

## **Curs 1 - Principiile de baza ale securitatii**

## Amenintari si obiectivele securitatii

- Triada C-I-A (confidentiality, integrity, availability)
  - Confidentialitate – bunurile protejate pot fi vazute doar de persoanele autorizate
  - Integritate - bunurile protejate pot fi modificate doar de persoanele autorizate
  - Disponibilitate (availability) - bunurile protejate pot fi utilizate doar de persoanele autorizate

## **Principiile securitatii**

- Criptografice

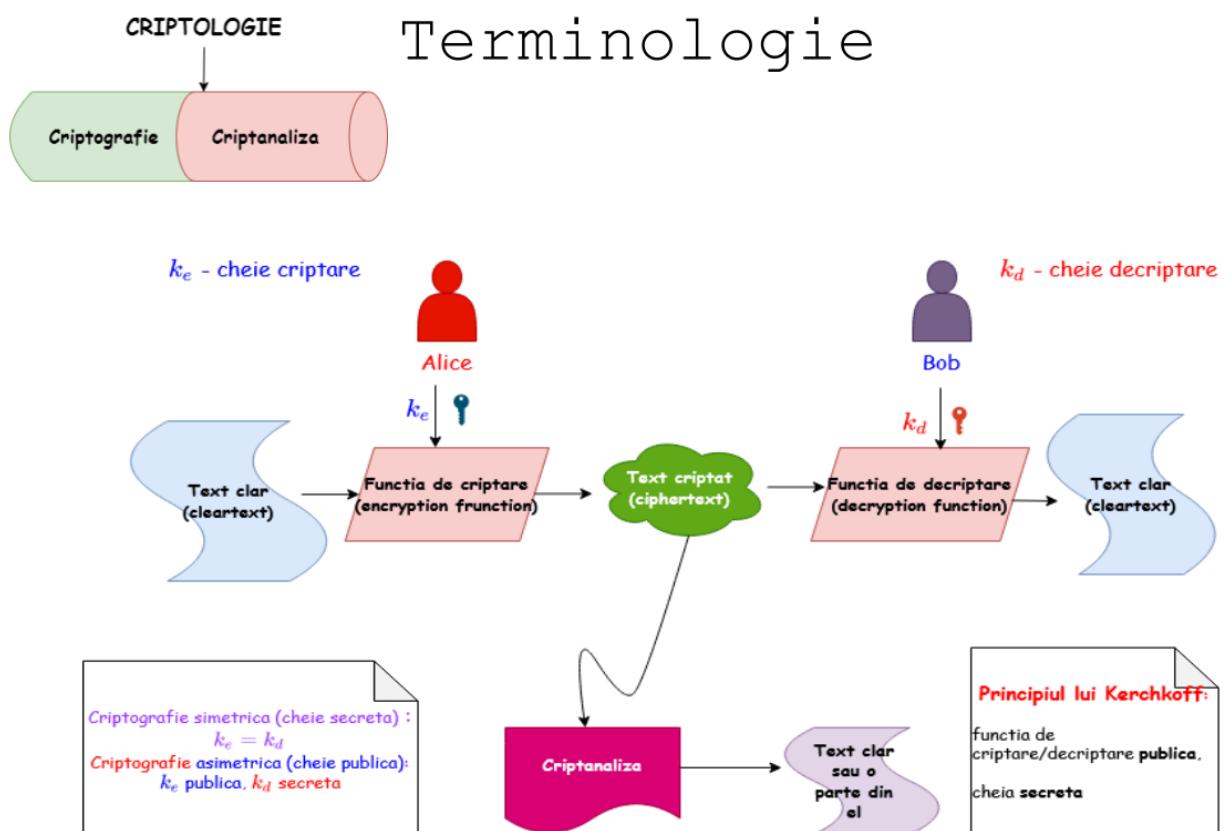
Principiul lui Kerkoff: doar cheia este secreta, constructia (algoritmul) e publică

Principiul separării cheilor: chei diferite pentru scopuri diferite

Principiul diversitatii: foloseste tipuri diferite de algoritmi

## Principiile securitatii

- Alte principii
  - Principiul simplitatii: keep it simple
  - Securitate implicita (security by default): trebuie gandita de la inceput, iar nu adaugata mai tarziu
  - Principiul increderii minime (principle of minimal trust): minimizarea numarului de entitati carora le acordam incredere
  - Principiul celei mai slabe verigi (principle of the weakest link): securitatea unui sistem este data de punctul sau cel mai slab
  - Principiul celui mai mic privilegiu (least privilege): se acorda exact privilegiul necesar pentru efectuarea unei activitati
  - Principiul modularitatii: totul trebuie pastrat modular
  - Defence in depth – securitate la diverse nivele



- Criptografie: cum folosim  $ke$  si  $kd$  ca sa asiguram securitatea comunicatiilor pe un canal nesigur
  - Securitate: cum protejeaza calculatorul/sistemul cheilestocate  $ke$  si  $kd$  de diverse atacuri (virusi, viermi etc.)

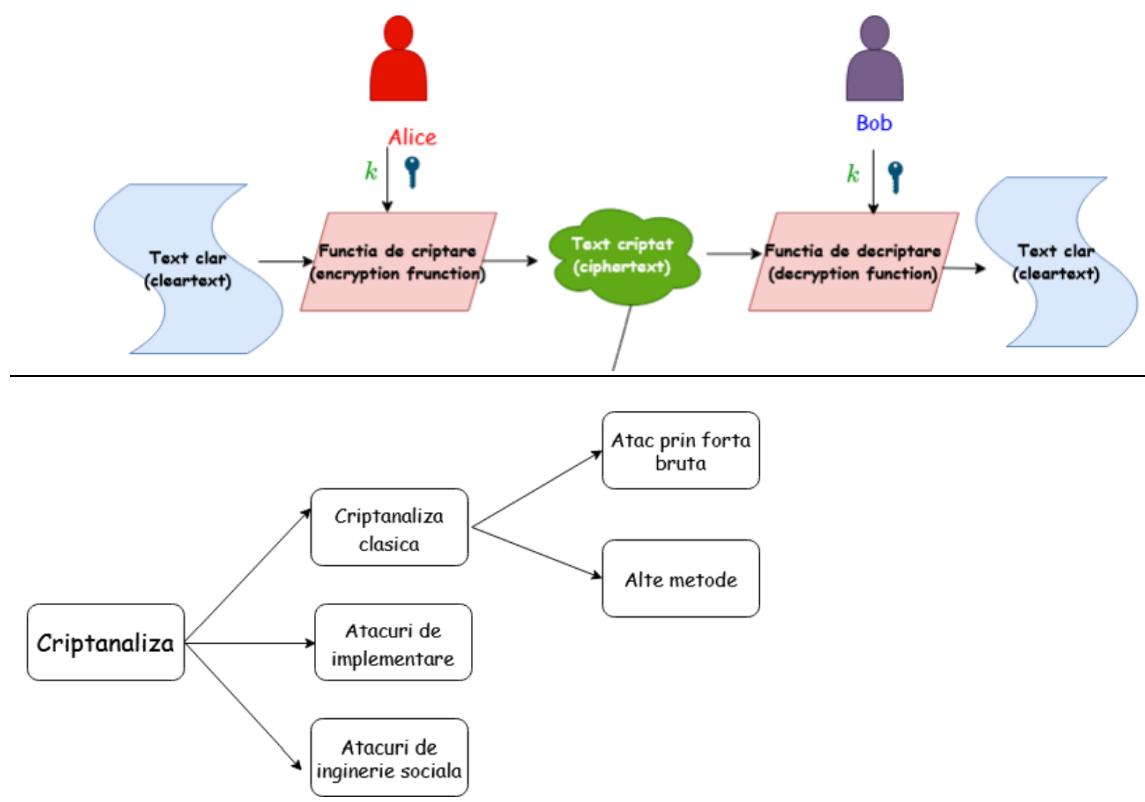
**Ce proprietati de securitate asigura criptografia?**

- Confidentialitatea (mesajelor) - adversarul nu vede sau nu poate obtine mesajul M

- Integritatea (mesajelor) – Alice (sau Bob) trebuie să își dea seama dacă mesajul primit a fost modificat – asigurată de MAC sau semnaturi digitale
- Autentificarea (expeditorului și a mesajului) – Bob trebuie să poată verifica că mesajul provine de la Alice
- Ne-reputierea - Alice nu poate nega că a trimis mesajul lui Bob

## Curs 2 - Sisteme istorice de criptare. Securitate perfectă.

# Criptarea simetrică (cu cheie secreta)



Def: Un cifru de permutare presupune rearanjarea literelor în textul clar pentru a obține textul criptat.

Rail Fence:

<span style="color: red;">→</span>				
<span style="color: red;">↓</span>	M	A	R	T
<span style="color: red;">↓</span>	E	J	I	A

<span style="color: red;">↓</span>	S	C	P	T
------------------------------------	---	---	---	---

**Text clar:** mesaj criptat

**Cheia:**  $k = 3$

**Text criptat:** MARTEJIASCPT

### Cifrul lui cezar:

a	b	c	d	e	f	g	h	i	j	k	l	m
D	E	F	G	H	I	J	K	L	M	N	O	P
n	o	p	q	r	s	t	u	v	w	x	y	z
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Text clar: mesaj criptat

Text criptat: PHVDM FULSWDW

►  $|\mathcal{K}| = 26$

- **atac prin forță brută (căutare exhaustivă):** încercarea, pe rând, a tuturor cheilor posibile până când se obține un text clar cu sens

**Principiul cheilor suficiente:** O schemă sigură de criptare trebuie să aibă un spațiu al cheilor suficient de mare a.î. să nu fie vulnerabilă la căutarea exhaustivă.

### Substitutia simplă:

a	b	c	d	e	f	g	h	i	j	k	l	m
F	I	L	O	R	U	X	A	D	G	J	M	P
n	o	p	q	r	s	t	u	v	w	x	y	z
S	V	Y	B	E	H	K	N	Q	T	W	Z	C

Text clar: mesaj criptat

Text criptat: PRHFG LEDYKFK

►  $|\mathcal{K}| = 26!$

- atacul prin forță brută devine mai dificil
- **analiza de frecvență:** determinare corespondenței între alfabetul clar și alfabetul criptat pe baza frecvenței de apariție a literelor în text, cunoscând distribuția literelor în limba textului clar
- se cunoaște limba textului clar
  - lungimea textului permite analiza de frecvență

### Criptografia modernă

- Se bazează pe 3 principii moderne

Principiul 1 – orice problema criptografică necesită o definitie clasică și riguroasă – discutat în Curs 1  
(scheme construite după acest principiu sunt folosite azi în TLS, SSH, IPSec)

Principiul 2 – securitatea primitivelor criptografice se bazează pe presupuse clare de securitate

Principiul 3 – orice construcție criptografică trebuie să fie insotită de o demonstrație de securitate conform principiilor anterioare

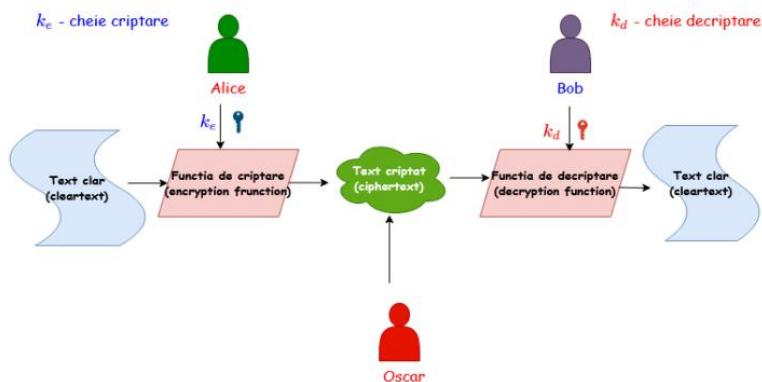
### Principiul 2 – presupuse (ipoteze) de securitate

- majoritatea constructiilor criptografice moderne nu pot fi demonstreate ca fiind sigure neconditonal
- ipotezele de securitate trebuie sa fie explicite:
  - permit cercetatorilor sa valideze aceste ipoteze
  - permit comparatia intre doua scheme bazate pe ipoteze diferite de securitate
  - implicatii practice in cazul unor erori aparute in cadrul ipotezei de securitate
  - necesare pentru demonstratiile de securitate

Exemplu de ipoteza de securitate(problema dificila)

- Factorizarea numerelor mari
  - Se da un numar compus N si se cere descompunerea lui in factori primi. Expl:  $85 = 17 * 5$
  - Astazi nu se cunoaste nici un algoritm care sa factorizeze un numar de 400 cifre intr-un timp practic
  - Totusi:
    1. Un algoritm mai rapid ar putea exista
    2. Un calculator cuantic factorizeaza rapid (inca nu a fost construit dar se fac eforturi in acest sens)
    3. Criptografia "post-cuantica" este in plina dezvoltare –competitia de standardizare post-cuantica NIST, criptografia bazata pe latici etc.

## Securitate perfecta (Shannon 1949)



**Ipoteza: Oscar cunoaste distributia peste M**

**Securitate perfectă:**

1. daca **Oscar** afla textul criptat nu are nici un fel de informatie in plus decat daca nu l-ar fi aflat (nu schimba ceea ce stie el despre distributia peste M).  
(textul criptat nu ofera nici un fel de informatie despre textul clar)
2. Oscar nu poate ghici care din doua posibile mesaje clare a fost criptat, doar vazand textul criptat

**Definiție**

O schemă de criptare peste un spațiu al mesajelor  $\mathcal{M}$  este perfectă 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:

$$Pr[M = m | C = c] = Pr[M = m]$$

- ▶  $Pr[M = m]$  - probabilitatea *a priori* ca Alice să aleagă mesajul  $m$ ;
- ▶  $Pr[M = m | C = c]$  - probabilitatea *a posteriori* ca Alice să aleagă mesajul  $m$ , chiar dacă textul criptat  $c$  a fost văzut ;
- ▶ **securitate perfectă** - dacă Oscar afla textul criptat nu are nici un fel de informatie în plus decât dacă nu l-ar fi aflat.

## Definitie

O schemă de criptare ( $Enc, Dec$ ) este perfect sigură dacă pentru orice mesaje  $m_0, m_1 \in \mathcal{M}$  cu  $|m_0| = |m_1|$  și  $\forall c \in \mathcal{C}$  următoarea egalitate este îndeplinită:

$$\Pr[Enc_k(m_0) = c] = \Pr[Enc_k(m_1) = c]$$

unde  $k \in \mathcal{K}$  este o cheie aleasă uniform.

- ▶ fiind dat un text criptat, este imposibil de ghicit dacă textul clar este  $m_0$  sau  $m_1$
- ▶ cel mai puternic adversar nu poate deduce nimic despre textul clar dat fiind textul criptat

## One Time Pad (OTP)

mesaj clar:	0	1	1	0	0	1	1	1	1	⊕
cheie:	1	0	1	1	0	0	1	1	0	
text criptat:	1	1	0	1	0	1	0	0	1	

- ▶ avantaj - criptare și decriptare rapide
- ▶ dezavantaj - cheia foarte lungă (la fel de lungă precum textul clar)

mesaj clar:	0	1	1	0	0	1	1	1	1	⊕
cheie:	1	0	1	1	0	0	1	1	0	
text criptat:	1	1	0	1	0	1	0	0	1	

mesaj clar:	1	1	0	0	0	0	1	1	0	⊕
cheie:	0	0	0	1	0	1	1	1	1	
text criptat:	1	1	0	1	0	1	0	0	1	

- ▶ Același text criptat poate să provină din orice text clar cu o cheie potrivită
- ▶ Dacă adversarul nu știe decât textul criptat, atunci nu știe nimic despre textul clar!

- Securitatea perfectă nu este imposibila dar..

- ☒ cheia trebuie să fie la fel de lungă precum mesajul

- ☒ inconveniente practice (stocare, transmisie)

- ☒ cheia trebuie să fie folosita o singura data – one time pad

- Exercitiu: Ce se întâmplă dacă folosim o același cheie de două ori cu sistemul OTP ?

- Dacă un adversar obține + - xor

$C = M + K$  și  $C' = M' + K$

- atunci el poate calcula

$C + C' = (M + K) + (M' + K) = (M + M')$

ceea ce invalidează proprietatea de securitate perfectă

# Limitarile securitatii perfecte

## Teorema

Fie o schemă ( $Enc, Dec$ ) de criptare perfect sigură peste un spatiu al mesajelor  $\mathcal{M}$  și un spațiu al cheilor  $\mathcal{K}$ . Atunci  $|\mathcal{K}| \geq |\mathcal{M}|$ .

Sau altfel spus

## Teorema

Nu există nici o schemă de criptare ( $Enc, Dec$ ) perfect sigură în care mesajele au lungimea  $n$  biți iar cheile au lungimea (cel mult)  $n - 1$  biți.

## Curs 3-Securitate computationala si aleatorism

### Schema de criptare OTP este perfecta sigura.

Majoritatea constructiilor criptografice moderne → securitate computationala;

Schemele moderne pot fi sparte daca un atacator are la dispozitie suficient spatiu si putere de calcul.

### Securitate perfecta vs. Criptografie computationala

- ▶ Securitatea computationala mai slaba decat securitatea informational-teoretica;
- ▶ Prima se bazeaza pe presupusiile de securitate; a doua este neconditionata;
- ▶ Intrebare: de ce renuntam la securitatea perfecta?
- ▶ Raspuns: datorita limitarilor practice!
- ▶ Preferam un compromis de securitate pentru a obtine constructii practice

### Securitate computationala

- ▶ Ideea de baza: principiul 1 al lui Kerckhoffs

Un cifru trebuie sa fie practic, daca nu matematic, indescifrabil.

- ▶ Sunt de interes mai mare schemele care practic nu pot fi sparte desi nu beneficiaza de securitate perfecta;

#### 1. Adversari limitati computational/eficienti/timp polinomial

Exemplu: Un atacator care realizeaza un atac prin forta bruta peste spatiul cheilor si testeaza o cheie/ciclu de ceas

- ▶ calculator desktop - se pot testa aprox. 257 chei/an

- ▶ supercalculator - se pot testa aprox. 280 chei/an

- ▶ supercalculator, varsta universului - 2112 chei

#### 2. Adversarii pot efectua un atac cu succes cu o probabilitate foarte mica;

Exemplu: un adversar afla textul clar cu probabilitate  $2^{-60}$  ^intr-un an

### Indistinctibilitate perfecta

- ▶ Pentru securitatea perfecta am dat doua definitii echivalente, a doua sublinia ideea de indistinctibilitate:

adversarul nu poate distinge intre criptarile a doua mesaje diferite

- ▶ Vom defini indistinctibilitatea pe baza unui experiment

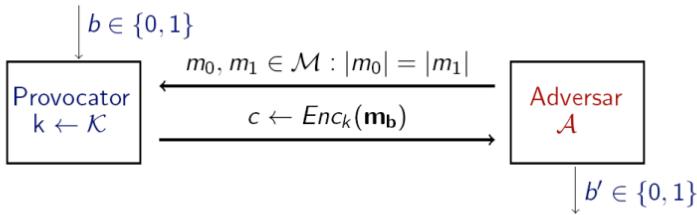
$$Priv_{A,\pi}^{eav}$$

unde  $\pi = (\text{Enc}, \text{Dec})$  este schema de criptare

- ▶ Personaje participante: adversarul A care incearca sa spargă schema si un provocator (challenger).

▶ Trebuie sa definim capacitatile adversarului: el poate vedea un singur text criptat cu o anume cheie, fiind un adversary pasiv care poate rula atacuri in timp polinomial, si nu are nici o alta interacțiune cu Alice sau Bob

## Experimentul $Priv_{\mathcal{A}, \pi}^{eav}$



- ▶ Output-ul experimentului este 1 dacă  $b' = b$  și 0 altfel. Dacă  $Priv_{\mathcal{A}, \pi}^{eav} = 1$ , spunem că  $\mathcal{A}$  a efectuat experimentul cu succes.
- ▶ Schema  $\pi$  este *perfect indistinctibilă* dacă

$$Pr[Priv_{\mathcal{A}, \pi}^{eav}(n) = 1] = \frac{1}{2}$$

- ▶ Reamintim ca *indiscreționalitatea perfectă* este doar o definitie alternativa pentru *securitatea perfectă*

O schemă de criptare este  $(t, \epsilon)$ -indistinctibila dacă orice adversar care rulează în timp cel mult  $t$

$$Pr[Priv_{\mathcal{A}, \pi}^{eav} = 1] \leq \frac{1}{2} + \epsilon$$

- ▶ probabilitatea de succes a adversarului  $\leq \epsilon$
- ▶ adversarul ruleaza in timp  $\leq t$
- ▶ dezavantaj: am dori să avem scheme in care utilizatorul își poată ajusta nivelul de securitate dorit

## Neglijabil și ne-neglijabil

- ▶ **în practică:**  $\epsilon$  este scalar și
  - ▶  $\epsilon$  ne-neglijabil dacă  $\epsilon \geq 1/2^{30}$
  - ▶  $\epsilon$  neglijabil dacă  $\epsilon \leq 1/2^{80}$
- ▶ **în teorie:**  $\epsilon$  este funcție  $\epsilon : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  și  $p(n)$  este o funcție polinomială în  $n$  (ex.:  $p(n) = n^d$ ,  $d$  constantă)
  - ▶  $\epsilon$  ne-neglijabilă în  $n$  dacă  $\exists p(n) : \epsilon(n) > 1/p(n)$
  - ▶  $\epsilon$  neglijabilă în  $n$  dacă  $\forall p(n), \exists n_d$  a.î.  $\forall n \geq n_d : \epsilon(n) \leq 1/p(n)$

### Definiție

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

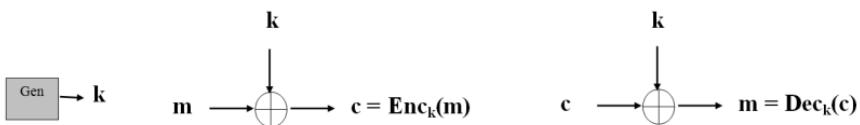
- ▶  $\mathcal{K}$  = spațiul cheilor
- ▶  $\mathcal{M}$  = spațiul textelor clare (mesaje)
- ▶  $\mathcal{C}$  = spațiul textelor criptate

este un triplet  $(Gen, Enc, Dec)$ , unde:

1. **Gen( $1^n$ )**: este algoritm probabilistic de generare a cheilor care întoarce o cheie  $k$  conform unei distribuții
2. **Enc**: primește o cheie  $k$  și un mesaj  $m \in \{0,1\}^*$  și întoarce  $c \leftarrow Enc_k(m)$
3. **Dec**: primește cheia  $k$  și textul criptat și întoarce  $m$  sau "eroare".

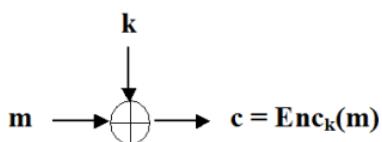
## Aleatorism

- ▶ Încercăm criptare în stil OTP: cheia va masca mesajul dar
  - ▶ masca nu va fi doar cheia ci masca =  $f(\text{cheie})$  unde  $f$  este o funcție de extindere a cheii
  - ▶ pentru securitate perfectă masca trebuie să fie perfect aleatoare
  - ▶ pentru securitate computațională, este suficient ca masca să para aleatoare pentru un adversar PPT chiar dacă nu este
- ▶ Vom avea nevoie întâi să definim noțiunea de *generatoare de numere pseudoaleatoare* ca element important de construcție pentru schemele de criptare simetrice

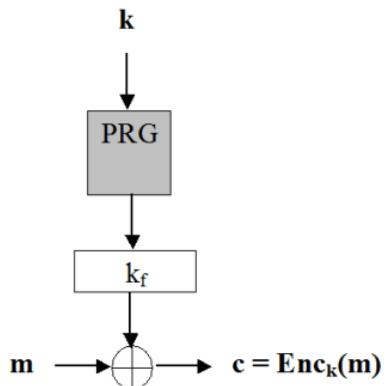


pseudoaleatorismul este o relaxare a aleatorismului perfect astă cum securitatea computatională este o relaxare a securității perfecte

### OTP (One Time Pad)



### Sistem de cripare



PRG (PseudoRandom Generator);

- ▶ Aceasta este un algoritm determinist care primește o semantă relativ scurtă  $s$  (seed) și generează o secvență pseudoaleatoare de biti;
- ▶ Notam  $|s| = n$ ,  $|\text{PRG}(s)| = l(n)$
- ▶ PRG prezintă interes dacă:  $l(n) \geq n$  (altfel NU generează aleatorism)

Notări

- ▶ D = Distinguisher
- ▶ PPT = Probabilistic Polynomial Time
- ▶  $x \leftarrow R X = x$  este ales uniform aleator din  $X$
- ▶  $\text{negl}(n) =$  o funcție neglijabilă în (parametrul de securitate)  $n$
- ^În plus:
- ▶ Vom nota  $A$  un adversar (Oscar / Eve), care (în general) are putere polinomială de calcul

- ▶ Consideram următorul PRG:  $G(s) = s || \oplus_{i=1}^n s_i$
- ▶ factorul de expansiune  $l(n) = n + 1$
- ▶ Consideram algoritmul D astfel:  $D(w) = 1$  dacă și numai dacă ultimul bit al lui  $w$  este egal cu xor-ul tuturor biților precedenți
- ▶ Se verifică ușor că  $\Pr[D(G(s)) = 1] = 1$
- ▶ Dacă  $r$  este uniform, atunci bitul final al lui  $r$  este uniform și deci  $\Pr[D(r) = 1] = \frac{1}{2}$
- ▶  $|\frac{1}{2} - 1|$  nu e neglijabilă și deci  $G$  nu este PRG

## Observații

- ▶ Distributia output-ului unui PRG este departe de a fi uniformă
- ▶ Exemplificăm pentru un  $G$  care dublează lungimea intrării i.e.  $l(n) = 2n$
- ▶ Pentru distributia uniformă peste  $\{0, 1\}^{2n}$ , fiecare din cele  $2^{2n}$  este ales cu probabilitate ...
- ▶ ...  $\frac{1}{2^{2n}}$
- ▶ Considerăm distributia output-ului lui  $G$  când primește la intrare un sir uniform de lungime  $n$
- ▶ Numarul de siruri diferite din codomeniul lui  $G$  este cel mult ...
- ▶ ...  $2^n$
- ▶ Probabilitatea ca un sir de lungime  $2n$  să fie output al lui  $G$  este  $2^n / 2^{2n} = 1/2^n$

Obs:

- ▶ Seed-ul unui PRG este analogul cheii unui sistem de criptare
- ▶ seed-ul trebuie ales uniform și menținut secret
- ▶ seed-ul trebuie să fie suficient de lung astă incat un atac prin forță bruta să nu fie fezabil

## Definiție

*Un sistem de criptare  $(Enc, Dec)$  definit peste  $(\mathcal{K}, \mathcal{M}, \mathcal{C})$  se numește **sistem de criptare bazat pe PRG** dacă:*

1.  $Enc : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$

$$c = Enc_k(m) = G(k) \oplus m$$

2.  $Dec : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

$$m = Dec_k(c) = G(k) \oplus c$$

*unde  $G$  este un generator de numere pseudoaleatoare cu factorul de expansiune  $l$ ,  $k \in \{0, 1\}^n$ ,  $m \in \{0, 1\}^{l(n)}$*

- ▶ OTP este perfect sigur;

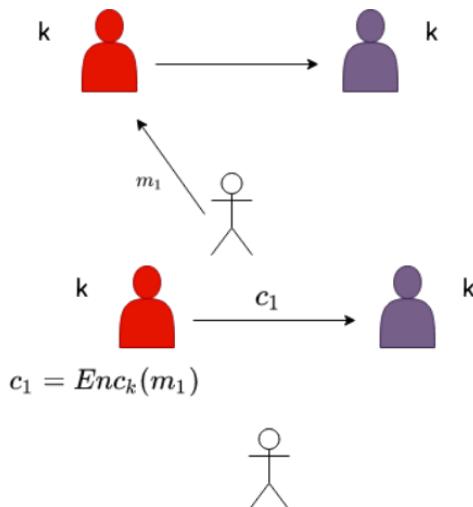
- ▶ Criptarea bazata pe PRG se obtine din OTP prin inlocuirea pad cu  $G(k)$ ;
- ▶ Daca  $G$  este PRG, atunci pad si  $G(k)$  sunt indistinctibile pentru orice  $A$  adversar PPT;
- ▶ In concluzie, OTP si sistemul de criptare bazat pe PRG sunt indistinctibile pentru  $A$ .

#### **Curs 4: Notiuni de securitate mai puternice, Sisteme fluide - stream ciphers, Sisteme de criptare bloc**

- ▶ Reamintim cateva dintre scenariile de atac pe care le-am mai intalnit:
- ▶ Atac cu text criptat: Atacatorul stie doar textul criptat – poate încerca un atac prin forta bruta prin care se parcureg toate cheile pana se gaseste cea corecta;
- ▶ Atac cu text clar: Atacatorul cunoaste una sau mai multe perechi (text clar, text criptat);
- ▶ Atac cu text clar alese: Atacatorul poate obtine criptarea unor texte clare alese de el;
- ▶ Atac cu text criptat alese: Atacatorul are posibilitatea sa obtina decriptarea unor texte criptate alese de el.

### Securitate CPA

- ▶ **CPA (Chosen-Plaintext Attack)**: adversarul poate să obțină criptarea unor mesaje alese de el;



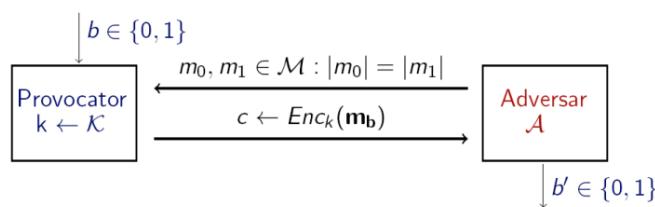
adversarul poate cere criptarea unor mesaje alese de el repetitiv (polinomial de multe ori)

mai tarziu adversarul observa criptarea unui mesaj necunoscut

dorim ca adversarul sa nu afle nici un fel de informatie despre mesajul  $m$

- ▶ Capabilitatile adversarului: el poate interactiona cu un oracol de criptare, fiind un adversar activ care poate rula atacuri în timp polinomial;
- ▶ Adversarul poate transmite catre oracol orice mesaj  $m$  și primește înapoi textul criptat corespunzător;
- ▶ Daca sistemul de criptare este nedeterminist, atunci oracolul folosește de fiecare data o valoare aleatoare nouă și neutilizată anterior.

