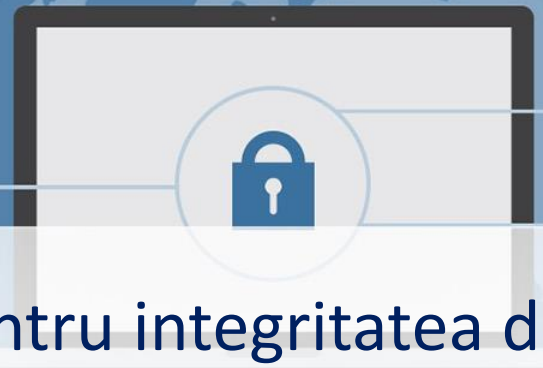
Criptografie și Securitate Cibernetică

RCC - CSC 3

Conținut

- Algoritmi criptografici pentru integritatea datelor
  - Integritatea datelor
  - Funcții criptografice *hash*
  - Coduri de autentificare
  - Semnături digitale
- Managementul si distribuirea cheilor de securitate
  - Distribuirea cheilor în sisteme simetrice
  - Distribuirea cheilor în sisteme asimetrice
  - Distribuirea cheilor publice
  - Certificate X.509
  - Infrastructura sisteme cu chei publice

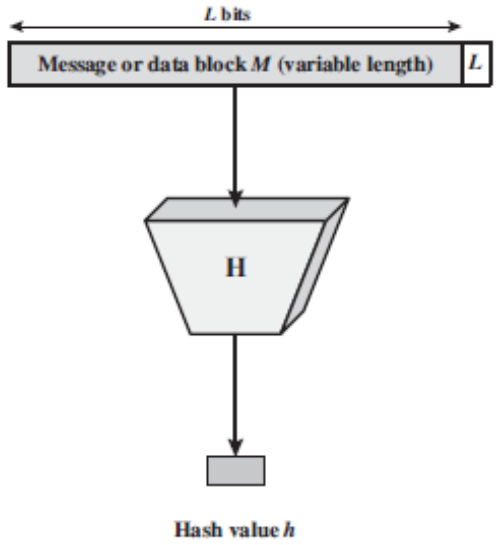
Integritatea datelor

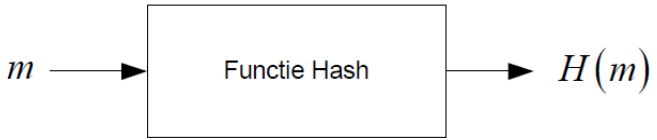


- Algoritmi criptografici pentru integritatea datelor
  - Integritatea datelor
  - Funcții criptografice *hash*
  - Coduri de autentificare
  - Semnături digitale

Funcții criptografice *hash*

- Funcția *hash*
  - la intrare mesaje de dimensiune variabilă
  - returnează un mesaj de lungime fixă
  - mesajul inițial nu poate fi recuperat
  - nu se utilizează niciun fel de cheie
  - ieşirea se mai numeşte şi **etichetă** (**tag**)
- Sumă criptografică fără cheie (nu necesită o cheie criptografică)
  - cele mai cunoscute – MD5 şi SHA-1
  - altele: MD4, HAVAL şi Snefru





Funcții *hash*
Proprietăți

- Proprietăţi de securitate
  - rezistenţa imaginii (*preimage resistance*)
    având *y* o ieşire a funcţiei, nu se poate găsi *x* astfel încât
    $y = H(x)$
  - rezistenţă secundară a imaginii (secondary preimage resistance)
    având x, H(x) nu se poate găsi x' astfel încât
    $H(x) = H(x')$
  - rezistenţă la coliziune (collision resistance)
    nu se poate găsi o pereche x, x' astfel încât
    $H(x) = H(x')$

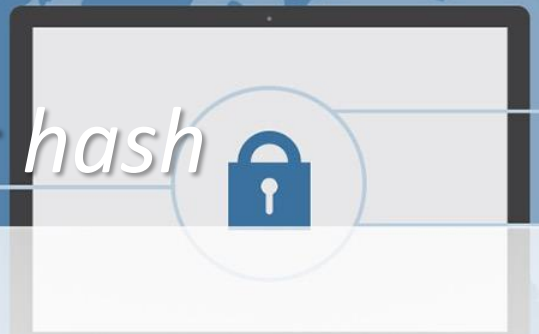
Funcții *hash*
Construcție

- Construcţia funcţiilor hash
  - conceptul de funcţie hash iterată
  - spargerea intrării în blocuri de dimensiune fixă
  - care sunt trecute printr-o funcţie de compresie
  - în cadrul căreia se efectuează operaţii specifice

    Ex.: construcţia Merkle-Damgard

IV - vector de iniţializare

Aplicații ale funcțiilor *hash*



- Autentificarea mesajelor
  - mecanism sau serviciu folosit pentru a verifica integritatea
  - asigură că datele primite sunt exact la fel ca cele trimise (fără modificare, inserare, ștergere, sau repetare)
  - funcția mai este denumită *message digest (*rezumat)
  - coduri de autentificare mesaj (MAC), funcție *hash* cu cheie
- Semnături digitale
  - valoarea *hash* a unui mesaj este criptata cu cheia privată
  - folosind cheia publică, se poate verifica integritatea mesajului
- Alte aplicații
  - fișiere cu parole (stocate *hash*, in loc de parolele efective)
  - detectarea intruziunilor și detectarea virușilor (*hash* pentru fișiere)
  - funcții pseudo-aleatoare (generarea de numere pseudo-aleatoare)

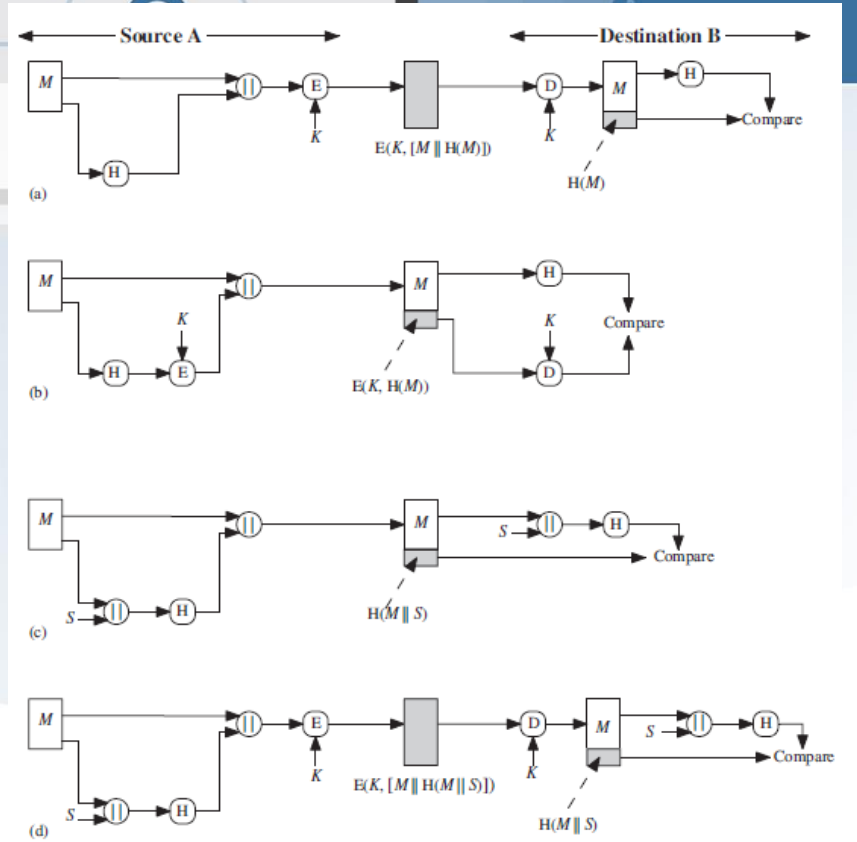
Utilizare HASH autentificare mesaje

Mesajul și codul de distribuire concatenate sunt criptate prin criptare simetrică

Numai codul de distribuire este criptat folosind criptarea simetrică

Este posibil să se utilizeze o funcție *hash* dar fără criptare

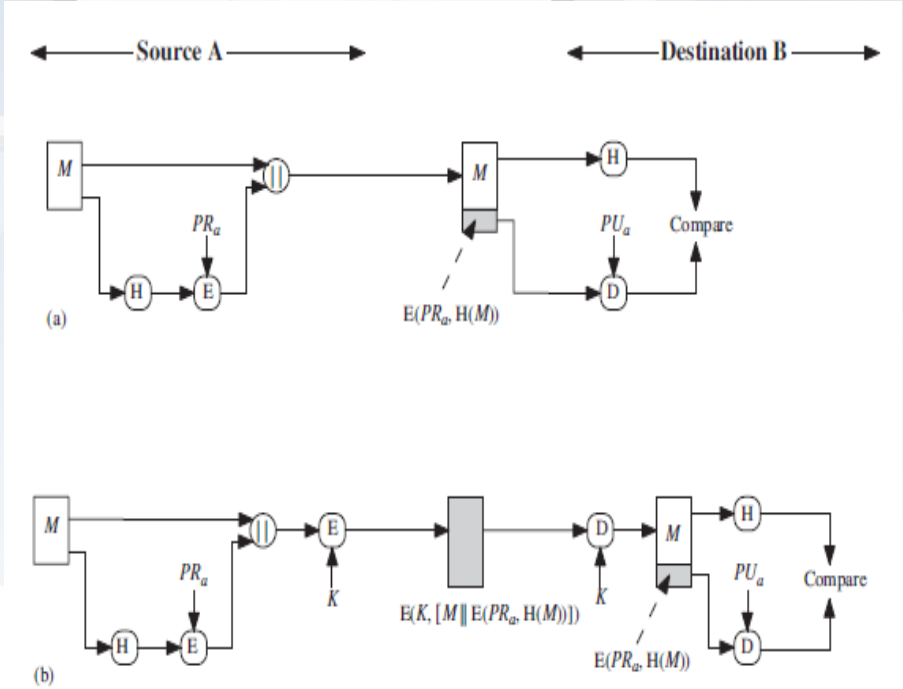
Confidențialitatea poate fi asigurată, prin criptarea întregului mesaj, inclusiv codul de distribuire.



