Acces control 01

Useri, subiect, obiect, operații, permisiuni -

Acces control = modul în care userii pot accesa resursele într-un sistem
→ cel mai filosit, fundamental în securitate atî
→ forme: { hardware, OS, Aplicații -

Scop → Păstrarea integrității și confidențialității inf.
+ accesibilității inf.

Autorizarea și autentificarea sunt fundamentale în A.Control.
ce îți este permis — cine ești.

User = persoană care folosește un sistem
Subiect = proces care acționează pe seama unui user
Obiect = resursă accesibilă pe un sistem.
Operație = proces activ invocat de un subiect.
Permisiune (drept, privilegiu) = autorizarea de a realiza diverse acțiuni pe sist

- Obiectele = fișiere, directoare, programe, relații în baze de date, etc
- Subiecții - entități active / cer acces la obiecte/
- Obiectele = entități pasive; conțin sau primesc informație; sunt protejate de subiecți
- Subiecții pot juca rol de obiect

Principiul privilegiului minim → Orice program și fiecare user ar trebui să opereze utilizând (apelând) la drepturile minime necesare îndeplinirii act.
- Beneficii: stabilitate, ~~securitate~~ securitate îmbunătățite, .
În practică, principiul e greu de definit și respectat.

privat = al meu/tău → block, stream
public = al nostru

SISTEM de Securitate

1) Politică de securitate = cereri de nivel înalt care specifică modul în care se gestionează accesul și cine și în ce împrejurări poate accesa (ce) informație dorește.
2) Model de securitate = o reprezentare formală a politicilor de control a accesului și modul cum acestea lucrează. -> Probleme proprietății
3) Mecanism de securitate = nivelul de implementare a politicilor de securitate - modul cum sunt impl (hard + soft).

Politici de securitate

DiscreționareDAC = realizate pe baza identității entităților ce solicită acces. Discreționară: pt că un utilizator poate acorda drepturi altui utilizator după cum dorește.
- are reguli de acces explicite → care stabilesc cine poate sau nu să execute (ce) acțiuni asupra (căror) resurse.

Mandatare: MAC - pe baza unor reguli dictate de o autoritate centrală

RBAC - Role Based Access Control - utilizatorii nu pot trece permisiunile la alți useri la propria lor voință

→ funcțiile de control al accesului pt un user sunt realizabile în interiorul unei organizații.

Modele de securitate: bazate pe Matrici de acces
- grafuri
- mulțimi parțial ordonate
- logici ~~de ordine~~

Mecanisme. - de punere în practică a politicilor de securitate
→ bazate pe monitoare referențiale = porțiune hard + soft dintr-un sistem de operare responsabilă de asigurarea politicilor de securitate pe un sistem (security kernel)

Monitor referențial: 1) Complet intuitiv → tot timpul invocat și impos de ignorat;
2) izolare - nu se poate modifica de entitățile din sistem
3) verificabil → să poată fi verificat. (se pot propaga erori în tot sistemul).

- să aibă dimens. reduse;
- flexibilitate, ~~nu~~ ne ușor de folosit, intuitiv; scalabil - (la ur de resu (ai useri).

→ implementări: la niv de sistem, de resurse sau de aplicație

Control Acces 02 — DAC: Discretionary acces Control: Take/grant; Matrice de acces; Model schematic.

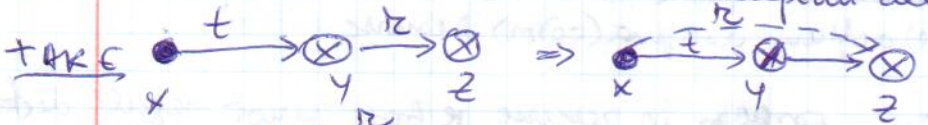
DISCR.
- Controlul accesului pe baza identității entităților (ofeu)
- discreționare = pt că useri pot avea posibilitatea de a trece drepturi spre alți useri
- includ conceptul de proprietate (ownership).

