea de chei oricare poate fi utilizată pt. criptare şi a doua pt. decriptare

Metoda

- Utilizatorul generează cele două chei
- Una din chei este făcută publică (*E*)
- Cealaltă rămîne privată (*D*)

- Pt. a trimite mesaj la destinatar se criptează cu cheia publică a acestuia
- Destinatarul decriptează cu cheia sa privată

- Mesajul clar M → mesajul criptat E(M) → mesajul decriptat D(E(M)) = M

Semnătură digitală

- Posibilă dacă D şi E sînt simetrice - exemplu RSA:
  D(E(M)) = E(D(M))
- Expeditorul criptează mesajul cu cheia sa privată
- Destinatarul poate să decripteze cu cheia publică a expeditorului
- Nu se asigură protecţia datelor
  - Cheia de decriptare este publică - oricine, nu numai destinatarul, poate să decripteze
- Aceasta autentifică expeditorul, fiind unicul care posedă cheia privată D
- nu există alt D' a.î. D'(E(M)) = M

Algoritmul RSA [2]

Generarea cheilor

- se selectează *p,q* prime
- se calculează $n = p \cdot q$
- se calculează $\Phi(n) = (p-1)(q-1)$
- se alege *e* a.î. $\Phi(n)$ şi *e* prime între ele (adică cmmdc($\Phi(n)$,*e*)=1)
- se calculează $d=e^{-1} \bmod \Phi(n)$

- cheia publică este KU = {*e,n*}
- cheia privată este KD = {*d,n*}

Algoritmul RSA

Justificare: $\Phi(n) = (p-1)(q-1)$, *funcţia totient a lui Euler;* $\Phi(n)$ reprezintă numărul de numere întregi < *n*, relativ prime faţă de *n*

Criptarea se face astfel:

- textul clar $M < n$
- textul cifrat $C = M^e \pmod n$

Decriptarea se face astfel:

- textul cifrat C
- textul clar $M = C^d \pmod m$

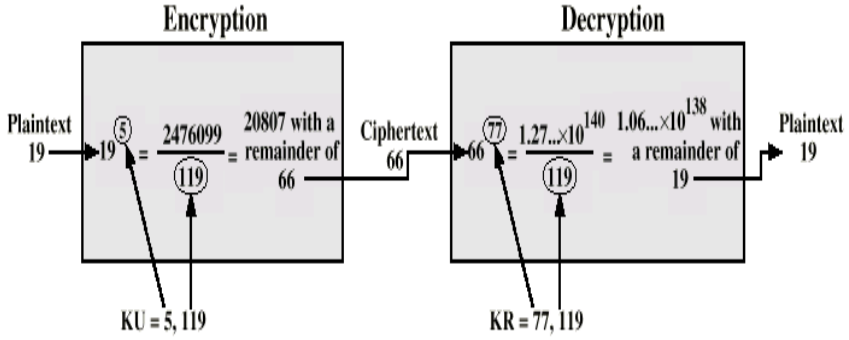
Exemplu 1 RSA

- alegem p, q prime: p=7, q=17
- calculăm $n=pq$ = 7*17 = 119
- calc Φ(n) = $(p-1)(q-1)$ = 96
- alegem cheile e şi d :
- alegem e a.î. (e, 96) prime între ele şi e < 96; de exemplu e=5 (de obicei se alege e mic)
- alegem d a.î. $e*d$ mod $96 = 1$ şi d < 96; de exemplu d=77, 77*5=385=4*96+1

Cheia publică {e,n}= {5,119}

Cheia privată {d,n}= {77,119}

Exemplu 2 RSA



“Tăria” algoritmului RSA

- ex: cheia publică {5,119}; cheia privată {77,119}
- 119 produs de 2 nr. prime, ales de 3 cifre
- factorizarea unui nr. mare (> 100 cifre, > 300 biţi) în numere prime - timp de lucru f. mare
- n=119 cunoscut - factorii p, q necunoscuţi

“Tăria” algoritmului RSA - Factorizarea

Timp de descompunere în factori primi a unui număr de *N* cifre zecimale, dacă o operaţie durează 1μs (algoritmul lui Schroeppel)

N	nr. de operaţii	Timp
50	$1.4*10^{10}$	3.9 ore
75	$9.0*10^{12}$	104 zile
100	$2.3*10^{15}$	74 ani
200	$1.2*10^{23}$	$3.8*10^{9}$ ani
300	$1.5*10^{29}$	$4.9*10^{15}$ ani
500	$1.3*10^{39}$	$4.2*10^{25}$ ani

Demonstraţie chei publice (GPG)

- se instalează GPG (*Gnu Privacy Guard,* versiune *free* a PGP)
- userii asi1 şi asi2 îşi crează fiecare cheile, folosind
  asi1@matrix:~$ gpg --gen-key
- fiecare user îşi exportă cheia **publică** şi i-o trimite celuilalt, care o importă
  asi1@matrix:~$ gpg --export -a -o public1
  asi2@matrix:~$ gpg --import public1
- asi1 criptează fişierul *fisier1* şi-l trimite lui asi2:
  asi1@matrix:~$ gpg --encrypt -a fisier1
  (-a = ascii; se obţine *fisier1.asc;* gpg va cere specificarea destinatarului şi va folosi cheia **publică** a acestuia*)*
- asi2 decriptează fişierul primit
  asi2@matrix:~$ gpg --decrypt fisier1.asc
  (gpg va cere cheie **privată** a userului asi2)

Demonstraţie chei publice (GPG) [2]

- se foloseşte GPG pentru semnături
- asi1 semnează fişierul fisier1, fără a-l cripta
  asi1@matrix:~$ gpg --clearsign fisier1
  (clearsign produce varianta ascii, nu binară)

- asi2 primeşte fişierul şi-i verifică semnătura:
  gpg --verify fisier1.asc
  gpg: Good signature from "asi 1 <asi1@matrix>“

- dacă fişierul este alterat:
  gpg: BAD signature from "asi 1 <asi1@matrix>"

- *OBS: opţiunile --encrypt şi --sign se pot combina.*

Bibliografie

[1] William Stallings, *Data and Computer Communications,* Capitolul 18

[2] Algoritmul RSA:

R.L. Rivest, A. Shamir, and L. Adleman*, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*

[3] Tanenbaum, *Computer Networks,* 4ed.