Utilizare HASH semnături digitale

Codul de distribuire este criptat, utilizând criptarea cu cheie publică cu cheia privată a expeditorului. Se oferă autentificare și de asemenea, o semnătură digitală, pentru că numai expeditorului ar fi putut produce codul de distribuire criptata.

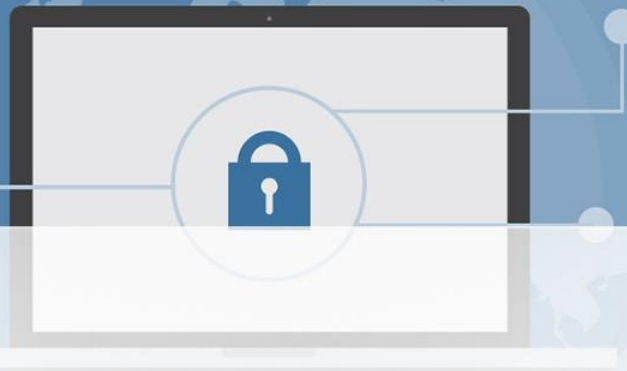
Pentru confidențialitate și semnătură digitală, mesajul împreuna cu codul *hash* criptat cu cheia privată, pot fi criptate folosind o cheie secretă simetrică.



Funcții *hash* Versiuni

- Funcții *hash* frecvent utilizate
  - MD - *Message Digest*
    - MD5 (foarte utilizată)
  - SHA - *Secure Hash Algorithm*
    - SHA-1
    - SHA2 (cea mai utilizată)
      - dimensiunea ieşirii este 224, 256, 384, 512 biţi
      - indiferent de dimensiunea datelor de intrare
- MD5 si SHA1 – nesigure
  - (nu mai oferă rezistenţă secundară a imaginii)
- Se recomandă folosirea SHA-256

Message Digest MD5



- MD2, MD4, MD5 (Ronald Rivest)
  - Produce un rezumat de 128 biti;
  - MD2 este cel mai sigur, mai greu de calculat (rar folosit)
  - MD4 – alternativa rapidă
- MD5 (MD4 modificat)
  - Realizat in 1991
  - urmărește construcția Merkle-Damgard
  - operează cu blocuri de mesaj de câte 512 biți.
  - mesajul inițial se concatenează cu un bit de 1 și apoi cu numărul necesar de 0-uri.
  - ultimii 64 de biți din mesajul preprocesat reprezintă lungimea mesajului inițial.

MD5 - algoritm

- Algoritm MD5
  - constă în 64 de iteraţii, grupate în 4 runde de câte 16 iteraţii
  - fiecare rundă se efectuează de 16 ori deoarece blocul procesat este de 512 biţi, în timp ce funcţia de rundă procesează 32 de biţi la un moment dat (16x32=512).
  - în fiecare rundă folosindu-se una din 4 funcțiile neliniare:

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

  - vectorii de inițializare (blocuri de stare)

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

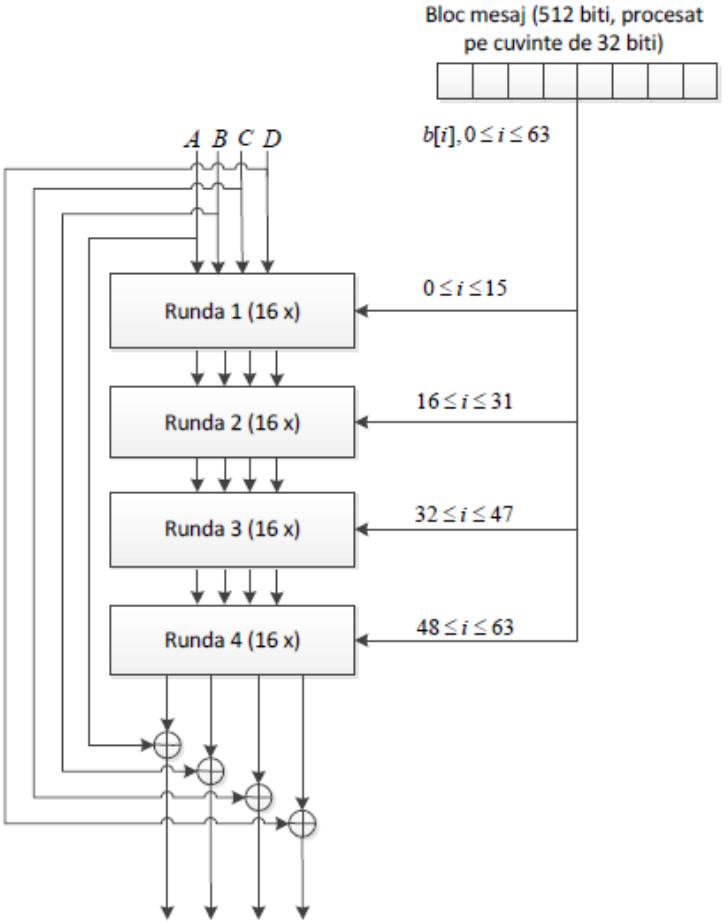
MD5 structură

Funcţii (in cadrul rundelor)

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

Blocuri de stare

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



MD5 Funcționare

- Funcțiile MD5
  - toate cele 4 funcții sunt neliniare
  - se bazează pe operaţii simple la nivel de bit
  - efectuează aceeaşi operaţie de 16 ori,

$$B + ((A + FR(B,C,D) + M + K) <<< s)$$

$$D \leftarrow C,$$
$$C \leftarrow B,$$
$$B \leftarrow B + ((A + FR(B,C,D) + M + K) <<< S),$$
$$A \leftarrow D.$$

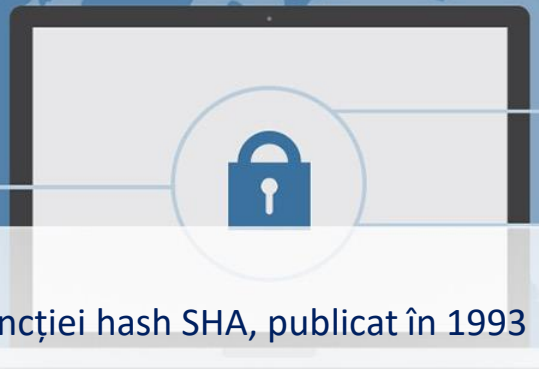
  - rezultatul se depune în *B*
  - valorile de stare se interschimbă
  - *FR* funcţia de rundă (pe 32 de biţi)
    - (*F* în runda 1, *G* în runda 2, *H* în runda 3 şi *I* în runda 4),
  - M este un bloc de 32 de biţi al mesajului
  - *K* şi *S* sunt valori numerice predefinite

MD5 exemple

- Vectori de test pentru MD5
  - RFC 1321
  - rezultatul are întotdeauna 128 de biţi

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

Secure Hash Algorithm
SHA



- Versiuni SHA
  - SHA-0 - versiunea originală a funcției hash SHA, publicat în 1993
    - 160 de biți
  - SHA-1 - funcție *hash* de 160 de biți,
    - seamănă cu algoritmul MD5
    - proiectat să facă parte din Digital Signature Algorithm.
    - nu a mai fost utilizat după 2010.
  - SHA-2 - o familie de două funcții *hash* similare,
    - SHA-256 și SHA-512, cu diferite dimensiuni bloc,
    - diferă în mărime SHA-256 - 32 biți iar SHA-512 - 64 de biți,
    - versiuni trunchiate, SHA-224, SHA-384, SHA-512/224, SHA-512/256.
  - SHA-3: - funcție *hash* numit anterior Keccak,
    - ales în 2012, după o competiție publică
    - susține aceleași lungimi *hash* ca SHA-2
    - structura sa internă diferă semnificativ de restul familiei SHA.