### Experimentul $\text{Priv}_{\mathcal{A}, \pi}^{\text{CPA}}(n)$



- ▶ Pe toată durata experimentului,  $\mathcal{A}$  are acces la oracolul de criptare  $\text{Enc}_k(\cdot)$ !

### Definiție

O schemă de criptare  $\pi = (Enc, Dec)$  este **CPA-sigură** dacă pentru orice adversar PPT  $\mathcal{A}$  există o funcție neglijabilă  $negl$  așa încât

$$\Pr[\mathsf{Priv}_{\mathcal{A}, \pi}^{\mathsf{cpa}}(n) = 1] \leq \frac{1}{2} + negl(n).$$

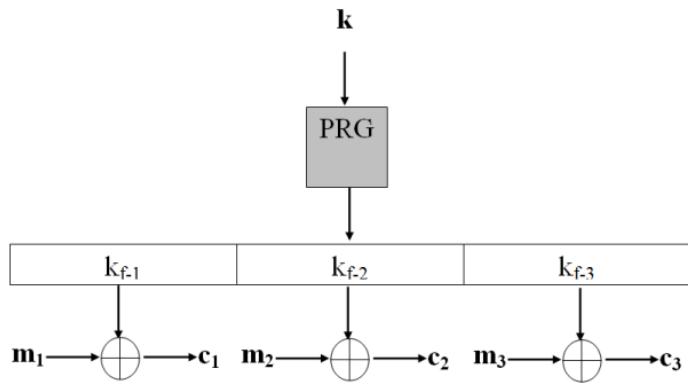
- ▶ Un adversar nu poate determina care text clar a fost criptat cu o probabilitate semnificativ mai mare decât dacă ar fi ghicit (în sens aleator, dat cu banul), chiar dacă are acces la oracolul de criptare.
- ▶ **Întrebare:** Un sistem de criptare CPA-sigur are întotdeauna proprietatea de indistinctibilitate?
- ▶ **Răspuns:** DA! Experimental  $\mathsf{Priv}_{\mathcal{A}, \pi}^{\mathsf{eav}}(n)$  este  $\mathsf{Priv}_{\mathcal{A}, \pi}^{\mathsf{cpa}}(n)$  în care  $\mathcal{A}$  nu folosește oracolul de criptare.
- ▶ **Întrebare:** Un sistem de criptare determinist poate fi CPA-sigur?
- ▶ **Răspuns:** NU! Adversarul cere oracolului criptarea mesajului  $m_0$ . Dacă textul criptat este egal cu  $c$ , atunci  $b' = 0$ , altfel  $b' = 1$ . În concluzie,  $\mathcal{A}$  câștigă cu probabilitate 1.

### Sisteme fluide

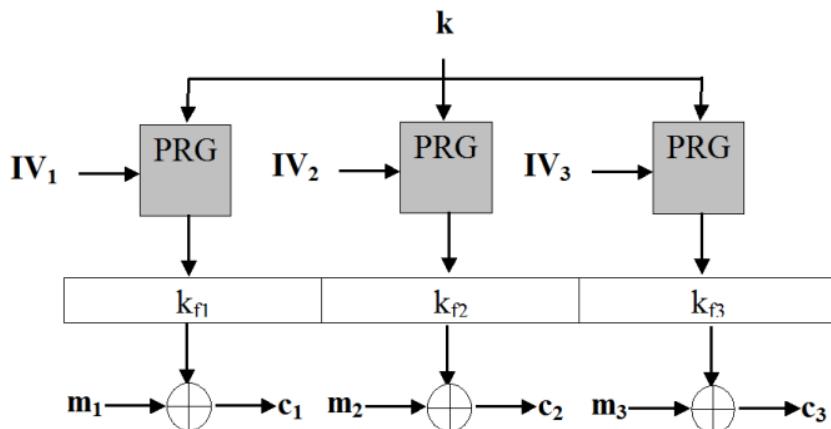
- ▶ sistemele fluide produc biți de output (pseudo-aleatori) gradual și la cerere, fiind mai **eficiente și flexibile**
- ▶ criptarea cu un sistem fluid presupune 2 faze:
  - ▶ **Faza 1:** se generează o secvență pseudoaleatoare de biți, folosind un **generator de numere pseudoaleatoare (PRG)**
  - ▶ **Faza 2:** secvența obținută se XOR-ează cu mesajul clar
- ▶ **Atenție!** De multe ori când ne referim la un sistem de criptare fluid considerăm doar Faza 1

### Securitate - interceptare multipla

- ▶ Un sistem de criptare fluid în varianta prezentată este determinist: unui text clar îi corespunde întotdeauna același mesaj criptat;
- ▶ În consecință, utilizarea unui sistem fluid în forma prezentată pentru criptarea mai multor mesaje (cu aceeași cheie) este nesigură;
- ▶ Un sistem de criptare fluid se folosește în practică în 2 moduri: sincronizat și nesincronizat.
- ▶ **modul sincronizat:** partenerii de comunicație folosesc pentru criptarea mesajelor parti succeseive ale secvenței pseudoaleatoare generate;



► **modul nesincronizat:** partenerii de comunicatie folosesc pentru criptarea mesajelor secvente pseudoaleatoare diferite.



### Modul sincronizat

- ▶ mesajele sunt criptate în mod **succesiv** (participanții trebuie să știe care părți au fost deja folosite)
- ▶ necesită **păstrarea** stării
- ▶ mesajele succeseive pot fi percepute ca un *singur mesaj clar lung*, obținut prin concatenarea measajelor succeseive
- ▶ se pretează unei singure sesiuni de comunicații

### Modul nesincronizat

- ▶ mesajele sunt criptate în mod **independent**
- ▶ NU necesită **păstrarea** stării
- ▶ valorile  $IV_1, IV_2, \dots$  sunt alese uniform aleator pentru fiecare mesaj transmis
- ▶ valorile  $IV_1, IV_2, \dots$  (dar și  $IV$  în modul sincronizat) fac parte din mesajul criptat (sunt necesare pentru decriptare)

Fie  $G(s, IV)$  un PRG cu 2 intrari:

▶  $s =$  seed

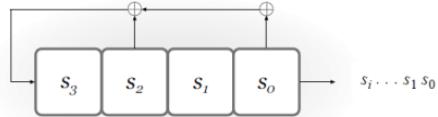
▶  $IV =$  Initialization Vector

PRG trebuie sa se satisfaca (cel putin):

1.  $G(s, IV)$  este o secventa pseudoaleatoare chiar daca  $IV$  este public (i.e. securitatea lui  $G$  consta in securitatea lui  $s$ );

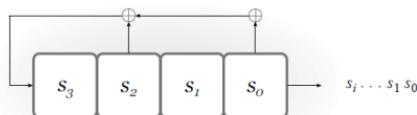
2. daca  $IV_1$  si  $IV_2$  sunt valori uniform aleatoare, atunci  $G(s, IV_1)$  si  $G(s, IV_2)$  sunt indistinctibile

### Linear-Feedback Shift Registers (LFSR)



In exemplul de mai sus avem

- ▶  $c_0 = c_2 = 1$  și  $c_1 = c_3 = 0$
- ▶ fiecare bit de la ieșire este calculat după formula  
 $c_0s_0 \oplus \dots \oplus c_3s_3$
- ▶ la fiecare tact de ceas, LFSR scoate la ieșire valoarea din registrul  $s_0$  iar valorile din ceilalți registri sunt deplasate la dreapta cu o poziție



Pentru starea inițială (0,0,1,1), biții de la ieșire vor fi ...

(0,0,1,1)  
(1,0,0,1)

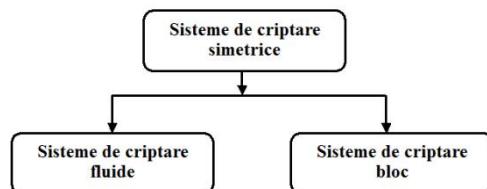
...

In general

- ▶ starea LFSR constă din  $n$  biți (conținutul regiștrilor la un moment dat)
- ▶ există cel mult  $2^n$  stări posibile până când output-ul LFSR-ului se repetă

Vulnerabilitati LFSR ▶ LFSR-urile sunt liniare iar liniaritatea induce vulnerabilitati (sistemele liniare de ecuatii permit aflarea informatiilor sensibile) ▶ Insa combinatiile de mai multe LFSR-uri pot produce sisteme de criptare sigure

- ▶ Am studiat sisteme simetrice care criptează **bit cu bit** - **sisteme de criptare fluide**;
- ▶ Vom studia sisteme simetrice care criptează **câte n biți simulan** - **sisteme de criptare bloc**;



... d.p.d.v. al modului de criptare:

#### Sisteme fluide

- ▶ criptarea biților se realizează **individual**
- ▶ criptarea unui bit din textul clar este **independentă** de orice alt bit din textul clar

#### Sisteme bloc

- ▶ criptarea se realizează în **blocuri** de câte  $n$  biți
- ▶ criptarea unui bit din textul clar este **dependentă** de biții din textul clar care aparțin aceluiași bloc

### Sisteme fluide

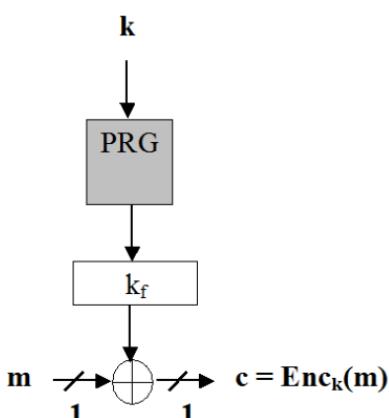
- ▶ necesități computaționale reduse
- ▶ utilizare: telefoane mobile, dispozitive încorporate, PDA
- ▶ par să fie mai puțin sigure, multe sunt sparte

### Sisteme bloc

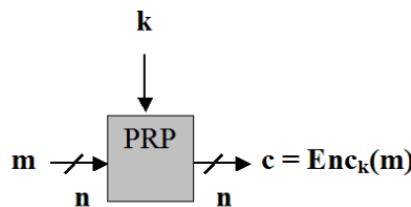
- ▶ necesități computaționale mai avansate
- ▶ utilizare: internet
- ▶ par să fie mai sigure, prezintă încredere mai mare

► Introducem noțiunea de permutare pseudoaleatoare sau PRP(PseudoRandom Permutation) ► În analogie cu ce stim deja: ► PRP sunt necesare pentru construcția sistemelor bloc asa cum ► PRG sunt necesare pentru construcția sistemelor fluide

### Sisteme fluide



### Sisteme bloc



PRP (PseudoRandom Permutation); ► Aceasta este o funcție **determinista și bijectiva** care pentru o cheie fixată produce la ieșire o **permutare** a intrării ... ► ... **indistinctibila** față de o permutare aleatoare; ► În plus, atât funcția cat și inversa sa sunt **eficient calculabile**.

funcție pseudoaleatoare sau PRF (PseudoRandom Function)... ► ... ca o generalizare a noțiunii de permutare pseudoaleatoare; ► Aceasta este o funcție cu cheie care este indistinctibila față de o funcție aleatoare (cu același domeniu și multime de valori).

$$PRP \subseteq PRF$$

► **Întrebare:** De ce PRF poate fi privită ca o generalizare a PRP?

► **Răspuns:** PRP este PRF care satisface:

1.  $\mathcal{X} = \mathcal{Y}$
2. este inversabilă
3. calculul funcției inverse este eficient

► PRF  $\Rightarrow$  PRG

Pornind de la PRF se poate construi PRG

► PRG  $\Rightarrow$  PRF

Pornind de la PRG se poate construi PRF

► PRP  $\Rightarrow$  PRF

Pornind de la PRP se poate construi PRF

► PRF  $\Rightarrow$  PRP

Pornind de la PRF se poate construi PRP

Intrebare: Care dintre aceste constructii este triviala?

PRP  $\Rightarrow$  PRF

► Intrebare: Ce se intampla daca lungimea mesajului clar este mai mica decat dimensiunea unui bloc?

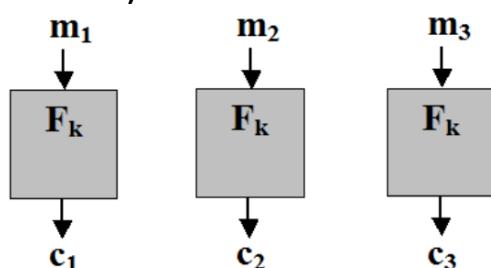
► Raspuns: Se completeaza cu biti: 1 0 . . . 0;

► Intrebare: Ce se intampla daca lungimea mesajului clar este mai mare decat lungimea unui bloc?

► Raspuns: Se utilizeaza un mod de operare (ECB, CBC, OFB, CTR);

► Notam cu  $F_k$  un sistem de criptare bloc (i.e. PRP) cu cheia  $k$  fixata.

### ECB (Electronic Code Book)



► Pare modul cel mai natural de a cripta mai multe blocuri;

► Pentru decriptare,  $F_k$  trebuie sa fie inversabila;

► Este paralelizabil;

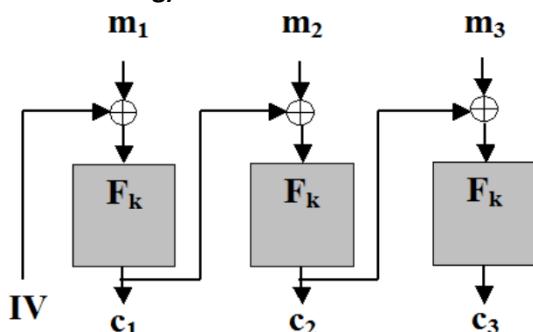
► Este determinist, deci este nesigur;

► Intrebare: Ce informatii poate sa ofere modul de criptare ECB unui adversar pasiv?

► Raspuns: Un adversar pasiv detecteaza repetarea unui bloc de text clar pentru ca se repeta blocul criptat corespunzator;

► Modul ECB NU trebuie utilizat in practica!

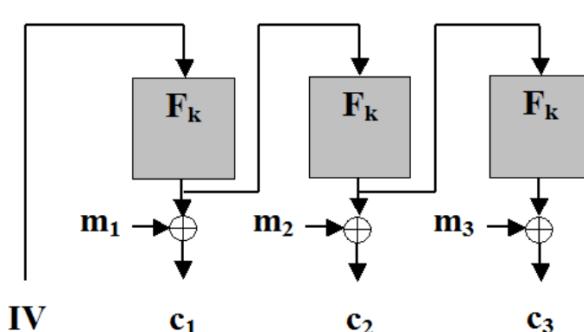
### CBC (Cipher Block Chaining)



► IV este ales in mod aleator la criptare; ► IV se transmite in clar pentru ca este necesar la decriptare;

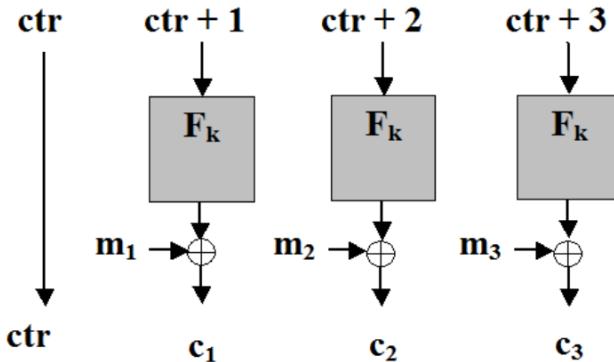
► Pentru decriptare,  $F_k$  trebuie sa fie inversabila; ► Este secvential, un dezavantaj major daca se poate utiliza procesarea paralela.

### OFB (Output FeedBack)



- Genereaza o secventa pseudoaleatoare care se XOR-eaza mesajului clar; ► IV este o ales în mod aleator la criptare; ► IV se transmite în clar pentru ca este necesar la decriptare; ►  $F_k$  nu trebuie neapărat să fie inversabila; ► Este secvential, însă secventa pseudoaleatoare poate fi pre-procesată anterior decriptării.

### CTR (Counter)



- Genereaza o secventa pseudoaleatoare care se XOR-eaza mesajului clar; ►  $ctr$  este o ales în mod aleator la criptare; ►  $ctr$  se transmite în clar pentru ca este necesar la decriptare; ►  $F_k$  nu trebuie neapărat să fie inversabila; ► Este paralelizabil; ► În plus, secvența pseudoaleatoare poate fi pre-procesată anterior decriptării.

- Generează o secvență pseudoaleatoare care se XOR-ează mesajului clar;
- $ctr$  este o ales în mod aleator la criptare și se transmite în clar pentru ca este necesar la decriptare;
- Este **paralelizabil**;  $F_k$  nu trebuie neapărat să fie inversabilă;
- În plus, secvența pseudoaleatoare poate fi pre-procesată anterior decriptării.
- CTR poate fi văzut și ca un sistem fluid nesincronizat:
  - pentru criptarea unui mesaj de lungime  $l < 2^{n/4}$  blocuri, se alege un IV uniform din  $\{0, 1\}^{2n/4}$
  - fiecare bloc de text criptat este calculat  $y_i = F_k(IV||i)$  unde  $i$  este codificat ca un string pe  $n/4$  biți

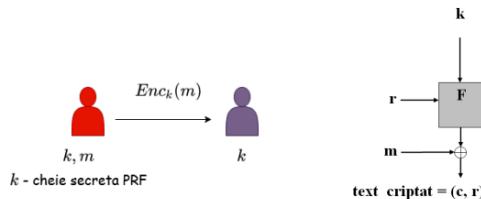
!!!!

- modurile CTR, OFB și CBC sunt CPA-signure
- modurile CBC, OFB și CTR folosesc un IV uniform aleator - asigură faptul că  $F_k$  este mereu evaluat pe intrări diferite (previne situația în care adversarul află informații la vederea de intrări identice)
- CTR - IV ales uniform de lungime  $3n/4$  înseamnă că IV se repetă după criptarea aprox.  $q(n) = 2^{2n/8}$  mesaje
- Dacă  $n = 64$  atunci  $q \approx 17.000.000$  ceea ce e puțin pentru zilele noastre
- Dacă  $n = 128$  și vrem să folosim CTR având garanția că IV se repetă cu probabilitate cel mult  $2^{-40}$ , rezultă  $q \approx 2^{28}$  mesaje (calculand  $q$  din  $\frac{q^2}{2^{3n/4}+1} \leq 2^{-40}$ )

- Ce se întampla dacă IV se repeta? ► Pentru modurile OFB și CTR, întregul stream pseudoaleator (cu care se face xor pe mesaj) se repeta ► Dacă IV nu este uniform aleator (deci este predictibil), CTR este sigur dar CBC nu este sigur.

## Curs 5 - Scheme de criptare CPA-sigure bazate pe PRF, Securitate CCA, Constructii practice PRF

- Sistemele de criptare bloc sunt instanțieri sigure ale PRP



- Fie  $F_k$  o funcție cu cheie
- Gen( $1^n$ ): alege uniform cheie  $k \in \{0, 1\}^n$
- $Enc_k(m)$ : pentru  $|m| = |k|$ , alege  $r$  uniform în  $\{0, 1\}^n$

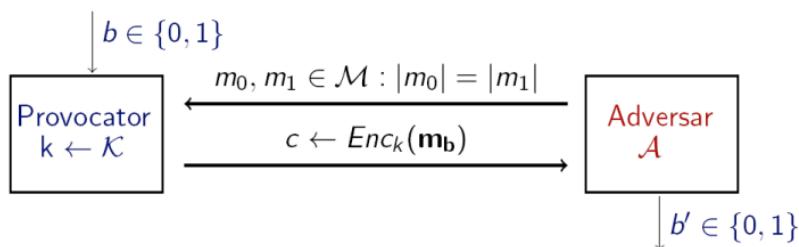
$$Enc_k(m) = (r, F_k(r) \oplus m)$$

►  $Dec_k(c = (c_0, c_1))$ : întoarce  $c_1 \oplus F_k(c_0)$

3 / 7

### CCA

- **CPA (Chosen-Plaintext Attack)**: adversarul poate să obțina criptarea unor mesaje alese de el; - discutată în cursul anterior
- **CCA (Chosen-Ciphertext Attack)**: adversarul poate să obțina criptarea unor mesaje alese de el și decriptarea unor texte criptate alese de el.
- Capabilitățile adversarului: el poate interacționa cu un oracol de criptare și cu un oracol de decriptare, fiind un adversary activ care poate rula atacuri în timp polinomial;
- Adversarul poate transmite către oracolul de criptare orice mesaj  $m$  și primește înapoi textul criptat corespunzător sau poate transmite către oracolul de decriptare anumite mesaje  $c$  și primește înapoi mesajul clar corespunzător;
- Dacă sistemul de criptare este nedeterminist, atunci oracolul de criptare folosește de fiecare dată o valoare aleatoare nouă și neutilizată anterior



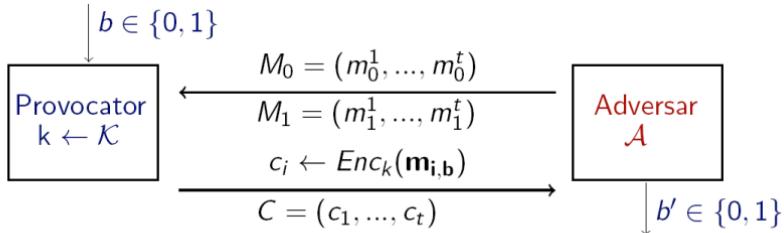
- Output-ul experimentului este 1 dacă  $b' = b$  și 0 altfel. Dacă  $Priv_{A,\pi}^{cca}(n) = 1$ , spunem că  $A$  a efectuat experimentul cu succes.

## Definiție

O schemă de criptare  $\pi = (Enc, Dec)$  este **CCA-sigură** dacă pentru orice adversar PPT  $\mathcal{A}$  există o funcție neglijabilă negl astfel încât

$$\Pr[\text{Priv}_{\mathcal{A}, \pi}^{\text{cca}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

- ▶ Un adversar nu poate determina care text clar a fost criptat cu o probabilitate semnificativ mai mare decât dacă ar fi ghicit (în sens aleator, dat cu banul), chiar dacă are acces la oracolele de criptare și decriptare.
- ▶ Intrebare: Un sistem de criptare CCA-sigur este întotdeauna CPA-sigur?
- ▶ Raspuns: DA! Experimentul  $\text{Priv}_{\mathcal{A}, \pi}$  este  $\text{Priv}_{\mathcal{A}, \pi}$  în care  $\mathcal{A}$  nu folosește oracolul de decriptare.
- ▶ Intrebare: Un sistem de criptare determinist poate fi CCA-sigur?
- ▶ Raspuns: NU! Sistemul nu este CPA-sigur, deci nu poate fi CCA-sigur.
- ▶ În definitia precedenta am considerat cazul unui adversar care primește un singur text criptat; ▶ În realitate, în cadrul unei comunicatii se trimit mai multe mesaje pe care adversarul le poate intercepta; ▶ Definim ce înseamna o schema sigura chiar si in aceste conditii



- ▶ Pe toată durata experimentului,  $\mathcal{A}$  are acces la oracolul de criptare  $\text{Enc}_k(\cdot)$  și la oracolul decriptare  $\text{Dec}_k(\cdot)$  cu restricția că nu poate decripta  $c_1, \dots, c_t$ !

**Nici una din schemele de criptare de pana acum nu sunt CCA-sigure.**

Schemele deterministe nu sunt semantic / CPA / CCA sigure

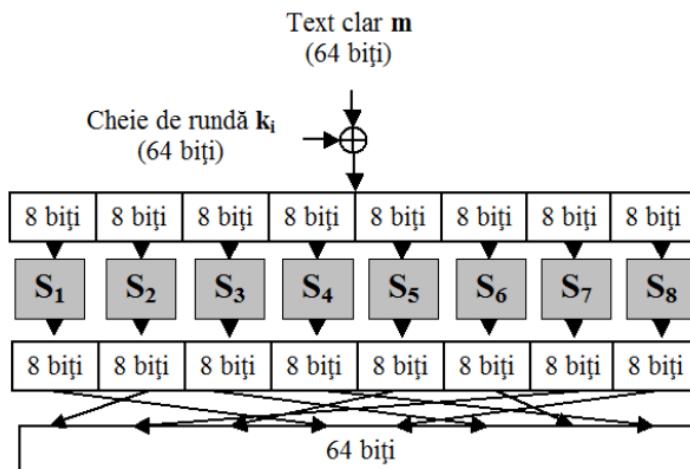
- ▶ Se construiește funcția  $F$ , pe baza mai multor funcții aleatoare  $f_i$  de dimensiune mai mică;
- ▶ Considerăm  $F$  pe 128 biți și 16 funcții aleatoare  $f_1, \dots, f_{16}$  pe câte 8 biți;
- ▶ Pentru  $x = x_1 || \dots || x_{16}$ ,  $x \in \{0, 1\}^{128}$   $x_i \in \{0, 1\}^8$ :

$$F_k(x) = f_1(x_1) || \dots || f_{16}(x_{16})$$

- ▶ Spunem că  $\{f_i\}$  introduc confuzie în  $F$ .

**Retele de substitutie - permutare** F încă nu este PRF dar ▶ F se transformă în PRF în 2 pași: ▶ Pasul 1: se introduce difuzie prin amestecarea (permutearea) bitilor de ieșire; ▶ Pasul 2: se repetă o runda (care presupune confuzie și difuzie) de mai multe ori; ▶ Repetarea confuziei și difuziei face ca modificarea unui singur bit de intrare să fie propagată asupra tuturor bitilor de ieșire;

- ▶ O rețea de **substituție-permutare** este o implementare a construcției anterioare de **confuzie-difuzie** în care funcțiile  $\{f_i\}$  sunt **fixe** (i.e. nu depind de cheie) și se numesc permutări;
- ▶  $\{f_i\}$  se numesc **S-boxes** (Substitution-boxes);
- ▶ Cum nu mai depind de cheie, aceasta este utilizată în alt scop;
- ▶ Din cheie se obțin mai multe **chei de rundă** (*sub-chei*) în urma unui proces de derivare a cheilor (*key schedule*);
- ▶ Fiecare cheie de rundă este XOR-ată cu valorile intermediare din fiecare rundă.



- ▶ Există 2 principii de bază în proiectarea rețelelor de substituție - permutare:
  - ▶ **Principiul 1:** Inversabilitatea S-box-urilor;
    - ▶ dacă toate S-box-urile sunt inversabile, atunci rețeaua este inversabilă;
    - ▶ necesitate funcțională (pentru decriptare)
  - ▶ **Principiul 2:** Efectul de avalanșă
    - ▶ Un singur bit modificat la intrare trebuie să afecteze toți biții din secvența de ieșire;
    - ▶ necesitate de securitate.

#### AES

- ▶ AES este o rețea de substituție - permutare pe 128 biți care poate folosi chei de 128, 192 sau 256 biți;
- ▶ Lungimea cheii determină numărul de runde:

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

- ▶ Folosește o matrice de octeți  $4 \times 4$  numită **stare**;
- ▶ Starea inițială este mesajul clar ( $4 \times 4 \times 8 = 128$ );
- ▶ Starea este modificată pe parcursul rundelor prin 4 tipuri de operații: *AddRoundKey*, *SubBytes*, *ShiftRows*, *MixColumns*;
- ▶ Ieșirea din ultima rundă este textul criptat.

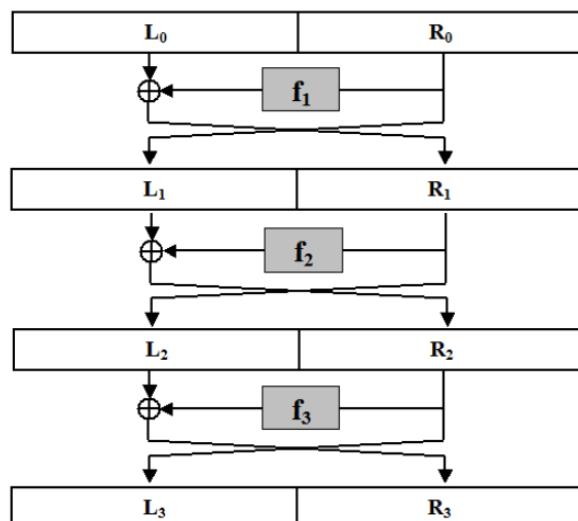
- ▶ Singurele atacuri netriviale sunt asupra AES cu număr redus de runde:

- ▶ AES-128 cu 6 runde: necesită  $2^{72}$  criptări;
- ▶ AES-192 cu 8 runde: necesită  $2^{188}$  criptări;
- ▶ AES-256 cu 8 runde: necesită  $2^{204}$  criptări.

- ▶ Nu există un atac mai eficient decât căutarea exhaustivă pentru AES cu număr complet de runde.

#### RETELE FEISTEL

- ▶ Se aseamănă rețelelor de substituție-permutare în sensul că păstrează aceleași elemente componente: S-box, permutare, procesul de derivare a cheii, runde;
- ▶ Se diferențiază de rețelele de substituție-permutare prin proiectarea de nivel înalt;
- ▶ Introduc avantajul major că S-box-urile NU trebuie să fie inversabile;
- ▶ Permit aşadar obținerea unei structuri *inversabile* folosind elemente *neinversabile*.



- ▶ Intrarea în runda  $i$  se împarte în 2 jumătăți:  $L_{i-1}$  și  $R_{i-1}$  (i.e. *Left* și *Right*);
- ▶ ieșirile din runda  $i$  sunt:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f_i(R_{i-1})$$

- ▶ Funcțiile  $f_i$  depind de cheia de rundă, derivând dintr-o funcție publică  $\hat{f}_i$ :

$$f_i(R) = \hat{f}_i(k_i, R)$$

- Rețelele Feistel sunt inversabile indiferent dacă funcțiile  $f_i$  sunt inversabile sau nu;
- Fie  $(L_i, R_i)$  ieșirile din runda  $i$ ;
- Intrările  $(L_{i-1}, R_{i-1})$  în runda  $i$  sunt:

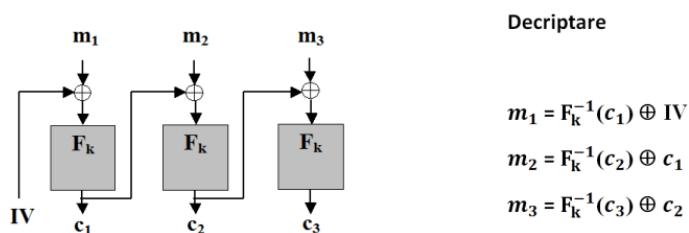
$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus f_i(R_{i-1})$$

## Curs 6 - Padding-oracle attack, Constructii practice PRF, Data Encryption Standard – DES

### Atacul bazat pe oracol de padding

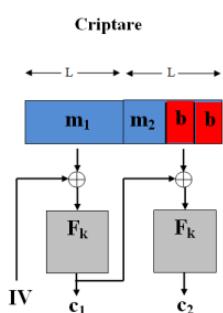
- Am văzut cum funcționează modul CBC când lungimea mesajului clar este multiplu de lungimea L blocului de criptat (suportat de  $F_k$ ) în octeți



- Ce se întâmplă când  $|m| \neq L$ ?
- Folosim padding-ul PKCS#7:
  - Fie  $b$  numărul de octeți de adăugat la ultimul bloc pentru a avea  $|m|$  multiplu de L ( $1 \leq b \leq L$ )
  - Se adaugă  $b$  octeți la ultimul bloc din  $m$ , fiecare reprezentând valoarea lui  $b$

### CBC cu padding PKCS7

Considerăm situația în care un client trimite mesaje criptate în modul CBC către un server.



