Notite

Cursul 1 - Principiile de baza ale securitatii

- **Domenii de securitate:**
    - **Computer Security**: protejeaza date si resuse (security policies, access control, malware etc.)
    - **Network Security:** protejeaza datele in timpul transmisiei si comunicarii
    - **Software Security:** projeaza software dintr-un sistem computerizat
    - **Web Security:** protectie fata de un atacator web
- **Confidentiality (1), Integrity (2), Availability (3):** bunurile pot fi *vazute (1), modificate (2) sau utilizate (3)* doar de persoanele autorizate

Cursul 2 - Sisteme istorice de criptografie. Securitate perfecta

- Procesul de determinare a cheii aferente unui sistem de criptare, cunoscand doar textul criptat si eventual alte informatii auxiliare se numeste **criptanaliza**
- Decriptarea si criptanaliza au acelasi scop: gasirea textului clar. Diferenta consta in faptul ca la criptanaliza nu se cunoaste cheia de decriptare

Definitie

Un sistem de criptare simetric definit peste $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, cu:

- $\mathcal{K}$ = *spațiul cheilor*
- $\mathcal{M}$ = *spațiul textelor clare (mesaje)*
- $\mathcal{C}$ = *spațiul textelor criptate*

este un triplet $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, *unde:*

1. Gen: *este algoritmul probabilistic de generare a cheilor care întoarce o cheie k conform unei distribuții*
2. $\mathrm{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
3. $\mathrm{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

a.î. $\forall m \in \mathcal{M}, k \in \mathcal{K} : Dec_k(Enc_k(m)) = m$.

1. **Sisteme de criptare istorice:**
   - Pot fi spate cu un efort computational foarte mic
   - **Cifruri de permutari / transpozitie:** un cifre de permutare presupune rearanjarea literelor in textul clar pentru a obtine textul criptat (exemplu: Rail Fence)

	→			
	M	A	R	T
↓	E	J	I	A
	S	C	P	T

Text clar: mesaj criptat
Cheia: $k = 3$
Text criptat: MARTEJIASCPT

- **Brute Force Attack (cautare exhaustiva):** incercarea, pe rand, a tuturor cheilor posibile pana cand se obtine un text clar cu sens
  - **Principiul cheilor suficiente:** o schema sigura de criptare trebuie sa aiba un spatiu al cheilor suficient de mare a.i. sa nu fie vulnerabila la cautarea exhaustiva
- **Analiza de frecventa:** determinarea corespondentei intre alfabetul clar si alfabetul criptat pe baza frecventei de aparitie a literelor in text, cunoscand distributia literelor in limba textului clar
  - Cifruri de subtitutie monoalfabetice: cifrul lui Cezar, substitutia simpla etc.
  - Cifruri de subtitutie polialfabetice: sistemul Playfair, sistemul Hill, sistemul **Vinegere**

    unde $Enc_{k_j}(m_i) = m_i + k_j (mod 26)$ si $Dec_{k_j}(c_i) = c_i - k_j (mod 26)$

Text clar:	c	u	r	s	c	r	i	p	t	o	g	r	a	f
Cheie:	c	h	e	i	e	c	h	e	i	e	c	h	e	i
Text criptat:	E	B	V	A	G	T	P	T	B	S	I	Y	E	N

- **Factorizarea numerelor mari:** astazi nu se cunoaste nici un algoritm care sa factorizeze un numar de 400 de cifre intr-un timp practic (descompunere in numere prime)

2. **Sistemele de criptare moderne**

Cursul 3

Cursul 3.1 - Securitate computationala

- **Securitatea perfecta:** schema care rezista in fata unui adversar cu putere computationala nelimitata (are insa alte limitari)
  - Textul criptat nu ofera nici un fel de informatie despre textul clar

> Definiție
>
> *O schemă de criptare peste un spațiu al mesajelor $\mathcal{M}$ este perfect sigură dacă pentru orice probabilitate de distribuție peste $\mathcal{M}$, pentru orice mesaj $m \in \mathcal{M}$ și orice text criptat c pentru care $Pr[C = c] > 0$, următoarea egalitate este îndeplinită:*
>
> $$Pr[M = m|C = c] = Pr[M = m]$$

  - **One Time Pad (OTP)**
    - acelasi text criptat poate sa provina din orice alt text clar cu o cheie potrivita
    - are cheia la fel de lunga ca si mesajul criptat
    - cheia trebuie sa fie folosita o singura data
    - este o ***schema perfect sigura***

mesaj:	0	1	1	0	0	1	1	1	1	$\oplus$
cheie:	1	0	1	1	0	0	1	1	0	
text criptat:	1	1	0	1	0	1	0	0	1	

m, k, c de lungime n

$$E_k(m) = m \oplus k = c$$
$$D_k(c) = c \oplus k = m \oplus k \oplus k = m$$

unde

m = mesajul in clar

E_k = criptatea mesajului

k = cheia de criptare D_k = decriptarea mesajului

c = mesajul criptat

- **Scheme de criptografice moderne:** majoritatea de bazeaza pe ***securitate computationala*** si pot fi sparte daca atacatorul are la dispozitie suficient spatiu si putere de calcul
  - *Securitatea computationala*: un cifru trebuie sa fie practic, daca nu matematic, indescrifrabil (**principiul lui Kerckhoffs)**
    - calculator desktop: se pot testa aproximativ 2^{57} chei / an
    - supercalculator: se pot testa aproximativ 2^{80} chei / an
    - supercalculator, varsta universului : 2^{112} chei

Cursul 3.2 - Aleatorism

- **Pseudo Random Generator (PRG)** este un algoritm determinist care primeste un **seed** relativ scurt si genereaza o secventa pseudoaleatoare de biti. Notam:
  - lungimea seedului s = n
  - lungimea $PRG(s)$ = $l(n)$

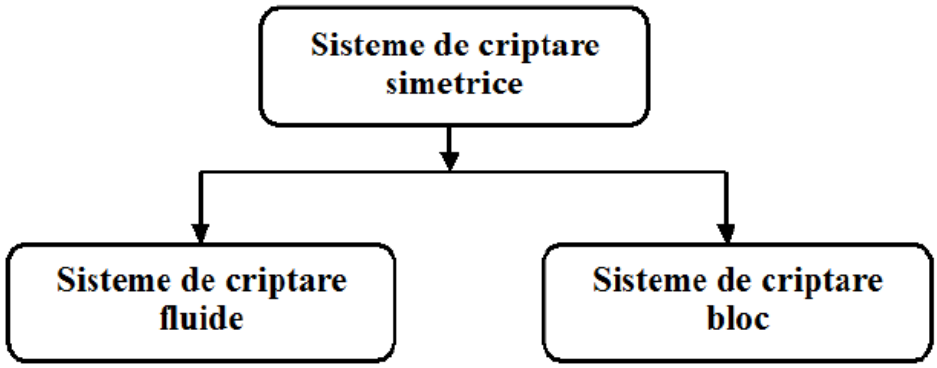
  PRG prezinta interes daca: $l(n) \geq n$, astfel *nu* genereaza aleatorism
- Un ***sir pseudoaleator*** arata similar unui ***sir uniform aleator*** din punctul de vedere al oricarui algoritm polinomial. Astfel, algoritmul polinimial nu poate face diferenta intre cele doua
- Distributia outputului unui **PRG** este departe de a fi uniforma
- **Proprietatile unui seed:**
  - Seed-ul trebuie sa fie suficient de lung incat un atac brute force sa nu fie fezabil
  - Seed-ul unui **PRG** este analogul cheii unui sistem de criptare
  - Seed-ul trebuie ales uniform si mentinut secret
- **Criptarea baza pe PRG** se obtine din OTP prin inlocuirea *pad* cu $G(k)$
  - Daca G este PRG arunci *pad* si $G(k)$ sunt indistinctibile pentru orice adversar PPT
  - OTP si sistemul de criptare bazat pe PRG sunt indistinctibile pentru un adversar PPT
- **Sistemul OTP si sistemul de criptare bazat pe PRG** sunt sigure doar daca cheia este folosita o singura data
- **Vulnerabilitati ale seed-ului:**
  - CWE-336**: Same Seed in Pseudo-Random Number Generator**
  - CWE-339**: Small Seed Space in PRNG:** spațiul seed-urilor posibile este mic
  - CWE-337**: Predictable Seed in Pseudo-Random Number Generator:** a PRNG is initialized from a predictable seed, such as the process ID or system time.
  - CWE-338**: Use of Cryptographically Weak PRNG:** the product uses a PRNG in a security context, but the PRNG's algorithm is not cryptographically strong.