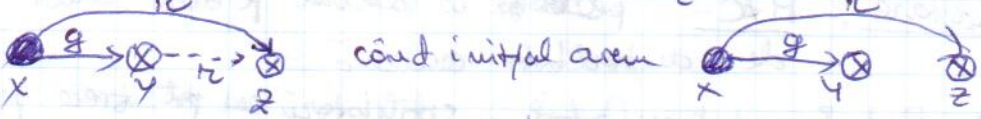
1) TAKE-GRANT: + eficient, verificarea descărcării de drepturi se face în timp liniar în raport cu dimens. stării inițiale a sist.

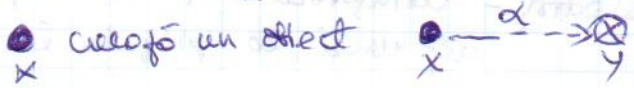
Scurgere de drepturi: un utilizator își cere drepturi de la alt utilizator pe care n-ar trebui să le dețină.
- Subiecții nu pot fi obiecte.

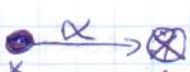
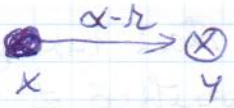
Stările = grafuri direct orientate. în care noduri sunt subiecți, obiecte, și de către arce sunt etichetate cu reguli (drepturi).

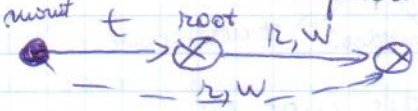
→ 2 drepturi speciale: 1) Take = un subiect poate lua drepturi de la alt to entitate din sistem.
2) Grant = dreptul unui subiect de a acorda drepturi altui subiect. (sau utilizator)

TAKE: 

GRANT: 

Create: 

Remove :  x remunță la o parte din drepturi 

mount : 

Graf take grant = graf orientat, în care nodurile sunt subiecți sau obiecte - arcele sunt etichetate cu mulțimi de drepturi -

DOAR Subiecții (x) pot inițializa acțiuni (sunt activi).

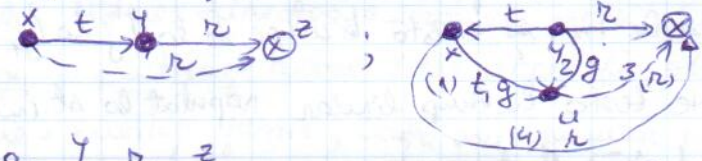
Scurgerea de drepturi = modelată prin predicatul can share.

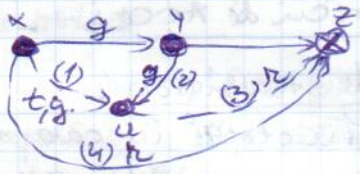
Can share (r, x, p, G) -
$x \to y$ = tg conectate dacă $\exists$ arc orientat t sau g.

Dacă într-un graf $x \to y$ nu sunt conectate, x și y nu vor putea fi conectate cu regulile t, g, nu adaugă arce între noduri neconectate. oricum ar rescrie graful

Șablon: $\vec{t}$, $\vec{g}$, $\overleftarrow{t}$, $\overleftarrow{g}$;

- Dacă 2 subiecți x, y sunt tg conectați (direct) - atunci toate drepturile unuia pot fi obținute de către celălalt.

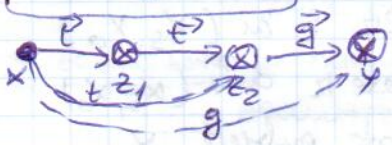
Cazuri:  ; x conectat cu subiect ce nu îi dă drepturile acestuia lui z.



INSULĂ = subgraf format doar din subiecți tg conectați între ei
Dacă un subiect din insulă are drepturi față de alți subiecți din insulă obține aceleași drepturi -

INITIAL SPAN : x se întinde inițial spre y (initially spans to y) dacă) x e subiect și 2) $\exists$ un tg drum de la x la y cu șablonul.