- în urma decriptării se obțin  $m_1||m_2$
- se citește octetul final  $b$
- dacă ultimii  $b$  octeți au toți valoarea  $b$ , atunci se scoate padding-ul și se obține mesajul original  $m$
- altfel întoarce mesajul *padding gresit* și cere retransmiterea mesajului

Server-ul acționează ca un oracol de padding - adversarul îi trimite texte criptate și află dacă padding-ul este corect sau nu

## Ideea atacului cu oracol de padding

- ▶ Unui text criptat  $(IV, c)$  îi corespunde textul clar cu padding  $m' = F_k^{-1}(c) \oplus IV$
- ▶ Dacă un adversar modifică octetul  $i$  din IV atunci modificarea se va reflecta și în octetul  $i$  din  $m'$

$$F_k^{-1}(c) \quad \boxed{\text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx}}$$

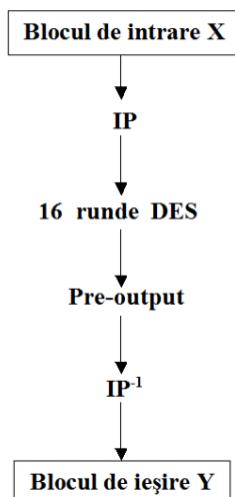


$$IV \quad \boxed{\text{CD} \quad 02 \quad A7 \quad 19 \quad 23 \quad 7B \quad 00 \quad E8}$$

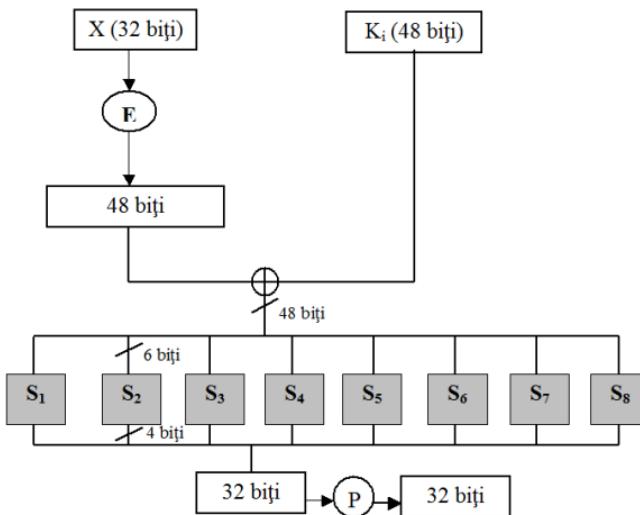
=

$$\begin{array}{l} \text{mesajul cu} \\ \text{padding} \end{array} \quad \boxed{\text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx} \quad \text{xx}}$$

Complexitatea atacului cu oracol de padding ▶ sunt necesare cel mult L încercari pentru a afla b ▶ cel mult  $2^8 = 256$  încercari pentru a afla fiecare octet din mesajul initial ▶ în total sunt necesare  $256 * bt$  încercari (unde bt reprezintă numărul de octeți din mesajul original) pentru a găsi întregul text clar DES



- ▶ DES este o rețea de tip Feistel cu 16 runde și o cheie pe 56 biți;
- ▶ Procesul de derivare a cheii (*key schedule*) obține o sub-cheie de rundă  $k_i$  pentru fiecare rundă pornind de la cheia master  $k$ ;
- ▶ Funcțiile de rundă  $f_i(R) = f(k_i, R)$  sunt derivate din aceeași funcție principală  $\hat{f}$  și nu sunt inversabile;
- ▶ Fiecare sub-cheie  $k_i$  reprezintă permutarea a 48 biți din cheia master;
- ▶ Întreaga procedură de obținere a sub-cheilor de rundă este fixă și cunoscută, singurul secret este cheia master .



- ▶ Funcția  $f$  este, în esență, o rețea de substituție-permutare cu doar o rundă.
- ▶ Pentru calculul  $f(k_i, R)$  se procedează astfel:
  1.  $R$  este expandat la un bloc  $R'$  de 48 biți cu ajutorul unei funcții de expandare  $R' = E(R)$ .
  2.  $R'$  este XOR-at cu  $k_i$  iar valoarea rezultată este împărțită în 8 blocuri de câte 6 biți.
  3. Fiecare bloc de 6 biți trece printr-un SBOX diferit rezultând o valoare pe 4 biți.
  4. Se concatenează blocurile rezultate și se aplică o permutare, rezultând în final un bloc de 32 biți.
- ▶ De remarcat: Toată descrierea lui DES, inclusiv SBOX-urile și permutările sunt publice.

## SBOX-urile din DES

- ▶ Formează o parte esențială din construcția DES;
- ▶ DES devine mult mai vulnerabil la atacuri dacă SBOX-urile sunt modificate ușor sau dacă sunt alese aleator
- ▶ Primul și ultimul bit din cei 6 de la intrare sunt folosiți pentru a alege linia din tabel, iar biții 2-5 sunt folosiți pentru coloană; output-ul va consta din cei 4 biți aflați la intersecția liniei și coloanei alese.

		Cei 4 biți din mijloc															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Biți din margină	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

- ▶ Modificarea **unui bit** de la intrare întotdeauna afectează cel puțin **doi biți** de la ieșire.
- ▶ DES are un puternic efect de avalanșă generat de ultima permutare menționată mai sus și de permutările folosite:

## Securitatea sistemului DES

- ▶ Încă de la propunerea sa, DES a fost criticat din două motive:
    1. Spațiul cheilor este prea mic făcând algoritmul vulnerabil la forță brută;
    2. Criteriile de selecție a SBOX-urilor au fost ținute secrete și ar fi putut exista atacuri analitice care explorau proprietățile matematice ale SBOX-urilor, cunoscute numai celor care l-au proiectat.
  - ▶ Cu toate acestea....
    - ▶ După 30 ani de studiu intens, cel mai bun atac practic rămâne doar o căutare exhaustivă pe spațiul cheilor;
    - ▶ O căutare printre  $2^{56}$  chei este fezabilă azi (dar netrivială);
    - ▶ În 1977, un calculator care să efectueze atacul într-o zi ar fi costat 20.000.000\$;
  - ▶ Primul atac practic a fost demonstrat în 1997 cand un număr de provocari DES au fost propuse de RSA Security și rezolvate; ▶ Prima provocare a fost spartă în 1997 de un proiect care a folosit sute de calculatoare coordonate prin Internet; a durat 96 de zile; ▶ A doua provocare a fost spartă anul următor în 41 de zile; ▶ Impresionant a fost timpul pentru a treia provocare: 56 de ore; ▶ S-a construit o mașină în acest scop, Deep Crack cu un cost de 250.000\$
  - ▶ O alta problema a sistemului DES, mai puțin importantă, este lungimea blocului relativ scurtă (64 biti);
  - ▶ Securitatea multor construcții bazate pe cifruri bloc depinde de lungimea blocului; ▶ În modul de utilizare CTR, dacă un atacator are  $2^{27}$  perechi text clar/text criptat, securitatea este compromisa cu probabilitate mare; ▶ Concluzionând, putem spune că insecuritatea sistemului DES nu are a face cu structura internă sau construcția în sine (care este remarcabilă), ci se datorează numai lungimii cheilor prea mici.
- Criptanaliza avansată ▶ La sfârșitul anilor '80, Biham și Shamir au dezvoltat o tehnică numita criptanaliza diferențială pe care au folosit-o pentru un atac împotriva DES; ▶ Atacul presupune complexitate timp  $2^{37}$  (memorie neglijabilă) dar cere ca atacatorul să analizeze  $2^{36}$  texte criptate obținute dintr-o multime de  $2^{47}$  texte clare alese; ▶ Din punct de vedere teoretic, atacul a fost o inovație, dar practic este aproape imposibil de realizat; ▶ La începutul anilor '90, Matsui a dezvoltat criptanaliza liniară aplicată cu succes pe DES; ▶ Desi necesită  $2^{43}$  texte criptate, avantajul este că textele clare nu trebuie să fie alese de atacator, ci doar cunoscute de el; ▶ Problema însă ramane aceeași: atacul este foarte greu de pus în practică

### Criptanaliza diferențială

- ▶ Cataloghează diferențe specifice între texte clare care produc diferențe specifice în textele criptate cu probabilitate mai mare decât ar fi de așteptat pentru permutările aleatoare;
- ▶ Fie un cifru bloc cu blocul de lungime  $n$  și  $\Delta_x, \Delta_y \in \{0, 1\}^n$ ;
- ▶ Spunem că diferențiala  $(\Delta_x, \Delta_y)$  apare cu probabilitate  $p$  dacă pentru intrări aleatoare  $x_1, x_2$  cu

$$x_1 \oplus x_2 = \Delta_x$$

și o alegere aleatoare a cheii  $k$

$$\Pr[F_k(x_1) \oplus F_k(x_2) = \Delta_y] = p$$

- ▶ Pentru o funcție aleatoare, probabilitatea de apariție a unei diferențiale nu este mai mare decât  $2^{-n}$ ;
- ▶ La un cifru bloc slab, ea apare cu o probabilitate mult mai mare;

## Criptanaliza liniară

- ▶ Metoda consideră relații liniare între intrările și ieșirile unui cifru bloc;
- ▶ Spunem că porțiunile de biți  $i_1, \dots, i_l$  și  $i'_1, \dots, i'_{l'}$  au distanța  $p$  dacă pentru orice intrare aleatoare  $x$  și orice cheie  $k$

$$\Pr[x_{i_1} \oplus \dots \oplus x_{i_l} \oplus y_{i'_1} \oplus \dots \oplus y_{i'_{l'}} = 0] = p$$

unde  $y = F_k(x)$ .

- ▶ Pentru o funcție aleatoare se așteaptă ca  $p = 0.5$ ;
- ▶ Matsui a arătat cum se poate folosi o diferență  $p$  mare pentru a sparge complet un cifru bloc;
- ▶ Necesită un număr foarte mare de perechi text clar/text criptat.

## Criptare dublă

- ▶ Fie  $F$  un cifru bloc (in particular ne vom referi la DES); definim un alt cifru bloc  $F'$  astfel

$$F'_{k_1, k_2}(x) = F_{k_2}(F_{k_1}(x))$$

cu  $k_1, k_2$  chei independente;

- ▶ Lungimea totală a cheii este 112 biți, suficient de mare pentru căutare exhaustivă;
- ▶ Însă, se poate arăta un atac în timp  $2^{56}$  unde  $|k_1| = 56 = |k_2|$  (față de  $2^{2 \cdot 56}$  cât necesită o căutare exhaustivă);
- ▶ Atacul se numește **meet-in-the-middle**;

## Atacul meet-in-the-middle

- ▶ Iată cum funcționează atacul dacă se cunoaște o pereche text clar/text criptat  $(x, y)$  cu  $y = F_{k_2}(F_{k_1}(x))$ :
  1. Pentru fiecare  $k_1 \in \{0, 1\}^n$ , calculează  $z := F_{k_1}(x)$  și păstrează  $(z, k_1)$ ;
  2. Pentru fiecare  $k_2 \in \{0, 1\}^n$ , calculează  $z := F_{k_2}^{-1}(y)$  și păstrează  $(z, k_2)$ ;
  3. Verifică dacă există perechi  $(z, k_1)$  și  $(z, k_2)$  care coincid pe prima componentă;
  4. Atunci valorile  $k_1, k_2$  corespunzătoare satisfac

$$F_{k_1}(x) = F_{k_2}^{-1}(y)$$

adică  $y = F'_{k_1, k_2}(x)$

- ▶ Complexitatea timp a atacului este  $O(2^n)$ .

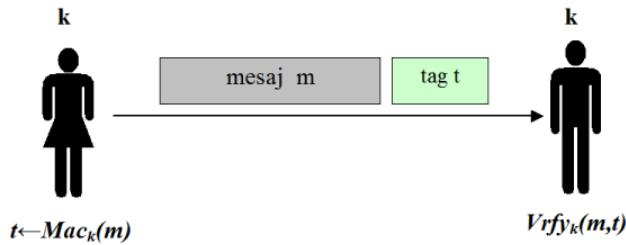
**Triplu-DES (3DES)** ► Se bazeaza pe tripla invocare a lui DES folosind doua sau trei chei; ► Este considerat sigur si in 1999 l-a inlocuit pe DES ca standard; ► 3DES este foarte eficient in implementarile hardware (la fel ca si DES) dar totusi lent in implementari software; ► 3DES cu 3 chei inca se mai foloseste dar recomandarea este de a inlatura datorita lungimii mici a blocurilor si a faptului ca este lent ► Este popular in aplicatiile financiare si in protejarea informatiilor biometrice din pasapoartele electronice;

- DES a fost sistemul simetric dominant de la mijlocul anilor '70 pana la mijlocul anilor '90;
  - DES cu cheia pe 56 biti poate fi spart relativ usor astazi prin forta bruta;
  - Insă, este foarte greu de spart folosind criptanaliza diferențială sau liniară;
  - Pentru 3DES nu se cunoaste nici un atac practic.
- 

#### Curs 7- Coduri de autentificare a mesajelor – MAC, Functii hash

##### Criptare vs. autentificarea mesajelor

- Criptarea, in general, NU ofera integritatea mesajelor!
  - Nu folositi criptarea cu scopul de a obtine autentificarea mesajelor
  - Daca un mesaj este transmis criptat de-a lungul unui canal de comunicare, nu inseamna ca un adversar nu poate modifica/altera mesajul asa incat modificarea sa aiba sens in textul clar;
  - Verificam, in continuare, ca nici o schema de criptare studiata nu ofera integritatea mesajelor;
  - Asa cum am vazut, criptarea nu rezolva problema autentificarii mesajelor;
  - Vom folosi un mecanism diferit, numit cod de autentificare a mesajelor - MAC (Message Authentication Code);
  - Scopul lor este de a impiedica un adversar sa modifice un mesaj trimis fara ca partile care comunic sa nu detecteze modificarea;
  - Vom lucra in continuare in contextul criptografiei cu cheie secreta unde partile trebuie sa prestageasca de comun acord o cheie secreta.
-



- ▶ Alice și Bob stabliesc o cheie secretă  $k$  pe care o partajează;
- ▶ Când Alice vrea să îi trimită un mesaj  $m$  lui Bob, calculează mai întâi un tag  $t$  (o etichetă) pe baza mesajului  $m$  și a cheii  $k$  și trimite perechea  $(m, t)$ ;

► Tag-ul este calculat folosind un algoritm de generare a tag-urilor numit Mac; ► La primirea perechii  $(m, t)$  Bob verifica dacă tag-ul este valid (în raport cu cheia  $k$ ) folosind un algoritm de verificare Vrfy; ► În continuare prezentăm definitia formală a unui cod de autentificare a mesajelor

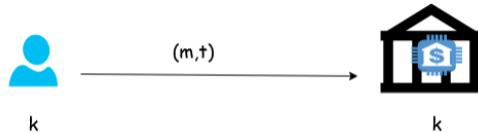
### Definiție

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

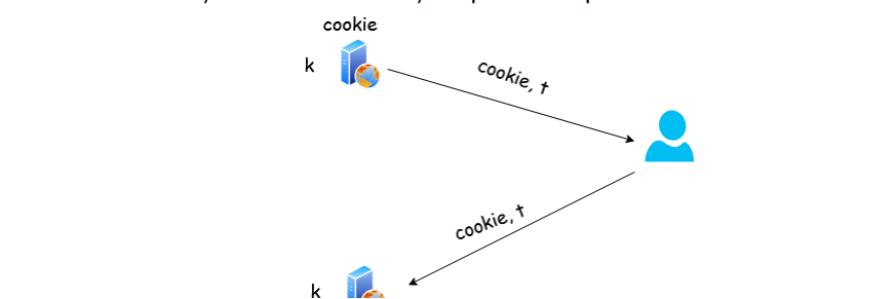
1.  $\text{Gen}(1^n)$ : este algoritmul de generare a cheii  $k$  (aleasă uniform pe  $n$  biți)
  2.  $\text{Mac} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$  este algoritmul de generare a tag-urilor  $t \leftarrow \text{Mac}_k(m)$ ;
  3.  $\text{Vrfy} : \mathcal{K} \times \mathcal{M}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

### Cazuri de folosire ale MAC

1. când cele două părți care comunică partajează o cheie secretă în avans

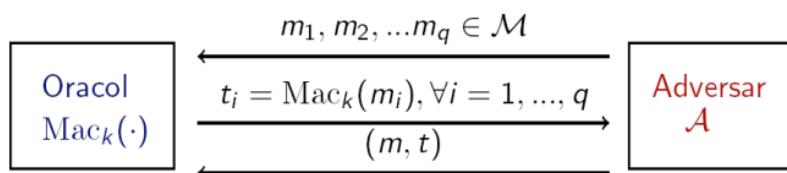


2. când avem o singură parte care autentifică o comunicație, interacționând cu ea însăși după un timp



Securitate MAC - discutie ► Intuitie: nici un adversar polinomial nu ar trebui sa poata genera un tag valid pentru nici un mesaj "nou" care nu a fost deja trimis (si autentificat) de partile care comunica; ► Trebuie sa definim puterea adversarului si ce inseamna spargerea sau un atac asupra securitatii; ► Adversarul lucreaza in timp polinomial si are acces la mesajele trimise intre parti impreuna cu tag-urile aferente. ► Adversarul este activ, poate influent, a autentificarea unor mesaje alese de el... ► ...dar nu trebuie sa poata falsifica tag-ul aferent unor mesaje care nu au fost autentificate de expeditor

- Formal, ii dăm adversarului acces la un oracol  $\text{Mac}_k(\cdot)$ ;
- Adversarul poate trimite orice mesaj  $m$  dorit către oracol și primește înapoi un tag corespunzător  $t \leftarrow \text{Mac}_k(m)$ ;
- Considerăm că securitatea este impactată dacă adversarul este capabil să producă un mesaj  $m$  împreună cu un tag  $t$  aşa încât:
  1.  $t$  este un tag valid pentru mesajul  $m$ :  $\text{Vrfy}_k(m, t) = 1$ ;
  2. Adversarul nu a solicitat anterior (de la oracol) un tag pentru mesajul  $m$ .
- Despre un MAC care satisface nivelul de securitate de mai sus spunem că *nu poate fi falsificat printr-un atac cu mesaj ales*;
- Aceasta înseamnă că un adversar nu este capabil să falsifice un tag valid pentru nici un mesaj ...
- ... deși poate obține tag-uri pentru orice mesaj ales de el, chiar *adaptiv* în timpul atacului.
- Pentru a da definiția formală, definim mai întâi un experiment pentru un MAC  $\pi = (\text{Mac}, \text{Vrfy})$ , în care considerăm un adversar  $\mathcal{A}$  și parametrul de securitate  $n$ ;



- Output-ul experimentului este 1 dacă și numai dacă:
  - (1)  $\text{Vrfy}_k(m, t) = 1$  și (2)  $m \notin \{m_1, \dots, m_q\}$ ;
- Dacă  $\text{Mac}_{\mathcal{A}, \pi}^{\text{forge}}(n) = 1$ , spunem că  $\mathcal{A}$  a efectuat experimentul cu succes.

► Intrebare: De ce este necesara a doua conditie de la securitatea MAC (un adversar nu poate intoarce un mesaj pentru care anterior a cerut un tag)? ► Raspuns: Pentru a evita atacurile triviale în care un adversar cere tag-ul aferent unui mesaj și apoi intoarce chiar acel mesaj împreună cu tag-ul primit. ► Intrebare: Definitia MAC ofera protectie la atacurile prin replicare (in care un adversar copiaza un mesaj

impreuna cu tag-ul aferent trimise de partile comunicante)? ► Raspuns: NU! MAC-urile nu ofera nici un fel de protectie la atacurile prin replicare.

#### MAC-uri sigure

- Avem nevoie de o functie cu cheie, pentru care, dandu-se  $Mac_k(m_1), Mac_k(m_2)...$
  - ... sa nu fie usor (in timp polinomial) a gasi  $Mac_k(m)$  pentru orice  $m \notin \{m_1, m_2, \dots\}$
  - Functia MAC ar putea fi un PRF
- 
- *Functiile pseudoaleatoare (PRF) sunt un instrument bun pentru a construi MAC-uri sigure;*

#### Construcție

*Fie  $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  o PRF. Definim un MAC in felul urmator:*

- Mac : pentru o cheie  $k \in \{0, 1\}^n$  si un mesaj  $m \in \{0, 1\}^n$ , calculeaza tag-ul  $t = F_k(m)$  (dacă  $|m| \neq |k|$  nu întoarce nimic);
- Vrfy : pentru o cheie  $k \in \{0, 1\}^n$ , un mesaj  $m \in \{0, 1\}^n$  si un tag  $t \in \{0, 1\}^n$ , întoarce 1 dacă si numai dacă  $t = F_k(m)$  (dacă  $|m| \neq |k|$ , întoarce 0).

**Teorema** Daca F este PRF, constructia de mai sus reprezinta un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales).

#### MAC-uri pentru mesaje de lungime variabila

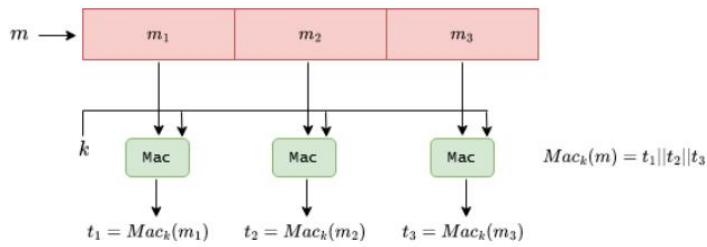
- Construcția prezentată anterior funcționează doar pe mesaje de lungime fixă (PRF-urile sunt instanțiate cu sisteme de criptare bloc care, cel mai adesea, acceptă input-uri de 128 biți);
- Însă în practică avem nevoie de mesaje de lungime variabilă;
- Arătăm cum putem obține un MAC de lungime variabilă pornind de la un MAC de lungime fixă;
- Fie  $(\pi' = (\text{Mac}', \text{Vrfy}'))$  un MAC sigur de lungime fixă pentru mesaje de lungime  $n$ ;
- Pentru a construi un MAC de lungime variabilă, putem sparge mesajul  $m$  în blocuri  $m_1, \dots, m_d$  și autentificam blocurile folosind  $\pi'$ ;
- Iată câteva modalități de a face aceasta:

#### 1. XOR pe toate blocurile cu autentificarea rezultatului:

$$t = \text{Mac}'_k(\oplus_i m_i)$$

- **Intrebare:** Este sigură această metodă?
- **Răspuns:** NU! Un adversar poate modifica mesajul original  $m$  a.i. XOR-ul blocurilor nu se schimbă, el obținând un tag valid pentru un mesaj nou;

## 2. Autentificare separată pentru fiecare bloc:



► **Intrebare:** Este sigură această metodă?

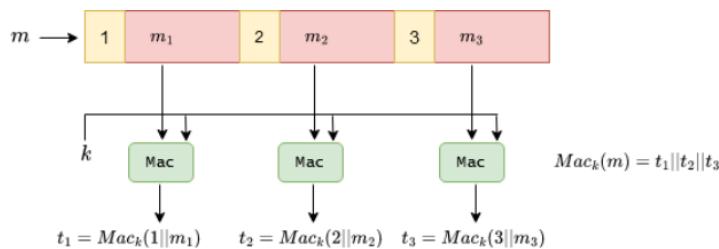
► **Răspuns:** NU!

► Fiind dat  $(m, t)$  cu  $m = m_1 || m_2 || m_3$  și  $t = t_1 || t_2 || t_3$

► Atunci  $(m', t')$  este o pereche validă cu  $m' = m_2 || m_1 || m_3$  și  $t' = t_2 || t_1 || t_3$

## 3. Autentificare separată pentru fiecare bloc folosind o secvență de numere:

În felul acesta prevenim atacul anterior

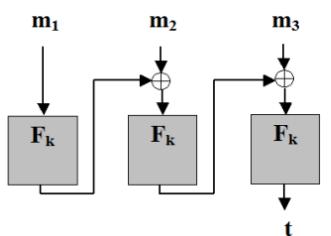


► **Intrebare:** Este sigură această metodă?

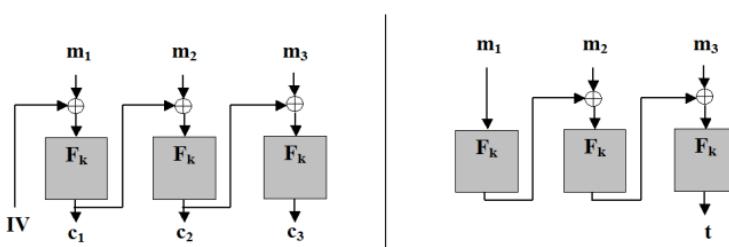
► **Răspuns:** NU! Un adversar poate scoate blocuri de la sfârșitul mesajului:  $t = t_1 || t_2$  este un tag valid pentru mesajul  $m = m_1 || m_2$ ;

## CBC-MAC

- O soluție mult mai eficientă este să folosim **CBC-MAC**;
- CBC-MAC este o construcție similară cu modul CBC folosit pentru criptare;
- Folosind CBC-MAC, pentru un tag aferent unui mesaj de lungime  $l \cdot n$ , se aplică sistemul bloc doar de  $l$  ori, iar tag-ul are numai  $n$  biți.



## CBC-MAC vc. Criptare în mod CBC



### Criptare în mod CBC

- ▶  $IV = 0^n$  este aleator pentru a obține securitate;
- ▶ toate blocurile  $c_i$  constituie mesajul criptat.

### CBC-MAC

- ▶  $IV = 0^n$  este fixat pentru a obține securitate;
- ▶ doar ieșirea ultimului bloc constituie tag-ul  
(întoarcerea tuturor blocurilor intermediare duce la pierderea securității)

## Confidențialitate și integritate - criptare autentificată

- ▶ Iată trei abordări uzuale pentru a combina criptarea și autentificarea mesajelor:
  1. *Criptare-și-autentificare*: criptarea și autentificarea se fac independent. Pentru un mesaj clar  $m$ , se transmite mesajul criptat  $\langle c, t \rangle$  unde

$$c \leftarrow \text{Enc}_{k_1}(m) \text{ și } t \leftarrow \text{Mac}_{k_2}(m)$$

La recepție,  $m = \text{Dec}_{k_1}(c)$  și dacă  $\text{Vrfy}_{k_2}(m, t) = 1$ , atunci întoarce  $m$ ; altfel întoarce  $\perp$ .

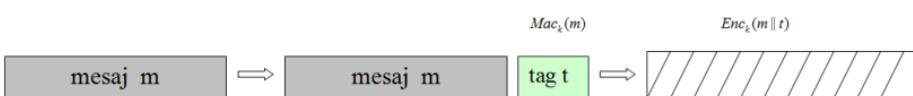


- ▶ Combinăția aceasta **nu** este neapărat sigură; un MAC sigur nu implică nici un fel de confidențialitate;

- ▶ 2. *Autentificare-apoi-criptare*: întâi se calculează tag-ul  $t$  apoi mesajul și tag-ul sunt criptate împreună

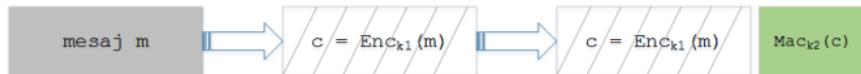
$$t \leftarrow \text{Mac}_{k_2}(m) \text{ și } c \leftarrow \text{Enc}_{k_1}(m \parallel t)$$

La recepție,  $m \parallel t = \text{Dec}_{k_1}(c)$  și dacă  $\text{Vrfy}_{k_2}(m, t) = 1$ , atunci întoarce  $m$ ; altfel întoarce  $\perp$ .