Adversat PPT: probabilistic polynomial time adversary.

Securitatea computationala este o versiune mai slaba a securitatii perfecte.

Cursul 4

Cursul 4.0 - Notiuni de securitate mai puternice

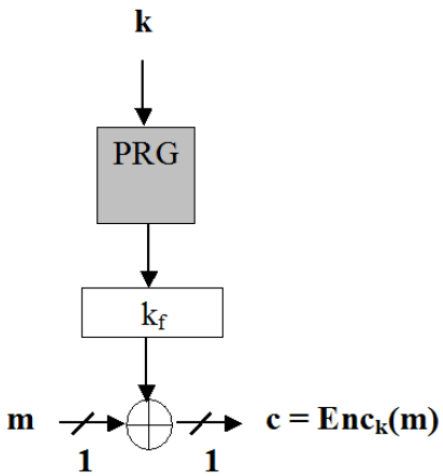


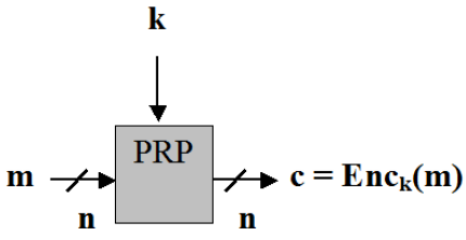
Sisteme de criptare fluide

- criptarea bitilor se realizeaza individual
- criptarea unui bit din textul clar este **independenta** de orice alt bit din text
- necesatari computationale reduse
- utilizare: telefoane mobile, dispozitive incorporate, PDA
- par sa fie mai putin sigure, multe sunt sparte

Sisteme de criptare bloc

- criptarea bitilor se realizeaza in blocuri de cate n biti
- criptarea unui bit din textul clar este **dependenta** de bitii din text care apartin aceluiasi bloc
- necesatari computationale mai avansate
- utilizare: internet
- par sa fie mai sigure, incredere mai mare





- Permutare pseudoaleatoare sau **Pseudo Random Permutation (PRP)**

    - **PRP** sunt necesare pentru constructia **sistemelor bloc**
    - **PRG** unt necesare pentru constructia **sistemelor fluide**
    - o functie **determinista** si **bijectiva** care pentru o cheie fixata produce la iesire o permutare a intrarii, **indistinctibila** fata de o permutare aleatoare. Aceasta este **eficient calculabila**
- Functie pseudoaleatoare sau **Pseudo Random Function (PRF)**
    - o generalizare a notiunii de permutare aleatoare
    - o functie **cu cheie** care este **indistinctibila** fata de o functie aleatoare
- Sistemele de criptare bloc sunt **instantieri sigure** ale PRP. Pentru n suficient de mare, un PRP este **indistinctibil** de un PRF. Pentru PRP avem nevoie si de invertibilitate.
- **Atacul nasterilor** necesita $O(2^{n/2})$ evaluari ale functie H

Cursul 5

Cursul 5.2 - Securitatea CCA

- Atac cu text clar ales - **Chosen Plain-text Attack (CPA):**
    - Atacatorul are posibilitatea sa obtina criptarea unor texte clare alese de el
    - Adversarul este **pasiv**: interactioneaza cu un **oracol de criptare**
- Atac cu text criptat ales - **Chosen Ciphertext Attack (CCA)**:
    - Atacatorul are posibilitatea sa obtina decriptarea unor texte criptate alese de el
    - Adversarul este **activ** si are putere crescuta: interactioneaza cu un **oracol de criptare** si cu un **oracol de decriptare** si poate rula atacuri in timp polinomial
- Un sistem de criptare **CCA-sigur este intotdeauna CPA-sigur** fiindca CPA este CCA in care atacatorul nu foloseste oraculul de decriptare
    - Un sistem de **criptare determinist NU** poate fi CCA-sigur fiindca sistemul nu e CPA sigur deci nu poate fi CCA-sigur
- Securitatea pentru **criptare simpla implica** securitate pentru **criptare multipla**
- Securitate CCA $\Rightarrow$ Securitate CPA $\Rightarrow$ Securitate Semantica

Cursul 5.3 - Constructii practice PRF

- **Constructia practica a PRF-ului:**

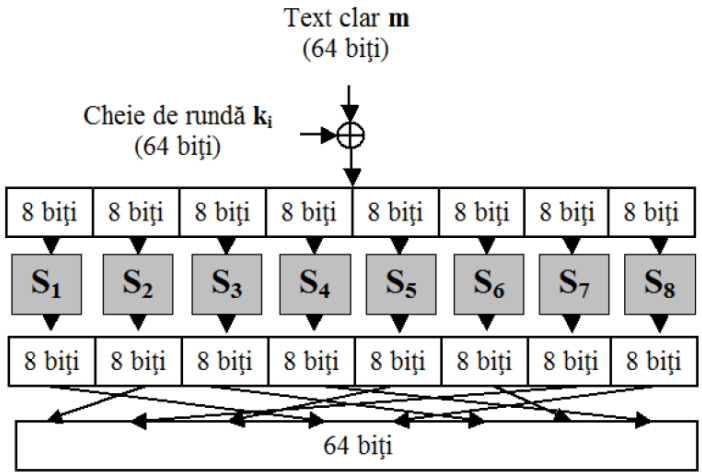
1. Se construieste functia F, pe baza mai multor functii aleatoare f_i de dimensiune mai mica
    - Consideram F pe 128 de biti si 16 functii aleatoare $f_1, f_2, ..., f_{16}$ pe cate 8 biti

$$F_k(x) = f_1(x_1) \,||\, f_2(x_2) \,||\, ... \,||\, f_{16}(x_{16})$$

    - Spunem ca functiile $\{f_i\}$ introduc **confuzie** in functia F
2. Se introduce **difuzie** prin amestecarea (permutarea) bitilor la iesirea din functie
3. Se repeta o **runda** (care presupune confuzie si difuzie) de mai multe ori

- Repetarea ***confuziei*** si ***difuziei*** face ca modificarea unui singur bit la intrare sa fie propagata asupra tuturor bitilor de iesire

- O **retea de subtitutie - permutare** este o implementare a constructiei anterioare de *confuzie-difuzie* in care functiile $\{f_i\}$ sunt fixe si se numesc permutari
  - $\{f_i\}$ se numesc **Substitution Boxes - S-boxes**