SHA2

- SHA2 este construit pe principii similare MD5
  - dispune de 4 funcții neliniare, mai complexe, cu 3 intrări
  - se folosesc 8 blocuri de stare (*A, B, C, D, E, F, G, H*), 32 de biți
- Pentru SHA-256 cele 4 funcții neliniare sunt:

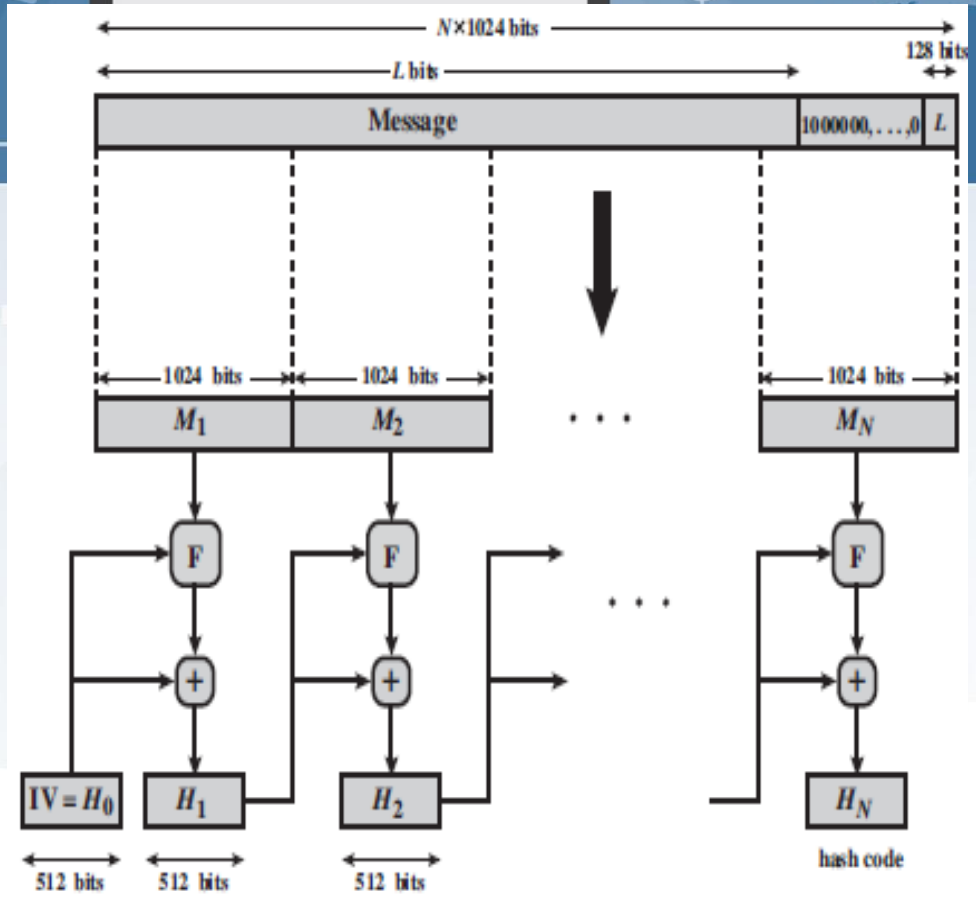
$$Ch(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G),$$
$$Ma(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C),$$
$$\Sigma_0(A) = (A >>> 2) \oplus (A >>> 13) \oplus (A >>> 22),$$
$$\Sigma_1(E) = (E >>> 6) \oplus (E >>> 11) \oplus (E >>> 25).$$

- Numărul de runde SHA2
  - 64 - SHA-256 şi SHA-224
  - 80 - SHA-384 şi SHA-512

SHA-512 - Algoritm

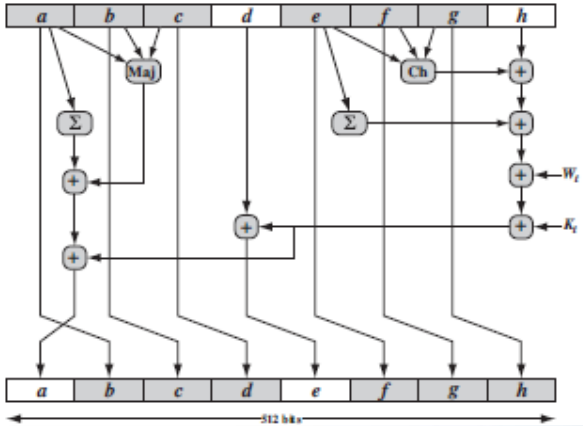
1. Adăugare biți de umplutură
   - Lungime, (L)=896(mod 1024)
   - “1” + (L-1) biți “0”.
2. Adăugare bloc de 128 biți
   - lungimea mesajului original
3. Inițializare buffer (512 biți)
   - regiștri a, b, c, d , e, f, g, h
4. Procesare mesaj
   - blocuri 1024 biți (128 cuvinte)
5. Obținere *hash*
   - ieșirea din ultimul bloc

```
a = 6A09E667F3BCC908   e = 510E527FADE682D1
b = BB67AE8584CAA73B   f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B   g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1   h = 5BE0CD19137E2179
```

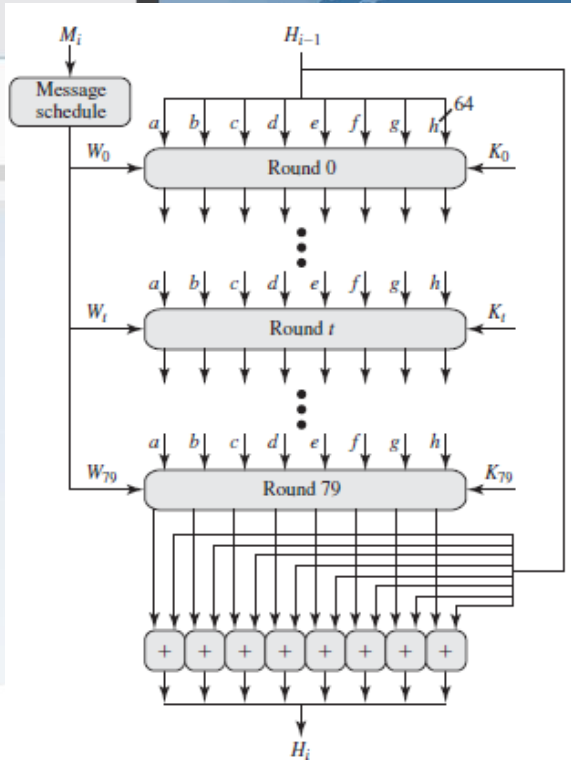


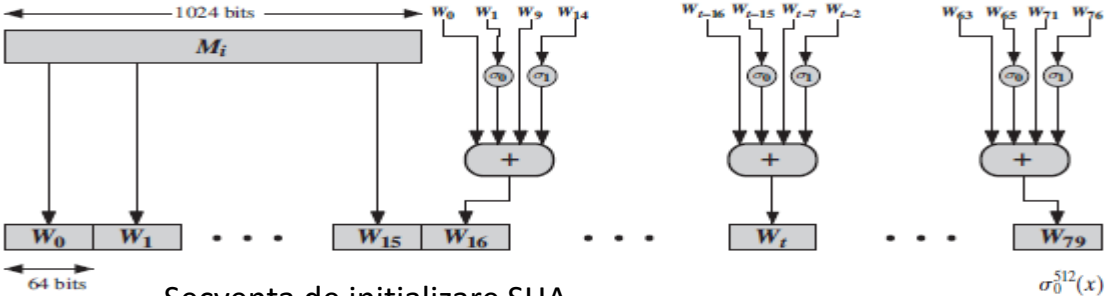
SHA-512 – procesare

Runda SHA



Funcția SHA





Secvența de inițializare SHA

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$

SHA - Exemplu

- Mesaj inițial = “abc” (ASCI) → 01100001 01100010 01100011 (binar)
- Adăugare biți de umplutură
- 896 modulo 1024 → 896 - 24 = 872 biți (“1” + 871 “0” biți.)
- Adăugare lungime mesaj original
- Lungimea (24 biți) este adăugata la sfârșit → 18 (hexa)

```
6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018
```

- Masaj (bloc de 1024 biți)
- Secvența de inițializare SHA

$W_0 = 6162638000000000$
$W_1 = 0000000000000000$
$W_2 = 0000000000000000$
$W_3 = 0000000000000000$
$W_4 = 0000000000000000$
$W_5 = 0000000000000000$
$W_6 = 0000000000000000$
$W_7 = 0000000000000000$
$W_8 = 0000000000000000$
$W_9 = 0000000000000000$
$W_{10} = 0000000000000000$
$W_{11} = 0000000000000000$
$W_{12} = 0000000000000000$
$W_{13} = 0000000000000000$
$W_{14} = 0000000000000000$
$W_{15} = 0000000000000018$

SHA - Exemplu

- Buffer registri a, b, c, d , e, f, g, h după primele 2 runde

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

După runda 80

```
73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9
d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326
```

SHA - Exemplu

- Procesare hash

$$H_{1,0} = 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba$$
$$H_{1,1} = bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131$$
$$H_{1,2} = 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2$$
$$H_{1,3} = a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a$$
$$H_{1,4} = 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8$$
$$H_{1,5} = 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd$$
$$H_{1,6} = 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e$$
$$H_{1,7} = 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f$$