- ▶ Combinăția aceasta **nu** este neapărat sigură;
- ▶ Se poate construi o schemă de criptare CPA-sigură care împreună cu orice MAC sigur nu poate fi CCA-sigură;

### 3. Criptare-apoi-autentificare



- ▶ Combinația aceasta este întotdeauna sigură; se folosește în IPsec;
- ▶ Deși folosim aceeași construcție pentru a obține securitate CCA și transmitere sigură a mesajelor, scopurile urmărite sunt diferite în fiecare caz;

► Orice schema de criptare autentificata este și CCA-sigură dar există și scheme de criptare CCA-negligibile care nu sunt scheme de criptare autentificata ► Totuși, cele mai multe aplicații de scheme de criptare cu cheie secretă în prezența unui adversar activ necesită integritate

CA in	bazată pe	care în general este	dar în acest caz este
SSH	C_si_A	nesigură	sigură
SSL	A_apoi_C	nesigură	nesigură
IPSec	C_apoi_A	sigură	sigură
WinZip	C_apoi_A	sigură	nesigură

- ▶ CA = criptare autentificată;
- ▶ C-apoi-A = criptare-apoi-autentificare;
- ▶ C-si-A = criptare-si-autentificare

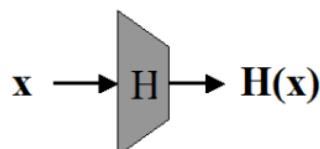
#### Functii hash

Acstea primesc ca argument o secvență de lungime variabilă pe care o comprimă într-o secvență de lungime mai mică, fixată

Căutare O(1)

**Intrebare:** Ce se întampla dacă pentru 2 valori  $x \neq x'$ ,  $H(x) = H(x')$ ? ► Raspuns: În acest caz, apare o coliziune - ambele valori se vor stoca în aceeași celula

- ▶ **Întrebare:** Există funcții hash fără coliziuni?
- ▶ **Răspuns:** NU! Funcțiile hash (cel puțin cele interesante d.p.d.v criptografic) comprimă, deci funcția nu poate fi injectivă;



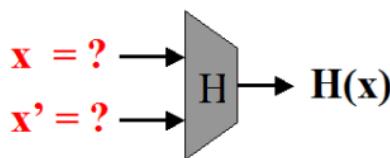
#### Definiție

$H : \{0,1\}^* \rightarrow \{0,1\}^{l(n)}$  se numește **rezistentă la coliziuni (collision-resistant)** dacă pentru orice adversar PPT  $\mathcal{A}$  există o funcție neglijabilă negl astfel încât

$$\Pr[\text{Hash}_{\mathcal{A}, H}^{\text{coll}}(n) = 1] \leq \text{negl}(n).$$

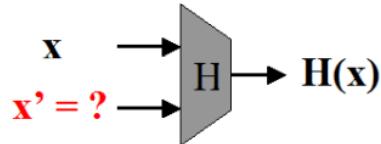
- În practică, rezistența la coliziuni poate fi dificil de obținut;
- Pentru anumite aplicații sunt utile noțiuni mai relaxate de securitate;
- Există 3 nivele de securitate:
  1. **Rezistență la coliziuni**: este cea mai puternică noțiune de securitate și deja am definit-o formal;
  2. **Rezistență la a doua preimagine**: presupune că fiind dat  $x$  este dificil de determinat  $x' \neq x$  a.î.  $H(x) = H(x')$
  3. **Rezistență la prima preimagine**: presupune că fiind dat  $H(x)$  este imposibil de determinat  $x$ .

#### Coliziuni



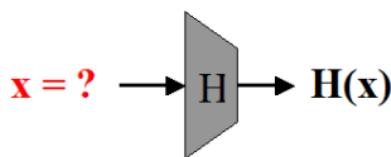
- **Provocare**: Se cer 2 valori  $x \neq x'$  a.î.  $H(x) = H(x')$ ;
- **Atac**: "Atacul zilei de naștere" necesită  $\approx 2^{l(n)/2}$  evaluări pentru  $H$ .

#### A doua preimagine



- **Provocare**: Fiind dat  $x$ , se cere  $x'$  a.î.  $H(x) = H(x')$ ;
- **Atac**: Un atac generic necesită  $\approx 2^{l(n)}$  evaluări pentru  $H$ .

#### Prima preimagine



- **Provocare**: Fiind dat  $y = H(x)$ , se cere  $x$  a.î.  $H(x) = y$ ;
- O astfel de funcție se numește și *calculabilă într-un singur sens (one-way function)*;
- **Atac**: Un atac generic necesită  $\approx 2^{l(n)}$  evaluări pentru  $H$ .

- ▶ **Întrebare:** De ce o funcție care satisface proprietatea de **rezistență la coliziuni** satisface și proprietatea de **rezistență la a doua preimagine**?
- ▶ **Răspuns:** Pentru  $x$  fixat, adversarul determină  $x' \neq x$  pentru care  $H(x) = H(x')$ , deci găsește o coliziune;
- ▶ **Întrebare:** De ce o funcție care satisface proprietatea de **rezistență la a doua preimagine** satisface și proprietatea de **rezistență la prima imagine**?
- ▶ **Răspuns:** Pentru  $x$  oarecare, adversarul calculează  $H(x)$ , o inversează și determină  $x'$  a.î.  $H(x') = H(x)$ . Cu probabilitate mare  $x' \neq x$ , deci găsește o a doua preimagine.

**Atacul "zilei de nastere"** ▶ Analizam posibilitatea de a determina o coliziune pornind de la un exemplu clasic: paradoxul nasterilor; ▶ Intrebare: Care este dimensiunea unui grup de oameni pentru ca 2 dintre ei să fie născuți în aceeași zi cu probabilitate  $1/2$ ? ▶ Răspuns: 23!

- ▶ Generalizând, considerăm o mulțime de dimensiune  $n$  și  $q$  elemente uniform aleatoare din această mulțime  $y_1, \dots, y_q$ ;
- ▶ Atunci pentru  $q \geq 1.2 \times 2^{n/2}$  probabilitatea să existe  $i \neq j$  a.î.  $y_i = y_j$  este  $\geq 1/2$ .
- ▶ Această rezultat conduce imediat la un atac asupra funcțiilor hash cu scopul de a determina coliziuni:
  - ▶ Adversarul alege  $2^{n/2}$  valori  $x_i$ ;
  - ▶ Calculează pentru fiecare  $y_i = H(x_i)$ ;
  - ▶ Caută  $i \neq j$  cu  $H(x_i) = H(x_j)$ ;
  - ▶ Dacă nu găsește nici o coliziune, reia atacul.
- ▶ Cum probabilitatea de succes a atacului este  $\geq 1/2$ , atunci numărul de încercări este  $\approx 2$ .

## Transformarea Merkle-Damgård

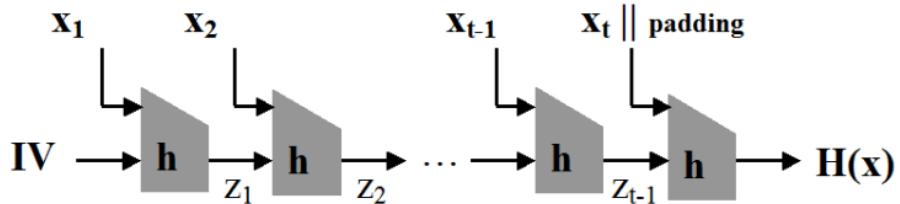
- ▶ Numim **funcție de compresie** o funcție hash rezistentă la coliziuni de intrare de lungime fixă;
- ▶ **Întrebare:** Intuitiv, ce se construiește mai ușor,  $H_1$  sau  $H_2$ ?

$$H_1 : \{0, 1\}^{m_1} \rightarrow \{0, 1\}^{n_1}, m_1 > n_1, m_1 \approx n_1$$

$$H_2 : \{0, 1\}^{m_2} \rightarrow \{0, 1\}^{n_2}, m_2 \gg n_2$$

- ▶ **Răspuns:** Cu cât domeniul și codomeniul funcției sunt mai apropiate ca dimensiune, numărul de coliziuni scade  $\Rightarrow$  coliziunile devin mai dificil de determinat (considerăm că funcția distribuie valorile uniform aleator).

- ▶ Scopul este să construim o funcție hash (cu intrare de lungime variabilă), pornind de la o funcție de compresie (de lungime fixă);
- ▶ Pentru aceasta se aplică **transformarea Merkle-Damgård**;



Daca h prezinta rezistenta la coliziuni, atunci si H prezinta rezistenta la coliziuni.

- ▶ **MD5** (Message Digest 5)
  - ▶ definit în 1991 de R.Rivest ca înlocuitor pentru MD4
  - ▶ produce o secvență hash de 128 biți
  - ▶ nesigur din 1996
  - ▶ utilizat în versiuni mai vechi de Moodle

- ▶ **SHA** (Secure Hash Algorithm)
  - ▶ o familie de funcții hash publicate de NIST
  - ▶ SHA-0 și SHA-1 sunt nesigure
  - ▶ SHA-2 prezintă 2 variante: SHA-256 și SHA-512
  - ▶ SHA-3 adoptat în 2012 pe baza Keccak

- 
- ▶ Este  $H(x) = x \pmod{N}$  o funcție hash rezistenta la coliziuni? Daca da, explicati de ce, daca nu, gasiti o coliziune.
  - ▶ **Raspuns:** Funcția nu e rezistenta la coliziuni: pentru orice  $x < N$  si orice  $a \in \mathbb{N}$ ,  $x$  si  $aN + x$  formeaza o coliziune.
  - ▶ Este  $H(x) = ax + b \pmod{N}$  cu  $(a, N) = 1$  o functie hash rezistenta la coliziuni? Daca da, explicați de ce, daca nu, gasiți o coliziune.
  - ▶ **Raspuns:** Funcția nu e rezistenta la coliziuni: cautam  $x_1$  si  $x_2$  asa incat  $H(x_1) = H(x_2)$  adica  $ax_1 + b = ax_2 + b \pmod{N}$ . Obținem  $a(x_1 - x_2) = 0 \pmod{N}$ . Cum  $(a, N)$  obținem ca  $x_1 - x_2 = 0 \pmod{N}$ .
- 

#### Curs 8 – SHA 3, Aplicatii ale functiilor hash, Criptografie cu cheie publica (asimetrica), Teoria numerelor pentru criptografie, Prezumptii criptografice dificile

- ▶ **MD5** ▶ output pe 128 biti ▶ propusa în 1991 si considerata rezistenta la coliziuni o perioada de timp
- ▶ 2004 - atac pentru gasirea de coliziuni + coliziuni explicite ▶ astazi se pot gasi coliziuni in mai putin de un minut pe un calculator desktop

► **SHA-1**

- face parte din seria de algoritmi standardizați SHA (Secure Hash Algorithm) și a fost standardul aprobat de NIST până în 2011
- output pe 160 biti
- 2005 - atac teoretic pentru găsirea coliziunilor; necesită  $2^{69}$  evaluări ale funcției hash
- 2017 - prima coliziune practică -  $2^{63}$  evaluări ale funcției hash
- atac pentru găsirea de *coliziuni cu prefix* - aprox.  $2^{67}$  evaluări
- *coliziuni cu prefix* - pornind de la prefixele  $P$  și  $P'$ , se cere găsirea mesajelor  $M \neq M'$  cu  $H(P||M) = H(P'||M')$
- În practică, acestea sunt mai periculoase, pot duce la diverse atacuri incluzând generarea de certificate digitale false și atacuri asupra TLS, SSH

► **SHA-2** ► prezintă două versiuni: SHA-256 și SHA-512 în funcție de lungimea output-ului ► nu se cunosc vulnerabilități; atât SHA-2 cât și SHA-3 sunt siguri de folosit acolo unde rezistența la coliziuni este necesar

Criterii de selecție sha3 ► Lungimea secvenței de ieșire:  $n = 224, 256, 384$  și  $512$  biti; ► Alte dimensiuni ale secvenței de ieșire sunt optionale; ► Eficiența crescută față de SHA-2; ► Utilizare în HMAC; ► Rezistența la coliziuni, prima și a doua preimagine (conform cu atacurile generice, traditionale); ► Demonstrație de securitate; ► Parametrizabilă, număr de runde variabil; ► Simplicitate, claritate.

► Keccak este câștigătorul competiției SHA-3 ► SHA-2 ramane în continuare sigur

**Atac pentru găsirea de coliziuni**

- Considerăm funcții hash

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

- **Rețineți!** Cel mai bun atac generic pentru găsirea de coliziuni ne spune că avem nevoie de funcții hash cu output-ul pe  $2n$  biti pentru securitate împotriva adversarilor care rulează în timp  $2^n$ ...
- ... spre deosebire de sistemele de criptare bloc, unde avem nevoie de  $n$  biti pentru securitate împotriva adversarilor care rulează în timp  $2^n$
- **Exemplu.** Dacă dorim ca găsirea de coliziuni să necesite timp  $2^{128}$  atunci trebuie să alegem funcții hash cu output-ul pe 256 biti
- Am văzut că funcțiile hash pot fi folosite pentru a construi MAC-uri de lungime variabilă (în HMAC)

$$x \rightarrow H(x) \rightarrow Mac_k(H(x))$$

pentru MAC de lungime fixă

- Funcțiile hash au și multe alte aplicații atât în criptografie cât și în securitate
- Pentru un fișier  $x$ ,  $H(x)$  poate servi ca identificator unic (amprentă) - existența unui fișier diferit cu același hash implică găsirea de coliziuni în  $H$ .
- Prezentăm câteva aplicații care decurg de aici

► Amprentarea virusilor

- scanner-ele de virusi pastreaza hash-urile virusilor cunoscuti
- verificarea fisierelor potential malitioase se face comparand hash-ul lor cu hash-ul virusilor cunoscuti.

► Deduplicarea datelor - stocarea in cloud partajata de mai multi utilizatori - se verifica daca hash-ul unui fisier nou (de incarcat) exista deja in cloud, caz in care se adauga doar un pointer la fisierul respectiv

**Arbori Merkle**

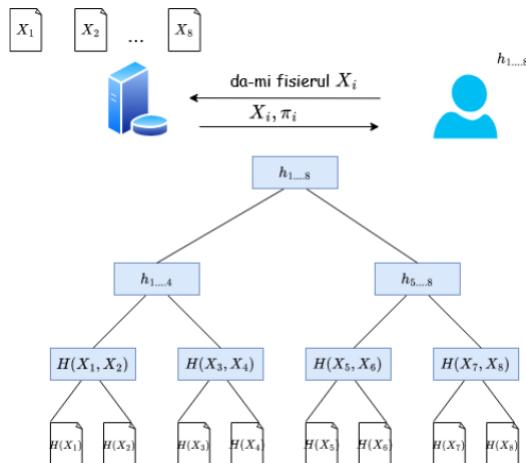
1. Prima solutie:

- clientul stocheaza local  $h_1 = H(x_1), \dots, h_n = H(x_n)$  si verifică dacă  $h_i = H(x'_i)$  pentru fiecare fisier  $x'_i$  recuperat
- **Dezavantaj:** spațiul necesar stocării crește liniar în  $n$

2. A doua soluție

- clientul stocheaza local un singur hash  $h = H(x_1, \dots, x_n)$
- **Dezavantaj:** pentru a verifica  $x_i$ , clientul trebuie să recupereze toate fisierele  $x_1, \dots, x_n$
- **Soluție:** arborii Merkle oferă un compromis între cele două soluții de mai sus

**Arbori Merkle - stocarea fisierelor în cloud**



- $\pi_i$  - conține nodurile adiacente drumului de la  $X_i$  la rădăcină
- pe baza lui  $\pi_i$ , user-ul poate re-calcula valoarea rădăcinii pentru a verifica dacă este aceeași cu valoarea stocată de el

**Stocarea parolelor** ► Cea mai utilizata metoda de autentificare este bazata pe nume de utilizator si parola; ►

Metoda presupune o etapa de înregistrare, în care utilizatorul alege un username si o parola (pwd); ► Ulterior, la fiecare logare, utilizatorul introduce cele 2 valori; ► Daca username se regaseste in lista de utilizatori înregistrati si parola introdusa este corecta atunci autentificarea se realizeaza cu succes; ► In caz contrar, autentificarea esuaza. ► O greseala majora este stocarea parolelor in clar! ► Intrebare: De ce parolele NU trebuie stocate in clar? ►

Raspuns: Pentru ca daca adversarul capata acces la fisierul de parole atunci afla direct parolele tuturor utilizatorilor! ► Pentru stocarea parolelor se utilizeaza functiile hash; ► In fisierul de parole (sau baza de date) se stocheaza, pentru fiecare utilizator perechi de forma: (username, H(pwd))

► Pentru autentificare, utilizatorul introduce numele de utilizator username si parola pwd; ► Sistemul de autentificare calculeaza H(pwd) si se verifica daca valoarea obtinuta este stocata in fisierul de parole pentru utilizatorul indicat prin username; ► Daca da, atunci autentificarea se realizeaza cu succes; in caz contrar, autentificarea esuaza; ► Functiile hash sunt functii one-way: cunoscand H(pwd) nu se poate determina pwd; ► Aceasta metoda de stocare a parolelor introduce deci avantajul ca nu ofera adversarului acces direct la parola, chiar daca acesta detine fisierul de parole.

**Atac de tip dictionar**

► Adversarul poate însă să verifice, pe rand, toate parolele cu probabilitate mare de apariție; ► Acestea sunt de obicei cuvinte cu sens sau parole uzuale; ► Se consideră că formează termenii unui dicționar, de unde și numele atacului: atac de tip dicționar; ► Pentru a minimiza sansele unor astfel de atacuri: ► se blochează procesul de autentificare după un anumit număr de încercări nereusite; ► se obligă utilizatorul să folosească o parola care satisfac anumite criterii: o lungime minima, utilizarea a cel puțin 3 tipuri de simboluri (litere mici, litere mari, cifre, caracter speciale); ► În caz de succes, adversarul determină parola unui singur utilizator;

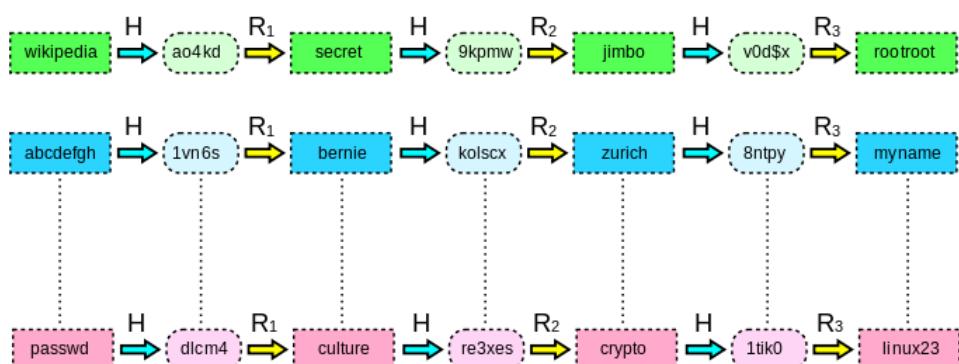
#### Atac folosind tabele hash precalculat

► Pentru a determina parolele mai multor utilizatori simultan, un adversar poate precalcula valorile hash ale parolelor din dicționar; ► Dacă adversarul capătă acces la fișierul de parolă, atunci verifică valorile care se regăsesc în lista precalculată; ► Toate conturile utilizatorilor pentru care se potrivesc valorile sunt compromise; ► Atacul necesită capacitate de stocare mare: trebuie stocate toate perechile ( $pwd, H(pwd)$ ) unde  $pwd$  este o parola din dicționar;

## Rainbow tables

- **Rainbow tables** introduc un compromis între capacitatea de stocare și timp;
- Se compun lanțuri de lungime  $t$ :  

$$pwd_1 \xrightarrow{H} H(pwd_1) \xrightarrow{f} pwd_2 \xrightarrow{H} H(pwd_2) \xrightarrow{f} \dots \xrightarrow{f} pwd_{t-1} \xrightarrow{H} H(pwd_{t-1}) \xrightarrow{f} pwd_t \xrightarrow{H} H(pwd_t)$$
- $f$  este o funcție de mapare a valorilor hash în parolă;
- Atenție! Funcția  $f$  nu este inversa funcției  $H$  (s-ar pierde proprietatea de *unidirectionalitate* a funcției hash)
- Se memorează doar capetele lanțurilor, valorile intermediare se generează la nevoie;
- Dacă  $t$  este suficient de mare, atunci capacitatea de stocare scade semnificativ;



- Algoritmul de determinare a unei parolă:

  1. Se caută valoarea hash a parolăi în lista de valori hash a tabelei stocate;
    - 1.1 Dacă se găsește, atunci lanțul căutat este acesta și se trece la pasul 2;
    - 1.2 Dacă nu se găsește, se reduce valoarea hash prin funcția de reducere  $f$  încrănată și se aplică funcția  $H$  și se reia căutarea de la pasul 1;
  2. Se generează întreg lanțul plecând de la valoarea inițială stocată. Parola corespunzătoare este cea situată în lanț înainte de valoarea hash căutată.

## Salting

- ▶ Pentru a evita atacurile precedente, se utilizează tehnica de **salting**;
  - ▶ La înregistrare, se stochează pentru fiecare utilizator:  
 $(username, salt, H(pwd||salt))$
  - ▶ *salt* este o secvență aleatoare de  $n$  biți, distinctă pentru fiecare utilizator;
  - ▶ Adversarul nu poate precalcula valorile hash înainte de a obține acces la fișierul de parole...
  - ▶ ... decât dacă folosește  $2^n$  valori posibile *salt* pentru fiecare parolă;
  - ▶ Atacurile devin deci impracticabile pentru  $n$  suficient de mare.
- ▶ În plus, în practică se folosesc funcții hash lente; ▶ Astfel verificarea unui număr mare de parole devine impracticabilă în timp real; ▶ Un alt avantaj introdus de tehnica de salting este că desigură că 2 utilizatori folosesc aceeași parola, valorile stocate sunt diferite: (Alice, 1652674,  $H(parolatest || 1652674)$ ) (Bob, 3154083,  $H(parolatest || 3154083)$ ) ▶ Prin simpla citire a fișierului de parole, adversarul nu își poate da seama că 2 utilizatori folosesc aceeași parola.

### Parole de unică folosintă

- ▶ Cazul extrem de schimbare periodică a parolei îl reprezintă parolile de unică folosintă; ▶ Utilizatorul folosește o listă de parole, la fiecare logare utilizând următoarea parolă din listă; ▶ Această listă se calculează pornind de la o valoare  $x$  folosind o funcție hash  $H$ ; ▶ Se consideră o listă de 3 parole de unică folosintă:  $P_0 = H(H(H(x)))$ ,  $P_1 = H(H(H(x)))$ ,  $P_2 = H(H(x))$ ,  $P_3 = H(x)$
- ▶ La înregistrare, în fișierul de parole se păstrează tripletul: (*username*, 1,  $P_0 = H(H(H(x))))$ ) ▶ Utilizatorul introduce o parolă  $P_1$  și autentificarea se realizează cu succes dacă  $H(P_1) = P_0$ ; ▶ Se actualizează fișierul de parole cu noua parolă introdusă: (*username*, 2,  $P_1 = H(H(H(x)))$ ) ▶ Procesul continua până se ajunge la  $H(x)$ ; ▶ Fiind cunoscută o parolă din secvență, se poate calcula imediat parola anterioară, dar NU se poate calcula parola următoare.

### Criptografie cu cheie publică (asimetrică)

Limitările criptografiei simetrice ▶ Am studiat până acum criptografia simetrică; ▶ Aceasta asigură confidențialitatea și integritatea mesajelor transmise pe canale nesecurizate; ▶ Însă rămân multe probleme nerezolvate...

### Problema 1 - Distribuirea cheilor

- ▶ Criptarea simetrică necesită o cheie secretă comună partilor comunicante; ▶ întrebare: Cum se obțin și se distribuie aceste chei?
- ▶ Varianta 1. Se transmit printr-un canal de comunicație nesecurizat; ▶ NU! Un adversar pasiv le poate intercepta și utiliza ulterior pentru decriptarea comunicației.
- ▶ Varianta 2. Se transmit printr-un canal de comunicație sigur care presupune un serviciu de mesagerie de încredere; ▶ O opțiune posibilă la nivel guvernamental sau militar; ▶ Dar nu va fi niciodată posibilă în cazul organizațiilor numeroase; ▶ Presupunem doar cazul în care fiecare manager trebuie să partajeze o cheie secretă cu fiecare subordonat; ▶ Problemele care apar sunt multiple: pentru fiecare nou angajat este necesară stabilirea cheilor, organizația are sedii în mai multe țări, ...

## Problema 2 - Stocarea secretă a cheilor

- ▶ Rămânem la exemplul anterior al unei organizații numeroase cu  $N$  angajați;
- ▶ Întrebare: Câte chei sunt necesare pentru ca fiecare 2 angajați să poată comunica criptat?
- ▶ Răspuns:  $C_N^2 = N(N - 1)/2$ ;
- ▶ La acestea se adaugă cheile necesare pentru accesul la resurse (servere, imprimante, baze de date ...);
- ▶ Apare o problemă de **logistică**: foarte multe chei sunt dificil de menținut și utilizat;
- ▶ și apare o problemă de **securitate**: cu cât sunt mai multe chei, cu atât sunt mai dificil de stocat în mod sigur, deci cresc şansele de a fi furate de adversari;

Problema 3 - Medii de comunicare deschise

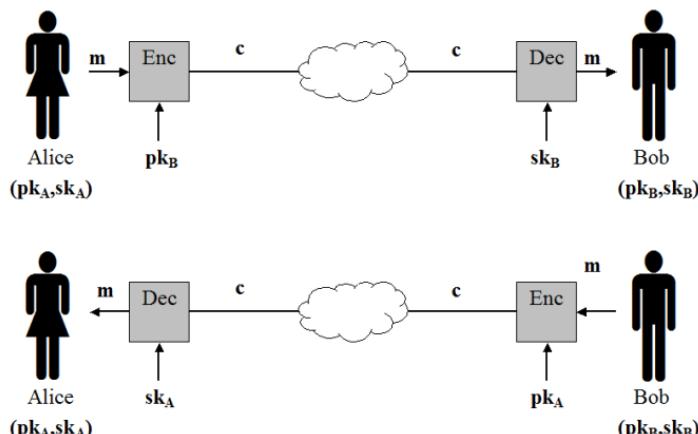
- ▶ Desi dificil de stocat sau utilizat, criptografia simetrică ar putea (cel puțin în teorie) să rezolve aceste probleme;
- ▶ Dar este insuficientă în medii deschise, în care participanții nu se întâlnesc niciodată;
- ▶ Astfel de exemple includ: o tranzacție prin internet sau un e-mail transmis unei persoane necunoscute;

Problema 4 - Imposibilitatea non-repudierii

- ▶ O cheie simetrică este detinută de cel puțin 2 parti;
- ▶ Este imposibil de demonstrat de exemplu că un MAC a fost produs de una dintre cele 2 parti comunicante;
- ▶ De aceea nu se poate utiliza autentificarea simetrică pentru a atesta sursa unui mesaj sau document.

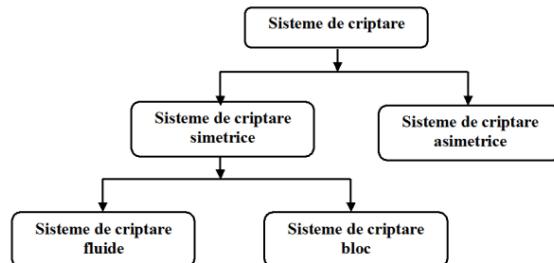
**Criptografia asimetrică**

- ▶ Criptografia cu cheie publică este introdusă de W. Diffie și M. Hellman în 1976 ca o soluție la problemele enumerate anterior:



## Sisteme de criptare asimetrice

- ▶ Am studiat sisteme de criptare care folosesc **aceeași cheie** pentru criptare și decriptare - [sisteme de criptare simetrice](#);
- ▶ Vom studia sisteme de criptare care folosesc **chei diferite** pentru criptare și decriptare - [sisteme de criptare asimetrice](#);



**Terminologie** ▶ Spre deosebire de criptarea simetrică, criptarea asimetrică folosește o pereche de chei:

► Cheia publică pk este folosită pentru criptare; ► Cheia secreta sk este folosită pentru decriptare; ► Cheia publică este larg răspândită pentru a asigura posibilitatea de criptare oricui dorește să transmită un mesaj către entitatea careia îi corespunde; ► Cheia secreta este privată și nu se cunoaște decât de entitatea careia îi corespunde; ► Considerăm (pentru simplitate) ca ambele chei au lungimea cel puțin n biti.

### Criptografia simetrică

- ▶ necesită secretizarea întregii chei
- ▶ folosește aceeași cheie pentru criptare și decriptare
- ▶ rolurile emițătorului și ale receptorului pot fi schimbate
- ▶ pentru ca un utilizator să primească mesaje criptate de la mai mulți emițători, trebuie să partajeze cu fiecare câte o cheie

### Avantaje

- ▶ număr mai mic de chei
- ▶ simplifică problema distribuirii cheilor
- ▶ fiecare participant trebuie să stocheze o singură cheie secretă de lungă durată
- ▶ permite comunicarea sigură pe canale publice
- ▶ rezolvă problema mediilor de comunicare deschise

### Criptografia asimetrică

- ▶ necesită secretizarea unei jumătăți din cheie
- ▶ folosește chei distincte pentru criptare și decriptare
- ▶ rolurile emițătorului și ale receptorului nu pot fi schimbate
- ▶ o pereche de chei asimetrice permite oricui să transmită informație criptată către entitatea căreia îi corespunde

### Dezavantaje

- ▶ criptarea asimetrică este mult mai lentă decât criptarea simetrică
- ▶ compromiterea cheii private conduce la compromiterea tuturor mesajelor criptate primite, indiferent de sursă
- ▶ necesită verificarea autenticității cheii publice (PKI rezolvă această problemă)

## Teoria numerelor pentru criptografie

- ▶  $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ▶  $\mathbb{N} = \{0, 1, 2, \dots\}$
- ▶  $\mathbb{Z}_+ = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ▶ Pentru  $a, n \in \mathbb{N}$  notăm  $\gcd(a, n)$  ca fiind cel mai mare divizor comun (greatest common divisor) al lui  $a$  și  $n$ .
- ▶ **Exemplu:**  $\gcd(30, 50) = 10$ .
- ▶ pentru  $n \in \mathbb{Z}_+$  notăm
  - ▶  $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$
  - ▶  $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$
  - ▶  $\phi(n) = |\mathbb{Z}_n^*|$
- ▶ **Exemplu:**  $n=12$ 
  - ▶  $\mathbb{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
  - ▶  $\mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$
  - ▶  $\phi(12) = 4$
- ▶  $a = qn + r$  cu  $0 \leq r < n$ .  
Considerăm împărțirea lui  $a$  la  $n$ .  
Atunci  $q$  este catul împărțirii iar  $r$  este restul și notăm

$$a \bmod n = r$$

- ▶ **Exemplu:**  $17 \bmod 3 = 2$ .
- ▶  $a = b \bmod n$  dacă  $a \bmod n = b \bmod n$ .

## Grupuri și invers

- ▶ Dacă  $n \in \mathbb{Z}_+$  atunci  $G = \mathbb{Z}_n^*$  împreună cu operația "\*" definită

$$a * b = ab \bmod n$$

pentru  $a, b \in G$  formează un grup și are cele trei proprietăți:

- ▶ asociativitatea: operația \* este asociativă
- ▶ element neutru: există un element  $1 \in G$  așa încât  $a * 1 \bmod n = 1 * a \bmod n = a, \forall a \in G$ .
- ▶ element inversabil: pentru orice  $a \in G$  există un unic  $b \in G$  așa încât  $a * b = b * a = 1 \bmod n$ .  
 $b$  se numește inversul lui  $a$  și îl notăm cu  $a^{-1} \bmod n$ .
- ▶ **Exemplu:**  $5^{-1} \bmod 12$  este acel număr  $b \in G$  care satisfacă  $5b \bmod 12 = 1$ 
  - ▶ deci  $b = 5$ .