  - **Principii de baza** in proiectarea retelelor de substitutie - permutare:
    - **Principiul 1:** Inversibilitatea S-box-urilor - daca toate S-box-urile sunt inversibile atunci reteaua este inversibila (*necesitate functionala - descriptare*)
    - **Principiul 2:** Efectul de avalansa - un singur bit modificat la intrare trebuie sa afecteze toti bitii din secventa de iesire (*necesitate de securitate*)

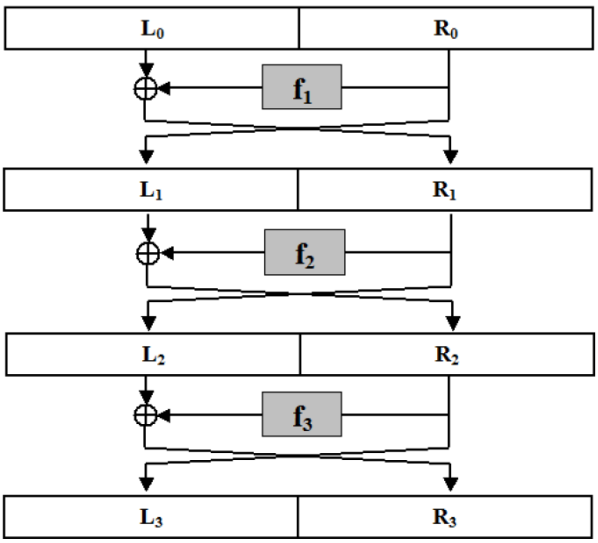
- **Advanced Encryption Standard (AES)**
  - Este o **retea de subtitutie - permutare** pe 128 de biti care poate folosi chei de 128, 192 si 256 de biti. Lungimea cheii determina numarul de runde:

Lungime cheie (biți)	128	192	256
Număr runde	10	12	14

  - Foloseste o matrice de octeti 4×4 numita **stare,** starea initiala fiind mesajul clar
  - **Starea este modificata** pe parcursul rundelor prin 4 tipuri de operatii: *AddRoundKey, SubBytes, ShiftRows, MixColumns*
  - Singurele **atacuri netriviale** sunt asupra AES cu numar redus de runde
    - **AES-128** cu 6 runde: necesita 2^{72} criptari
    - **AES-192** cu 8 runde: necesita 2^{188} criptari

        - **AES-256** cu 8 runde: necesita 2^{204} criptari
    - Nu exista un atac mai eficient decat **brute force** pentru AES cu numar complet de runde

- **Retele Feistel**
    - Se aseamana retelelor de substitutie - permutare avand componentele: S-box, permutare, procesul de derivare a cheii, runde
    - Permite obtinerea unei **structuri inversibile** folosind **elemente neinversibile** (S-box-uri)



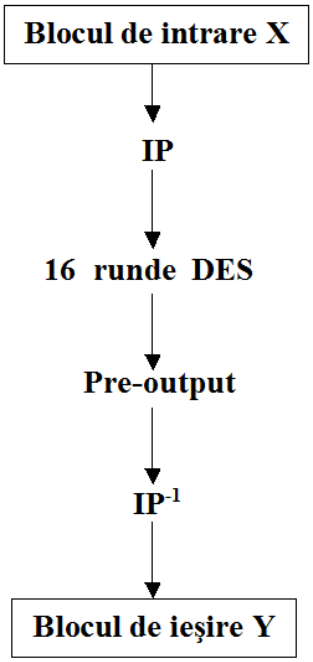
Cursul 6

Cursul 6.1 - Padding Oracle Attack

- Atacul cere ca un adversar sa poata afla numai daca un text criptat modificat este unul valid (care se poate decripta corect), nefiind necesara intreaga functionalitate a unui oracol de decriptare (care intoarce textul clar corespunzator unui text criptat).
    - Acesta poate fi exploatat pentru a afla intregul text clar. Sunt necesare $256 \times bt$ pentru a gasi intregul mesaj clar, unde bt este numarul de octeti din mesajul original
- **CBC cu padding PKCS7:** serverul actioneaza ca un oracol de padding - adversarul ii trimite texte criptate si afla daca padding-ul este corect sau nu

Cursul 6.3 - Data Encryption Standard (DES)

- **DES** este o **retea Feistel** cu 16 runde si o cheie de 56 de biti
- Procesul de derivare a cheii (*key schedule)* obtine o sub-cheie de runda k_i pentru fiecare runda,pornind de la cheia master k
  - Fiecare sub-cheie k_i reprezinta permutarea a 48 de biti din cheia k
  - Procedura de obtinere a sub-cheilor este fixa si cunoscuta, singurul secret este cheia master
- Modificarea unui bit de intrare afecteaza mereu **cel putin** 2 biti de iesire
- **DES** poate fi spart din **cautare exhaustiva**
  - *Problema*: lungimea mica a cheii

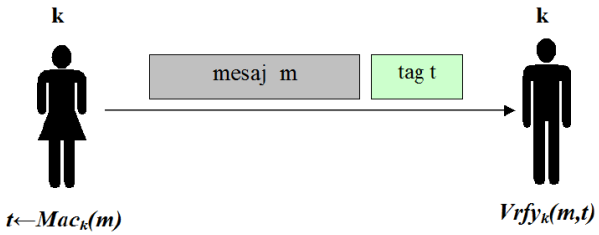


- **Criptanaliza avansata**
  - *Criptanaliza diferentiala*: presupune complexitate de timp 2^{37} dar cere ca atacatorul sa analizeze 2^{36} texte criptate obtinute dintr-o multime de 2^{47} texte clare alese
  - *Criptanaliza liniara:* necesita 2^{43} texte criptate dar textele clare nu trebuie sa fie alese de atacator, ci doar cunoscute de acesta
- **Triplu DES (3DES):** tripla invocare a lui DES folosind doua sau trei chei
  - Foarte **eficient** in implementarile *hardware*, **lent** la implementari *software*
  - Popular in aplicatii financiare si protejarea informatiilor biometrice din pasapoarte
  - Nu se cunoaste niciun atac practic

Cursul 7

Cursul 7.1 - Coduri de autentificare a mesajelor (MAC)

- Scopul de baza al criptografiei este sa asigure **comunicarea sigura** de-a lungul unui canal public de comunicare
  - **Confidentialitatea** se obtine prin **schemele de criptare**
  - **Integritatea** se obtine prin **coduri de autentificare a mesajelor (MAC)**
    - Criptarea, in general, nu ofera integritatea mesajelor
- Scopul MAC-urilor este de a impredica un adversar sa modifice un mesaj trimis fara ca partile care comunica sa nu detecteze modificarea



1. Alice si Bob stabilesc o cheie secreta k care este partajata
2. Cand Alice ii trimite un mesaj lui Bob, calculeaza intai un tag t pe baza mesajului m si a cheii k si trimite perechea (m, t). Tagul este generat printr-un algoritm numit *Mac*
3. La primirea perechii (m, t) Bob verifica daca tagul este valid cu un algoritm de verificare *Vrfy*

- **Definitia formala** a unui cod de autentificare a mesajelor:

Un cod de autentificare a mesajelor (MAC) definit peste $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ *este format dintr-un triplet de algoritmi polinomiali* $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ *unde:*