$\vec{t}^* \vec{g} + \lambda$

exemplu: 

un initial span poate duce toate drepturile subiectului x până la destinație.

TERMINAL SPAN : x se întinde terminal spre y (T.S to y) dacă
1) x este subiect și
2) $\exists$ între x și y un drum cu șablonul $\vec{t}^*$
x ia toate drepturile de la y.
x tre să fie subiect ca să poată inițializa acțiuni (duce toate drepturile de la destinație la sursă)

BRIDGE
Capetele x și y sunt subiecți ; iar drumul de la x la y poate avea următoarele șabloane : 1) TS de la x la y sau TS de la y la x, sau 2) compunere între IS de la y la x și TS de la x la y.
$\vec{t}^* \vec{g} \vec{t}^*$ sau $\vec{t}^* \vec{g} \overleftarrow{t}^*$ sau $(\vec{t}^*)$ sau $(\overleftarrow{t}^*)$

Definiție: Take grant algoritm = dacă predicatul can share nu se verifică pt (R, x, P, G)
drept nodul → start, G).

- x, y direct connected dacă ∃ arc între ele.
- direct tg connected = dacă ∃ arc între ele etichetat t sau g.
- cale (tg cale) = secvență de noduri $x_0, x_1, \dots, x_n$ a.î. x_i, x_{i+1} connected (tg drum direct).
- x, y - connected dacă ∃ drum între ele.

(T5) **Decizia securității**

G = state take grant, R = drept, x, p noduri în G;
can share (R, x, P, G) = true dacă și numai dacă $R \in (p, x)$ sau ∃ un nod s, doi subiecți s' și p' și impulele $i_1, \dots, i_n$ a.î.

1) $R \in G(s, x)$
2) $p' = p$ sau p' initially spans to p;
3) $s' = s$ sau s' terminally spans to s.
4) p' e în i_1, s' e în i_n, și există bridge de la i_j la i_{j+1}, pt toți $i = \overline{1, n-1}$

(T5) Verifică can share doar în st inițială, nu tre să mai repet pt alte stări

- Can share se poate testa în timp liniar raportat la st inițială.

2) **Modelul Matricii de Acces** - cel mai general model Dtc.

- E un sistem state-transitions.
- ~~Pasul~~ starile sunt matrici în care fiecare linie coresp unui subiect, fiecare coloană e un obiect; O celulă (s, O) specifică dreptul pe care îl are s asupra lui O.
- tranzițiile sunt date de comenzi; subiecții sunt de asemenea obiecte. Comenzile = teste + op primitive.
- Un subiect poate avea drept de a executa alt proces -

Def: Stare: triplet (S, O, A) S-subiecți, O obiecte, $A = S \times O$ matrice.

Operații primitive: 1) enter R în (X_s, X_o)
2) delete R din (X_s, X_o).
3) create subject X_s
4) Create object X_o
5) Destroy subject X_s
6) Destroy object X_o.

SI ST
- Comenzi: teste + op primitive expuse mai sus (if - cond then com)
- Un sistem de protecție = un set finit de comenzi -
- Substituții: oricând subiect - subiect; obiect - obiect.
- Rel de tranziție → dacă $A \to B$ și $B \to C$ at $A \to C$.

de protecție **Securitate**: securizat dacă inf dacă ∃ o substituție a.î.
(dacă C este un sist de protecție, Q o stare în C, R un drept., securizat =
⇒ ∃ de la Q spre C dacă ∃ o comandă în C care permite R altor Q

(2) Sistem nesigur: Q e nesigur pt r dacă ∃ o stare accesibilă Q' a î C scurge r dinspre Q'.
Scurgerile nu sunt "rele" cu totul, doar cele neautorizate de Problema sec. în sisteme cu o singură condiție e decidabilă.