- ▶ calculati  $5 * 8 * 10 * 16 \bmod 21$ . ▶ Prima varianta: Calculam mai intai  $5 * 8 * 10 * 16 = 6400$  si apoi calculam  $6400 \bmod 21 = 16$  ▶ A doua varianta (mai rapida): ▶  $5 * 8 \bmod 21 = 40 \bmod 21 = 19$  ▶  $19 * 10 \bmod 21 = 190 \bmod 21 = 1$  ▶  $1 * 16 \bmod 21 = 16$

## Ordinul unui grup

- ▶ Ordinul unui grup  $G$  este numărul de elemente din acel grup, îl notăm cu  $|G|$ .
- ▶ **Exemplu:** Ordinul lui  $\mathbb{Z}_{21}^* = 12$  pentru că

$$\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

- ▶ Fie  $G$  un grup de ordin  $m$  și  $a \in G$ . Atunci:
  - ▶  $a^m = 1$ .
  - ▶ Pentru orice  $i \in \mathbb{Z}$ ,  $a^i = a^{i \bmod m}$ .
  - ▶ **Exemplu:** Calculați  $5^{74} \bmod 21$ .
  - ▶ **Răspuns:** Fie  $\mathbb{Z}_{21}^*$  și  $a = 5$ . Atunci  $m = 12$  și
  - ▶  $5^{74} \bmod 21 = 5^{74 \bmod 12} \bmod 21 = 5^2 \bmod 21 = 4$ .

### Prezumtii criptografice dificile

#### Prezumptii criptografice dificile

- ▶ Criptografia modernă se bazează pe prezumptia că *anumite* probleme nu pot fi rezolvate în timp polinomial;
- ▶ Până acum am văzut că schemele de criptare și de autentificare se bazează pe prezumptia existenței permutărilor pseudoaleatoare;
- ▶ Dar această presupție e nenaturală și foarte puternică;
- ▶ În practică, PRF pot fi instanțiate cu cifruri bloc;
- ▶ Însă metode pentru a demonstra pseudoaleatorismul construcțiilor practice relativ la alte prezumptii "mai rezonabile" nu se cunosc;
- ▶ Dar e posibil să demonstreze existența permutărilor pseudoaleatoare pe baza unei prezumptii mult mai slabe, cea a existenței funcțiilor one-way;
- ▶ In continuare vom introduce câteva două considerate "dificele": **problema factorizării și problema logaritmului discret** și vom prezenta funcții conjecturate ca fiind one-way bazate pe aceste probleme; ▶ Tot materialul ce urmează se bazează pe noțiuni de teoria numerelor; ▶ La criptografia simetrică (cu cheie secreta) am văzut primitive criptografice (i.e. funcții hash, PRG, PRF) care pot fi construite eficient fără a implica teoria numerelor; ▶ La criptografia asimetrică (cu cheie publică) construcțiile cunoscute se bazează pe probleme matematice dificile din teoria numerelor;

- ▶ O primă problemă conjecturată ca fiind dificilă este problema factorizării numerelor întregi sau mai simplu problema factorizării;
  - ▶ Fiind dat un număr compus  $N$ , problema cere să se găsească două numere prime  $x_1$  și  $x_2$  pe  $n$  biți așa încât  $N = x_1 \cdot x_2$ ;
  - ▶ Cele mai greu de factorizat sunt numerele care au factori primi foarte mari.
- Pentru a putea folosi problema în criptografie, trebuie să generăm numere prime aleatoare în mod eficient; ► Putem genera un număr prim aleator pe  $n$  biti prin alegerea repetată de numere aleatoare pe  $n$  biti până când gasim unul prim; ► Pentru eficiență, ne interesează două aspecte: 1. probabilitatea ca un număr aleator de  $n$  biti să fie prim; 2. cum testăm eficient ca un număr dat să este prim.
- ▶ Pentru distribuția numerelor prime, se cunoaște următorul rezultat matematic:

### Teorema

*Pentru orice  $n > 1$ , proporția de numere prime pe  $n$  biți este de cel puțin  $1/3n$ .*

- ▶ Rezultă imediat că dacă testăm  $t = 3n^2$  numere, probabilitatea ca un număr prim să nu fie ales este cel mult  $e^{-n}$ , deci neglijabilă.
- ▶ Deci avem un algoritm în timp polinomial pentru gasirea numerelor prime

**Testarea primalitatii** ► Cei mai eficienți algoritmi sunt probabilisti: ► Dacă numărul  $p$  dat este prim atunci algoritmul întotdeauna întoarce rezultatul prim; ► Dacă  $p$  este compus, atunci cu probabilitate mare algoritmul va întoarce compus; ► Concluzie: dacă outputul este compus, atunci  $p$  sigur este compus, dacă outputul este prim, atunci cu probabilitate mare  $p$  este prim dar este posibil să se fie produs o eroare; ► Un algoritm determinist polinomial a fost propus în 2002, dar este mai lent decât algoritmii probabilisti; ► Un algoritm probabilist foarte răspândit este Miller-Rabin care acceptă la intrare un număr  $N$  și rulează în timp polinomial în  $|N|$

► Deocamdată nu se cunosc algoritmi polinomiali pentru problema factorizării; ► Dar există algoritmi mult mai buni decât forța bruta; ► Prezentăm în continuare câteva algoritmi de factorizare.

- ▶ **Reamintim:** Fiind dat un număr compus  $N$ , problema factorizării cere să se găsească 2 numere prime  $p$  și  $q$  a.î.  $N = pq$ ;
- ▶ Considerăm  $|p| = |q| = n$  și deci  $n = O(\log N)$ ;
- ▶ Metoda cea mai simplă este împărțirea numărului  $N$  prin toate numerele  $p$  impare din intervalul  $p = 3, \dots, \lfloor \sqrt{N} \rfloor$ .
- ▶ Complexitatea timp este  $O(\sqrt{N} \cdot (\log N)^c)$  unde  $c$  este o constantă, adică exponentială în numărul  $b$  de biti al lui  $N$ ;  $b = \log_2 N$ . Complexitatea este  $O(N^{1/2}) = O((2^{\log_2 N})^{1/2})$
- ▶ Pentru  $N < 10^{12}$  metoda este destul de eficientă.

- ▶ Există însă algoritmi mai sofisticăți, cu timp de execuție mai bun, între care:
  - ▶ Metoda **Pollard p – 1**: funcționează atunci când  $p - 1$  are factori primi "mici";
  - ▶ Metoda **Pollard rho**: timpul de execuție este  $O(N^{1/4} \cdot (\log N)^c)$ , deci tot exponențial în lungimea lui  $N$ ;
  - ▶ **Algoritmul sitei pătratice** - rulează în timp *sub-exponențial* în lungimea lui  $N$ .
- ▶ Deocamdată, cel mai rapid algoritm de factorizare este o îmbunătățire a sitei pătratice care factorizează un număr  $N$  de lungime  $O(n)$  în timp  $2^{O(n^{1/3} \cdot (\log n)^{2/3})}$ .

**RSA Challenge** ▶ În 1991, Laboratoarele RSA lansează RSA Challenge; ▶ Aceasta presupune factorizarea unor numere  $N$ , unde  $N = p \cdot q$ , cu  $p, q$  2 numere prime mari; ▶ Au fost lansate mai multe provocări, cate 1 pentru fiecare dimensiune (în biti) a lui  $N$ : ▶ Exemple includ: RSA-576, RSA-640, RSA-768, ..., RSA-1024, RSA-1536, RSA-2048; ▶ Provocarea s-a încheiat oficial în 2007; ▶ Multe provocări au fost sparte în cursul anilor (chiar și ulterior închiderii oficiale), însă există numere încă nefactorizate:

## 2. Problema logaritmului discret

- ▶ O altă prezumție dificilă este **DLP** (Discrete Logarithm Problem) (sau **PLD** (Problema Logaritmului Discret));
- ▶ Considerăm  $\mathbb{G}$  un grup ciclic de ordin  $q$ ;
- ▶ Există un generator  $g \in \mathbb{G}$  a.î.  $\mathbb{G} = \{g^0, g^1, \dots, g^{q-1}\}$ ;
- ▶ Echivalent, pentru fiecare  $h \in \mathbb{G}$  există un *unic*  $x \in \mathbb{Z}_q$  a.î.  $g^x = h$ ;
- ▶  $x$  se numește **logaritmul discret** al lui  $h$  în raport cu  $g$  și se notează

$$x = \log_g h$$

### Experimentul logaritmului discret $DLog_{\mathcal{A}}(n)$

1. Generează  $(\mathbb{G}, q, g)$  unde  $\mathbb{G}$  este un grup ciclic de ordin  $q$  (cu  $|q| = n$ ) iar  $g$  este un generator al lui  $\mathbb{G}$ .
2. Alege  $h \leftarrow^R \mathbb{G}$ . (se poate alege  $x' \leftarrow^R \mathbb{Z}_q$  și apoi  $h := g^{x'}$ .)
3.  $\mathcal{A}$  primește  $\mathbb{G}, q, g, h$  și întoarce  $x \in \mathbb{Z}_q$ ;
4. Output-ul experimentului este 1 dacă  $g^x = h$  și 0 altfel.

#### Definiție

Spunem că problema logaritmului discret (DLP) este dificilă dacă pentru orice algoritm PPT  $\mathcal{A}$  există o funcție neglijabilă negl așa încât

$$\Pr[DLog_{\mathcal{A}}(n) = 1] \leq negl(n)$$

Grupuri ciclice de ordin prim ► Există câteva clase de grupuri ciclice pentru care DLP este considerată dificila; ► Una dintre ele este clasa grupurilor ciclice de ordin prim (în aceste grupuri, problema este "cea mai dificila"); ► DLP nu poate fi rezolvată în timp polinomial în grupurile care nu sunt de ordin prim, ci doar este mai usoara; ► În aceste grupuri căutarea unui generator și verificarea că un număr dat este generator sunt triviale.

- DLP este considerată dificilă în grupuri ciclice de forma  $\mathbb{Z}_p^*$  cu  $p$  prim;
  - Însă pentru  $p > 3$  grupul  $\mathbb{Z}_p^*$  NU are ordin prim;
  - Aceasta problemă se rezolvă folosind un *subgrup* potrivit al lui  $\mathbb{Z}_p^*$ ;
- Cel mai rapid algoritm de factorizare necesită timp sub-exponential; ► Problema logaritmului discret este dificila.

### Curs 9 - Notiuni de securitate în criptografia asimetrică, Criptare hibridă, RSA, PKCS

Securitate perfectă ► Incepem studiul securitatii în același mod în care am început la criptografia simetrică: cu securitatea perfectă; ► Definiția e analoagă cu diferența că adversarul cunoaște, în afara textului criptat, și cheia publică;

- **Întrebare:** Securitatea perfectă este posibilă în cadrul criptografiei cu cheie publică?
- **Răspuns:** NU! Indiferent lungimea cheilor și a mesajelor;
- Având  $pk$  și un text criptat  $c = Enc_{pk}(m)$ , un adversar nelimitat computațional poate determina mesajul  $m$  cu probabilitate 1.

#### Indistinctibilitatea

- Indistinctibilitatea în criptografia cu cheie publică este corespondența noțiunii similare din criptografia cu cheie secretă;
- Vom defini această noțiune pe baza unui experiment de indistinctibilitate  $PubK_{\mathcal{A}, \pi}^{eav}(n)$  unde  $\pi = (Enc, Dec)$  este schema de criptare iar  $n$  este parametrul de securitate al schemei  $\pi$ ;
- Personaje participante: **adversarul**  $\mathcal{A}$  care încearcă să spargă schema și un **challenger**.

#### Definiție

O schemă de criptare  $\pi = (Enc, Dec)$  este indistinctibilă în prezența unui atacator pasiv dacă pentru orice adversar  $\mathcal{A}$  există o funcție neglijabilă negl astfel încât

$$\Pr[PubK_{\mathcal{A}, \pi}^{eav}(n) = 1] \leq \frac{1}{2} + negl(n).$$

## Securitate pentru interceptare simplă

- ▶ Principala diferență față de definiția similară studiată la criptografia cu cheie secretă este că  $\mathcal{A}$  primește cheia publică  $pk$ ;
- ▶ Adică  $\mathcal{A}$  primește acces *gratuit* la un oracol de criptare, ceea ce înseamnă că el poate calcula  $Enc_{pk}(m)$  pentru orice  $m$ ;
- ▶ Prin urmare, definiția este echivalentă cu cea pentru securitate CPA (nu mai este necesar oracolul de criptare pentru că  $\mathcal{A}$  își poate crita singur mesajele);
- ▶ Reamintim că în criptografia simetrică există scheme indistinctibil sigure dar care nu sunt CPA-sigure .

### Insecuritatea schemelor deterministe

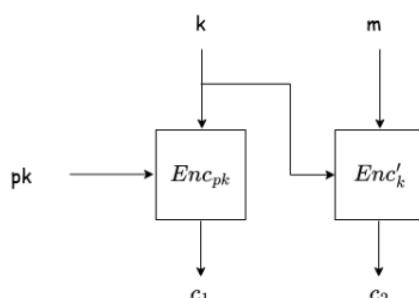
- ▶ După cum am vazut la criptografia simetrică, nici o schema determinista nu poate fi CPA sigură;
- ▶ Datorita echivalentei între notiunile de securitate CPA și indistinctibilitate pentru interceptare simplă (în criptografia asimetrică) concluzionăm că: **Teorema** Nici o schema de criptare cu cheie publică determinista nu poate fi semantic sigură pentru interceptarea simplă.

**Securitate CCA** ▶ Notiunea de securitate CCA ramane identica cu cea de la sistemele simetrice; ▶ Capabilitatile adversarului: el poate interactiona cu un oracol de decriptare, fiind un adversar activ care poate rula atacuri în timp polinomial; ▶ Adversarul poate transmite catre oracolul de decriptare anumite mesaje și primește înapoi mesajul clar corespunzător; ▶ Ca și în cazul securitatii CPA, adversarul nu mai necesita acces la oracolul de criptare pentru ca detine cheia publică  $pk$  și poate realiza singur criptarea oricărui mesaj  $m$ .

▶ **In criptografia cu cheie publică:** ▶ NU există securitate perfectă ▶ indistinctibilitate = securitate CPA

### Criptarea hibridă

- ▶ Criptarea cu cheie secreta este mult mai rapida decât criptarea cu cheie publică
- ▶ Pentru mesajele care sunt suficient de lungi, se foloseste criptare cu cheie secreta în tandem cu criptarea cu cheie publică;
  - ▶ Rezultatul acestei combinații se numește **criptare hibridă** și este folosită extensiv în practică;



- ▶ Pentru criptarea unui mesaj  $m$ , se urmează doi pași:
  1. Expeditorul alege aleator o cheie  $k$  pe care o criptează folosind cheia publică a destinatarului, rezultând  $c_1 = Enc_{pk}(k)$ ; Numai destinatarul va putea decripta  $k$ , ea rămânând secretă pentru un adversar;
  2. Expeditorul criptează  $m$  folosind o schemă de criptare cu cheie secretă ( $Enc'$ ,  $Dec'$ ) cu cheia  $k$ , rezultând  $c_2 = Enc'_k(m)$ ;
- ▶ Mesajul criptat este  $c = (c_1, c_2)$ ;

### Teoremă

Dacă  $\Pi$  este o schemă de criptare cu cheie publică CPA-sigură iar  $\Pi'$  este o schemă de criptare cu cheie secretă sigură semantic, atunci construcția hibridă  $\Pi^{hyb}$  este o schemă de criptare cu cheie publică CPA-sigură.

- ▶ Este suficient ca  $\Pi'$  să satisfacă noțiunea mai slabă de securitate semantică (care nu implică securitate CPA)...
  - ▶ ...deoarece cheia secretă  $k$  este una "nouă" și aleasă aleator de fiecare dată când se criptează un mesaj;
  - ▶ Cum o cheie  $k$  este folosită o singură dată, e suficientă noțiunea de securitate la interceptare simplă pentru securitatea schemei hibrile.
- ▶ Pentru criptarea mesajelor lungi, în practică se folosește criptarea hibridă ▶ Aceasta îmbina avantajele criptării simetrice și criptării asimetrice

### RSA

#### Functii one-way

- ▶ Reprezintă o primitivă criptografică minima, necesară și suficientă pentru criptarea cu cheie secreta dar și pentru codurile de autentificare a mesajelor
- ▶ O funcție  $f$  one-way este ușor de calculat și dificil de inversat

#### Definiție

O funcție  $f : \{0,1\}^* \rightarrow \{0,1\}^*$  este one-way dacă următoarele două condiții sunt îndeplinite:

1. Ușor de calculat: Există un algoritm polinomial pentru calculul lui  $f$
2. Dificil de inversat: Pentru orice algoritm polinomial  $\mathcal{A}$ , există o funcție neglijabilă negl asa incat

$$\Pr[\text{Invert}_{\mathcal{A},f}(n) = 1] \leq \text{negl}(n)$$

▶ Problema factorizării numerelor mari în produs de două numere prime de aceeași lungime este one-way ... ▶ ... însă nu poate fi folosită direct pentru criptografie ▶ În schimb, introducem o problemă apropiată de problema factorizării pe baza căreia putem construi sisteme de criptare

#### Problema RSA

- ▶ Problema RSA se bazează pe dificultatea factorizării numerelor mari:  $N = p \cdot q$ ,  $p$  și  $q$  prime;
- ▶ Fie  $\mathbb{Z}_N^*$  un grup de ordin  $\phi(N) = (p-1)(q-1)$ ;
- ▶ Dacă se cunoaște factorizarea lui  $N$ , atunci  $\phi(N)$  este ușor de calculat
- ▶ Fixăm  $e$  cu  $\gcd(e, \phi(N)) = 1$ . Atunci  $(x^e)^d = x^{ed} \bmod \phi(N) = x \bmod N = \text{mod } N = (x^d)^e$
- ▶  $x^d$  se numește radacina de ordin  $e$  a lui  $x$  modulo  $N$
- ▶ Dacă  $p$  și  $q$  se cunosc, atunci putem calcula  $\phi(N)$  și  $d = e^{-1} \bmod \phi(N)$
- ▶ Dacă  $p$  și  $q$  nu se cunosc
  - ▶ calculul lui  $\phi(N)$  este la fel de dificil precum factorizarea lui  $N$
  - ▶ calculul lui  $d$  este la fel de dificil precum factorizarea lui  $N$

## Experimentul RSA $RSA - \text{inv}_{\mathcal{A}, \text{GenRSA}(n)}$

- ▶ Consideram algoritmul GenRSA( $1^n$ ) care are ca output  $(N, e, d)$  unde  $ed = 1 \pmod{\phi(N)}$
- ▶ Considerăm experimentul RSA pentru un algoritm  $\mathcal{A}$  și un parametru  $n$ .
  1. Execută GenRSA și obține  $(N, e, d)$ ;
  2. Alege  $y \leftarrow \mathbb{Z}_N^*$ ;
  3.  $\mathcal{A}$  primește  $N, e, y$  și întoarce  $x \in \mathbb{Z}_N^*$ ;
  4. Output-ul experimentului este 1 dacă  $x^e = y \pmod{N}$  și 0 altfel.

### Definiție

Spunem că problema RSA este dificilă cu privire la GenRSA dacă pentru orice algoritm PPT  $\mathcal{A}$  există o funcție neglijabilă negl așa încât

$$\Pr[RSA - \text{inv}_{\mathcal{A}, \text{GenRSA}(n)} = 1] \leq negl(n)$$

- ▶ Prezumăția RSA este că există un algoritm GenRSA pentru care problema RSA este dificilă;
- ▶ Un algoritm GenRSA poate fi construit pe baza unui număr compus împreună cu factorizarea lui;

---

### Algorithm 1 GenRSA

---

**Input:**  $n$

**Output:**  $N, e, d$

- 1: **generează**  $p$  și  $q$  prime pe  $n$ -biti;  $N = p \cdot q$
  - 2:  $\phi(N) = (p-1)(q-1)$
  - 3: **găsește**  $e$  a.î.  $\gcd(e, \phi(N)) = 1$
  - 4: **calculează**  $d := e^{-1} \pmod{\phi(N)}$
  - 5: **return**  $N, e, d$
- 

- ▶ Valoarea lui  $e$  aleasa pare ca nu afecteaza dificultatea problemei RSA
- ▶ În practică se folosește  $e = 3$  sau  $e = 16$  pentru exponențiere eficientă
- ▶ Daca  $N$  este usor de factorizat, atunci problema RSA este usoara
- ▶ Pentru ca problema RSA să poată fie dificilă, trebuie ca  $N$ -ul ales în GenRSA să fie dificil de factorizat în produs de două numere prime;
- ▶ Nu se cunoaște nici o dovedă că nu există o altă metodă de a rezolva problema RSA care să nu implice calculul lui  $\phi(N)$  sau al lui  $d$ .

**Exemplu**

- ▶ Presupunem  $(N, p, q) = (143, 11, 13)$ . Atunci  $\phi(N) = 120$ .
- ▶ Alegem  $e$  asa incat  $\gcd(e, \phi(N)) = 1$ , fie  $e = 7$ .
- ▶ Calculăm  $d = e^{-1} \bmod \phi(N)$  si obtinem  $d = 103$ . Deci output-ul algoritmului GenRSA este  $(143, 7, 103)$ .

## Textbook RSA

- ▶ Definim sistemul de criptare *Textbook RSA* pe baza problemei prezentată anterior;
  1. Se rulează GenRSA pentru a determina  $N, e, d$ .
    - ▶ Cheia publică este:  $pk = (N, e)$ ;
    - ▶ Cheia privată este  $sk = d$ ;
  2. **Enc:** dată o cheie publică  $(N, e)$  și un mesaj  $m \in \mathbb{Z}_N$ , întoarce  $c = m^e \bmod N$ ;
  3. **Dec:** dată o cheie secretă  $(N, d)$  și un mesaj criptat  $c \in \mathbb{Z}_N$ , întoarce  $m = c^d \bmod N$ .
- ▶ Sistemul de criptare este corect pentru ca  
 $\text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m$  astfel:  
 $(m^e)^d \bmod N = m^{ed \bmod \phi(N)} \bmod N = m^1 \bmod N = m$

### Problema 1: Determinismul

- ▶ Intrebare: Este Textbook RSA CPA-sigur sau CCA-sigur? ▶ Raspuns: NU! Sistemul este determinist, deci nu poate rezista definitiilor de securitate!

### Problema 2: Utilizarea multiplă a modulului

- ▶ Cunoscând  $e, d, N$  cu  $(e, \phi(N)) = 1$  se poate determina eficient factorizarea lui  $N$ ;
- ▶ **Întrebare:** Este corect să se utilizeze mai multe perechi de chei care folosesc același modul?
- ▶ **Răspuns:** NU! Fie 2 perechi de chei:
 
$$pk_1 = (N, e_1); sk_1 = (N, d_1)$$

$$pk_2 = (N, e_2); sk_2 = (N, d_2)$$
- ▶ Posesorul perechii  $(pk_1, sk_1)$  factorizează  $N$ , apoi determină  $d_2 = e_2^{-1} \bmod \phi(N)$ .

- ▶ RSA este cel mai cunoscut și mai utilizat algoritm cu cheie publică; ▶ Textbook RSA NU trebuie utilizat!

### PKCS

#### Padded RSA!

- ▶ Am vazut că Textbook RSA este nesigur; ▶ Eliminam una dintre problemele principale, aceea este că sistemul este determinist; ▶ Introducem în acest sens Padded RSA; ▶ Ideea este să se adauge un număr aleator (pad) la mesajul clar înainte de criptare; ▶ Notam în continuare cu  $n$  parametrul de securitate (conform GenRSA).

1. Se rulează GenRSA pentru a determina  $N, e, d$ .
  - ▶ Cheia publică este:  $(N, e)$ ;
  - ▶ Cheia privată este  $(N, d)$ ;
  
2. **Enc**: dată o cheie publică  $(N, e)$  și un mesaj  $m \in \{0, 1\}^{l(n)}$ , alege  $r \leftarrow^R \{0, 1\}^{|N|-l(n)-1}$ , interpretează  $r||m$  ca un element în  $\mathbb{Z}_N$  și întoarce  $c = (r||m)^e \pmod N$ ;
  
3. **Dec**: dată o cheie secretă  $(N, d)$  și un mesaj criptat  $c \in \mathbb{Z}_N$ , calculează  $c^d \pmod N$  și întoarce ultimii  $l(n)$  biți.
  - ▶ Pentru  $l(n)$  foarte mare, atunci este posibil un atac prin forță brută care verifică toate valorile posibile pentru  $r$ ;
  - ▶ Pentru  $l(n)$  mic se obține securitate CPA:

### Teoremare

Dacă problema RSA este dificilă, atunci Padded RSA cu  $l(n) = O(\log n)$  este CPA-sigură.

- ▶ martie 1998 - Laboratoarele RSA introduc PKCS #1 v1.5; ▶ PKCS = Public-Key Cryptography Standard;
- ▶ Foloseste Padded RSA; ▶ Utilizat în HTTPS, SSL/TLS, XML Encryption.

### PKCS #1 v1.5

- ▶ Notăm  $k$  lungimea modulului  $N$  în bytes:  $2^{8(k-1)} \leq N < 2^{8k}$ ;
- ▶ Mesajele  $m$  care se criptează se consideră multiplii de 8 biți, de maxim  $k - 11$  bytes;
- ▶ Criptarea se realizează astfel:

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

- ▶  $r$  este ales aleator, pe  $k - D - 3$  bytes nenuli, unde  $D$  este lungimea lui  $m$  în bytes;

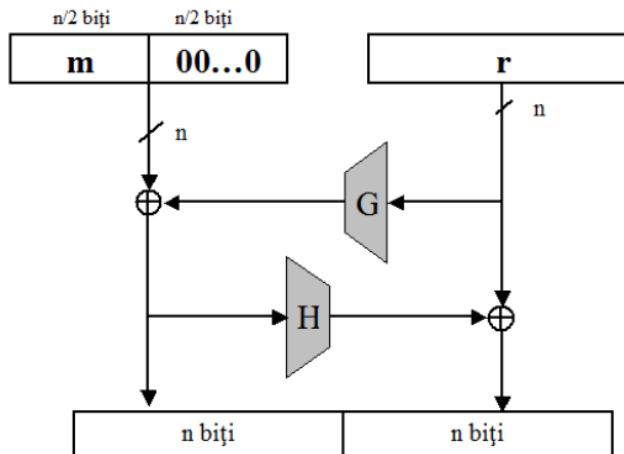
### Securitatea PKCS #1 v1.5

- ▶ Se crede că este CPA-sigur, dar acest lucru nu este demonstrat;
- ▶ Cu siguranță însă nu este CCA-sigur;
- ▶ În 1998, D.Bleichenbacher publică un atac care bazându-se pe faptul că serverul web (HTTPS) întoarce eroare dacă primii 2 octeți nu sunt **02**;
- ▶ Scopul adversarului este să decripteze un text  $c$ ;
- ▶ Adversarul transmite către server  $c' = r^e \cdot c \pmod N$ ;
- ▶ Răspunsul serverului indică adversarului dacă  $c'$  este valid (i.e. începe cu **02**);
- ▶ Adversarul folosește răspunsul primit pentru a determina informații despre  $m$ ;
- ▶ Repetă atacul până determină mesajul  $m$ .

## OAEP

- ▶ octombrie 1998 - Laboratoarele RSA introduc un nou standard PKCS demonstrat CCA-sigur...
  - ▶ ...în modelul  $\mathcal{ROM}$  (Random Oracle Model);
  - ▶ Este vorba despre PKCS #1 v2.0 sau **OAEP = Optimal Asymmetric Encryption Standard**;
  - ▶ OAEP este de fapt o modalitate de padding;
  - ▶ OAEP este o metodă **nedeterministă și inversabilă** care transformă un mesaj  $m$  de lungime  $n/2$  într-o secvență  $m'$  de lungime  $2n$ ;
  - ▶ Notăm  $m' = OAEP(m, r)$ , unde  $r$  este o secvență aleatoare (de lungime  $n$ );
  - ▶ RSA-OAEP cripteză  $m \in \{0, 1\}^{n/2}$  folosind cheia publică  $(N, e)$  ca:
- $$(OAEP(m, r))^e \mod N$$
- ▶ RSA-OAEP decripteză  $c$  folosind cheia secretă  $(N, d)$  ca:

$$(m, r) = OAEP^{-1}(c^d \mod N)$$



- ▶  $G, H =$  funcții hash
- ▶  $m \in \{0, 1\}^{n/2}$  mesajul
- ▶  $r \leftarrow^R \{0, 1\}^n$
- ▶ se obține  $OAEP(m, r)$  pe  $2n$  biți
- ▶ Utilizarea RSA în practică: PKCS #1 v1.5 și PKCS #1 v2.0 (OAEP)

**Curs 10 - Problema logaritmului discret, Schimbul de chei Diffie-Hellman, ElGamal, Criptografia bazata pe curbe eliptice, Schimbul de chei Diffie-Hellman si ElGamal pe curbe eliptice**

- ▶ Reamintim PLD:
  - ▶ Fie  $\mathbb{G}$  un grup ciclic de ordin  $q$  (cu  $|q| = n$ ) iar  $g$  este generatorul lui  $\mathbb{G}$ .
  - ▶ Pentru fiecare  $h \in \mathbb{G}$  există un unic  $x \in \mathbb{Z}_q$  a.î.  $g^x = h$ .
  - ▶ PLD cere găsirea lui  $x$  știind  $\mathbb{G}, q, g, h$ ; notăm  $x = \log_g h$ ;
  - ▶ Atenție! Atunci când  $g^{x'} = h$  pentru un  $x'$  arbitrar (deci NU neapărat  $x \in \mathbb{Z}_q$ ), notăm  $\log_g h = [x' \bmod q]$

- 
- ▶ Problema PLD se poate rezolva, desigur, prin forță brută, calculând pe rând toate puterile  $x$  ale lui  $g$  până când se găsește una potrivită pentru care  $g^x = h$ ;
  - ▶ Complexitatea timp este  $\mathcal{O}(q)$  iar complexitatea spațiu este  $\mathcal{O}(1)$ ;
  - ▶ Dacă se precalculează toate valorile  $(x, g^x)$ , căutarea se face în timp  $\mathcal{O}(1)$  și spațiu  $\mathcal{O}(q)$ ;
  - ▶ Sunt de interes algoritmii care pot obține un timp mai bun la rulare decât forță brută, realizând un compromis spațiu-timp.
  - ▶ Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
    - ▶ algoritmi *generici* care funcționează în grupuri arbitrale (i.e. orice grupuri ciclice);
    - ▶ algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri
  - ▶ Dintre algoritmii generici enumerăm:
    - ▶ Metoda **Baby-step/giant-step**, datorată lui Shanks, calculează logaritmul discret într-un grup de ordin  $q$  în timp  $\mathcal{O}(\sqrt{q} \cdot (\log q)^c)$ ;
    - ▶ pentru  $g \in \mathbb{G}$  generator, elementele lui  $\mathbb{G}$  sunt
$$1 = g^0, g^1, g^2, \dots, g^{q-1}, g^q = 1$$
    - ▶ știm că  $h = g^x$  se află între aceste valori

## Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanță  $t = \lfloor \sqrt{q} \rfloor$  (giant-steps)

$$g^0, g^t, g^{2t}, \dots, g^{\lfloor q/t \rfloor \cdot t}$$

- știm că  $h = g^x$  se află în unul din aceste intervale
- calculând, cu baby-steps, valorile

$$h \cdot g^1, h \cdot g^2, \dots, h \cdot g^t$$

- una din ele va fi egală cu unul din punctele marcate i.e.  $h \cdot g^i = g^{k \cdot t}$
- Complexitatea timp este  $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q))$  iar complexitatea spațiu este  $\mathcal{O}(\sqrt{q})$

► Metoda Baby-Step/Giant-Step este optimă ca timp de rulare, însă există alți algoritmi mai eficienți d.p.d.v. al complexității spațiu; ► Algoritmul Pohlig-Hellman poate fi folosit atunci când se cunoaște factorizarea ordinului  $q$  al grupului iar timpul de rulare depinde de factorii primi ai lui  $q$ ; ► Pentru ca algoritmul să nu fie eficient, trebuie ca cel mai mare factor prim al lui  $q$  să fie de ordinul  $2^{160}$ .