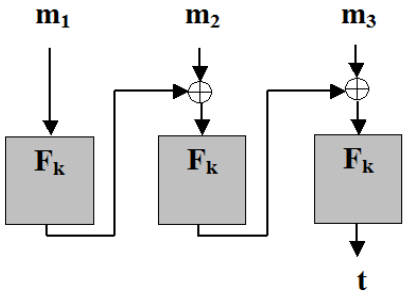
1. $\mathrm{Gen}(1^n)$*: este algoritmul de generare a cheii k (aleasă uniform pe n biți)*
2. $\mathrm{Mac} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ *este algoritmul de generare a tag-urilor* $t \leftarrow \mathrm{Mac}_k(m)$*;*
3. $\mathrm{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0, 1\}$
   este algoritmul de verificare ce întoarce un bit $b = \mathrm{Vrfy}_k(m, t)$ *cu semnificația că:*
   - $b = 1$ *înseamnă* valid
   - $b = 0$ *înseamnă* invalid

a.î : $\forall m \in \mathcal{M},\ k \in \mathcal{K}\ \mathrm{Vrfy}_k(m, \mathrm{Mac}_k(m)) = 1.$

- Niciun adversar polinomial **nu ar trebui** sa poata genera un tag **valid** pentru niciun mesaj nou care nu a fost deja trimis si autentificat de partile care comunica
  - Adversarul are *acces* la *mesajele* trimise intre parti impreuna cu tag-urile aferente
  - Este activ si poate influenta autentificarea unor mesaje **alese de el**, dar nu mesaje trimise
- Definitia MAC **nu ofera protectia impotriva atacurilor prin replicare** - in care un adversar copiaza mesajul impreuna cu tagul trimis de partile de comunicare
  - Proiectia impotriva replicarii trebuie facuta la nivel inalt de aplicatiile care folosesc MAC-uri
- **Functiile pseudoaleatoare (PRF)** sunt un instrument bun pentru a construi MAC-uri **sigure**, care nu poate fi falsificat prin atacurile cu mesaj ales
- **Constructii MAC:**
  - Exista doua constructii de baza care se folosesc in practica:
    - **CBC-MAC**: folosit pe larg in industia bancara
    - **HMAC:** pentru protocoale de internet (SSL, IPsec, SSH, ..)

1. **CBC-MAC**
2. **HMAC (Hash MAC)**

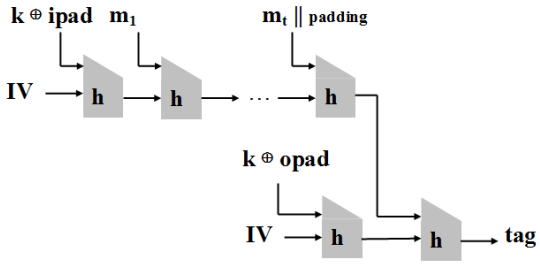
Doar iesirea ultimului bloc constituie tag-ul. Intoarcerea tuturor blocurilor intermediare duce la pierderea securatatii.



unde i_{pad} consta in byte-ul 0x5C repetat,

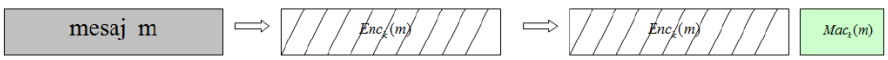
o_{pad} consta in byte-ul 0x36 repetat iar

IV reprezinta o constanta fixa



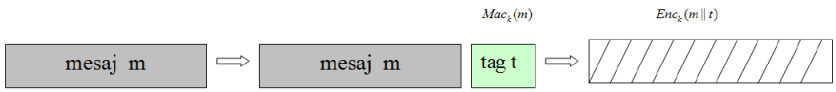
- **Authenticated encryption:** combinarea *confidentialitatii* si a *integritatii datelor*
  - **Nu orice combinatie** de criptare sigura si MAC sigur ofera ambele proprietati de securitate
  - Trei abordari uzuale pentru combinarea criptarii si autentificarii mesajelor:

1. *Criptare-și-autentificare*: criptarea și autentifcarea se fac independent. Pentru un mesaj clar m, se transmite mesajul criptat $\langle c, t \rangle$ unde
$$c \leftarrow \mathrm{Enc}_{k_1}(m) \text{ și } t \leftarrow \mathrm{Mac}_{k_2}(m)$$
La recepție, $m = \mathrm{Dec}_{k_1}(c)$ și dacă $\mathrm{Vrfy}_{k_2}(m, t) = 1$, atunci întoarce m; altfel întoarce $\perp$.



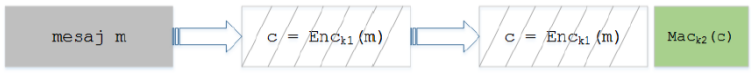
Combinatia **nu este neaparat sigura**. Un MAC sigur nu implica confidentialitate

2. *Autentificare-apoi-criptare*: întâi se calculează tag-ul t apoi mesajul și tag-ul sunt criptate împreună
$$t \leftarrow \mathrm{Mac}_{k_2}(m) \text{ si } c \leftarrow \mathrm{Enc}_{k_1}(m||t)$$
La recepție, $m||t = \mathrm{Dec}_{k_1}(c)$ și dacă $\mathrm{Vrfy}_{k_2}(m, t) = 1$, atunci întoarce m; altfel întoarce $\perp$.



Combinatia **nu este neaparat sigura**. Se poate construi o schema de criptare CPA-sigura care impreuna cu orice MAC sigur nu poate fi CCA-sigura

3. *Criptare-apoi-autentificare*



Combinatia **este intotdeauna sigura**. Se foloseste in IPsec

- Pentru scopuri de criptare diferite trebuie folosite **chei diferite** (criptare si autentificare)
- Scheme de criptare autentificare: OCB, GCM, CCM, EAX. *TLS* foloseste GCM

Cursul 7.2 - Functii hash

- **Functii** care primesc ca argument o secventa de **lungime variabila** pe care o comprima intr-o secventa **de lungime fixata, mai mica**
  - Complexitate de cautare intr-o functie hash este de $O(1)$
- **Coliziune:** pentru 2 valori $x \neq x'$ avem $H(x) = H(x')$
  - In criptografie, o functie hash impune ca determinarea unei coliziuni sa fie dificila
- Exista 3 niveluri de securitate:
  - **Rezistenta la coliziuni:** cea mai puternica notiune de securitate
  - **Rezistenta la a doua preimagine:** presupunem ca fiind dat x este dificil de determinat $x' \neq x$ a.i. $H(x) = H(x')$
  - **Rezintenta la prima preimagine:** presupunem ca fiind dat $H(x)$ este imposibil de determinat x

Cursul 8

Cursul 8.1 - SHA-3

- **Message Digest 5 (MD5)**: output pe 128 biti, nu e rezistent la coliziuni
- **Secure Hash Algorithm:**
  - **SHA-1**: output 160 biti, prima coliziune practica 2^{63} evaluari ale functiei hash, nesigur
  - **SHA-2**: prezinta 2 versiuni - **SHA-256 si SHA-512** in functie de lungimea output-ului. Nu se cunosc vulnerabilitati iar **SHA-2 si SHA-3** sunt **rezistente la coliziuni**
  - **SHA-3:** ales printr-o competitie publica (echipa Keccak)
    - Lungimea secventei de iesire $n = 224, 256, 384, 512$ biti
    - Eficienta crescuta fata de SHA-2, utilizare in HMAC
    - Rezistenta la coliziuni, prima si a doua preimagine
    - Parametrizabila, numar de runde variabil si foloseste **sponge functions**