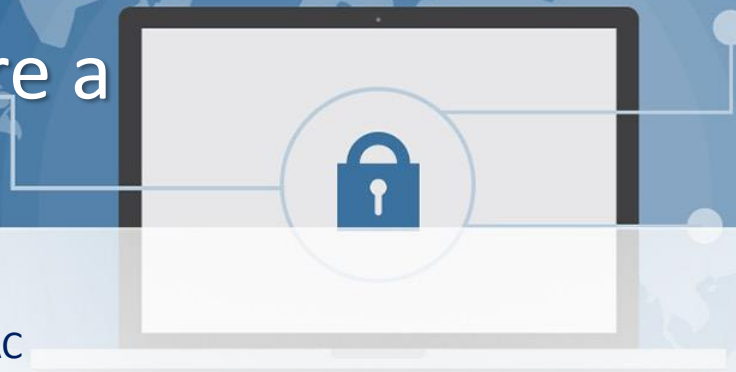
– Rezultat

```
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f
```

– Alt hash “cbc”

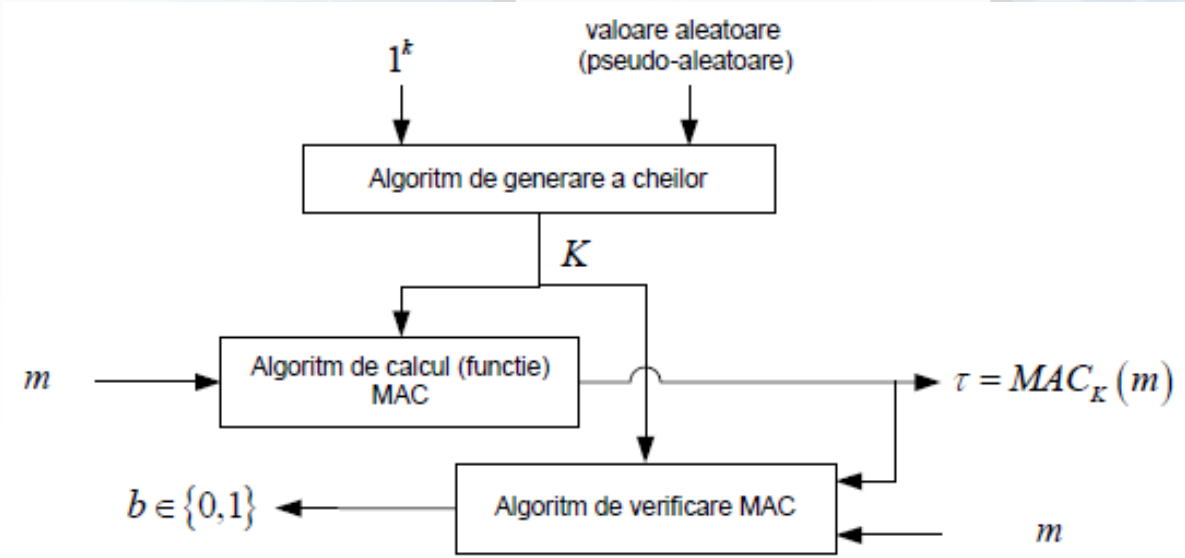
```
531668966ee79b70 0b8e593261101354 4273f7ef7b31f279 2a7ef68d53f93264
319c165ad96d9187 55e6a204c2607e27 6e05cdf993a64c85 ef9e1e125c0f925f
```

Coduri de autentificare a mesajelor



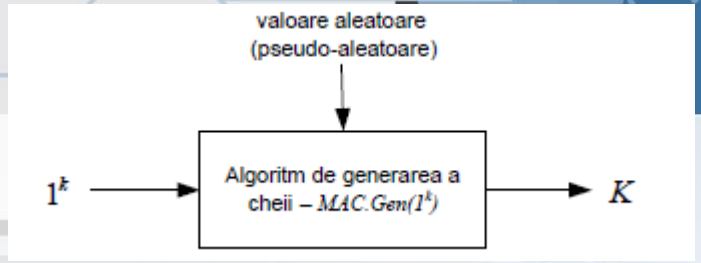
- Codurile de autentificare a mesajelor
  - Message Authentication Codes - MAC
  - Hash-uri cu cheie
  - Pentru a testa autenticitatea unei informații
  - pentru a verifica sursa de proveniență
  - Versiuni HMAC şi NMAC
- Un cod de autentificare a mesajelor (MAC) constă în trei algoritmi:
  - algoritmul de generare a cheii
    primește ca intrare nivelul de securitate k şi returnează cheia K
    $K \leftarrow MAC.Gen(1^k)$
  - algoritmul de etichetare
    primește mesajul m şi cheia K returnând eticheta
    $MAC_K(m) \leftarrow MAC.Tag(m, K)$
  - algoritmul de verificare
    primește mesajul, cheia şi eticheta returnând o valoare binară
    care este 1 dacă şi numai dacă eticheta corespunde perechii cheie-mesaj.
    $MAC.Ver(m, K, \tau)$

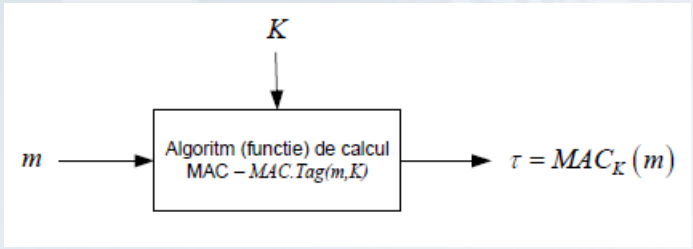
Sistem MAC
- schema bloc

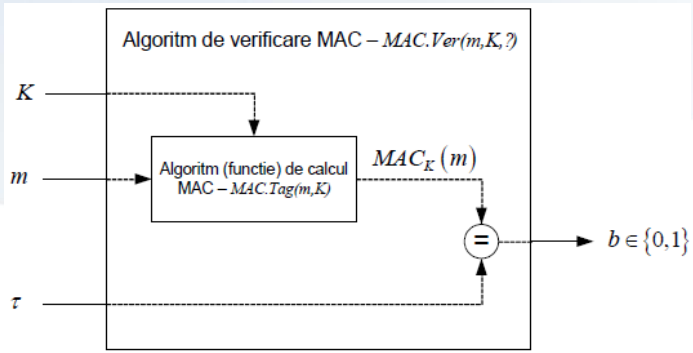


Sistem MAC - componente

- Algoritm de generare cheie →
- Algoritm de calcul MAC →
- Algoritm de verificare MAC →

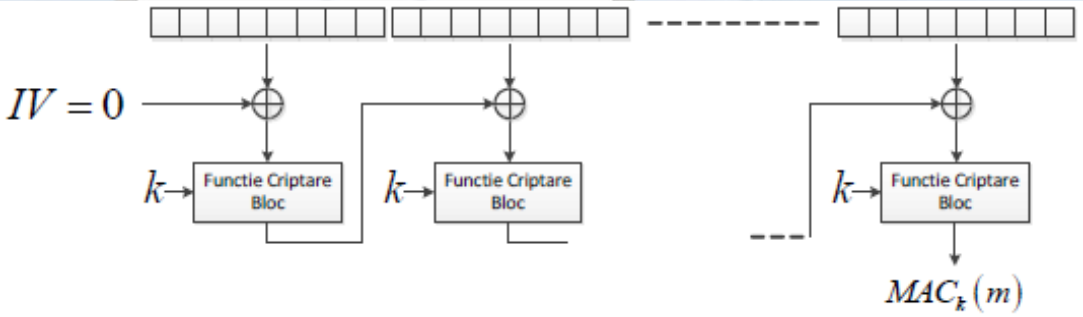






Sisteme MAC

- Sisteme criptografice cu coduri MAC
  - CBC-MAC (Cipher Block Chaining MAC)
    permite construirea unui MAC folosind un simplu cod bloc.

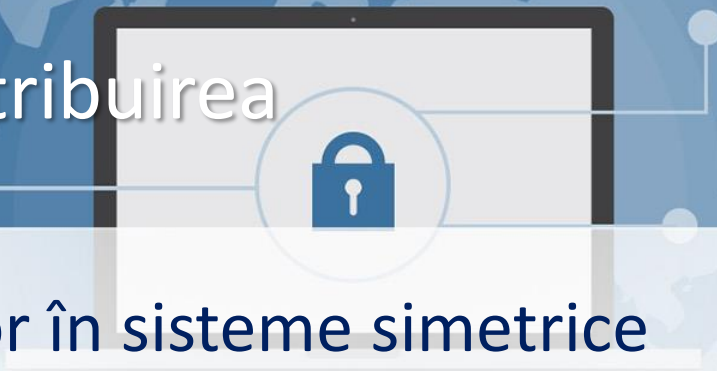


  - H-MAC
    se bazează pe o funcţie *hash* (simplitate şi eficienţă)
    aplicarea de două ori a unei funcţii hash folosind de fiecare dată alt padding,
    primul este denumit *ipad* (inner-padding) şi cel de-al doilea *opad* (outer-padding)