## Algoritmi non-generici pentru calculul logaritmului discret

- Algoritmii non-generici sunt potențial mai puternici decât cei generici;
- Cel mai cunoscut algoritm pentru PLD în  $\mathbb{Z}_p^*$  cu  $p$  prim este algoritmul GNFS (General Number Field Sieve) cu complexitate timp  $2^{\mathcal{O}(n^{1/3} \cdot (\log n)^{2/3})}$  unde  $|p| = \mathcal{O}(n)$ ;
- Există și un alt algoritm non-generic numit *metoda de calcul a indicelui* care rezolvă DLP în grupuri ciclice  $\mathbb{Z}_p^*$  cu  $p$  prim în timp sub-exponențial în lungimea lui  $p$ .
- Această metodă seamănă cu algoritmul sitei pătratice pentru factorizare;

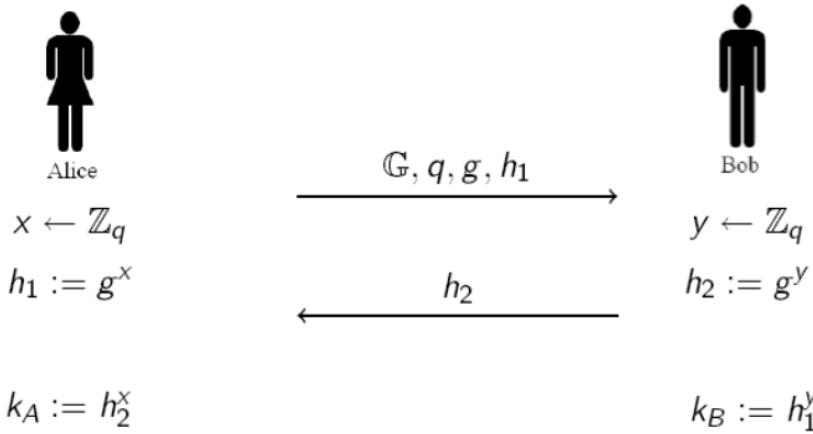
► Cel mai bun algoritm pentru DLP este sub-exponential; ► Se pot construi funcții hash rezistente la coliziuni bazate pe dificultatea DLP

### Schimbul de chei Diffie-Hellman

Primitive cu cheie publică ► Am văzut că bazele criptografiei cu cheie publică au fost puse de Diffie și Hellman în 1976 ... ► ... când au introdus 3 primitive cu cheie publică diferite: 1. sisteme de criptare cu cheie publică 2. semnaturi digitale 3. schimb de chei ► Desi au introdus 3 concepte diferite, Diffie și Hellman au introdus o singură construcție, pentru schimbul de chei.

- Sistemele de criptare cu cheie publică le-am studiat și le vom mai studia în detaliu;
- Semnaturile digitale sunt analogul MAC-urilor din criptografia simetrică (sau corespondentul digital unei semnaturi reale);
- Schimbul de chei îl introducem pentru a facilita introducerea sistemelor de criptare bazate pe DLP. Schimb de chei

- Un protocol de schimb de chei este un protocol prin care 2 persoane care nu partajeaza in prealabil nici un secret pot genera o cheie comună, secreta; ► Comunicarea necesara pentru stabilirea cheii se realizeaza printr-un canal public! ► Deci un adversar poate intercepta toate mesajele transmise pe canalul de comunicatie, dar NU trebuie sa afle nimic despre cheia secreta obtinuta in urma protocolului;
- Protocolele de schimb de chei reprezinta o primitiva fundamentala in criptografie; ► ^In continuare, ne vom rezuma strict la schimbul de chei Diffie-Hellman.



- Alice și Bob doresc să stabilească o cheie secretă comună;
- Alice generează un grup ciclic  $\mathbb{G}$ , de ordin  $q$  cu  $|q| = n$  și  $g$  un generator al grupului;
- Alice alege  $x \leftarrow^R \mathbb{Z}_q$  și calculează  $h_1 := g^x$ ;
- Alice îi trimit lui Bob mesajul  $(\mathbb{G}, g, q, h_1)$ ;
- Bob alege  $y \leftarrow^R \mathbb{Z}_q$  și calculează  $h_2 := g^y$ ;
- Bob îi trimit  $h_2$  lui Alice și întoarce cheia  $k_B := h_1^y$ ;
- Alice primește  $h_2$  și întoarce cheia  $k_A = h_2^x$ .

- Corectitudinea protocolului presupune ca  $k_A = k_B$ , ceea ce se verifică ușor:
- Bob calculează cheia

$$k_B = h_1^y = (g^x)^y = g^{xy}$$

- Alice calculează cheia

$$k_A = h_2^x = (g^y)^x = g^{xy}$$

- ▶ O condiție **minimală** pentru ca protocolul să fie sigur este ca DLP să fie dificilă în  $\mathbb{G}$ ;
  - ▶ **Întrebare:** Cum poate un adversar pasiv să determine cheia comună dacă poate sparge DLP?
  - ▶ **Răspuns:** Ascultă mediul de comunicație și preia mesajele  $h_1$  și  $h_2$ . Rezolvă DLP pentru  $h_1$  și determină  $x$ , apoi calculează  $k_A = k_B = h_2^x$ .
  - ▶ Aceasta nu este însă singura condiție necesară pentru a proteja protocolul de un atacator pasiv!
- 

## CDH (Computational Diffie-Hellman)

- ▶ O condiție mai potrivită ar fi că adversarul să nu poată determina cheia comună  $k_A = k_B$ , chiar dacă are acces la întreaga comunicație;
- ▶ Aceasta este **problema de calculabilitate Diffie-Hellman (CDH)**: Fiind date grupul ciclic  $\mathbb{G}$ , un generator  $g$  al său și 2 elemente  $h_1, h_2 \leftarrow^R \mathbb{G}$ , să se determine:

$$CDH(h_1, h_2) = g^{\log_g h_1 \log_g h_2}$$

- ▶ Pentru schimbul de chei Diffie-Hellman, rezolvarea CDH înseamnă că adversarul determină  $k_A = k_B = g^{xy}$  cunoscând  $h_1, h_2, \mathbb{G}, g$  (toate disponibile pe mediul de transmisiune nesecurizat).
- 

## DDH (Decisional Diffie-Hellman)

- ▶ Nici această condiție nu este suficientă: chiar dacă adversarul nu poate determina cheia exactă, poate de exemplu să determine părți din ea;
- ▶ O condiție și mai potrivită este că pentru adversar, cheia  $k_A = k_B$  să fie **indistinctibilă** față de o valoare aleatoare;
- ▶ Sau, altfel spus, să satisfacă **problema de decidabilitate Diffie-Hellman (DDH)**:

### Definiție

*Spunem că problema decizională Diffie-Hellman (DDH) este dificilă (relativ la  $\mathbb{G}$ ), dacă pentru orice algoritm PPT  $\mathcal{A}$  există o funcție neglijabilă negl astfel încât:*

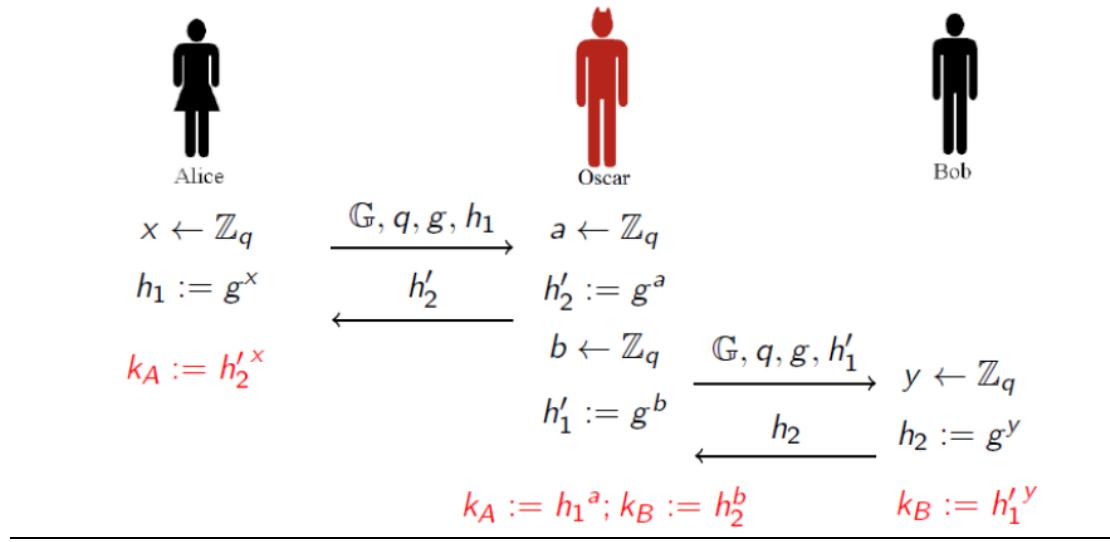
$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leq \text{negl}(n),$$

unde  $x, y, z \leftarrow^R \mathbb{Z}_q$

---

### Atacul Man-in-the-Middle

► Am analizat pana acum securitatea fata de atacatori pasivi; ► Aratam acum ca schimbul de chei Diffie-Hellman este total nesigur pentru un adversar activ ... ► ... care are dreptul de a intercepta, modifica, elimina mesajele de pe calea de comunicatie; ► Un astfel de adversar se poate interpune intre Alice si Bob, dand nastere unui atac de tip Man-in-the-Middle.



- Alice generează un grup ciclic  $\mathbb{G}$ , de ordin  $q$  cu  $|q| = n$  și  $g$  un generator al grupului;
- Alice alege  $x \leftarrow^R \mathbb{Z}_q$  și calculează  $h_1 := g^x$ ;
- Alice îi trimită lui Bob mesajul  $(\mathbb{G}, g, q, h_1)$ ;
- Oscar interceptează mesajul și răspunde lui Alice în locul lui Bob: alege  $a \leftarrow^R \mathbb{Z}_q$  și calculează  $h'_2 := g^a$ ;
- Oscar și Alice dețin acum cheia comună  $k_A = g^{xa}$ ;
- Oscar inițiază, în locul lui Alice, o nouă sesiune cu Bob: alege  $b \leftarrow^R \mathbb{Z}_q$  și calculează  $h'_1 := g^b$ ;
- Bob alege  $y \leftarrow^R \mathbb{Z}_q$  și calculează  $h_2 := g^y$ ;
- Oscar și Bob dețin acum cheia comună  $k_B = g^{yb}$ .

► Atacul este posibil pentru ca Oscar poate impersona pe Alice și pe Bob; ► De fiecare dată cand Alice va transmite un mesaj criptat catre Bob, Oscar îl interceptează și il previne să ajunga la Bob; ► Oscar îl decriptează folosind  $k_A$ , apoi îl recriptează folosind  $k_B$  și îl transmite catre Bob; ► Alice și Bob comunică fără să fie conștienți de existența lui Oscar

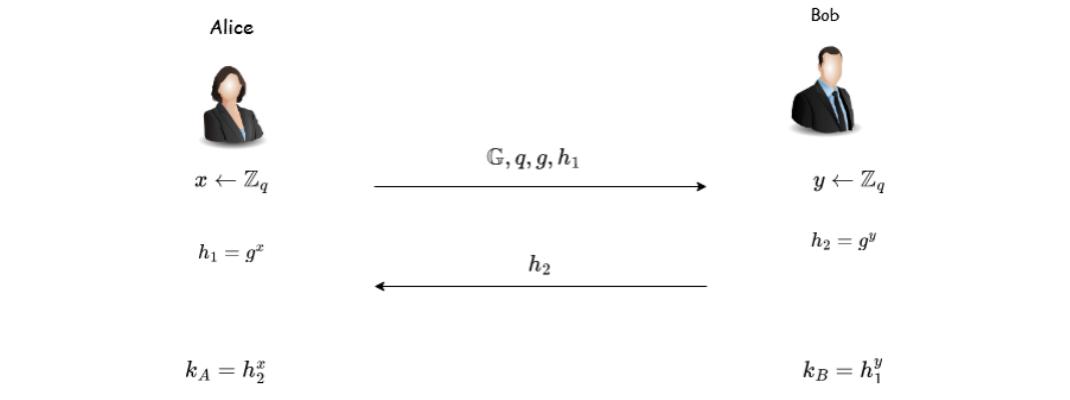
► Schimbul de chei - o primitiva criptografică importantă ► Prezumtii criptografice: CDH, DDH ► Schimbul de chei Diffie-Hellman nu reziste la atacuri active

### Sistemul de criptare ElGamal

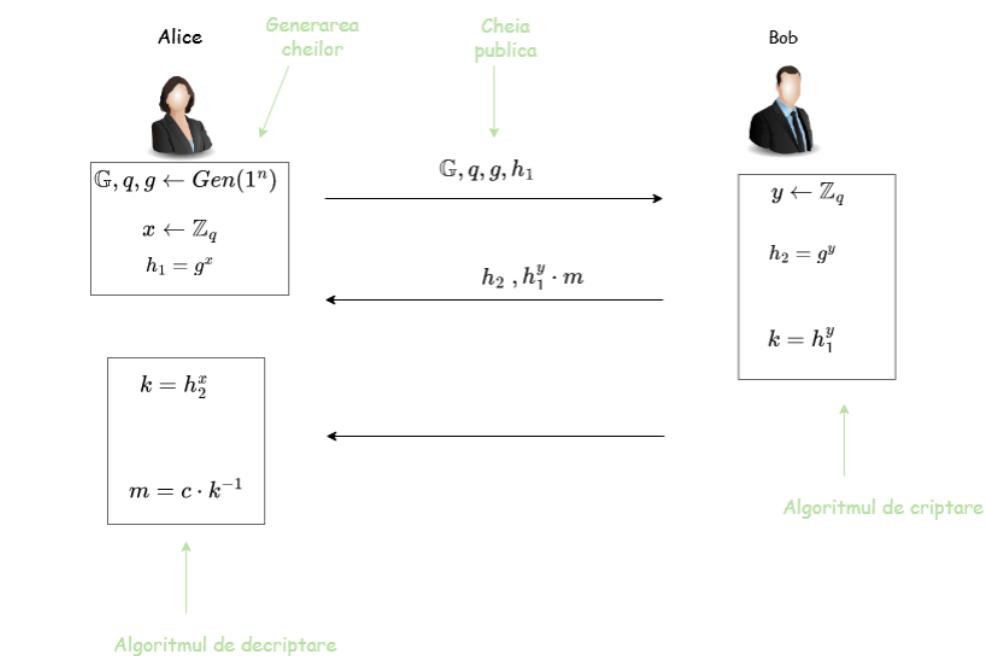
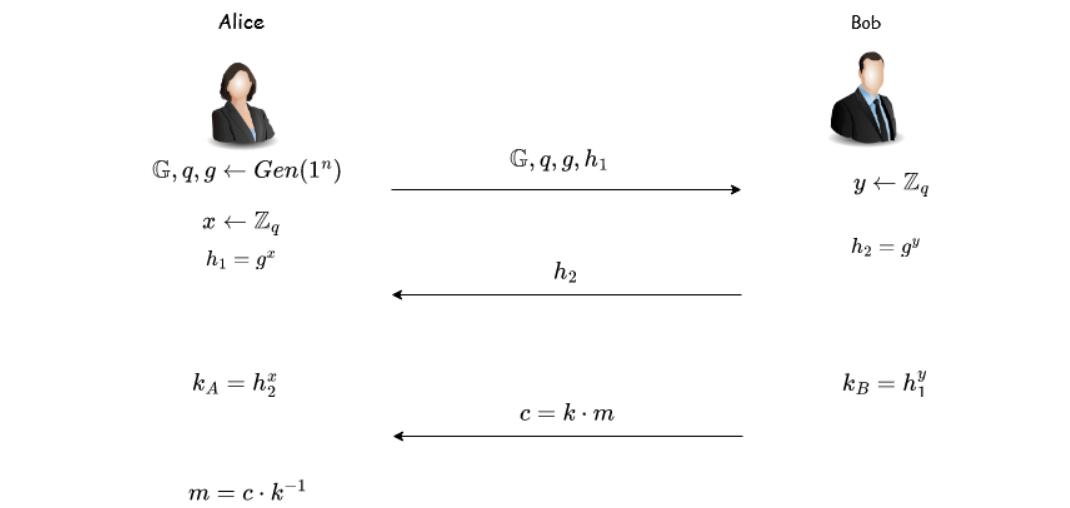
- 1976 - Diffie și Hellman definesc conceptul de criptografie asimetrică;
- 1977 - R. Rivest, A. Shamir și Leonard Adleman introduc sistemul RSA;
- 1985 - T. ElGamal propune un nou sistem de criptare.

## Sistemul de criptare ElGamal

- Reamintim schimbul de chei Diffie-Hellman



- Îl modificăm aşa încât să permită criptare



- Definim sistemul de criptare *ElGamal* pe baza ideii prezentate anterior;
    1. Se generează  $(\mathbb{G}, q, g)$ , se alege  $x \leftarrow^R \mathbb{Z}_q$  și se calculează  $h = g^x$ ;
      - Cheia publică este:  $(\mathbb{G}, q, g, h)$ ;
      - Cheia privată este  $x$ ;
    2. **Enc:** dată o cheie publică  $(\mathbb{G}, q, g, h)$  și un mesaj  $m \in \mathbb{G}$ , alege  $y \leftarrow^R \mathbb{Z}_q$  și întoarce  $c = (c_1, c_2) = (g^y, m \cdot h^y)$ ;
    3. **Dec:** dată o cheie secretă  $(\mathbb{G}, q, g, x)$  și un mesaj criptat  $c = (c_1, c_2)$ , întoarce  $m = c_2 \cdot c_1^{-x}$ .
- 

### Problema 1: Determinismul

- **Întrebare:** Este sistemul ElGamal determinist?
- **Răspuns:** NU! Sistemul este nedeterminist, datorită alegerii aleatoare a lui  $y$  la fiecare criptare.
- Un același mesaj  $m$  se poate cripta diferit, pentru  $y \neq y'$ :

$$c = (c_1, c_2) = (g^y, m \cdot h^y)$$

$$c' = (c'_1, c'_2) = (g^{y'}, m \cdot h^{y'})$$


---

### Problema 2: Dificultatea DLP

- **Întrebare:** Rămâne ElGamal sigur dacă problema DLP este simplă?
- **Răspuns:** NU! Se determină  $x$  a.î.  $h = g^x$ , apoi se decriptează orice mesaj pentru că se cunoaște cheia secretă.

### Problema 3: Proprietatea de homomorfism

- Fie  $m_1, m_2$  2 texte clare și  $c_1 = (c_{11}, c_{12}), c_2 = (c_{21}, c_{22})$  textele criptate corespunzătoare;
  - Atunci:
 
$$c_1 \cdot c_2 = (c_{11} \cdot c_{21}, c_{12} \cdot c_{22}) = (g^{y_1} \cdot g^{y_2}, m_1 h^{y_1} \cdot m_2 h^{y_2})$$
  - **Întrebare:** Dacă un adversar cunoaște  $c_1$  și  $c_2$  criptările lui  $m_1$ , respectiv  $m_2$ , ce poate spune despre  $c_1 \cdot c_2$ ?
  - **Răspuns:**  $c_1 \cdot c_2$  este criptarea lui  $m_1 \cdot m_2$  folosind  $y = y_1 + y_2$ :
 
$$c_1 \cdot c_2 = (g^{y_1+y_2}, m_1 m_2 h^{y_1+y_2})$$
  - Un sistem de criptare care satisface  $Dec_{sk}(c_1 \cdot c_2) = Dec_{sk}(c_1) \cdot Dec_{sk}(c_2)$  se numește sistem de criptare **homomorfic**.  
(homomorfismul este deseori o proprietate utilă în criptografie)
-

#### Problema 4: Utilizarea multiplă a parametrilor publici

- ▶ Este comun în practică pentru un administrator să fixeze parametrii publici  $(\mathbb{G}, q, g)$ , apoi fiecare utilizator să își genereze doar cheia secretă  $x$  și să publice  $h = g^x$ ;
- ▶ **Întrebare:** Este corect să se utilizeze de mai multe ori aceiași parametrii publici  $(\mathbb{G}, q, g)$ ?
- ▶ **Răspuns:** Se consideră că DA. Cunoașterea parametrilor publici pare să nu conducă la rezolvarea DDH.
- ▶ Atenție! Acest lucru nu se întâmplă și la RSA, unde modulul NU trebuie utilizat de mai multe ori.

#### Teoremă

Dacă problema decizională Diffie-Hellman (DDH) este dificilă în grupul  $\mathbb{G}$ , atunci schema de criptare ElGamal este CPA-sigură.

- ▶ Se poate vedea din securitatea schimbului de chei Diffie-Hellman
- ▶ În forma aceasta, sistemul ElGamal nu este CCA-sigur...pentru că este maleabil  
Însă poate fi modificat astfel încât să fie CCA-sigur

#### Criptografia bazată pe curbe eliptice

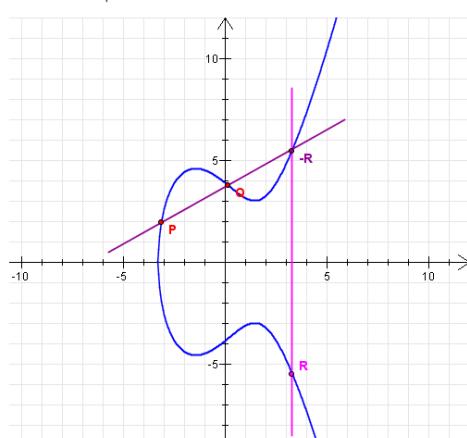
##### Definiție

O curbă eliptică peste  $\mathbb{Z}_p$ ,  $p > 3$  prim, este mulțimea perechilor  $(x, y)$  cu  $x, y \in \mathbb{Z}_p$  așa încât

$$y^2 = x^3 + Ax + B \text{ mod } p$$

împreună cu punctul la infinit  $\mathcal{O}$  unde  $A, B \in \mathbb{Z}_p$  sunt constante care respectă  $4A^3 + 27B^2 \neq 0 \text{ mod } p$

- ▶ Vom nota cu  $E(\mathbb{Z}_p)$  o curbă eliptică definită peste  $\mathbb{Z}_p$   
O curbă eliptică peste spațiul numerelor reale  $\mathbb{R}$   
 $E(\mathbb{R}) : y^2 = x^3 - x + 1$



## Grupul punctelor de pe o curba eliptică

- ▶ Pentru a arăta că punctele de pe o curbă eliptică formează un grup ciclic, definim o operație de grup peste aceste puncte:
  - ▶ Definim operația binară aditivă "+" astfel:
    - ▶ punctul la infinit  $\mathcal{O}$  este element neutru:  $\forall P \in E(\mathbb{Z}_p)$  definim

$$P + \mathcal{O} = \mathcal{O} + P = P.$$

- ▶ fie  $P = (x_1, y_1)$  și  $Q = (x_2, y_2)$  două puncte de pe curbă;  
atunci:
  - ▶ dacă  $x_1 = x_2$  și  $y_2 = -y_1$ ,  $P + Q = \mathcal{O}$
- ▶ altfel,  $P + Q = R$  de coordonate  $(x_3, y_3)$  care se calculează astfel:

$$\begin{aligned}x_3 &= [m^2 - x_1 - x_2 \text{ mod } p] \\y_3 &= [m(x_1 - x_3) - y_1 \text{ mod } p]\end{aligned}$$

- ▶ iar  $m$  se calculează astfel:
$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \text{ mod } p & \text{dacă } P \neq Q \\ \frac{3x_1^2 + A}{2y_1} \text{ mod } p & \text{dacă } P = Q \end{cases}$$
- ▶ dacă  $P = Q$  și  $y_1 = 0$ , atunci  $P + Q = 2P = \mathcal{O}$
- ▶ În practică, sunt căutate acele curbe eliptice pentru care ordinul grupului ciclic generat este prim;
- ▶ Pentru criptografie, sunt de interes curbe eliptice de ordin mare
- ▶ O curbă eliptică definită peste  $\mathbb{Z}_p$  are aproximativ  $p$  puncte.  
Mai precis [[Teorema lui Hasse](#)]:

$$p + 1 - 2\sqrt{p} \leq \text{card}(E(\mathbb{Z}_p)) \leq p + 1 + 2\sqrt{p}$$

- ▶ Există mai multe clase de curbe eliptice slabă d.p.d.v. criptografic, iar ele trebuie evitate.
- ▶ De pildă, curbe eliptice peste  $\mathbb{Z}_p$  cu  $\text{card}(E(\mathbb{Z}_p)) = p$
- ▶ În practică, se folosesc curbe eliptice standardizate

## Curbe eliptice folosite în practică

Curbe eliptice standardizate, folosite în practică, sigure și cu implementări eficiente:

- ▶ curba eliptică P-256 (sau *secp256r1*) este o curbă eliptică peste  $\mathbb{Z}_p$  cu  $p$  pe 256 biți de forma  $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ . Această curbă are ecuația  $y^2 = x^3 - 3x + B \pmod{p}$  iar  $p$ -ul ales astfel permite o implementare eficientă. Curbele P-384 (*secp384r1*) și P-521 (*secp521r1*) sunt definite în mod analog
- ▶ curba eliptică 25519 este definită peste  $\mathbb{Z}_p$  cu  $p$  pe 255 biți de forma  $p = 2^{255} - 19$  și permite o implementare eficientă. Grupul acestei curbe eliptice nu are ordin prim dar se poate lucra într-un subgrup de ordin mare prim

Curbe eliptice standardizate, folosite în practică, sigure și cu implementări eficiente:

- ▶ *secp256k1* este o curbă eliptică de ordin prim peste  $\mathbb{Z}_p$  cu  $p$  pe 256 biți de forma  $p = 2^{256} - 2^{232} - 2^{29} - 2^{28} - 2^{27} - 2^{26} - 2^{24} - 1$  și are ecuația  $y^2 = x^3 + 7 \pmod{p}$ . Aceasta curbă eliptică este folosită în Bitcoin.

## ECDLP - Problema logaritmului discret pe curbe eliptice

- ▶ ECDLP = Elliptic Curve Discrete Logarithm Problem
- ▶ Putem defini acum DLP în grupul punctelor unei curbe eliptice (ECDLP):
- ▶ Fie  $E$  o curbă eliptică peste  $\mathbb{Z}_p$ , un punct  $P \in \mathbb{Z}_p$  de ordin  $n$  și  $Q$  un element din subgrupul ciclic generat de  $P$ :

$$Q \in [P] = \{sP \mid 1 \leq s \leq n-1\}$$

- ▶ Problema ECDLP cere găsirea unui  $k$  așa încât  $Q = kP$ ;
- ▶ Notație:  $\underbrace{P + P + \dots + P}_{s \text{ ori}} = sP$ .

- ▶ Alegând cu grijă curbele eliptice, cel mai bun algoritm pentru rezolvarea ECDLP este considerabil mai slab decât cel mai bun algoritm pentru rezolvarea problemei DLP în  $\mathbb{Z}_p^*$ ;
- ▶ De exemplu, algoritmul de calcul al indicelui nu este deloc eficient pentru ECDLP;
- ▶ Pentru anumite curbe eliptice, singurii algoritmi de rezolvare sunt algoritmii generici pentru DLP, adică metoda baby-step giant-step și metoda Pollard rho;
- ▶ Cum numărul de pași necesari pentru un astfel de algoritm este de ordinul rădăcinii pătrate a cardinalului grupului, se recomandă folosirea unui grup de ordin cel puțin  $2^{160}$ .
- ▶ O consecință a teoremei lui Hasse este că dacă avem nevoie de o curbă eliptică cu  $2^{160}$  elemente, trebuie să folosim un număr prim  $p$  pe aproximativ 160 biți;
- ▶ Deci, dacă folosim o curbă eliptică  $E(\mathbb{Z}_p)$  cu  $p$  pe 160 biți, un atac generic asupra ECDLP are  $2^{80}$  complexitate timp;
- ▶ Un nivel de securitate de 80 biți oferă securitate pe termen mediu;
- ▶ În practică, curbe eliptice peste  $\mathbb{Z}_p$  cu  $p$  până la 256 biți sunt folosite, cu un nivel de securitate pe 128 biți.

## Comparatie între ECC, criptografia simetrică și asimetrică

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

**Figure:** [Understanding cryptography, Christoph Paar, Jan Pelzl, Springer 2010]

- ▶ Un algoritm are nivelul de securitate "security level" pe  $n$  biți dacă cel mai bun atac necesită  $2^n$  pași.
- ▶ Curbele eliptice ofera un suport bun pentru criptografie; ▶ ECDLP este dificila.