Cursul 8.2 - Aplicatii ale functiilor hash

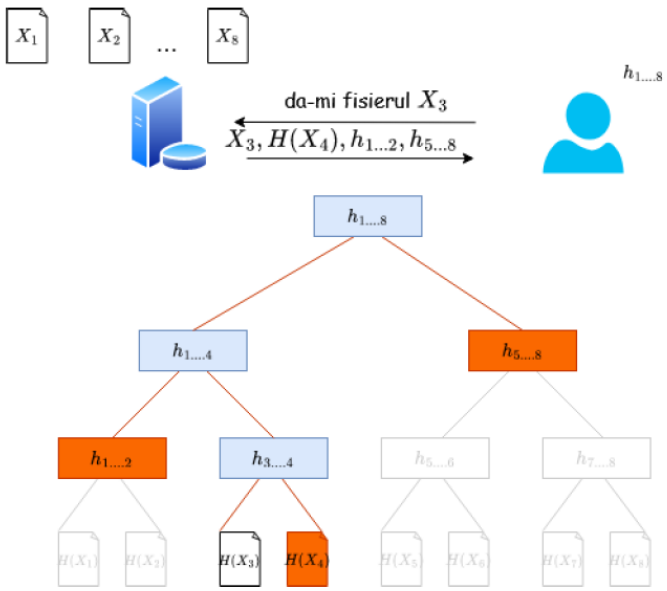
- Cel mai bun atac generic pentru gasirea de coliziuni ne spune ca avem nevoie de functii hash cu output-ul de $2n$ biti pentru securitate impotriva adversarilor care ruleaza in 2^n

- Exemplu: daca dorim **gasirea de coliziuni** sa necesite timp 2^{128}atunci trebuie sa alegem functii hash cu output-ului de 256 biti
- Pentru **sistemele de criptare bloc** avem nevoie de n biti pentru securitate impotriva adversarilor care ruleaza in timp 2^n

- Functiile hash pot fi folosite pentru a construi **MAC-uri de lungime variabila** in HMAC

$$x \to H(x) \to Mac_k(H(x))$$

- **Aplicatii** pentru functiile hash:
  - *Amprentarea virusurilor*: scannerele de virusi pastreaza hash-ul virusilor cunoscuti
  - *Deduplicarea datelor*: stocarea in cloud partajata de mai multi utilizatori
  - Un client incarca mai multe **fisiere** pe server si doreste ca atunci cand le recupereaza sa **verifice daca au fost modificate** folosind functii hash. Solutii:
    1. clientul stocheaza local $h_1 = H(x_1)$, ... , $h_n = H(x_n)$ si verifica daca $h_i = H(x_i')$ pentru fiecare fisier x_i' recuperat. *Dezavantaj:* spatiul creste liniar in n
    2. clientul stocheaza local un singur $h = H(x_1, ..., x_n)$. *Dezavantaj:* pentru a verifica x_i trebuie sa recupereze toate fisierele $x_1, ..., x_n$
    3. **Arborii Merkle**: compromis intre cele doua solutii de mai sus



Pe baza valorilor returnate, utilizatorul poate re-calcula valoarea radacinii pentru a verifica daca este aceeasi cu valoarea stocata de el:

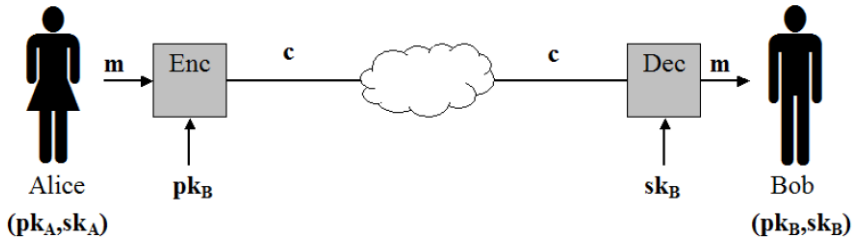
user-ul calculează $H(X_3)$, apoi $h'_{3...4} = H(H(X_3), H(X_4))$,
$h'_{1...4}$ si $h'_{1...8}$ si verifica daca $h'_{1...8} = h_{1...8}$

- **Atac** de tip **dictionar**: incercarea spargerii unei parole folosind cuvinte cu sens sau parole uzuale

- **Salting:** adaugarea unei secvente aleatoare de n biti, distincta pentru fiecare utilizator
  - Adversarul nu poate precalcula valorile hash inainte de a obtine acces la fisierul de parole
  - Prin citirea parolelor adversarul nu isi poate da seama ca doi utilizatori au aceiasi parola

Cursul 8.3 - Criptografie cu cheie publica (asimetrica)

- **Cheia simetrica:** asigura confidentialitatea si integritatea mesajelor transmise pe canale nesecurizate - necesita o cheie secreta comuna partilor care se poate trimite:
  1. printr-un canal nesecurizat - **NU,** poate fi interceptata si folosita la decriptare
  2. printr-un canal sigur - posibil doar la nivel guvernamental sau militar
- **Cheie asimetrica**: doua chei diferite folosite pentru criptare si decriptare



  - *Cheia publica* p_k*:* folosita pentru **criptare -** larg raspandita pentru a se putea cripta
  - *Cheia privata* s_k*:* folosita pentru **decriptare** - cunoscuta doar de entitatea corespunzatoare

Un sistem de criptare asimetric definit peste $(\mathcal{M}, \mathcal{C})$*, cu:*
- $\mathcal{M}$ *= spațiul textelor clare (mesaje)*
- $\mathcal{C}$ *= spațiul textelor criptate*

este un triplet $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$*, unde:*
1. $\mathrm{Gen}(1^n)$ *- generează cheile* (pk, sk)
2. Enc *- primește la intrare o cheie publică pk și un mesaj m și calculează* $c \leftarrow \mathrm{Enc}_p k(m)$
3. Dec *- primește la intrare o cheie secretă pk și un mesaj criptat c și intoarce mesajul clar sau eroare (simbolul* $\perp$*)*

a.î. $\forall m \in \mathcal{M}, (pk, sk)$ *generate cu algoritmul Gen*(1^n)
$Dec_{sk}(Enc_{pk}(m)) = m.$

Criptarea simetrica

- necesita secretizarea intregii chei
- rolurile emitatorului si receptorului pot fi schimbare intre ele
- pentru fiecare mesaj primit de la un emitator diferit trebuie sa partajeze o cheie secreta diferita

Criptarea asimetrica

- necesita secretizarea a jumatate de cheie
- rolurile emitatorului si receptorului **NU** pot fi schimbare intre ele
- o pereche de chei asimetrice permite oricui sa transmita informatie criptata catre entitatea corespunzatoare

Cursul 8.5 - Prezumtii criptografice dificile

- **Criptografia moderna** se bazeaza pe prezumtia ca anumite probleme **nu** pot fi rezolvate in timp **polinomial.** *Scheme de criptare* si autentificare se bazeaza pe prezumtia existentei *permutarilor pseudoaleatoare*
- La **criptografia simetrica** (cu cheie secreta) am vazut **primitive** criptografice (i.e. functii hash, PRG, PRF) care pot fi construite eficient *fara* a implica teoria numerelor;
    - La **criptografia asimetrica** (cu cheie publica) constructiile cunoscute se bazeaza pe probleme *matematice* dificile din *teoria numerelor*

1. **Problema factorizarii numerelor intregi**
    - Dat fiind un numar compus N, problema cere sa se gaseasca doua **numere prime** x_1 si x_2 pe n biti a.i. $N = x_1 \times x_2$ - problema devine dificila pentru numere foarte mari
    - Nu se cunosc algoritmi polinomiali pentru problema factorizarii
2. **Problema logaritmului discret**