Implementare:
- necesită un spațiu uriaș de procesare. useri x aplicații/procese
- utilizare de grupuri (roluri) → mai mulți useri simultan au ac. drepturi ⇒ RBAC
- a implementa matricea prin intermediul listelor ACL (access control lists) → pe linii sau coloane.
- Liste ACL = coloană din matricea de acces = un obiect + useri care au acc
  - bun pt medii unde useri își administrează singuri met de securitate.
  - dacă s mulți subiecți, listele e foarte uzi
  - complicat de făcut când un user dorește să acorde drepturi altor useri
  - Dacă un user părăsește sist, îi șterse și dr. lui în sistem

→ Se pot crea breșe de sec. tip cal troian.

(Modelele discreționare nu pot evita atacurile de tip cal troian)

folosire ACL în Unix și Windows. → un fișier cu x e plasat într-un folder în care y are acces deplin

Altă met: pe linii (subiecți sunt linii, coloanele repr. obiectele la care userul are acces) = listă de capacități = tichet → listă tuturor drepturilor pe care un user le poate avea într-un sistem → Ob tre să fie adusă de obiect → pot apare coliziuni; se folosește fct hash care să permită acces la ob rapid și fără coliziuni
⇔ tehnici de acordare de nume -
- tich. tre să fie complexe, să nu poată fi falsificate.

Sist take grant NU au condiții de garantare a accesului ca la modelul matriceal; La modelul matricei sarcinea de drepturi e nedecidabilă -; matricea = f complexă

UNIX; R, W, X
3 tipuri de useri: {- owner, - grup, - toți

3) - Model intermediar = Modelul schemotic -
- Drepturile se acordă pe tipuri de subiecți -; tipuri de obiecte TS, TO;
- drepturile { merge Ri; de control RC; - flag de copiere -; r = nu poate fi copiat; C = copiabil.
rc - r; C e (r sau C) = r sau rc.

Tichete = perechi (x, r:C) (x/r:C) Cine are acest tichet poate accesa entitatea x cu dreptul r.

Stare a unui sistem: mț de subiecți și obiecte ∪ domeniul subiecților din sist

Operații care schimbă starile curente: { copy - mută o copie a unui tichet din domeniul lui subo spre dom altui subiect, păstrând tich. original intact
create. → introduce noi subo sau ob în sist

DAC: acces control pe bază identității celui care cere accesul.
↗ bazat pe reguli de acces explicite.
→ ignoră distincția dintre sub ↗ obiect.
- vulnerabilă cal troian.

MAC • Acces control bazat pe reguli dictate de o autoritate centrală -
- Nu există ownership în MAC
- se face distincție S/O
- Bell-LaPadula, Biba, Bidul Chinezesc.

Modele latticeale:
- entitățile sunt grupate în clase de securitate.
- Obiectele = containere de informație → fișiere, etc.
→ Controlul accesului = se asignează obiectelor o clasă de sec

Model Latticeal: Clasă de securitate, relație binară (permisiune acces)
⊕ operator
A → B = inf poate curge de la A la B
A ⊕ B = inf din cele 2 clase se combină, rez. = apartine claselor A⊕B
Axiomele Denning: 1) Mț claselor de securitate e finită
2) rel de ordine parțială = rel de transf al informației - (Refl, simv, tranzitivă)
3) Mț claselor de sec SC are un cel mai mic elem (clasa inf publice).
4) Când se combină 2 clase de sec, rezultatul e cea mai mică clasă de sec →.
Orice model ce satisface cele 4 cond e o latice.

! Clasa A domină clasa B dacă inf poate curge de la B la A

MODELUL ASIGURĂ DOAR CONFIDENȚIALITATE -
- entități = plasate pe niv de securitate.
- Niv subiecților = gradul de încredere pe care-l acord subiectului de a nu divulga inf
Niv obiectelor = gradul de confidențialitate al inf din obiectul resp

1) MAC Politici ~~de~~ confidențialitate ~~bazate pe Modelul~~ Mandatore -
• Scopul = controlul curgerii de inf
- prevenirea scurgerilor de inf către subiecți neautorizați
Bell LaPadula: îmbogățire DAC cu MAC - pt a impune politicile de curgere a inf.
1) Matrice de control a accesului
2) Operațiile să fie autorizate de politicile de acces mandatorii MAC.