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

  - N-MAC (Nested MAC)
    necesită modificarea *IV*-ului pentru funcția *hash*

Managementul și distribuirea cheilor de securitate



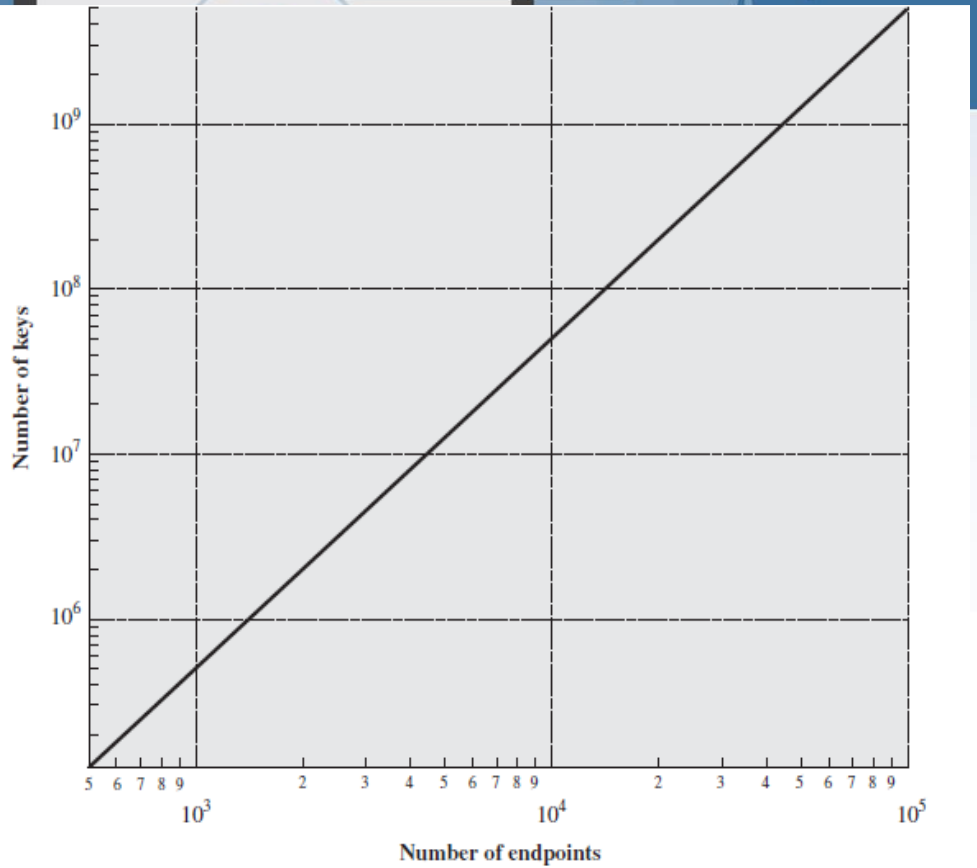
- Distribuirea cheilor în sisteme simetrice
- Distribuirea cheilor în sisteme asimetrice
- Distribuirea cheilor publice
- Certificate X.509
- Infrastructura sisteme cu chei publice

Sisteme criptografice simetrice

- Caracteristici
  - Cele 2 entități folosesc aceeași cheie
  - Cheia trebuie păstrată secretă
- Distribuirea cheilor
  - Tehnica prin care se realizează schimbul cheilor
  - Soluții de distribuire între 2 entități (A și B)
    1. A poate alege o cheie și o trimite către B
    2. O a treia entitate poate alege cheia și distribui către A și B
    3. Daca A și B au mai folosit recent o cheie, o entitate poate transmite celeilalte o noua cheie criptată pa baza celei vechi
    4. Dacă A și B au fiecare câte e conexiune criptată cu C, C poate distribui o cheie pe legăturile criptate către A și B.

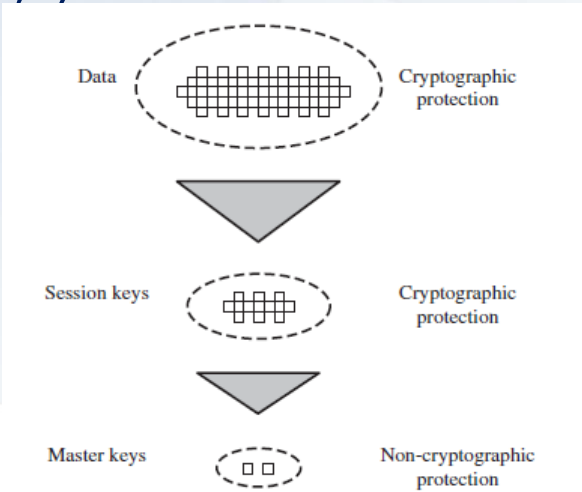
Criptarea punct la punct

- Complexitatea distribuirii
  - Depinde de numărul de perechi ce intră în comunicație
  - De ex. în rețeaua IP, este necesară o cheie pentru fiecare host din rețea cu care se dorește o comunicație criptată
  - Pentru N hosturi, vor fi necesare [N(N-1)]/2 chei
  - Pentru criptarea la nivel de aplicație, este necesară o cheie pentru fiecare pereche de utilizatori sau procese



Distribuirea centralizată

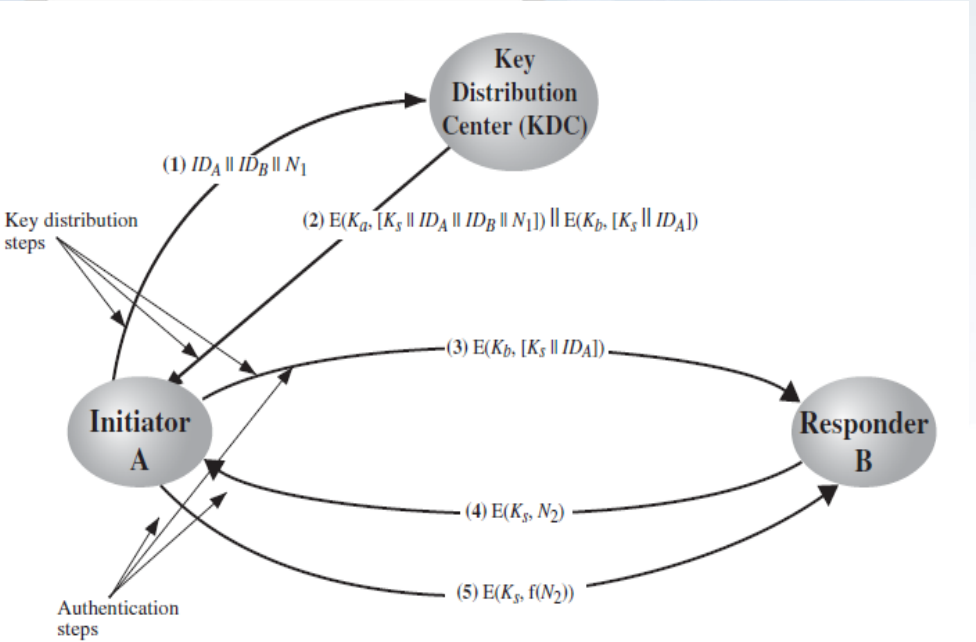
- Există un centru de distribuirea a cheilor
  - Responsabil cu distribuirea cheilor către perechile de entități (utilizatori, hosturi, procese, aplicații) ce intră în comunicație
  - Se bazează pe o ierarhie a cheilor
    - Minim 2 niveluri
      - Chei de sesiune
        comunicația este criptată cu o cheie temporară pe durata unei conexiuni logice (sesiune)
      - Chei master
        cheile de sesiune sunt transmise criptat pe baza unei chei master



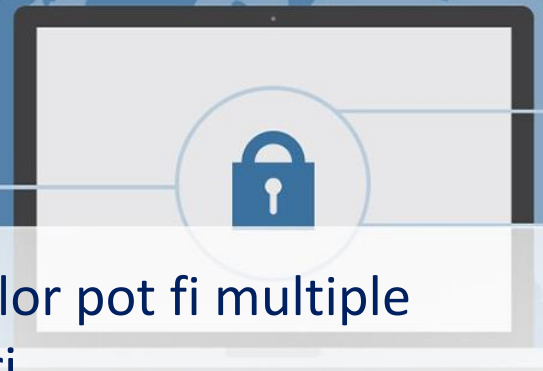
Scenariu de distribuire

- Fiecare utilizator folosește o cheie unică la nivelul centrului de distribuire a cheilor
  - 1. A cere la K, cheie sesiune
  - 2. K trimite criptat
    - cheia sesiunii, Ks
    - mesajul original (Ka)
    - un identificator A (IDa)

    pentru B (criptat cu Kb)
  - 3. A Păstrează Ks și
    - trimite IDa si Ks la B
  - 4. B răspunde cu N2
  - A confirmă f(N2)