Cursul 9

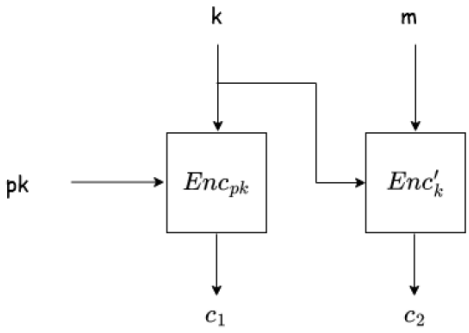
Cursul 9.0 - Notiuni de securitate in criptografia asimetrica

- **Securitatea perfecta** in criptografia asimetrica: analog cu cea simetrica doar ca adversarul cunoaste, in afara textului criptat, si cheia publica
    - **Securitatea perfecta NU** este posibila in **cadrul criptografiei cu cheie publica**
    - Nicio schema de criptare cu cheie publica **determinista** nu poate fi semnatic sigura pentru *interceptarea simpla* (nicio schema determinista din *criptografia simetrica* nu e *CPA sigura)*
- **Indistinctibilitatea** este echivalenta cu definita **securitatii CPA**
    - In cazul **criptarii asimetrice** adversarul nu mai are nevoie de oraculul de criptare fiindca are access la cheia publica p_k deci *isi poate criptata singur mesajele*
- Notiunea de **securitate CCA** ramane identica: adversarul poate interactiona cu un *oracol de decriptare*, fiind un adversar activ care poate rula atacuri in timp polinomial

Cursul 9.1 - Criptare hibrida

- Criptarea cu **cheie secreta** se realizeaza *mult mai rapid* decat criptarea cu cheie publica
    - Pentru mesajele *suficient de lungi* se foloseste criptarea cu cheie secreta in tandem cu criptarea cu cheie publica - **criptare hibrida**

1. Expeditorul alege o cheie k pe care o cripteaza folosind cheia publica a destinatarului, rezultand $c_1 = Enc_{pk}(k)$
2. Expeditorul cripteaza m folosind o schema de criptare cu cheie secreta Enc' cu cheia k, rezultand $c_2 = Enc'_k(m)$
3. Mesajul criptat este $c = (c_1, c_2)$



Cursul 9.2 - RSA

- Este cel mai cunoscut si utilizat algoritm cu cheie publica
  - Se bazeaza pe dificultatea factorizarii numerelor mari $N = p \cdot q$, p si q prime
  - Daca nu se cunosc p, q atunci calculul celorlalte valori este la fel de dificil ca factorizarea
- Prezumtia RSA este ca exista un algoritm $GenRSA$ pentru care problema RSA este dificila
  - **Algoritm GenRSA:** input = n, output = N, e, d
    1. genereaza p si q prime, n-biti, $N = p \cdot q$
    2. $\phi(N) = (p-1)(q-1)$
    3. gaseste e a.i. $gcd(e, \phi(N)) = 1$
    4. calculeaza $d = e^{-1} mod \phi(N)$
    5. returneaza N, e, d
- Sistemul de criptate **Textbook RSA** bazat pe problema anterioara:
  - Se ruleaza GenRSA pentru a determina N, e, d
    - **Cheia publica** este $p_k = (N, e)$
    - **Cheia privata** este $s_k = d$
    - **Enc**: data cheia publica (N, e) si un mesaj m intoarce $c = m^e mod N$
    - **Dec**: data cheia secreta (N, d) si un mesaj criptat c intoarce $m = c^d mod N$
  - *Sistemul de criptare* este **corect** dar **nu este CPA** sau **CCA**-sigur: **este determinist**

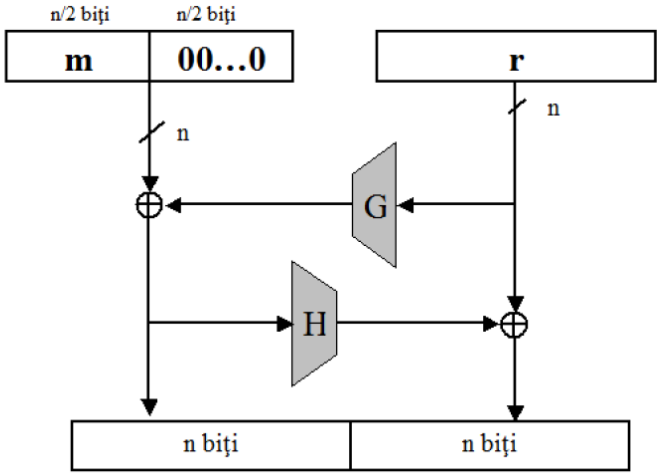
Cursul 9.3 - PKCS

- **Padded RSA:** *Eliminarea determinismului* din Textbook RSA prin adaugarea unui padding pad *generat aleator* la mesajul clar inainte de criptare
  - Pentru $l(n)$ foarte mare e posibil un atac brute force, pentru $l(n)$ mic este CPA-sigur
- **Public-Key Cryptography Standard (PKCS)** foloseste *Padded RSA* si este utilizat in HTTPS, SSL/TLS, XML Encryption (PKCS v1.5)
  - Se crede ca este CPA-sigur dar nu este demonstrat. Nu este CCA-sigur

- Criptarea se realizeaza altfel: unde r este ales aleator

$$(00000000||00000010||r||00000000||m)^e \mod N$$

- **Optimal Asymmetric Encryption Standard (OAEP)** un standard PKCS (v2.0) care e CCA-sigur
  - Este de fapt o metoda **nedeterminista** si **inversabila** de padding
  - Transforma un mesaj m de lungime $n/2$ intr-o secventa m' de lungime $2n$



unde G, H sunt functii hash si r este o secventa generata aleator

Cursul 10

Cursul 10.2 - Problema logaritmului discret (DLP)

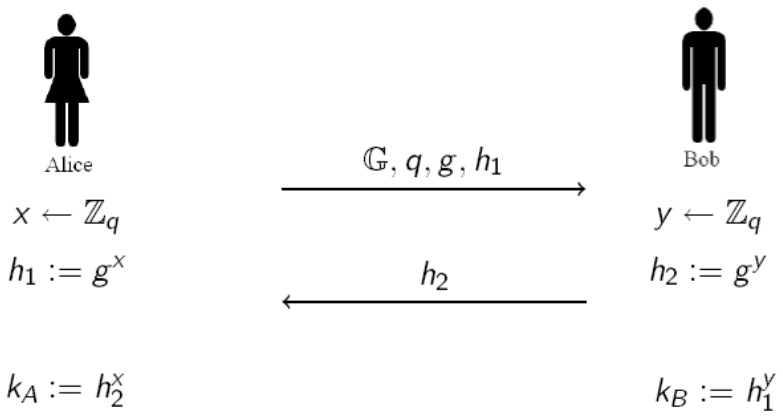
- Fie G un grup ciclic de ordin q cu $|q| = n$ iar g este generatorul lui G
  - pentru fiecare $h \in G$ exista un unic $x \in \mathbb{Z}_q$ a.i. $g^x = h$
  - se cere gasirea lui x stiind G, q, g, h unde $x = log_g h$
- Problema se poate rezolva prin brute force in complexitate de timp $O(q)$
  - Exista algoritmi mai eficienti decat brute force, imparititi in doua categorii:
    - algoritmi *generici* care functioneaza in grupuri arbitrare (ciclice)
    - algoritmi *non-generici* care lucreaza pe grupuri specifice
- Se pot construi **functii hash** rezistente la **coliziuni** bazate pe **dificultatea DLP**
- Cel mai bun algoritm pentru DLP este **sub-exponentiala**