Se verifică la acces 2 reguli:

$\lambda(s) \geq \lambda(o)$ — **No write DOWN** → un subiect s poate scrie în o, dacă o are un nivel superior lui s

$\lambda(s) \leq \lambda(o)$ — **No read UP** → subiectul s poate citi un obiect o doar dacă are un nivel de securitate superior lui o (securitate)

Nu poți scrie decât în obiecte cu niv de încredere mai mare decât cel pe care îl ai tu. (inf de calitate).

Nu poți citi decât din obiecte cu nivel de securitate mai mic - decât al tău. (confidențialitate).

→ În matricea de acces, linia x col. să conțină drepturile de citire a fișierului resp.

No write down * Proprietatea * = e autorizată cu scopul scrierii să se facă doar la același nivel de securitate (pt a nu altera informații de la nivele de securitate superioare, deci de importanță mai mare).

Dacă matricea de control a accesului nu autorizează specific, nu e voie de a verifica regulile Bell la Padula (→ No write down / No read up)

2) Politici MAC bazate pe integritate

- Scop: controlul curgerii de informație (flux)
- Prevenirea modificărilor reale/indirecte de subiecți asupra informațiilor asupra cărora nu au voie să scrie.
- Subiecții și obiectele pot dispune în clase de integritate

- Modelul Biba - **No read DOWN** - (s poate citi o doar dacă nivelul lui $s \leq$ nivelul lui o)
  No write UP (s poate scrie o doar dacă nivelul lui $s \geq$ nivelul lui o)

! invers cu Bell-La Padula

În BLP informația curge în jos
În Biba - inf curge în sus

1) • Combinație BLP + Biba → 2 latici, curgere de inf în direcții diferite; o latice pt confid. și una pt integritate, cea mai înaltă dpdv de sec = în vârf (high)

- s poate citi o doar dacă $\lambda(s) \geq \lambda(o)$ și $\omega(s) \leq \omega(o)$
  s poate scrie o doar dacă $\lambda(s) \leq \lambda(o)$ și $\omega(s) \geq \omega(o)$

2) comb. cu fluxul în din diferite; doar cu securitate maximă în ambele

te λ_{max} ω_{max}

s poate citi o dacă $\lambda(s) \geq \lambda(o)$ și $\omega(s) \geq \omega(o)$.
s poate scrie o dacă $\lambda(s) \leq \lambda(o)$ și $\omega(s) \leq \omega(o)$

Cea mai mare confid are integritate minimă și invers.

Biba => aplicat la Windows Vista

SELinux - Lotel - (Security Enhanced Linux)

Modelul zidului chinezesc — folosit în companii diverse

- Scop: evitarea conflictelor de interese.

exemplu: o firmă oferă consultanță la 2 firme aflate în competiție.

- Companiile sunt grupate în clase de interese.
- Subiecții au acces la inf a anumite restricții: dacă un subiect are acces la inf din clasa C_1 atunci trebuie să i se dea acces la altă dosar dacă e în conflict de interese cu C_2 în care e compania concurentă → se creează un „zid" chinezesc în jurul obiectului - peste care subiectul nu poate vedea mai departe.
- Dacă o un obiect din mulțimea O aflată în clasa de conflict de interese Col e ales, se creează un zid în jurul lui O și nici o mulțime din Col nu poate fi aleasă de același subiect.

Simple security Rule
- Subiectul S poate avea acces de read la un obiect O, doar dacă obiectul O: - e în același mulțime cu cele a companiei ce deține obiectele deja accesate de S; care e în interiorul zidului.
  - sau aparține unei alte clase de conflict de interese.
- Proprietatea (*) - un subiect S poate scrie într-un obiect O, doar dacă 1) S poate citi O prin regula S. Securitate
  2) nici un obiect care este într-o mulțime de interes a unei companii diferite nu poate fi citit -
  (un subiect poate scrie în O dacă poate citi inf din O, dar nici un alt subiect nu poate cit din O în care se scrie).