Controlul cheilor



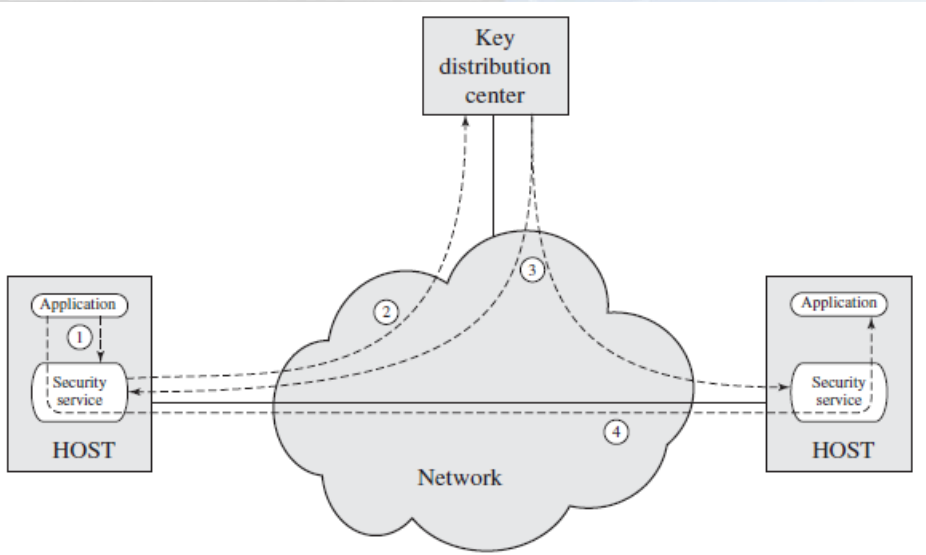
- Centrele de distribuie a cheilor pot fi multiple
  - Util în cazul rețelelor mari
  - Poate exista o structura ierarhică de centre de distribuire
    - **Centre locale** - la nivelul unui LAN, la nivelul unei clădiri
    - Pentru schimbul de mesaje între entitățile din același domeniu există un centru distribuire local
    - Pentru schimbul de mesaje între entități din domenii diferite, centrele de distribuire locale pot comunica printr-un centru de **distribuire global**
    - Pot exista mai multe niveluri ierarhice de distribuire
    - Permite minimizarea efortului de distribuire a cheilor master

Timpul de viață al cheilor de sesiune

- Cheile sesiune sunt folosite un timp limitat
  - De regulă la nivelul unei singure sesiuni
  - Schimbarea frecventă permite un nivel de securitate sporit
  - Distribuirea cheilor poate limita (întârzia) capacitatea rețelei
- Protocoale orientate pe conexiune
  - Pot folosi aceeași cheie pe timpul unei conexiuni
  - O nouă cheie pentru o altă conexiune
  - Pentru conexiuni lungi, cheia poate fi schimbată periodic (la resetarea secvenței PDU Protocol Data Unit)
- Protocoalele orientate pe tranzacție
  - O nouă cheie la fiecare tranzacție (sau un anumit număr)
  - Schimbarea cheilor la perioade fixe de timp

Distribuire automată a cheilor

- Pentru protocoale orientate pe conexiune (TCP)
- Modul de securitate sesiune
  - Criptare punct la punct
  - Obținere chei sesiune



- 1. Un host inițializează o conexiune
- 2. Pachetele sunt stocate în memorie - fiind solicitată și cheia sesiunii
- 3. Centrul distribuie cheia sesiune către ambele hosturi
- 4. Pachetele sunt transmise

Control descentralizat

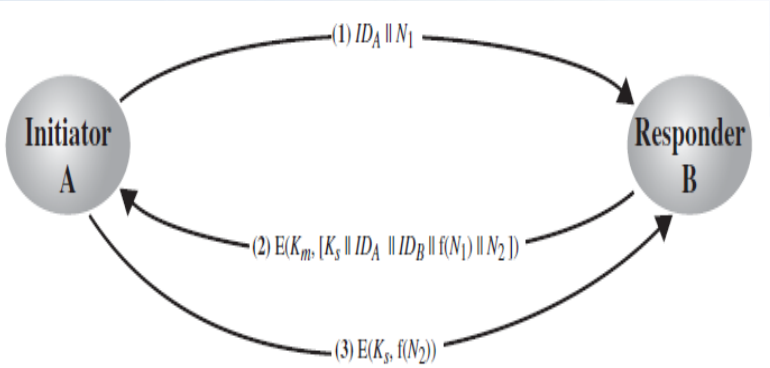
- Distribuirea descentralizată a cheilor
  - Presupune ca fiecare sistem să poată transmite securizat chei de sesiune cu toți potențialii parteneri
  - Evită înlocuirea malițioasă centrelor de distribuire
  - Nu e o soluție practică pentru rețele mari

1. A inițiază un mesaj N1 cu o cerere de cheie de sesiune către B
2. B răspunde cu mesaj criptat cu cheia master,

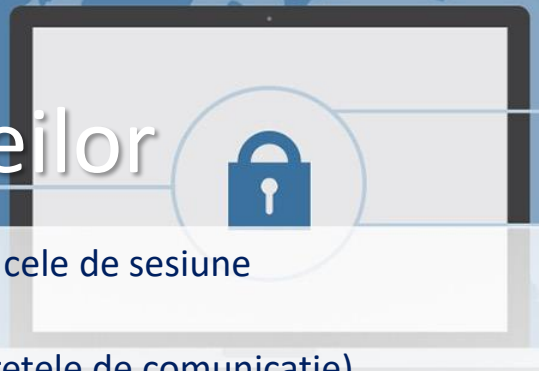
împreună cu cheia de sesiune, un identificator propriu,

o confirmare mesaj - f(N1) și un mesaj nou - N2

3. A confirmă cu f(N2), folosind cheia de sesiune Ks



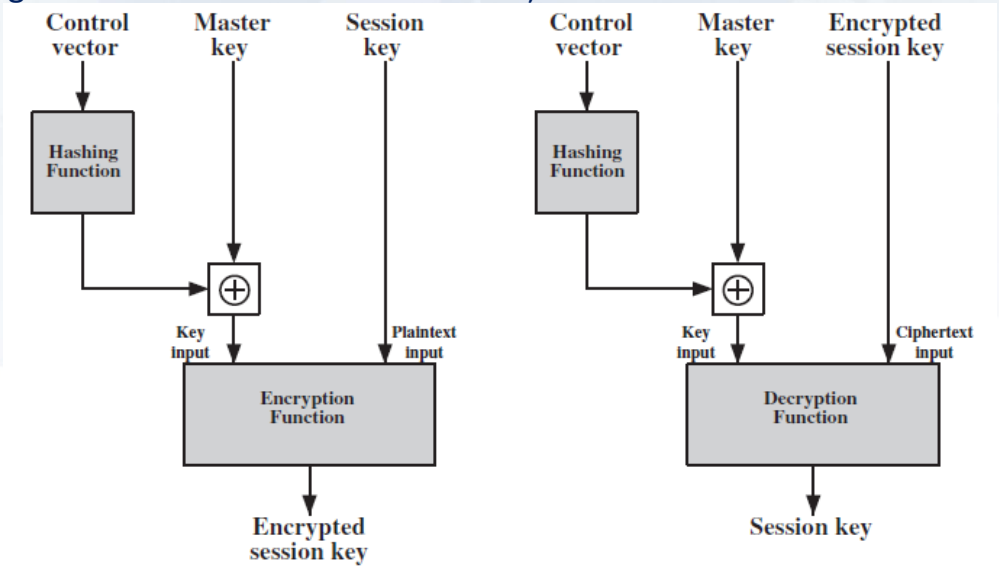
Controlul utilizării cheilor



- Pentru separarea cheilor master de cele de sesiune
- Diferite tipuri de chei de sesiune
  - Chei pentru criptarea datelor (rețele de comunicație)
  - Chei pe bază de PIN (*Personal Identification Numbers*)
    (transferuri electronice de fonduri, aplicații de comerț)
  - Chei pentru criptarea fișierelor
- Asocierea unui etichete (un tag) cu fiecare cheie
  - De ex. folosirea celor 8 biți de paritate dintr-o cheie DES
    - 1 bit poate indica dacă este cheie master sau sesiune
    - 1 bit poate indica dacă este folosită cheia la criptare
    - 1 bit poate indica dacă este folosită cheia la decriptare
    - Ceilalți biți pot avea alte interpretări
  - Eticheta este securizată, fiind transmisă împreună cu cheia
  - Limitare la doar 8 biți
  - Eticheta nefiind transmisă în clar, poate fi utilizată după decriptare

Controlul utilizării cheilor

- Control vectorial
  - Fiecărei chei sesiune i se asociază, la generare, un vector de control
  - Vectorul de control este generat de către centrul de distribuție



Criptare

$$\text{Hash value} = H = \text{h(CV)}$$
$$\text{Key input} = K_m \oplus H$$
$$\text{Ciphertext} = \text{E}([K_m \oplus H], K_s)$$

Decriptare

$$\text{D}([K_m \oplus H], \text{E}([K_m \oplus H], K_s))$$

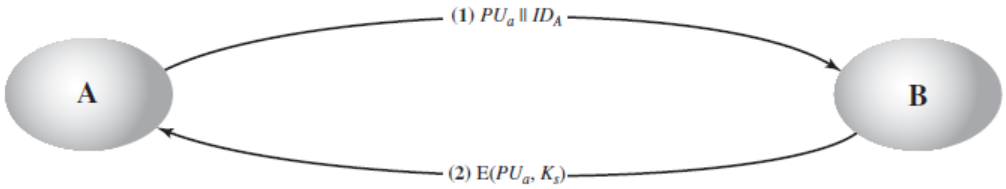
Sisteme criptografice Asimetrice

- Distribuirea simplă

1. **A** generează o pereche publică/privată de chei $\{PU_a, PR_a\}$

și trimite un mesaj către **B**, cu cheia publică și un identificator propriu ID_A