Cursul 10.3 - Schimbul de chei Diffie-Hellman

- **Diffie** si **Hellman** au introdus **3 primitive** cu *cheie publica* diferite, care folosesc aceiasi structura de schimb de chei
  1. **Sisteme de criptare cu cheie publica**
  2. **Semnaturi digitale:** analog MAC-urilor

3. **Schimbul de chei**

- Un **protocol de schimb de chei** se realizeaza intre doua persoane care nu partajeaza in prealabil niciun secret si pot **genera** o **cheie comuna, secreta**
  - Comunicarea necesara pentru stabilirea chestii se face printr-un *canal public*
    1. Alice genereaza un grup ciclic G de ordin q cu $|q| = n$ si g unde generator al grupului
    2. Alice alege $x \leftarrow^R \mathbb{Z}_q$ si calculeaza $h_1 = g^x$
    3. Alice ii trimite lui Bob mesajul (G, g, q, h_1)
    4. Bob alege $y \leftarrow^R \mathbb{Z}_q$ si calculeaza $h_2 = g^y$
    5. Bob ii trimite h_2 lui Alice si intoarce cheia $k_B = h_1^y$
    6. Alice primeste h_2 si intoarce cheia $k_A = h_2^x$
  - **Corectitudinea protocolului** presupune $k_A = k_B$ ceea ce se verifica usor



  - O conditie **minimala** pentru ca protocolul sa fie **sigur** este ca **DLP** sa fie **dificila** in G

- **Problema de calculabilitate Diffie-Hellman (CDH):** un adversar sa poata afla cheia comuna $k_A = k_B$ cunoscand h_1, h_2, G, g (din canalul public de comunicare)
- **Problema de decidabilitate Diffie-Hellman (DDH):** un adversar sa poata sa poata **distinge** cheia comuna $k_A = k_B$ fata de o valoarea aleatoare
- Schimbul de chei Diffie-Hellman este nesigur pentru un **adversar activ** deoarece acesta poate *intercepta informatiile* transmite pe **canalul public**:
  - Cand Alice ii trimite mesajul lui Bob acesta poate fi interceptat de un atacator ce il poate impersona pe Bob si poate comunicare cu Alice - atac de tipul **Man-in-the-Middle**

Cursul 10.4 - ElGamal

- Un sistem de criptare baza pe schimbul de chei Diffie-Hellman
  - Este **nedeterminist,** datorita alegerii aleatoare a lui y la fiecare criptare
  - Se bazeaza pe **dificultatea DLP**, altfel este nesigur
  - Are proprietatea de **homomorfism:** $Dec_sk(c_1 \cdot c_2) = Dec_sk(c_1) \cdot Dec_sk(c_2)$
  - Utilizarea **parametrilor publici** G, q, g apoi fiecare utilizator isi fenereaza cheia x

- Daca **problema decizionala DDH** este **dificila** in grupul G atunci schema de criptare ElGamal este *CPA-sigura*, dar *nu este CCA-sigura*

Cursul 11 - Semnaturi digitale si PKI

- Schemele de **semnatura digitala** reprezinta *echivalentul MAC-urilor* in criptografia cu cheie publica, desi exista cateva diferente importante intre ele
  - O schema de semnatura digitala ii permite unui **semnatar** S care a stabilit o cheie publica p_k sa semneze un mesaj incat oricine care cunoaste cheia p_k poate verifica originea mesajului ca fiind S si integritatea lui
- **MAC-urile** si **schemele de semnatura digitala** sunt folosite pentru **asigurarea integritatii** mesajelor cu urmatoarele **diferente**:
  - Scheme de semnaturi digitale sunt **public verificabile** si au proprietatea **non-repudierii** - un semnatar nu poate nega faptul ca a semnat un mesaj
  - MAC-urile au avantajul ca sunt de 2-3 ori **mai rapide** decat semnaturile digitale

O semnătură digitală definita peste $(\mathcal{K}, \mathcal{M}, \mathcal{S})$ *este formată din trei algoritmi polinomiali* $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ *unde:*

1. Gen*: este algoritmul de generare a perechii de cheie publică și cheie privată* (pk, sk)
2. $\mathrm{Sign} : \mathcal{K} \times \mathcal{M} \to \mathcal{S}$ *este algoritmul de generare a semnăturilor* $\sigma \leftarrow \mathrm{Sign}_{sk}(m)$*;*
3. $\mathrm{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{S} \to \{0,1\}$
   este algoritmul de verificare ce întoarce un bit $b = \mathrm{Vrfy}_{pk}(m, \sigma)$ *cu semnificația că:*
   - $b = 1$ *înseamnă* valid
   - $b = 0$ *înseamnă* invalid

a.î : $\forall m \in \mathcal{M},\ k \in \mathcal{K},\ \mathrm{Vrfy}_{pk}(m, \mathrm{Sign}_{sk}(m)) = 1.$

- Semnaturile digitale **nu** sunt rezistente la **atacuri prin replicare** i.e un atacator trimite un mesaj semnat mai departe, *fara sa fie autorizat de semnatar*
  - **Paradigma "hash-and-sign"** este *sigura*: inainte de semnare mesajul trece printr-o functie hash - *atata timp cat functia este rezistenta la coloziuni* (folosita larg in practica)
- **Semnatura digitala baza pe RSA**
  1. Gen genereaza N, e, d si $p_k = (N, e)$ iar $s_k = d$
  2. $Sign(s_k, m)$ semneaza mesajul m folosind cheia $s_k = d$ astfel $\sigma = m^d mod N$
  3. $Vrfy(p_k, m, \sigma)$ semneaza este valida daca si numai daca $m = \sigma^e mod N$
  - Aceasta **nu este sigura**, deoarece un atac poate fi generat astfel:
    - *scop adversar*: falsificarea semnaturii mesajului $m \in \mathbb{Z}_N^*$ e pentru $p_k = (N, e)$
    - *actiune adversar*: alege $m_1, m_2 \in \mathbb{Z}_N^*$ a.i. $m = m_1 \cdot m_2 mod N$
    - obtine *semnaturile* σ_1 si σ_2 pentru mesajele m_1, m_2
    - *intoarce* $\sigma = \sigma_1 \cdot \sigma_2$ mod N ca semnatura valida pentru m

- aceasta este *o semnatura valida* pentru ca:

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed} = m_1 \cdot m_2 = m \bmod N$$

- **Semnatura digitala baza pe RSA-FDH** (full domain hash)
  1. Gen genereaza N, e, d si $p_k = (N, e)$ iar $s_k = d$
  2. $Sign(s_k, m)$ semneaza mesajul m folosind cheia $s_k = d$ astfel $\sigma = H(m)^d mod N$
  3. $Vrfy(p_k, m, \sigma)$ semneaza este valida daca si numai daca $H(m) = \sigma^e mod N$
  - Aceasta schema **este sigura** daca H indeplineste doua conditii:
    1. H este rezistenta la coliziuni
    2. $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- **Scheme de identificare:** sunt protocoale interactive care permit unei parti (**Prover**) sa isi demonstreze identitatea in fata unei alte parti (**Verifier**)
  - Daca *problema logaritmului discret* este **dificila**, atunci schema de **identificare Schnorr** este sigura **impotriva atacurilor pasive**
  - Daca problema logaritmului discret este **dificila** si H este **modelata ca o functie aleatoare**, atunci **semnatura Schnorr** este **sigura**
- Alte scheme de semnatura digitala: **Digital Signature Algorithm (DSA)** bazat pe problema logaritmului discret si **ECDSA** care este varianta DSA bazata pe curbe eliptice (standard), ambele fiind incluse in **Digital Signature Standard (DSS)**
- **Certificatul digital** este o semnatura care ataseaza unei entitati o anumita cheie publica, pentru ca aceasta sa poata fi distribuita in mod sigur
  - Exemplu: Charlie doreste sa emita un certificat pentru Bob. Charlie are cheia generata (pk_C, sk_C) iar Bob are cheia generata (pk_B, sk_B) iar Charlie cunoaste pk_B atunci:

$$\text{cert}_{C \to B} = \text{Sign}_{sk_C}(\text{"}\textit{Cheia lui Bob este } pk_B\text{"})$$