Visto → MIC - mandatory integrity control -
- 6 niv de securitate -
- la logon, Visto angajează SID la tokenul de acces al userului
- niv de integrit al obiectelor → fișier, proces, → SACL (system ACL - stocate).

MAC → oferă protecție împotriva scurgerii indirecte de informații
→ e vulnerabil la canale acoperite -

curs 4 **RBAC**: role based Acc control
Drepturile de acces sunt grupate pe roluri - (o funcționalitate)
- fiecare rol are anumite drepturi / permisiuni -
- mecanism folosit pt administratori de sisteme pt a specifica privilegiile cerute de diverse funcții într-o firmă.
  - BASIC
  - ierarhic
  - Restrictiv
  - Constrângeri
- Un user care se mută pe alt rol este pus și simplu asignat rolului nou și i se șterge rolul vechi

(3) Componente RBAC

U = multime de utilizatori
R = multime de roluri
P = un set de permisiuni (mult. $(Op \times Obj)$ $(O \times Obj')$
Un drept = o functionalitate = o pereche de permisiuni de forma (Operatie × Obiect).

- Modelul utilizează o matrice de asignare (utilizator × rol) (produs cartezian) (relație de asignare) $U \times R$.
  - un utilizator poate juca mai multe roluri -
  - utilizatorii se exclud mutual.

$P \times R$ - matrice de asignare Permisiune × rol. $P \times R$.
- o funcție de mapare subiect → utilizator. $S \times U$
- o funcție de mapare subiect → rol. $S \times R$

BASIC RBAC: 1) autorizare de rol: un subiect nu poate avea niciodată un rol activ care nu este autorizat pentru userul său.
2) Autorizarea accesului la obiecte -
- un subiect poate realiza o operație asupra unui obiect doar dacă ∃ un rol inclus în setul de roluri active ale subiectului și există o permisiune asignată lui r a î. permisiunea să autorizeze realizarea operației asupra obiectului O.

RBAC ierarhic: Basic + ierarhie → există o rel. de ordine parțială între roluri → moștenire de roluri - (cel mai influent e în cf lăticii) ; Rel de dominare -

Modelul RBAC bazat pe constrângeri
- Model Basic + restricții -
  - bazat pe excluderea rolurilor - (nu se supun pasi)
  - cardinalitate ; nr maxim de useri pt un anumit rol.
  - interdependență - un user poate juca un rol doar dacă poate juca și alt rol - (îi oferă competențe necesare).

RBAC consolidat = ierarhii pe roluri + restricții -
→ implementare în oracle → DB management, securitate întreprinse.

Concluzii → userii se schimbă mai des decât atribuțiile din funcțiile pe care le au
- se simplifică administrarea securității prin utilizarea de roluri, ierarhii și constrângeri -
- RBAC poate fi configurat să susțină DAC și MAC.
- potrivit pt o varietate de aplicații și medii software

CONTROLUL ACCESULUI - se referă la autorizarea de a folo și admite resurse și NU la autentificare (acces prin autentificare).
- Nu e suficient pt a proteja fluxul de inf între 2 entități.
- de criptată informația cu ajutorul primitivelor criptografice

Criptografia simetrică -
Nu poate asig. securitate fără alte instr. ajutătoare: controlul accesului ; protocoale de securitate

Securitate: 1) Confidențialitate → caracterul privat al inf
2) integritate = inf trebuie să fie nealterată, ne modif
3) autenticitate - stabilirea identității entităților
4) nerepudiere → (nu se poate nega paternitatea inf)

- Criptografia 1) simetrică - cu chei private
2) asimetrică - cu chei publice.