2. **B** generează o cheie secretă, Ks și o transmite criptat către A, folosind cheie publică a lui **A**
3. **A** decriptează mesajul, folosind cheia privată,pentru a afla cheia de sesiune, Ks
4. **A** poate renunță la perechea de chei publică/privată, **B** renunță la cheia publică a lui **A**, ambele sisteme putând utiliza cheia secretă Ks



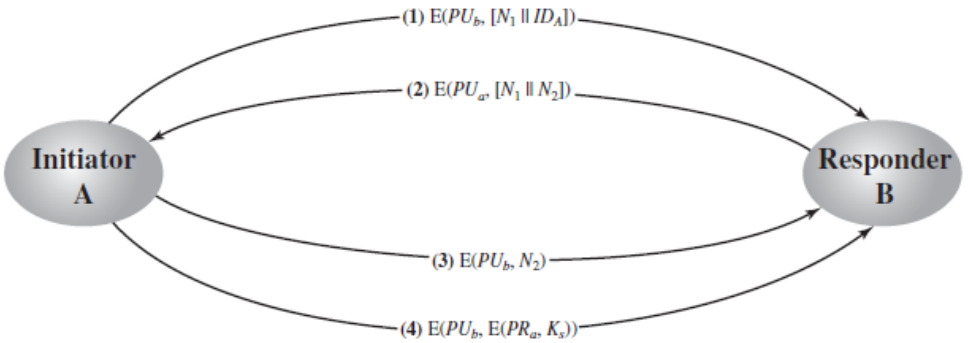
Distribuire cu confidențialitate și autentificare

- Asigură protecție împotriva atacurilor

Se presupune că A și B au schimbat deja cheile publice

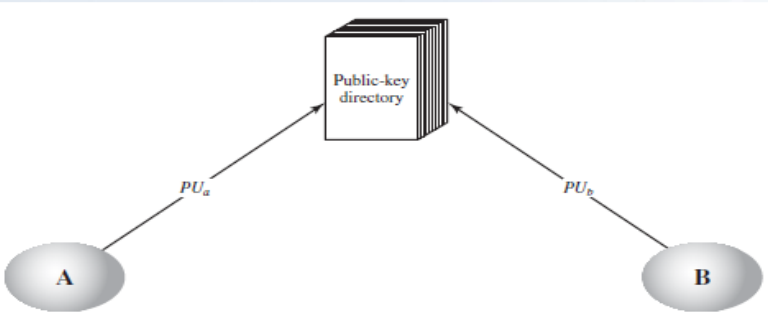
$$M = \mathrm{E}(PU_b, \mathrm{E}(PR_a, K_s))$$

1. A folosește cheia publică a lui B pentru a transmite mesaj N2 si IDa
2. B trimite mesaj criptat către A pentru confirmare N1 (autentificare) și mesaj N2
3. A răspunde la N2 cu cheia publică a lui B (astfel B va ști că discută cu A)
4. A alege cheia secretă și trimite mesaj către B (doar B îl poate citi)
5. B decriptează mesajul și află cheia secretă



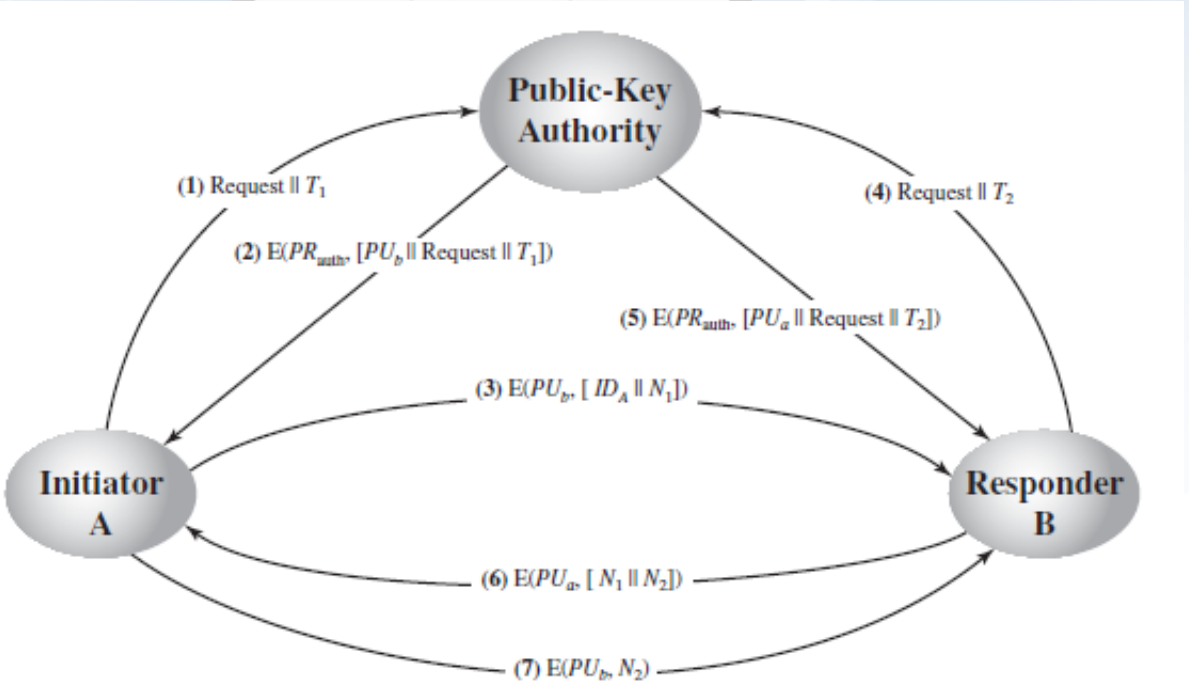
Distribuirea cheilor publice

- Scheme de distribuire
  - Anunțare publică
    - Ex. PGP (Pretty Good Privacy) (RSA) atașare cheie la mesaje
  - Director disponibil public
    - Menținerea unui director cu cheile tuturor participanților accesibil în mod public de fiecare
  - Autoritate de key publice
  - Certificate cu chei publice



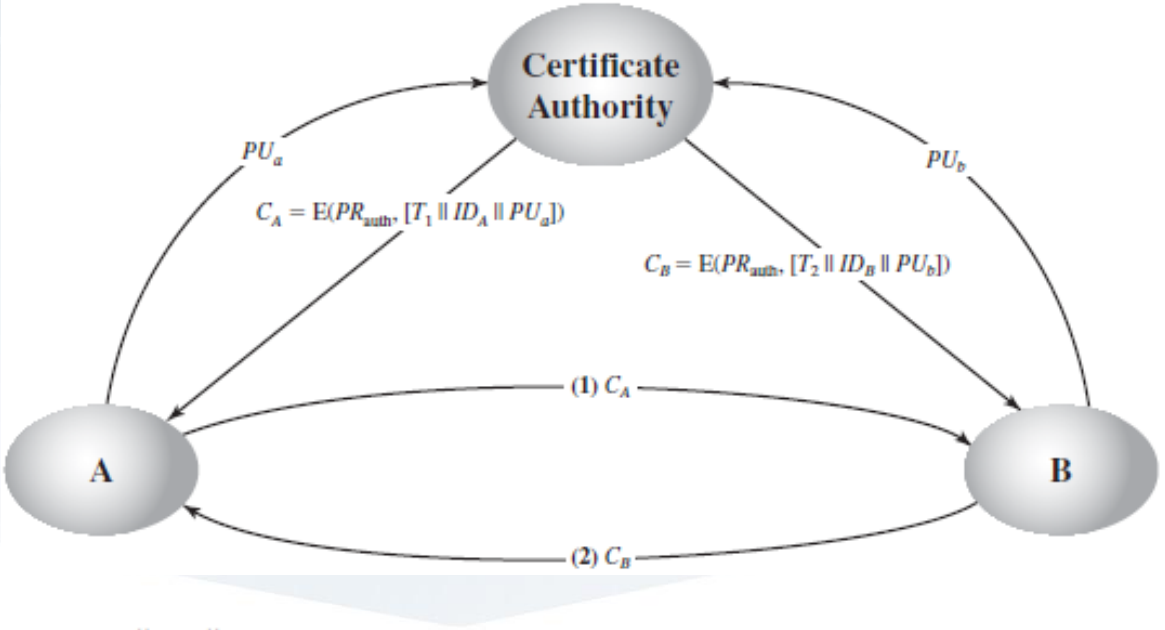
Autoritate de chei publice

A trimite mesaj *timestamp* (semnătura de timp), cerând cheia publică a lui B



Certificate cu chei publice

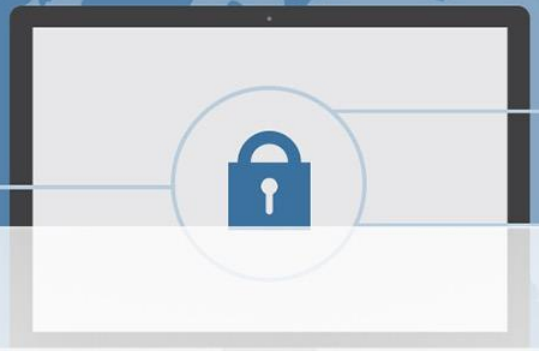
- Certificate ce pot fi utilizate pentru schimbare de chei fără contactarea autorității de chei publice



$$C_A = \mathrm{E}(PR_{\mathrm{auth}}, [T \| ID_A \| PU_a])$$

$$\mathrm{D}(PU_{\mathrm{auth}}, C_A) = \mathrm{D}(PU_{\mathrm{auth}}, \mathrm{E}(PR_{\mathrm{auth}}, [T \| ID_A \| PU_a])) = (T \| ID_A \| PU_a)$$