  - **Public Key Infractructe (PKI)**: permite distribuirea la scara larga a cheilor publice. Exista mai multe modele de PKI, dupa cum urmeaza:
    1. **Modelul cu o singura autoritate de certificare (CA)** in care toata lumea are incredere si care emite certificate pentru cheile publice
    - Oricine foloseste serviciile CA trebuie sa obtina o **copie legitima** a cheii ei publice pk_{CA}
    2. **Modelul cu autoritati multiple de certificare**: se bazeaza pe delagare si lanturi de certificare, toate certificarile se pot verifica pana se ajunge la **CA-ul root**

$$pk_A, \text{cert}_{B \to A}, pk_B, \text{cert}_{C \to B}$$

    3. **Modelul "web-of-trust":** *oricine poate emite certificate* pentru orice altcineva si fiecare utilizator decide cata incredere poate acorda certificatelor emise de alti utilizatori

- **Invalidarea certificatelor** se poate face prin mai multe metode
    1. **Expirarea:** se include data expirarii in certificat si se verifica cu validitatea semnaturii
    2. **Revocarea**: includerea unui numar serial, pe care CA-ul il poate include pe o lista de certificate revocate pe care o publica la finalul zilei

Cursul 12 - Criptografia post-cuantica

- In cadrul criptografiei de pana acum am discutat despre un **adversar PPT** care ruleaza in timp **polinomial** pe un calculator conventional (**clasic**)
- Aparitia unui **calculator cuantic** care sa poate fi folosit in practica va inseamna ca toti algoritmii cu cheie publica folositi in prezent dar si protocoalele asociate devin **vulnerabile**

Criptografia cuantica

- implementari folosind calculatoare cuantice, fenomene mecanice cuantice si canale de comunicare cuantice
- dificil de implementat la scara larga
- in unele cazuri este sigura neconditionat

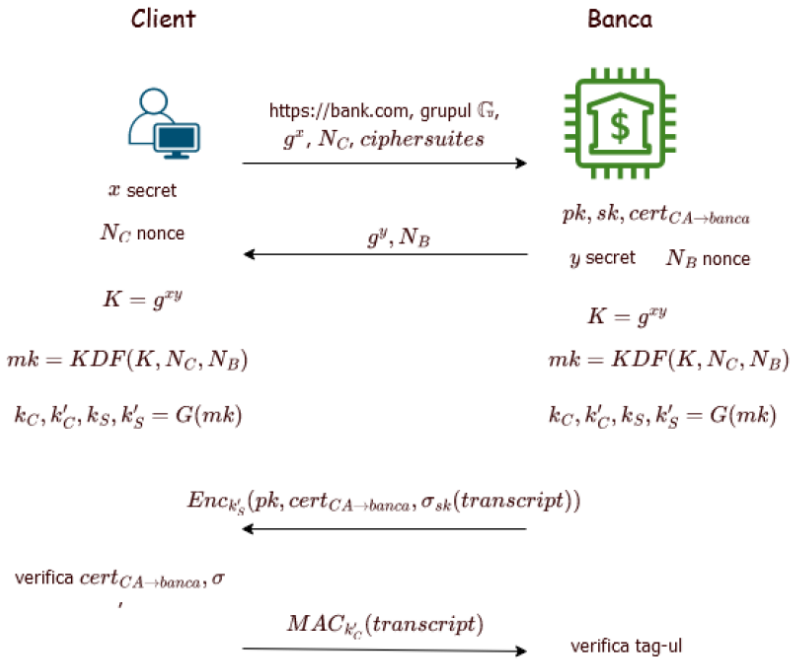
Criptografia post-cuantica

- implementari folosind calculatoare clasice
- este sigura si in fata unui adversar care are un calculator cuantic
- se bazeaza pe probleme matematice dificile computational chiar si pentru algoritmii cuantici

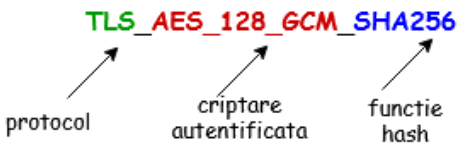
- Algoritmii cuantici presupun in general **cresterea dimensiunii cheii** si ofera imbunatatiri de ordin **polinomial** fata de cei mai buni algoritmi clasici
- Problema factorizarii si a logaritmului discret devin usoare pentru un calculator cuantic - timp polinomial (algoritmul lui Shor)
- **Problema dificila pentru un calculator cuantic**: problema Learning with Errors

Cursul 13 - TLS

- **Trasport Layer Security:** un protocol folosit de browser-ul web pentru **https**
    - Primele versiuni se numeau **Secure Sockets Layer (SSL)**
    - Versiuni: **SSL 3.0** (1995), **TLS 1.0** (1999), **TLS 1.1** (2006) - toate au probleme de securitate, **TLS 1.2** (2008), **TLS 3.0** (2018) cea mai sigura. Se recomanda folosirea *minim a TLS 1.2*
- **Protocolul TLS** permite unui client (browser) si un server (website) sa stabileasca un *set de chei* folosit la **comunicarea criptata si autentificata.** Consta in doua parti
    1. *protocolul handshake:* realizarea schimbului de chei care stabileste un set de chei comune
    2. *protocolul record-layer:* foloseste cheile stabilite pentru criptare si autentificare
- **Protocolul Handshake:**



- **Key Derivation Function (KDF)**: algoritm criptografic pentru derivarea de chei, pe baza unui PRF
- G: generator de numere **pseudo-aleatoare** (PRG)
- **Ciphersuites:** colectie de algoritmi criptografici.
- **TLS suporta 5 chipersuites**
  - *TLS_AES-128-GCM-SHA256*
  - *TLS_AES-256-GCM-SHA384*
  - *TLS_CHACHA20-POLY1305-SHA256*
  - *TLS_AES-128-CCMSHA256*
  - *TLS_AES-128-CCM-8-SHA256*



- **Transcript:** reprezinta mesajele trimise in cadrul protocolului pana la momentul curent
- **Cheile** k'_s si k'_c sunt folosite doar la handshake iar **clientul** si **serverul** au la final cheile k_s si k_c pe care le folosesc ulterior pentru criptare si autentificare
  - *Clientul* are garantia ca la final a partajat cheile cu **server-ul legitim** si ca acestea nu au fost **interceptate** sau **modificate** de o terta parte
  - **TLS 1.2:** clientul si server-ul foloseau o **schema de criptare** cu **cheie publica** in locul schimbului de chei Diffie-Hellman - compromiterea cheii secrete a serverului K duce la compromiterea tuturor cheilor de sesiune
  - **TLS 1.3:** are proprietatea de **forward secrecy** care presupune ca, in cazul compromiterii server-ului, cheile de sesiune anterioare nu sunt compromise
    - cheia K este calculata pe baza secretului y in fiecare sesiune, fiind nou de fiecare data