Parametri de securitate = lungimea unei chei de criptare - = dimens minimă a unei inf care să asigure securit la nivel.
- se notează cu 1^n
- Algoritmi utilizați = probabiliști - (suma tuturor prob. = 1)
- Canalele sigure (nu mai e nevoie de criptografie = ideal)
nesigure → atac pasiv → monitorizare canal, eavesdrop
activ → alterarea informației, injectare

Adversarul A = modelat de un algoritm probabilist cu complexitate timp polinomială
Oracol = entitate ce oferă informații algoritmului - cărui acesta i le cere.

Schemă de criptare simetrică: 3 uplu $\Pi(G, E, D)$
G = alg. probabilist de chei de criptare k, $k \leftarrow G(1^n)$ → generator de chei
E = alg probabilist de criptare $E(k, m)$
D = alg de decriptare
→ deci: 1 generator de chei, un alg de criptare, un alg de decriptare

M. de securitate Modele pasive
Criptosist să fie public, doar cheia de criptare să fie secretă
COA: ciphertext only attack: atacatorul vede criptotextele vrea să deducă mesajul sau cheia
KPA → known plaintext attack → atacatorul a învățat în timp că m_1 a fost criptat → c_1 (criptotext)
m_R - - - - - - - - c_e
și încearcă să găsească cheia știind și plaintextul inițial și ciphertextul rezultat.

Modele active:
CPA - chosen plaintext atack - adversarul poate obține criptotextul asociat unui plaintext ales de el
(Lunchtime attack).

CCA - chosen ciphertext attack.
- adversarul are posibilitatea de a afla mesaje originale asociate unor criptotexte (alese de el = adoptiv). → intuitionând oracolul.

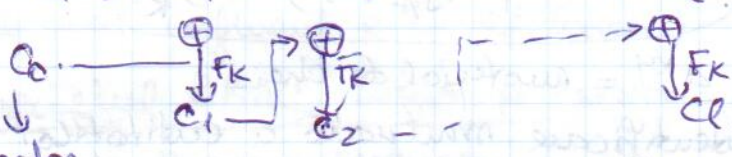
Criptosistemele simetrice (au chei private). { Stream / bloc.

Gr. (RC4) Stream → cheie scurtă de acces lung cu mesajul.
- cheia e generată dintr-una mică și apoi se expandează.
ex: RC4 -
→ vulnerabilitate: al doilea octet din cheia scurtă e 0 cu probab. de 1/128 (↑ mare). [Foarte rapide în practică]
• pt securitate, cheia de criptare trebuie schimbată de la mesaj la mesaj
• Criptosist bloc: mes. e împărțit în blocuri de criptare
se face bloc cu bloc → $k \leftarrow G(1^n)$; $\mathcal{E}(k,m)$; $c = G(k) \oplus m$
$\mathcal{D}(c,k)$, $m = G(k) \oplus c$
• se generează o cheie → avem o familie de fct pseudo-random;
; se criptează mesajul cu cheia k; cu cât generatorul e mai random, schema e CPA sigură.
- de decriptare, se face decr tot cu cheia k → Des - 64 biți, AES - 128 b.
3Des - 128 pt 2 key, 192 pt 3 key

Vulnerabilități: $c_1 = m_1 + G(k)$, $c_2 = m_2 + G(k)$, $m_1 \Rightarrow c_1 \oplus c_2 = (m_1 \oplus m_2)$ vulnerabil. NU!

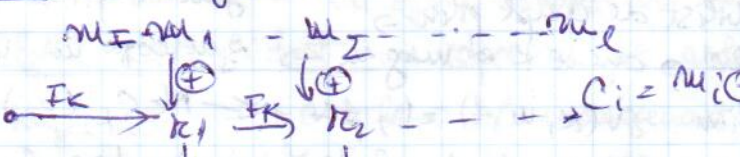
Moduri de criptare: ECB:
electronic code block.
→ $c_1 = F_k(m_1)$, ---$c_l = F_k(m_l)$
$r \leftarrow \{0,1\}^n$ random, $c_1 = F_k(m_1 \oplus r)$ un bloc care apare de mai multe ori e criptat la fel = vulnerab. majoră! → în practică nu se utilizează!