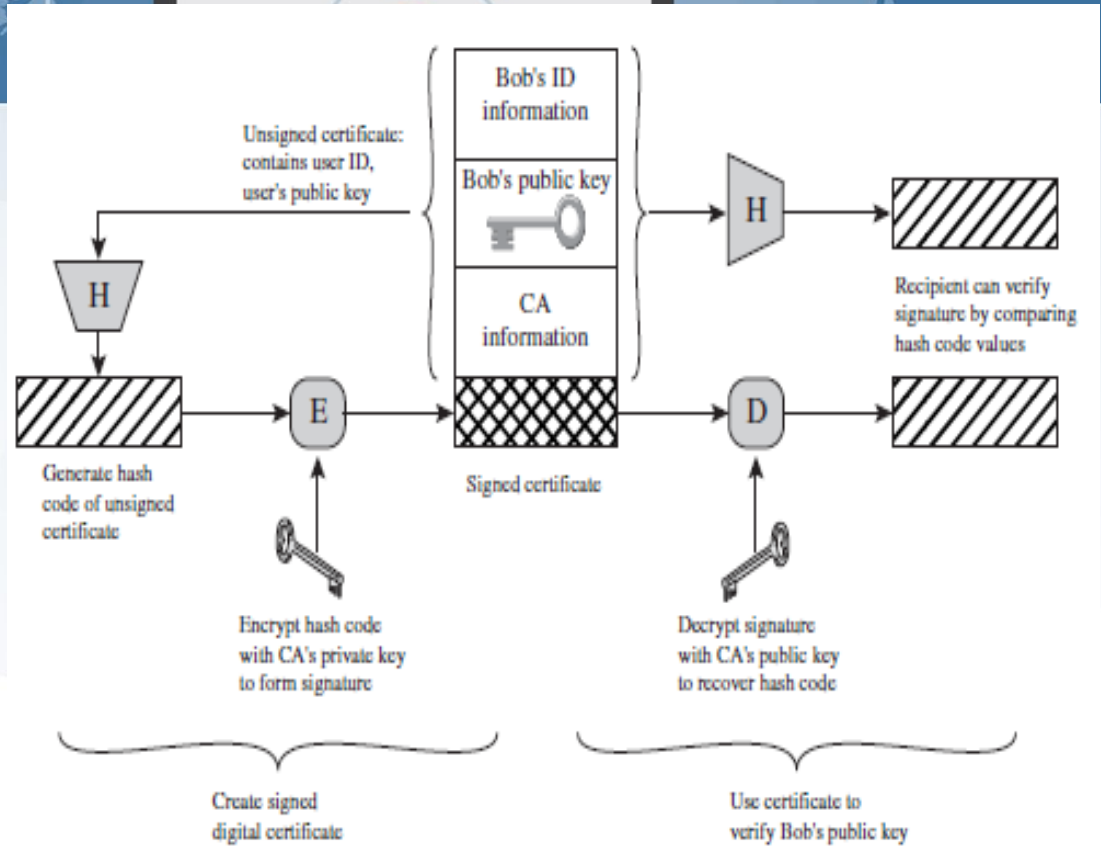
Certificate X.509



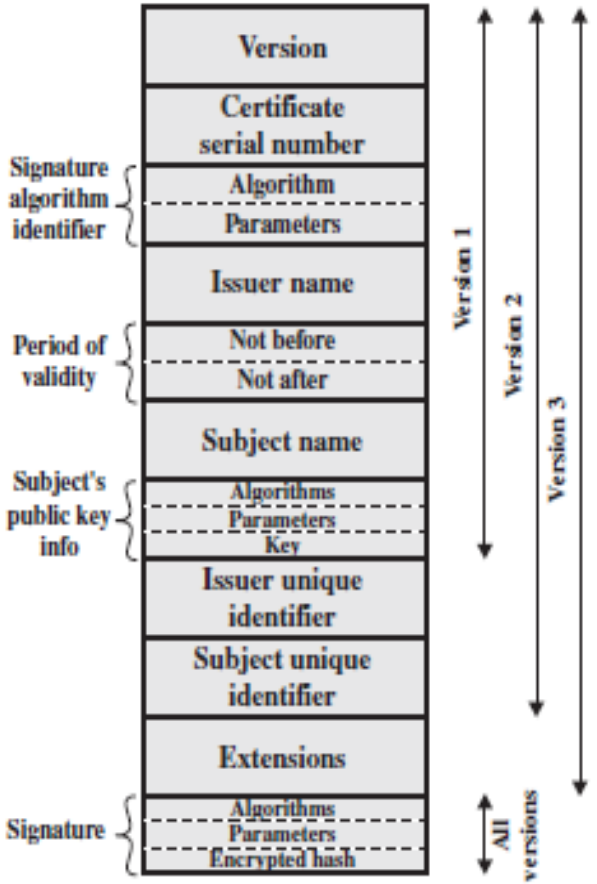
- Recomandările ITU-T X.509
  - ITU-T - ITU (*International Telecommunication Union*) *Telecommunication Standardization Sector*
  - Parte a recomandărilor ITU-T X.500 (*directory services*)
  - Standardul PKI *public key infrastructure*
  - Formatele standard pentru certificatele cu cheie publică
  - Listele de certificate revocate
  - Entitățile de autorizare a certificatelor (CA)
  - Algoritmul de certificare

Utilizare certificate cu cheie publică

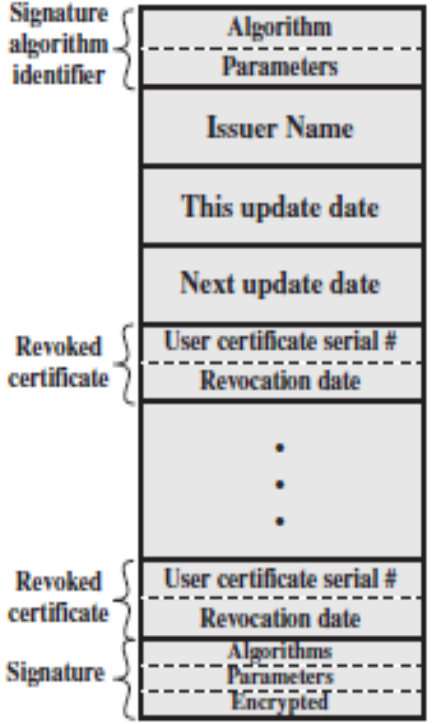
- Creare certificate digitale
- Verificare chei publice



Formate X.509

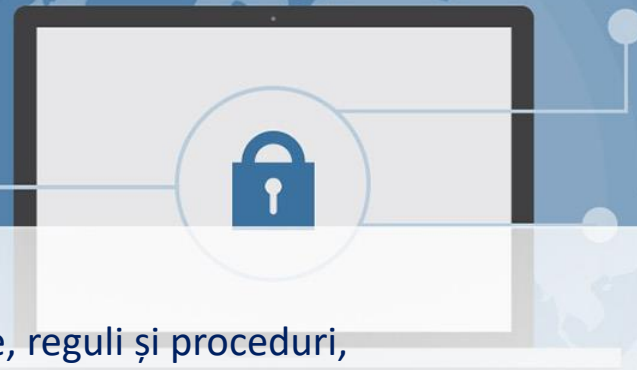


(a) X.509 certificate



(b) Certificate revocation list

Infrastructura cu chei publice - PKI



- PKI - *Public-Key Infrastructure* (RFC 2822)
  - Set de echipamente, software, persoane, reguli și proceduri,
  - pentru creare, administrare, stocare, distribuire și revocare,
  - certificate digitale bazate pe sisteme criptografice asimetrice.
  - Urmărește achiziția sigură, rapidă și eficientă a cheilor publice
  - Arhitectura de distribuire a certificatelor în Internet
  - Funcții de administrare specifice
    - Înregistrare (direct la CA sau prin RA)
    - Inițializare (generare chei, informații necesare pentru certificat)
    - Certificare (procesul prin care CA generează certificatul pe baza cheii publice client și îl pune la dispoziție)
    - Actualizare și recuperarea cheilor
    - Revocarea de certificate
    - Certificare între CA (schimbul de informații dintre CA privind certificatele)

Model arhitectural PKI

Elemente

- Utilizatori
- CA - *Certification Authority* (furnizează certificate)
- RA - *Registration Authority* (asigură operații de înregistrare a certificatelor)
- Entități CRL (publică liste de certificate revocate)
- Colecție de certificate (*repository)* – metodă de stocare a certificatelor sau listelor de certificate revocate astfel încât să poată fi accesate de utilizatori