## Schimbul de chei Diffie-Hellman pe curbe eliptice

- ▶ Alice și Bob doresc să stabilească o cheie secretă comună;
- ▶ Alice generează o curbă eliptică  $E(\mathbb{Z}_q)$ , și  $P$  un punct pe curbă (generator);
- ▶ Alice alege  $x \leftarrow^R \mathbb{Z}_q$  și calculează  $H_1 := xP$ ;
- ▶ Alice îi trimită lui Bob mesajul  $(E(\mathbb{Z}_q), P, H_1)$ ;
- ▶ Bob alege  $y \leftarrow^R \mathbb{Z}_q$  și calculează  $H_2 := yP$ ;
- ▶ Bob îi trimită  $H_2$  lui Alice și întoarce cheia  $k_B := yH_1$ ;
- ▶ Alice primește  $H_2$  și întoarce cheia  $k_A = xH_2$ .
- ▶ Corectitudinea protocolului presupune ca  $k_A = k_B$ , ceea ce se verifică ușor:
- ▶ Bob calculează cheia

$$k_B = yH_1 = y(xP) = (xy)P$$

- ▶ Alice calculează cheia

$$k_A = xH_2 = x(yP) = (xy)P$$

- ▶ O condiție **minimală** pentru ca protocolul să fie sigur este ca ECDLP să fie dificilă în  $\mathbb{G}$ ;
- ▶ **Întrebare:** Cum poate un adversar pasiv să determine cheia comună dacă poate sparge ECDLP?
- ▶ **Răspuns:** Ascultă mediul de comunicație și preia mesajele  $H_1$  și  $H_2$ . Rezolvă ECDLP pentru  $H_1$  și determină  $x$ , apoi calculează  $k_A = k_B = xH_2$ .
- ▶ Aceasta nu este însă singura condiție necesară pentru a proteja protocolul de un atacator pasiv!

## ECCDH (Elliptic Curve Computational Diffie-Hellman)

- ▶ O condiție mai potrivită ar fi că adversarul să nu poată determina cheia comună  $k_A = k_B$ , chiar dacă are acces la întreaga comunicație;
- ▶ Aceasta este problema de calculabilitate Diffie-Hellman pe curbe eliptice (ECCDH): Fiind date curba eliptică  $E(\mathbb{Z}_q)$ , un punct  $P$  pe curbă și 2 alte puncte  $H_1, H_2 \leftarrow^R E(\mathbb{Z}_q)$ , să se determine:

$$\text{ECCDH}(H_1, H_2) = (ECDLP(P, H_1)ECDLP(P, H_2))P$$

- ▶ Pentru schimbul de chei Diffie-Hellman, rezolvarea ECCDH înseamnă că adversarul determină  $k_A = k_B = xyP$  cunoscând  $H_1, H_2, E(\mathbb{Z}_q), P$  (toate disponibile pe mediul de transmisiune nesecurizat).

- ▶ Nici această condiție nu este suficientă: chiar dacă adversarul nu poate determina cheia exactă, poate de exemplu să determine părți din ea;
- ▶ O condiție și mai potrivită este ca pentru adversar, cheia  $k_A = k_B$  să fie **indistinctibilă** față de o valoare aleatoare;
- ▶ Sau, altfel spus, să satisfacă **problema de decidabilitate Diffie-Hellman pe curbe eliptice (ECDDH)**:

### Definiție

Spunem că problema decizională Diffie-Hellman (ECDDH) este dificilă (relativ la curba eliptică  $E(\mathbb{Z}_q)$ ), dacă pentru orice algoritm PPT  $\mathcal{A}$  există o funcție neglijabilă negl așa încât:

$$|\Pr[\mathcal{A}(E(\mathbb{Z}_q), P, xP, yP, zP) = 1] - \Pr[\mathcal{A}(E(\mathbb{Z}_q), P, xP, yP, xyP) = 1]| \leq negl(n), \text{ unde } x, y, z \leftarrow^R \mathbb{Z}_q$$

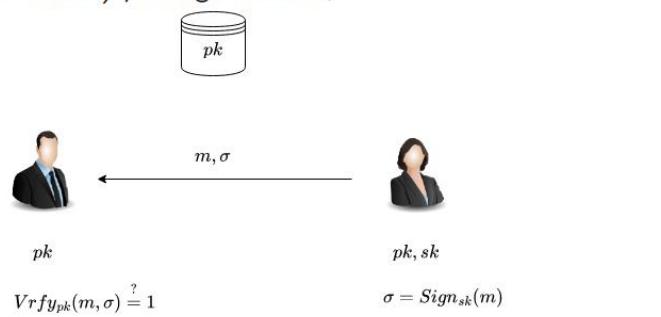
### Important de reținut!

- ▶ Modalitatea de trecere de la o construcție peste  $(\mathbb{Z}_q, \cdot)$  la  $(E(\mathbb{Z}_q), +)$
- ▶ Prezumții criptografice: ECCDH, ECDDH
- ▶ Schimbul de chei Diffie-Hellman pe curbe eliptice păstrează proprietățile schimbului de chei Diffie Hellman definit peste  $\mathbb{G}$  grup ciclic de ordin  $q$

### Curs 11 - Semnaturi digitale și PKI

#### Scheme de semnătură digitală

- ▶ Schemele de semnătură digitală reprezintă echivalentul MAC-urilor în criptografia cu cheie publică, deși există câteva diferențe importante între ele;
- ▶ O schemă de semnătură digitală îi permite unui semnatар S care a stabilit o cheie publică  $pk$  să semneze un mesaj în așa fel încât oricine care cunoaște cheia  $pk$  poate verifica originea mesajului (ca fiind S) și integritatea lui;



Aplicații ale semnaturilor digitale ▶ De pilda, o companie de software vrea să transmită patch-uri de software într-o manieră autentificată astă incat orice client să poată recunoaște dacă un patch e autentic; ▶ În schimb, o persoana malicioasa nu poate pacali un client să accepte un patch care a nu a fost realizat de compania respectiva.

► Pentru aceasta, compania genereaza o cheie publica pk impreuna cu o cheie secreta sk si distribuie cheia pk clientilor sai, pastrand cheia secreta; ► Atunci cand lanseaza un patch de software patch, compania calculeaza o semnatura digitala σ pentru patch folosind cheia sk si trimite fiecarui client perechea (patch, σ); ► Fiecare client stabileste autenticitatea lui patch verificand daca σ este o semnatura legitima pentru patch cu privire la cheia publica pk; ► Deci compania foloseste aceeasi cheie publica pentru toti clientii si calculeaza o singura semnatura pe care o trimite tuturor

#### **Avantaje semnaturi digitale fata de MAC-uri**

► MAC-urile si schemele de semnatura digitala sunt folosite pentru asigurarea integritatii (autenticitatii) mesajelor cu urmatoarele **diferente**: ► Schemele de semnatura digitala sunt public verificabile... ► ...ceea ce inseamna ca semnaturile digitale sunt transferabile - o terta parte poate verifica legitimitatea unei semnaturi si poate face o copie pentru a convinge pe altcineva ca aceea este o semnatura valida pentru m; ► Schemele de semnatura digitala **au proprietatea de non-repudiere** - un semnatar nu poate nega faptul ca a semnat un mesaj; ► MAC-urile au avantajul ca sunt cam de 2-3 ori mai eficiente (mai rapide) decat schemele de semnatura digitala.

#### **Definiție**

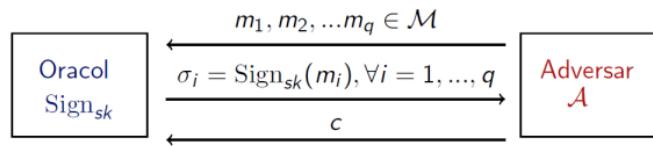
*O semnătură digitală definită peste ( $\mathcal{K}, \mathcal{M}, \mathcal{S}$ ) este formată din trei algoritmi polinomiali (Gen, Sign, Vrfy) unde:*

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

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

- 
- Formal, îi dăm adversarului acces la un oracol  $\text{Sign}_{sk}(\cdot)$ ;
  - Adversarul poate trimite orice mesaj  $m$  dorit către oracol și primește înapoi o semnătură corespunzătoare  $\sigma \leftarrow \text{Sign}_{sk}(m)$ ;
  - Considerăm că securitatea este impactată dacă adversarul este capabil să producă un mesaj  $m$  împreună cu o semnătură  $\sigma$  așa încât:
    1.  $\sigma$  este o semnătură validă pentru mesajul  $m$ :  
 $\text{Vrfy}_{pk}(m, \sigma) = 1$ ;
    2. Adversarul nu a solicitat anterior (de la oracol) o semnătură pentru mesajul  $m$ .
- 

**Securitate semnatura digitala - formalizare** ► Despre o semnatura care satisface nivelul de securitate de mai sus spunem ca nu poate fi falsificata printr-un atac cu mesaj ales; ► Aceasta inseamna ca un adversar nu este capabil sa falsifice o semnatura valida pentru nici un mesaj ... ► ... desi poate obtine semnaturi pentru orice mesaj ales de el, chiar adaptiv in timpul atacului. ► Pentru a da definitia formală, definim mai intai un experiment pentru o semnatura  $\pi = (\text{Sign}, \text{Vrfy})$ , in care consideram un adversar A si parametrul de securitate n;



- ▶ Adversarul întoarce o pereche de mesaj, semnătură  $(m, \sigma)$
- ▶ Output-ul experimentului este 1 dacă și numai dacă:
  - (1)  $\text{Vrfy}_{pk}(m, \sigma) = 1$  și (2)  $m \notin \{m_1, \dots, m_q\}$ ;
- ▶ Dacă  $\text{Sign}_{A, \pi}^{\text{forge}}(n) = 1$ , spunem că  $A$  a efectuat experimentul cu succes.

### Semnatura digitală bazată pe RSA

- ▶ Gen: generează  $N, e, d$  și  $pk = (N, e)$  iar  $sk = d$
- ▶ Sign( $sk, m$ ): semnează mesajul  $m$  folosind cheia  $sk = d$  astfel

$$\sigma = m^d \pmod{N}$$

- ▶ Vrfy( $pk, m, \sigma$ ): semnătura este validă dacă și numai dacă

$$m \stackrel{?}{=} \sigma^e \pmod{N}$$

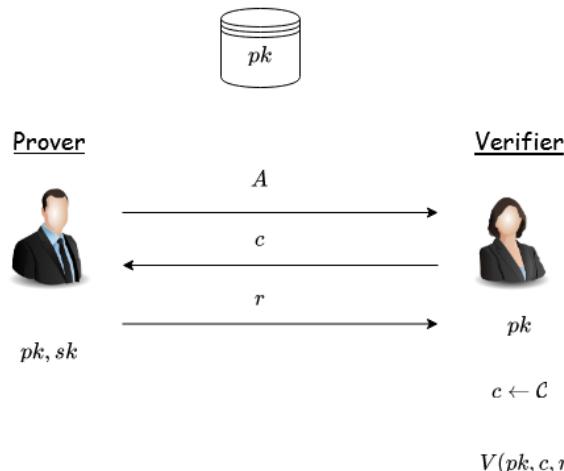
Această variantă de semnătură nu este sigură, se cunosc mai multe atacuri pentru ea

- ▶ scop adversar: falsificarea semnături mesajului  $m \in \mathbb{Z}_N^*$  pentru  $pk = (N, e)$
- ▶ acțiune adversar: alege  $m_1, m_2 \in \mathbb{Z}_N^*$  a.i.  $m = m_1 \cdot m_2 \pmod{N}$
- ▶ obține semnăturile  $\sigma_1$  și  $\sigma_2$  pentru mesajele  $m_1, m_2$
- ▶ întoarce  $\sigma = \sigma_1 \cdot \sigma_2 \pmod{N}$  ca semnătură validă pentru  $m$
- ▶ aceasta este o semnătură validă pentru că

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

## Scheme de identificare - identification schemes

- ▶ sunt protocole interactive care permit unei părți (Prover) să își demonstreze identitatea în fața unei alte părți (Verifier)
- ▶ sunt foarte importante ca building block pentru semnături digitale (dar, în sine, au aplicabilitate limitată)
- ▶ în continuare, abordare informală



- ▶ Securitate ▶ fata de adversarii pasivi - chiar daca are acces la mesajele trimise in mai multe executii ale protocolului, un adversar nu il poate convinge pe Verifier sa accepte ▶ Principala aplicatie ▶ identificarea persoanelor prezente fizic; de pilda, deschiderea unei usi securizate pe baza unei cartele de acces ▶ nu este potrivita pentru autentificarea la distanta (pe internet)

## Certificate și PKI

- ▶ O problemă a criptografiei cu cheie publică o reprezintă distribuirea cheilor publice;
- ▶ Se rezolvă tot cu criptografia cu cheie publică: e suficient să distribuim o singură cheie publică în mod sigur...
- ▶ Ulterior ea poate fi folosită pentru a distribui sigur oricât de multe chei publice;
- ▶ Ideea constă în folosirea unui *certificat digital* care este o semnătură care atașează unei entități o anume cheie publică;

- 
- ▶ De exemplu, dacă Charlie are cheia generată  $(pk_C, sk_C)$  iar Bob are cheia  $(pk_B, sk_B)$ , iar Charlie cunoaște  $pk_B$  atunci el poate calcula semnătura de mai jos pe care i-o dă lui Bob:

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

- ▶ Această semnătură este un *certificat* emis de Charlie pentru Bob;
- ▶ Atunci când Bob vrea să comunice cu Alice, îi trimită întâi cheia publică  $pk_B$  împreună cu certificatul  $\text{cert}_{C \rightarrow B}$  a cărui validitate în raport cu  $pk_C$  Alice o verifică;

## Curs 12- Criptografia post-cuantica

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). In evaluarea securitatii primitelor criptografie am considerat numai atacuri clasice. ► Nu am avut in vedere calculatoarele cuantice - care se bazeaza pe principiile mecanice cuantice si impactul lor asupra securitatii ► Algoritmii cuantici pot fi, in anumite cazuri, mult mai rapizi decat cei clasici si pot avea un impact zdrobitoare asupra securitatii primitelor criptografice studiate.

► D.p.d.v. teoretic, impactul calculatoarelor cuantice asupra criptografiei este recunoscut din anii 1990. ► Practic, un calculator cuantic generic, pe scara larga nu exista iar costurile pentru constructia lui ar fi urias. ► Totusi, un calculator cuantic dedicat care sa atace sistemele de criptare actuale ar putea aparea in decurs de cativa ani sau zeci de ani. ► Odata ce un astfel de calculator cuantic care sa poate fi folosit in practica devine disponibil, toti algoritmii cu cheie publica folositi in prezent dar si protocoalele asociate devin vulnerabile ► Aceasta inseamna ca toate emailurile, informatiile despre cardurile cu care facem plati online, semnaturi digitale, tranzactii online, datele sensibile, informatiile clasificate ale agentiilor de securitate si cele guvernamentale vor fi in pericol

### **Competitia NIST pentru standardizare post-cuantica**

- Iulie 2022 - NIST anunta primul grup de 4 finalisti
  - **criptare:** CRYSTALS - Kyber - pentru chei mici de criptare și operații rapide
  - **semnături digitale**
    - CRYSTALS-Dilithium
    - FALCON
    - SPHINCS+
- Dilithium și Falcon sunt foarte eficienți, cel din urmă fiind recomandat atunci când sunt necesare semnături mai mici decât cele oferite de Dilithium
- Primii 3 algoritmi se bazează pe probleme matematice de latici, iar SPHINCS+ folosește funcții hash
- Procesul de standardizare se va finaliza în anul 2024
- Alți 4 algoritmi sunt considerați pentru standardizare

#### **Criptografia cuantica**

- implementări folosind calculatoare cuantice, fenomene mecanice cuantice și canale de comunicare cuantice
- dificil de implementat la scara largă
- în unele cazuri este sigură necondiționat (nu se bazează pe ipoteze matematice)

#### **Criptografia post-cuantica**

- implementări folosind calculatoare clasice
- este sigură chiar și în fața unui adversar care are acces la un calculator cuantic
- se bazează pe probleme matematice dificile computațional chiar și pentru algoritmii cuantici

- ▶ Impactul calculatoarelor cuantice asupra criptografiei simetrice este minor; ilustram pe scurt
- ▶ Considerăm următoarea **problema abstractă**:
  - ▶ Se dă: funcție  $f : D \rightarrow \{0, 1\}$  cu acces de tip oracol (funcția poate fi interogată pe orice input și se primește output-ul corespunzător)
  - ▶ Se cere: să se găsească  $x$  a.î.  $f(x) = 1$ .
- ▶ Dacă există un singur  $x$  cu  $f(x) = 1$  atunci orice algoritm clasic necesită  $O(\|D\|)$  evaluări ale funcției  $f$  - corespunde unui atac prin forță brută
- ▶ **1996 - algoritmul cuantic al lui Grover:** găsește  $x$  folosind  $O(\|D^{1/2}\|)$  evaluări ale funcției  $f$ . Algoritmul este optim, și nu poate fi îmbunătățit

### Criptografia simetrică post-cuantică - sisteme bloc

- ▶ Trecem în revista impactul algoritmului asupra sistemelor de criptare simetrice
- ▶ Considerăm cazul unui sistem de criptare bloc  
 $F : \{0, 1\}^n \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  cu cheia  $k$  pe  $n$  biți pentru care cel mai bun atac (de găsire a cheii) este forță brută.
  - ▶ Un astfel de atac clasic necesită timp  $2^n$
  - ▶ Pentru securitate, alegem cheia  $k$  de lungime  $n$  biti aşa încât timpul pentru atac  $2^n$  să nu fie practic
  - ▶ Algoritmul lui Grover însă permite unui atacator să găsească cheia în timp  $2^{n/2}$ .
  - ▶ Pentru același nivel de securitate (precum în cazul clasic), alegem cheia  $k$  de lungime **dublă** față de cazul clasic.

### Criptografia simetrică post-cuantică - funcții hash

- ▶ Considerăm problema găsirii de coliziuni pentru o funcție hash  $H : \{0, 1\}^m \rightarrow \{0, 1\}^n$  cu  $m > n$ .
- ▶ În cazul **clasic**, am văzut că "atacul nașterilor" necesită  $O(2^{n/2})$  evaluări ale funcției  $H$ .
- ▶ Aceasta înseamnă că pentru a asigura rezistența la coliziuni față de un atac **în timp  $2^t$** , trebuie să alegem funcții hash cu **output-ul pe  $2t$  biți**.
- ▶ Însă în cazul **cuantic**, un atac pentru găsirea coliziunilor necesită  $O(2^{n/3})$  evaluări ale funcției  $H$ .
- ▶ Deci, pentru a asigura rezistența la coliziuni față de un atac **în timp  $2^t$** , trebuie să alegem funcții hash cu **output-ul pe  $3t$  biți**.

## Algoritmul lui Shor si impactul lui asupra criptografiei asimetrice

► Pana acum am vazut algoritmi cuantici care ofera o imbunatatire de ordin polinomial in comparatie cu cei mai buni algoritmi clasici pentru aceeasi problema. ► Acestia impun doar cresterea dimensiunilor cheilor fara a necesita alte schimbari majore ► In continuare vom vedea un algoritm care ofera o imbunatatire de ordin exponential - **algoritmi cuantici polinomiali pentru problema factorizarii si problema logaritmului discret**

► Algoritmul lui Shor poate fi folosit si pentru rezolvarea problemei logaritmului discret in timp polinomial ► Avand in vedere ca toate sistemele de criptare cu cheie publica se bazeaza pe problema factorizarii sau problema logaritmului discret, concluzionam ca

**Toate sistemele de criptare cu cheie publica prezentate la curs pot fi atacate in timp polinomial cu ajutorul unui calculator cuantic**

Sisteme de criptare cu cheie publica post-cuantice

► Problema factorizarii si problema logaritmului discret devin "usoare" pentru un calculator cuantic.  
► Avem nevoie de probleme matematice dificile computational chiar si pentru calculatoarele cuantice, dar care ruleaza pe calculatoare clasice ► Diferenta fata de cazul clasic este ca problemele considerate pentru criptografia post-cuantica sunt mai recente si nu au fost studiate la fel de mult ca problema factorizarii sau problema logaritmului discret ► In continuare vom prezenta o problema care a primit multa atentie si care este considerata dificila chiar si pentru calculatoarele cuantice. Aratam apoi cum se poate construi un sistem de criptare cu cheie publica bazat pe dificultatea acelei probleme.

## Problema LWE - Learning With Errors

- A fost introdusa in 2005 de Oded Regev
- Preliminarii:
  - $q$  număr prim. Vom nota cu  $\mathbb{Z}_q$  mulțimea  $\{-\lfloor(q-1)/2\rfloor, \dots, 0, \dots, \lfloor q/2 \rfloor\}$  (spre deosebire de  $\{0, \dots, q\}$ ) unde  $\lfloor x \rfloor$  este cel mai mare intreg mai mic sau egal cu  $x$ .
  - spunem că un element al lui  $\mathbb{Z}_q$  este "mic" dacă este "aproape" de 0.
- Problema cere găsirea lui  $\mathbf{s} \in \mathbb{Z}_q^n$  fiind dată o secvență de ecuații liniare "aproximative" în  $\mathbf{s}$ .
- Exemplu:

$$12s_1 + 10s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$$

$$3s_1 + 7s_2 + 9s_3 + s_4 \approx 4 \pmod{17}$$

$$16s_1 + 2s_2 + 8s_3 + 7s_4 \approx 3 \pmod{17}$$

- Exemplul

$$12s_1 + 10s_2 + 5s_3 + 2s_4 + 1 = 8 \pmod{17}$$

$$3s_1 + 7s_2 + 9s_3 + s_4 - 1 = 4 \pmod{17}$$

$$16s_1 + 2s_2 + 8s_3 + 7s_4 + 2 = 3 \pmod{17}$$

sub forma matriciala

12	10	5	2
3	7	9	1
16	2	8	7

$$\begin{matrix} * & s_1 \\ & s_2 \\ & s_3 \\ & s_4 \end{matrix} + \begin{matrix} 1 \\ -1 \\ 2 \end{matrix} \approx \begin{matrix} 8 \\ 4 \\ 3 \end{matrix}$$

sau, notand matricile corespunzator (unde  $\mathbf{s} = (s_1, s_2, s_3)$ ),  
ecuția matricială devine

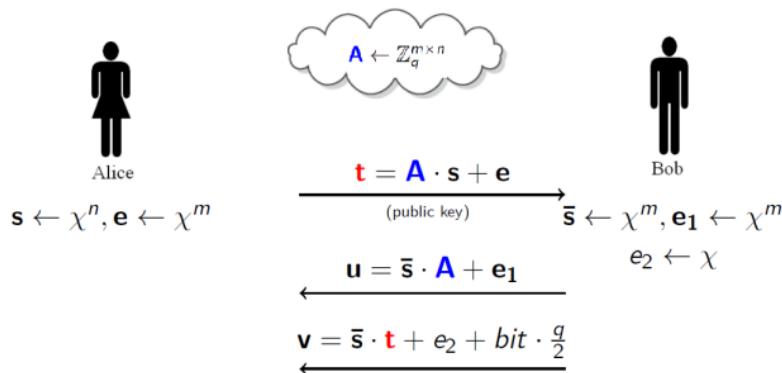
$$\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b} \pmod{q}$$

12	10	5	2	*	$s_1$	+	1	$\approx$	8
3	7	9	1		$s_2$		-1		4
16	2	8	7		$s_3$		2		3
					$s_4$				

- vectorul  $\mathbf{e} = (1, -1, 2)$  este format din elemente "mici" din  $\mathbb{Z}$  numite *noise* sau *error*.
- În lipsa lui  $\mathbf{e}$ , ecuația  $\mathbf{As} = \mathbf{b}$  devine usor de rezolvat cu tehnici clasice de algebra liniară
- Cand matricea  $\mathbf{A}$  are suficient de multe linii și parametrii sunt aleși corespunzător, problema devine dificilă.

## Sistem de criptare bazat pe LWE

- Protocolul de mai sus nu este sigur pentru că un atacator poate calcula  $\bar{\mathbf{s}}$  sau  $\mathbf{s}$  cu noțiuni de algebră liniară și poate afla și cheia.
- Însă el poate fi transformat într-un protocol sigur și adaptat ca un sistem de criptare adăugând "noise", sub ipoteza problemei decizionale LWE.



- Schema de mai sus criptează un bit iar decriptarea se face calculând  $x = \mathbf{v} - \mathbf{u} \cdot \mathbf{s}$ .
- Rezultatul va fi 1 dacă  $x$  este mai aproape de  $\frac{q}{2}$  decât de 0.
- Apropierea lui  $x$  de  $\frac{q}{2}$  este verificată calculând valoarea absolută a lui  $x - \frac{q}{2} \bmod q$

## Exerciții

Fie El Gamal cu  $pk = (g, h = g^a)$  și  $sk = (g, a)$  în  $\mathbb{G}$ .

- ▶ **Enc:** dată o cheie publică  $(\mathbb{G}, q, g, h)$  și un mesaj  $m \in \mathbb{G}$ , alege  $y \leftarrow^R \mathbb{Z}_q$  și întoarce  $c = (c_1, c_2) = (g^y, m \cdot h^y)$ ;
- ▶ **Dec:** dată o cheie secretă  $(\mathbb{G}, q, g, a)$  și un mesaj criptat  $c = (c_1, c_2)$ , întoarce  $m = c_2 \cdot c_1^{-a}$ .

Vrem să distribuim cheia secretă la două persoane aşa încât numai cele două persoane împreună pot decripta. O modalitate simplă de a rezolva această problemă este să alegem două numere aleatoare  $a_1, a_2 \in \mathbb{Z}_n$  aşa încât  $a_1 + a_2 = a$ . O persoană primeşte  $a_1$  iar cealaltă primeşte  $a_2$ . Pentru decriptarea  $(c_1, c_2)$ , trimitem  $c_1$  ambelor persoane.

Ce valori trebuie să calculeze cele două persoane și să ne trimită înapoi aşa încât să putem decripta textul criptat trimis?

### Solution

Persoana 1 trimite  $u_1 = c_1^{a_1}$  iar persoana 2 trimite  $u_2 = c_1^{a_2}$ .

Produsul  $u_1 \cdot u_2 = c_1^{a_1+a_2} = c_1^a$  împreună cu  $c_2$  poate fi folosit la

Societatea de Sistemele de Informatică

24 / 26

Se consideră  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$  o funcție hash rezistentă la coliziuni. Se definește o funcție  $H^* : \{0, 1\}^* \rightarrow \{0, 1\}^{n+1}$  astfel:

$$H'(x) = \begin{cases} x||1 & \text{dacă } x \in \{0, 1\}^n \\ H(x)||0 & \text{altfel} \end{cases}$$

Argumentați că  $H'$  este rezistentă la coliziuni.

### Solution

Fie  $H'(x_1) = H'(x_2)$  cu  $x_1 \neq x_2$ . Dacă  $H'(x_1) = x||1$  rezultă  $x_1 = x_2$ , contradicție. Dacă  $H'(x_1) = H(x_1)||0 = H(x_2)||0$  atunci se determină o coliziune pentru  $H$ , contradicție.

Fie  $(Mac, Vrfy)$  un MAC sigur definit peste  $(K, M, T)$  unde  $M = \{0, 1\}^n$  și  $T = \{0, 1\}^{128}$ . Este MAC-ul de mai jos sigur? Argumentați răspunsul.

$$Mac'(k, m) = Mac(k, m)$$

$$Vrfy'(k, m, t) = \begin{cases} Vrfy(k, m, t), & \text{dacă } m \neq 0^n \\ 1, & \text{altfel} \end{cases}$$

### Solution

MAC-ul nu este sigur pentru ca un adversar poate să întoarcă perechea validă  $(0^n, 0^s)$ .