CBC - Cypher block chaining)



vector de inițializare. ; Dacă c_0 = random și metoda de inițializ e random, schema e CPA sigură

OFB = output feedback -



$c_i = m_i \oplus F_k^i(r_0)$ } Ca un criptosistem de tip

CPA sigură dacă r_0 = random și met de inițializ = random
F_k = familie de fct pseudorandom. ex Des, AES, 3DES → 64, 128, 128/192.

CTR - counter →

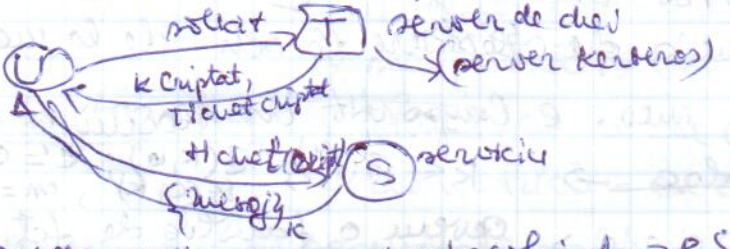
$m = m_1 \cdot m_2 \cdots m_l$

$r_0 \xrightarrow{F_K(r_0+1)} r_1 \longrightarrow r_2 = I_K(r_0+2)$

$c_i = m_i \oplus F_K(r_0+i)$

r_0 = random
F_K = pseudorandom. } metodo CPA sigură.

De ați făcute CPA sigure? → problema distribuției cheii →
→ Criptografia cu chei publice →
Centrul de distribuție de chei → Kerberos Protocol.



Diffie Hellman → protocol: $A \to B \{ x \leftarrow \{2 \cdots (p-2)\}, a^x \}$
$B \to A \; y \leftarrow \{2 \cdots (p-2)\} \to a^y$

Scop: A și B stabilesc materialul de chei (a^{xy})
Vulnerabilitate man in the middle.

STS → p = nr prim are, α = rădăcină primitivă publică modulo p; $sig_x(m)$ e semnătura RSA a lui x pe o valoare hash a lui m →

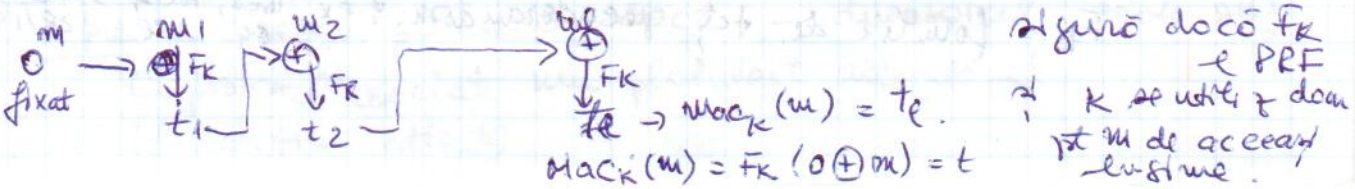
Protocol: $A \to B \; (\alpha^x)$, cu x ales între 2 și p-2
$B \to A \; \alpha^y$, cu $\{sig_B(\alpha^y, \alpha^x)\}_K$
$A \to B \; \{sig_A(\alpha^x, \alpha^y)\}_K$

$K = \alpha^{xy}$

a^{xy} = material de chei.

Proprietăți: autentificare mutuală a entităților.
și autentificare mutuală explicită a cheilor

MAC = message authentification code.
Schemă de autentificare: triplet $\Pi(G, Mac, V)$
gen de chei.
Mac = alg probabilist de compl polin → care pornind de la o cheie k și mesaj m → generează un mesaj de auth.
V = alg pob. polin. $V(k, m, t) = 1$, dacă $t \leftarrow Mac_K(m)$.
Mac tb să fie rezistent la falsificarea tagului (t)
CBC MAC = metodă de reducere a dimens. tagului (t)



sigură dacă F_K e PRF și k se ține secret și m de aceeași lungime.