## Curs 13 – TLS

TLS - Transport Layer Security

► Este un protocol folosit de browser-ul web de fiecare data cand ne conectam la un browser folosind https ► primele versiuni se numeau SSL - Secure Sockets Layer - dezvoltat de Netscape (1995) - SSL 3.0, cea mai cunoscuta versiune ► TLS 1.0 apare in 1999, TLS 1.1 in 2006, TLS 1.2 in 2008 si versiunea actuala, sigura si eficienta TLS 1.3 in 2018 ► folosirea lui SSL 3.0, TLS 1.0 si TLS 1.1 trebuie evitata, toate trei prezinta probleme de securitate ► se recomanda folosirea minim a lui TLS 1.2

- Protocolul TLS permite unui client (de ex. browser web) și unui server (de ex. website) să se pună de acord asupra unui set de chei pe care să le folosească ulterior pentru comunicare criptată și autentificare
- Protocolul TLS constă din 2 părți:
  1. *protocolul handshake* - realizează schimbul de chei care stabilește un set de chei comune
  2. *protocolul record-layer* - folosește cheile stabilite pentru criptare/autentificare ulterioară
- În continuare vom prezenta protocolul handshake din versiunea curentă TLS 1.3

TLS 1.3 ► clientul și serverul, partajează, la finalul protocolului, cheile kS și kC , pe care le folosesc ulterior pentru criptare și autentificare ► în plus, clientul are garantia că la final a partajat cheile cu server-ul legitim și că acestea nu au fost interceptate sau modificate de o terță parte de ce? ► în handshake-ul din TLS 1.2, clientul și serverul foloseau o schema de criptare cu cheie publică în locul schimbului de chei Diffie-Hellman ► clientul doar alegea o cheie K pe care o trimitea serverului criptată cu cheia lui publică

### **TLS 1.3**

- aceasta a fost eliminată în TLS 1.3 pentru că nu asigură proprietatea de *forward secrecy* - care presupune că, în cazul compromiterii server-ului, cheile de sesiune anterioare nu sunt compromise
- în TLS 1.3, în ceea ce privește server-ul, cheia K este calculată pe baza secretului y și în fiecare sesiune, secret care este unul nou de fiecare dată și care nu trebuie menținut între sesiuni; deci compromiterea server-ului (aflarea cheii lui secrete) nu duce la aflarea cheii K
- în TLS 1.2, cheia secretă K îl este trimisă server-ului criptată cu cheia lui publică; compromiterea cheii secrete server-ului duce la compromiterea celor de sesiune din toate sesiunile
- în protocolul *record*, clientul și serverul comunică în mod confidențial și autentificat folosind o schemă de criptare autentificată

### **TLS 1.3 vs. TLS 1.2**

- număr redus de runde

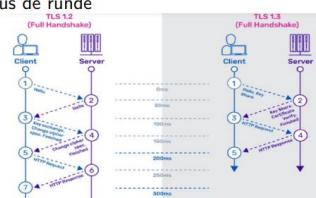


Figure: Sursa: [www.a10networks.com](http://www.a10networks.com)

- eliminarea algoritmilor criptografici vulnerabili precum SHA-1, RC4, DES, 3DES, AES-CBC, MD5
- 0-RTT (zero round-trip) – reluarea unei sesiuni mai vechi cu un website e mult mai rapidă

## EXAMEN ONLINE - Instrucțiuni generale

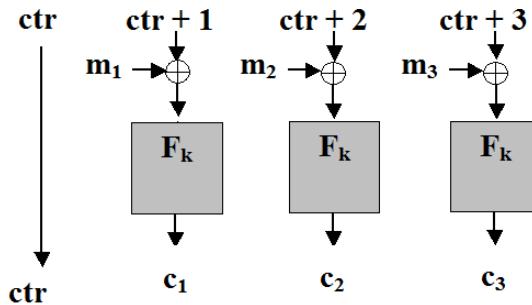
1. Transmiteți examenul **prin Moodle** până la termenul limită: **21 mai, ora 09:55.**
  - Transmiterea corectă a examenului este strict în responsabilitatea voastră.
  - Transmiteți în timp util, NU așteptați ultimele minute pentru a încărca examenul. Examenul poate fi transmis de oricâte ori doriți până la deadline, se ia în considerare doar ultima variantă transmisă. NU se accepă ca motivație pentru netransmiterea examenului niciun fel de probleme tehnice (încetinirea platformei, utilizarea incorectă, nesincronizări ale ceasului, etc.).
  - Studenții care nu transmit rezolvarea examenului scris sunt considerați absenți.
2. Răspunsul trebuie să fie în **format .pdf, încărcat prin contul instituțional Moodle** în secțiunea corespunzătoare sub numele **grupa\_nume\_prenume.pdf**. Prima pagină a fișierului de răspunsuri trebuie să conțină **nume, grupă, o listă a subiectelor ne tratate** (ex.: *Subiecte ne tratate: 1(a), 1(c), 3(b). sau -*).
3. Se acordă punctaje parțiale. Răspunsurile greșite la examenul scris NU depunțează suplimentar.
4. Pentru promovare, **este obligatoriu să participați la ambele probe (examen scris și oral) și să obțineți minim 45 de puncte** ca notă finală (include punctele obținute în timpul anului, fără bonus, care se acordă doar în caz de promovare).
5. Pentru examenul oral:
  - Este strict în responsabilitatea voastră să verificați repartizarea pe zile / ore (aprox.) și alte informații necesare referitoare la susținerea examenului oral.
  - Trebuie să vă conectați **audio-video, folosind contul instituțional Teams**.
  - Trebuie să arătați **un act de identitate** (CI, pașaport, permis de conducere, etc.) **sau legitimație de student cu poză**. Este în responsabilitatea voastră să ascundeți alte informații (altele decât numele și poza) de pe documentul prezentat, pe care nu doriți să le faceți publice!
  - Fiecare subiect rezolvat în scris, dar pe care nu știți să îl explicați (i.e., să arătați că l-ați rezolvat individual sau înțeles), **se depunțează cu dublul punctajului alocat subiectului respectiv**.
  - Studenții care transmit rezolvarea examenului scris dar nu participă la susținerea orală obțin nota finală 4.
  - Dacă există studenți care nu au posibilitatea unei conexiuni audio și video, trebuie să anunțe în prealabil, pe e-mail ([ruxandra.olimid@fmi.unibuc.ro](mailto:ruxandra.olimid@fmi.unibuc.ro)).

Dacă în timpul examenului aveți întrebări, le puteți posta pe forum, secțiunile *Examen* sau *Întrebări exerciții și probleme examen*. Urmăriți formul pentru informații. **NU postați indicii sau soluții!**

SUCCES!

### EXAMEN ONLINE - Probleme

1. Primiți de la Alice următorul mesaj criptat:  $C = 45A4562AB1C307F78ED2$  (în reprezentare hexa). Știți că mesajul este criptat cu One Time Pad (OTP) cu o cheie  $K$  stocată pe un stick, pe care din neglijență l-ați pierdut.
  - (a) Ce puteți spune despre cheia  $K$  stocată pe stick dacă știți că sistemul vă permite comunicarea perfectă sigură? (2.5p)
  - (b) Alice află că ați pierdut cheia. Vă transmite un alt mesaj criptat  $C'$ , corepunzător aceluiași mesaj clar  $M$ . Vă anunță (public) că pentru criptare a folosit o cheie  $K = M$  (i.e.,  $C'$  este criptarea mesajului  $M$  folosind cheia  $K = M$ ). Sistemul de criptare folosit rămâne OTP. Ce puteți afirma despre  $C'$  în această situație? Puteti decripta? (2.5p)
2. Se consideră modul de operare  $CTR_{modif}$  (Counter Mode Modificat), reprezentat în figura de mai jos pentru un mesaj clar  $m = (m_1, m_2, m_3)$  de 3 blocuri:



Bineînțeles, generalizând, modul de operare poate fi utilizat pentru criptarea unui mesaj de lungime oarecare. Notațiile trebuie să vă fie cunoscute de la modul de operare CTR.  $ctr$  este o valoare aleasă uniform aleator pentru fiecare mesaj.

- (a) Care este mesajul criptat? Scrieți formula de criptare. (2.5p)
- (b) Desenați schema de decriptare. Scrieți formula de decriptare. (2x2.5p)
- (c) Se folosește padding-ul *0-peste-tot*, adică se adaugă 0 pentru completarea ultimului bloc. Dacă mesajul  $m$  este multiplu de lungimea blocului, nu se mai realizează padding-ul. Ce puteți spune despre corectitudinea sistemului? Dar dacă s-ar realiza padding-ul indiferent de situație (i.e., dacă mesajul  $m$  este multiplu de lungimea blocului, se adaugă un bloc *0-peste-tot*)? (2x2.5p)
- (d) Este sistemul de criptare CCA sigur? Argumentați. (5p)
- (e) Se consideră lungimea blocului egală cu 48 de biți (6 bytes). Ce puteți afirma în plus despre securitatea sistemului în acest caz? (2.5p)

3. Sunteți angajat să verificați securitatea în cadrul unei companii. Observați că se folosesc următoarele:

- $MD5$ , funcție hash folosită pentru stocarea parolelor clientilor cu *salt*.
- *CryptStream*, un sistem de criptare de tip fluid care folosește ca generator  $G(x) = x^2 \pmod{x}$ , unde  $x$  este seed-ul de intrare, pentru comunicația criptată cu clientii.  $x$  se obține ca un derivat din parola *pwd* asociată clientului:  $x = F(MD5(pwd, salt))$ , cu  $F$  funcție *one-way* (deterministă) cunoscută.
- Protocolul de schimb de chei *Diffie-Hellman* neautentificat pentru generarea cheilor necesare securizării comunicației interne (i.e., între angajații firmei) într-un grup pentru care un adversar PPT poate rezolva *Problema Logaritmului Discret* (PLD, sau DLP în limba engleză) cu o probabilitate  $f(n) = 1/n^{65537}$ , cu  $n$  parametrul de securitate.
- *AuthMAC*, un sistem de autentificare proprietar utilizat pentru autentificarea părților în comunicația dintre manageri. În urma semnării unui *Non Disclosure Agreement (NDA)*, vi s-a dat acces la descrierea acestuia:  

$$Mac(k, m) = h(k||len(m)) \oplus h(m)$$
, unde  $h$  este o funcție hash rezistentă la coliziuni,  $||$  este concatenare,  $len(m)$  este lungimea în biți a mesajului  $m$ ,  $\oplus$  este operatorul pe biți XOR.

$$Vrfy(k, m, t) = \begin{cases} 1 & \text{dacă } Mac(k, m) = t \\ 0, & \text{altfel} \end{cases}$$

Vi se cere să completați un raport care să răspundă la următoarele întrebări:

- Sunt parolele clientilor stocate în mod sigur? Argumentați. **(2.5p)**
- Este  $G$  din *CryptStream* PRG (din punct de vedere criptografic)? Argumentați. **(2.5p)**
- Care este mesajul criptat  $c$  corespunzător unui mesaj  $m$  transmis unui client (se presupune  $x$  cunoscut) folosind *CryptStream*? Scrieți formula de criptare. **(2.5p)**
- Ce puteți spune despre funcția  $f$  și securitatea protocolului de schimb de chei *Diffie-Hellman* în acest caz? **(2.5p)**
- Presupunând că se setează niște parametru pentru care *problema decizională Diffie-Hellman (DDH)* este dificilă, la ce tip de atac rămâne vulnerabil protocolul de schimb de chei? Cum s-ar putea împedica un astfel de atac? **(2x2.5p)**
- Este *AuthMAC* un sistem de autentificare a părților sigur (din punct de vedere criptografic)? Argumentați. **(5p)**
- Există principii cunoscute în criptografie care sunt încălcate? Dacă da, dați un exemplu, specificând numele principiului și cum / de ce este încălcat. **(2.5p)**
- Ce obiective ale criptografiei ar trebui să fie satisfăcute în mod normal într-un astfel de scenariu (comunicația internă și externă a unei companii), dar în condițiile date sunt încălcate? Dați 2 exemple. **(2x2.5p)**

4. Se consideră sistemul de criptare RSA pentru care valoarea modulului  $N$  este (în reprezentare hexa):

$N = 22\ A1\ E6\ 5B\ 83\ 51\ 5A\ 43\ 47\ BC\ 69\ 14\ A3\ 00\ 13\ 7C\ 8E\ D0\ 80\ 43\ 00\ 8D\ 0C\ D5\ E1\ FE\ 44\ 4F\ DB\ A3\ 5E\ C4\ 1C\ B4\ 15\ 85\ 12\ BB\ B2\ AD\ DA\ FD\ FA\ 32\ EE\ B4\ 38\ A2\ 20\ 4E\ DD\ 64\ D6\ BC\ 78\ 7E\ 4B\ 42\ CC\ 37\ 09\ 77\ C6\ 23\ F4\ 46\ 96\ 61\ 8D\ D6\ CA\ E9\ 5D\ 71\ E2\ 97\ 84\ 1E\ BD\ 85\ 6D\ 39\ 21\ C1\ A5\ 92\ FD\ 5B\ E7\ 37\ 32\ C3\ 1C\ 04\ 33\ 69\ 2E\ 58\ 4F\ A4\ D0\ 1D\ D5\ BC\ 95\ 28\ ED\ AC\ 03\ 74\ AD\ 55\ 5D\ 7B\ 92\ 79\ 26\ 0A\ 51\ 5B\ 5A\ 20\ 9C\ 86\ 3C\ 14\ 91\ 9A\ C7\ 58\ 21\ 80\ 59\ A5\ EA\ 50\ C2\ A9\ 07\ 3A\ 67\ CD\ 9D\ 99\ CB\ E2\ 57\ C3\ 45\ EB\ 3C\ C9\ 2B\ 55\ 04\ 9E\ 8D\ FD\ 92\ 26\ 35\ 6D\ 5C\ 41\ F6\ 61\ 4B\ 0B\ 2D\ 96\ 92\ AF\ 92\ 8B\ 00\ 38\ 49\ 3F\ C2\ EC\ F7\ A8\ F1\ A9\ 24\ 43\ 98\ 7B\ 7D\ 22\ 87\ 31\ 82\ 27\ DF\ 9F\ DA\ 27\ 85\ EB\ 85\ 48\ E2\ D2\ 61\ 3F\ 09\ 0F\ 9B\ C7\ 31\ 56\ 8B\ C0\ 08\ 38\ 05\ D6\ DE\ 76\ 75\ E2\ 3D\ A1\ 33\ BC\ C2\ 90\ 05\ F8\ 3C\ DD\ FC\ 80\ B0\ BE\ 3E\ AB\ 45\ 22\ 46\ 1D\ 35\ 1D\ 0F\ 6E\ 34\ EA\ 8F\ A0\ 27\ 42\ 48\ 6A\ 8C\ D8$

- (a) Exponentul de criptare este  $e = 65537$ . Este sistemul corect definit dacă folosește acești parametrii? Argumentați. **(2.5p)**

Se decide înlocuirea RSA cu un sistem de criptare definit peste curba eliptică  $y^2 = x^3 + 17x + 3 \pmod{29}$ .

- (b) Care este inversul punctului  $(8, 10)$ ? Dar al punctului  $(8, 11)$ ? **(2x2.5p)**
- (c) Ce puteți spune despre curba eliptică dată din punct de vedere al securității? **(2.5p)**
- (d) Căutați un exemplu de curbă eliptică de această formă recomandată de o instituție cunoscută (NIST, ENISA, etc.). Specificați numele exact sub care este cunoscută curba și ecuația acesteia (specificând și modulul  $p$ ). **(2.5p)**

5. Completați următorul formular (accesibil și după deadline):

<https://forms.gle/YfpsXsawvqMMN1bT8> **(0p)**

**TOTAL disponibile: 65p**

## EXAMEN ONLINE - Instrucțiuni generale

1. Trimiteți examenul **prin Moodle** până la termenul limită: **19 mai, ora 09:59.**
  - Transmiterea corectă a examenului este strict în responsabilitatea studenților.
  - Trimiteți în timp util, NU așteptați ultimele minute pentru a încărca examenul. Examenul poate fi transmis de oricâte ori doriți până la deadline, se ia în considerare doar ultima variantă transmisă. NU se accepă ca motivele pentru netransmiterea examenului niciun fel de probleme tehnice (încetinirea platformei, utilizarea incorectă, nesincronizări ale ceasului platformei, etc.).
  - Studenții care nu transmit rezolvarea examenului scris sunt considerați absenți.
2. Răspunsul trebuie să fie în **format .pdf, încărcat prin contul instituțional Moodle** în secțiunea corespunzătoare sub numele **grupa\_nume\_prenume.pdf**. Prima pagină a fișierului de răspunsuri trebuie să conțină **nume, grupă, o listă a subiectelor ne tratate** (ex.: *Subiecte ne tratate: 1(a), 1(c), 3(b). sau -*).
3. Se acordă punctaje parțiale. Răspunsurile greșite la examenul scris NU depunțează suplimentar.
4. Pentru promovare, **este obligatoriu să participați la ambele probe (examen scris și oral) și să obțineți minim 45 de puncte** ca notă finală (include punctele obținute în timpul anului, fără bonus, care se acordă doar în caz de promovare).
5. Pentru examenul oral:
  - Este strict în responsabilitatea studenților să verificați repartizarea pe zile / ore (aprox.) și alte informații necesare referitoare la susținerea examenului oral.
  - Trebuie să vă conectați **audio-video, folosind contul instituțional Teams**.
  - Trebuie să arătați **un act de identitate**, de preferat **legitimătie / carnet de student cu poză**. Este în responsabilitatea studenților să ascundăteți alte informații (altele decât numele și poza) de pe documentul prezentat, pe care nu doriți să le faceți publice!
  - Fiecare subiect rezolvat în scris, dar pe care nu știți să îl explicați (i.e., să arătați că l-ați rezolvat individual sau înțeles), **se depunțează cu dublul punctajului alocat subiectului respectiv**.
  - Studenții care transmit rezolvarea examenului scris dar nu participă la susținerea orală obțin nota finală 4.
  - Dacă există studenți care nu au posibilitatea unei conexiuni audio și video, trebuie să anunțe în prealabil, pe e-mail ([ruxandra.olimid@fmi.unibuc.ro](mailto:ruxandra.olimid@fmi.unibuc.ro)).

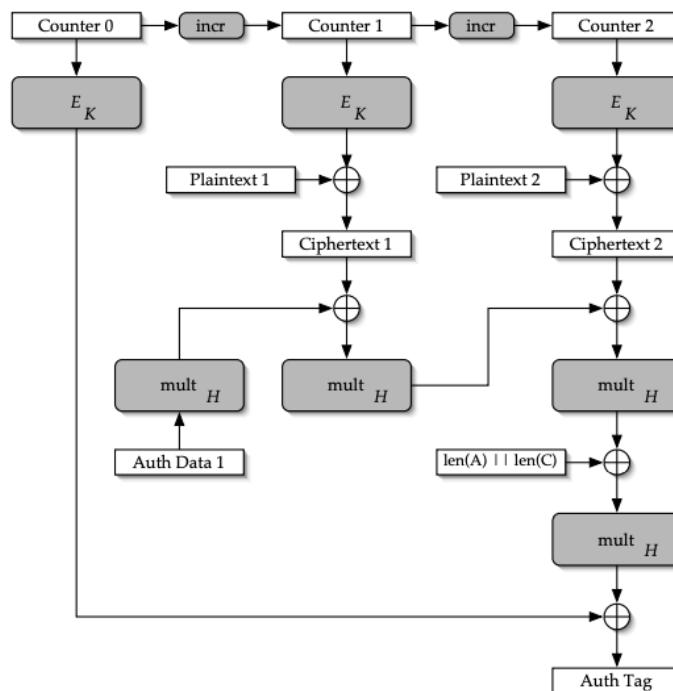
Dacă în timpul examenului aveți întrebări, le puteți posta pe forum, secțiunea *Examen*. Urmăriți formuł pentru informații. **NU postați indicii sau soluții!**

SUCES!

### EXAMEN ONLINE - Probleme

1. Există o tendință de a face confuzie între criptare și aplicarea unei funcții hash.
- (a) Explicați pe scurt câte un scenariu potrivit pentru fiecare dintre cele 2 situații (e.g., un exemplu de aplicație când această abordare este potrivită, etc.) și motivați de ce ați făcut această alegere (2 paragrafe, câte unul pentru fiecare subpunkt):
- (i) stocarea parolelor se realizează folosind o funcție hash (i.e., parolele sunt "hash-uite") **(2.5p)**
  - (ii) stocarea parolelor se realizează folosind o funcție de criptare (i.e., parolele sunt criptate) **(2.5p)**
- (b) Enunțați un aspect care diferențiază clar un sistem de criptare de o funcție hash. **(2.5p)**
- (c) Explicați pe scurt (1 paragraf) de ce nu putem considera securitate perfectă (ci doar computațională) în cazul funcțiilor hash. **(2.5p)**

2. Se consideră modul de operare GCM (Galois/Counter Mode), reprezentat în figura următoare pentru un mesaj clar (*plaintext*) de 2 blocuri (Plaintext 1, Plaintext 2):



Sursa imagine: McGrew, D. and Viega, J., 2004. The Galois/counter mode of operation (GCM). Submission to NIST Modes of Operation Process, 20, pp.0278-0070

Bineînțeles, generalizând, modul de operare poate fi utilizat pentru criptarea unui mesaj de lungime oarecare. Pentru simplificare, considerăm *Counter 0* o valoare aleatoare, aleasă la fiecare criptare și transmisă către destinație ca prima componentă a mesajului criptat (*ciphertext*).

- (a) Scrieți formula de criptare. **(2.5p)**
- (b) Scrieți formula de decriptare. **(2.5p)**
- (c) Considerați că  $E_K$  este o funcție de criptare bloc cu lungimea blocului de 128 biți. Mesajul clar are lungimea 320 biți. Care este lungimea în biți a mesajului criptat, fără să considerați și dimensiunea lui *Counter 0* (i.e., considerați doar (Cipertext 1, Ciphertext 2, ...))? Considerați soluția cea mai eficientă. **(2.5p)**
- (d) Considerăm în același context de la întrebarea precedentă și tag-ul de autentificare *Auth Tag*. Câți biți se adaugă mesajului criptat (i.e., care este lungimea tag-ului)? **(2.5p)**
- (e) Explicați pe scurt (1 paragraf) ce aduce GCM în plus față de modul de operare CTR din punct de vedere al securității. **(2.5p)**
- (f) Este adevărată următoarea afirmație: *"Pentru că lungimea blocului este mare (egală cu 128 biți), rezultă că sistemul definit mai sus este perfect sigur."*? Argumentați. **(2.5p)**
3. Se consideră cheia publică RSA stocată în fișierul *RSA\_public\_key.txt* (disponibil în Moodle, secțiunea *Examen*).
- (a) Folosiți un ASN.1 decoder (disponibil online) pentru a determina valoarea modulului  $N$  și a exponentului  $e$ . Scrieți valoarea lui  $e$  și dimensiunea în biți ai lui  $N$ . **(2.5p)**
- (b) Considerați că un alt user deține o cheie publică RSA pentru care  $e = 65537$ . Ce puteți spune despre  $N$ ? Argumentați. **(2.5p)**
- (c) Se consideră următoarea variantă a RSA Padded. Fie  $|m| \approx |N|/2$  (mesaje de aproximativ jumătate din lungimea modulului în biți). Definim  $\bar{m} = 0^8||r||m$ , unde  $0^8$  este un byte cu toți biții egali cu 0,  $r$  este ales uniform aleator (pe numărul de biți rămasi) și  $||$  este concatenare. Atunci  $c = \bar{m}^e \pmod N$ . Arătați că sistemul astfel definit nu este CCA-sigur. **(5p)**
- (d) Funcționează atacul de la punctul precedent și pentru PKCS#1 v1.5, definit în curs? Argumentați. **(2.5p)**
4. Vi se cere părerea în realizarea unui audit intern la locul de muncă.
- (a) O aplicație de comunicare externă folosește funcția hash SHA-256 pentru asigurarea integrității datelor. Cum vi se pare această abordare? Argumentați. **(2.5p)**
- (b) Găsiți în documente definiția a două funcții  $f$  și  $h$ .  $f : \{0, 1\}^m \rightarrow \{0, 1\}^m$  o funcție bijectivă rezistentă la prima preimagine. Pentru orice  $x \in \{0, 1\}^{2m}$  se definește  $h : \{0, 1\}^{2m} \rightarrow \{0, 1\}^m$  astfel:  $h(x) = f(x' \oplus x'')$ , unde  $x = x'||x''$  și  $x', x'' \in \{0, 1\}^m$ . Realizați că  $h$  nu este rezistentă la a doua preimagine. Argumentați. **(5p)**
- (c) Observați că  $h$  definită la punctul anterior este folosită pentru calculul amprentei fișierelor înainte de a fi semnate. Mai exact, pentru un fișier cu reprezentarea binară  $x$  se calculează  $h(x)$  care apoi se semnează folosind o semnatură digitală. Puteți evidenția cel puțin o problemă? **(2.5p)**

- (d) Propuneți o soluție ca să rezolvați problema pe care ati evidențiat-o la punctul anterior. **(2.5p)**
- (e) Pentru asigurarea confidențialității observați că se folosește un sistem de criptare fluid pentru care cheia fluidă este  $k = G(day)$ , unde  $G$  este un PRG (generator de numere pseudo-aleatoare sigur din punct de vedere criptografic), iar  $day$  este ziua curentă, sub forma  $yyymmdd$ . Cum vi se pare această abordare? Argumentați. **(2.5p)**
- (f) Vi se refuză accesul la implementarea schemei  $(Mac, Vrfy)$ , utilizată la nivel managerial, pe motiv că aceasta este proprietară și secretă. Ce principiu al criptografiei este încălcăt? **(2.5p)**
5. Considerăm următorul protocol de schimb de chei:
- (1) Alice alege uniform aleator  $k, a \leftarrow \{0, 1\}^n$  și îi trimite lui Bob  $s = k \oplus a$ ;
  - (2) Bob alege uniform aleator  $b \leftarrow \{0, 1\}^n$  și îi trimite lui Alice  $u = s \oplus b$ ;
  - (3) Alice calculează  $w = u \oplus a$  și îi trimite  $w$  lui Bob;
  - (4) Alice consideră drept cheie comună  $k$  iar Bob calculează drept cheie comună  $w \oplus b$ .
- (a) Arătați că Bob calculează aceeași cheie  $k$ . **(2.5p)**
- (b) Este schema astfel definită sigură față de un adversar pasiv? Dacă da, de ce? Dacă nu, explicați pe scurt un atac concret. **(5p)**
- (c) Propuneți o modalitate sigură în care Alice și Bob pot stabili o cheie comună chiar și în cazul unui adversar activ. **(2.5p)**
6. **(Optional)** Formular anonim de feedback: <https://forms.gle/DhBHMKadQo8SZCzSA>. Acest formular NU înlocuiește formularul de feedback oficial primit prin facultate, pe care vă încurajează să îl completați la momentul respectiv.

**TOTAL disponibile: 65p**

**EXAMEN ONLINE - Instrucțiuni generale**

1. Transmiteți examenul **prin Moodle** până la termenul limită: **27 ianuarie, ora 10:00**.
  - Transmiterea corectă a examenului este strict în responsabilitatea studenților.
  - Transmiteți în timp util, NU așteptați ultimele minute pentru a încărca examenul. Examenul poate fi transmis de oricâte ori doriți până la deadline, se ia în considerare doar ultima variantă transmisă. NU se accepă ca motivație pentru netransmiterea examenului niciun fel de probleme tehnice (încetinirea platformei, utilizarea incorectă, nesincronizări ale ceasului platformei, etc.).
  - Studenții care nu transmit rezolvarea examenului scris sunt considerați absenți.
2. Răspunsul trebuie să fie în **format .pdf, încărcat prin contul instituțional Moodle** în secțiunea corespunzătoare sub numele **grupa\_nume\_prenume.pdf**. Prima pagină a fișierului de răspunsuri trebuie să conțină **nume, grupă, o listă a subiectelor ne tratate** (ex.: *Subiecte ne tratate: 1(a), 1(c), 3(b). sau -*).
3. Se acordă punctaje parțiale. Răspunsurile greșite la examenul scris NU depunțează suplimentar.
4. Pentru promovare, **este obligatoriu să participați la ambele probe (examen scris și oral), să obțineți minim 10 puncte la examenul final și minim 45 de puncte** ca notă finală (include punctele obținute în timpul anului).
5. Pentru examenul oral:
  - Este strict în responsabilitatea studenților să verificați repartizarea pe zile / ore (aprox.) și alte informații necesare referitoare la susținerea examenului oral.
  - Trebuie să vă conectați **audio-video, folosind contul instituțional Teams**.
  - Trebuie să arătați **un act de identitate**, de preferat **legitimatie / carnet de student cu poză**. Este în responsabilitatea studenților să ascundeți alte informații (altele decât numele și poza) de pe documentul prezentat, pe care nu doriți să le faceți publice!
  - Fiecare subiect rezolvat în scris, dar pe care nu știți să îl explicați (i.e., să arătați că l-ați rezolvat individual sau înțeles), **se depunțează cu dublul punctajului alocat subiectului respectiv**.
  - Studenții care transmit rezolvarea examenului scris dar nu participă la susținerea orală obțin nota finală 4.
  - Dacă există studenți care nu au posibilitatea unei conexiuni audio și video, trebuie să anunțe în prealabil, pe e-mail ([ruxandra.olimid@fmi.unibuc.ro](mailto:ruxandra.olimid@fmi.unibuc.ro)).

Dacă în timpul examenului aveți întrebări, le puteți posta pe forum, secțiunea *Examen*. Urmăriți formul pentru informații. **NU postați indicii sau soluții!**

SUCCES!

## EXAMEN ONLINE - Probleme

### 1. Adevărat sau Fals

Răspundeți cu adevărat sau fals. Dacă afirmația este falsă, transformați-o într-o afirmație adevărată printr-o schimbare minimală (i.e., păstrați contextul, dar nu negați). Subliniați modificarea adusă.

*Exemplu: RSA este un sistem de criptare simetric.*

*Răspuns: Fals. RSA este un sistem de criptare **asimetric**.*

- (a) Decriptarea, folosind OTP, a textului criptat 0x253505ba folosind cheia 0x717056ee este mesajul clar MARE. **(2p)**
- (b) Niciun sistem determinist nu poate fi CCA-sigur. **(2p)**
- (c) Un atac de tip Man-in-the-Middle este un atac activ. **(2p)**
- (d) Un PRP presupune ca pentru fiecare intrare, ieșirea să conțină exact biții de intrare, permutați presudoaleator (ex. pentru o cheie fixată  $K$  și  $PRP_K$  :  $\{0, 1\}^4 \rightarrow \{0, 1\}^4$ ,  $PRP_K(1101) = 1011$  poate fi o atribuire corectă dar  $PRP_K(1101) = 0101$  este întotdeauna o atribuire incorectă). **(2p)**
- (e) Este recomandat să se folosească RSA pentru transmiterea fișierelor în mod criptat. **(2p)**
- (f) Pentru a asigura integritatea unor fișiere personale, este suficient să stocați pe calculatorul propriu fișierele și valoarea SHA256 corespunzătoare fiecărui sub forma  $(file1, SHA256(file1)), (file2, SHA256(file2))$ . . . . . **(2p)**
- (g)  $SHA256(PAROLA) = 0x1c65fc11a8651621\ 765d50083695b33b\ 4de0d253ff984adb\ 62f64e4c0504ed1f$ . **(2p)**
- (h) Scopul principal al unui adversar împotriva schimbului de chei Diffie-Hellman este să spargă Problema Logaritmului Discret (PLD). **(2p)**
- (i) Faptul că majoritatea certificatelor digitale bazate pe RSA folosesc exponentul de criptare = 65537 nu este o problemă de securitate dacă valoarea factorilor  $p$  și  $q$  este întotdeauna mare. **(2p)**
- (j) SSL/TLS implementează principiul diversității (*principle of diversity*) pentru că folosește în *handshake protocol* criptografia asimetrică (ex. certificate pentru autentificare) și în *record protocol* criptografia simetrică (ex. criptarea cu AES). **(2p)**

### 2. Vi se cere să faceți un audit al unei aplicații web de comerț electronic. Observați următoarele:

- Credențialele utilizatorilor se stochează în baza de date sub forma  $(username, e-mail, H(password,salt))$ , unde  $H$  este o funcție hash proprietară. În caz de pierdere a parolei, aceasta se poate reseta prin accesarea unui link transmis pe e-mail. Valabilitatea acestui link este de 1 oră de la momentul generării, link-ul fiind generat folosind un PRNG cunoscut, care primește ca seed username-ul și ziua curentă.

- Conexiunea client-server este securizată prin TLS (la accesarea aplicației web din browser puteți vizualiza un certificat digital valid, emis de o autoritate recunoscută). Conform TLS, comunicarea client-server folosește două chei pentru asigurarea confidențialității și două pentru integritate, câte una pentru fiecare sens de comunicație.
- Stocarea locală a fișierelor confidențiale (ex. a facturilor) se realizează direct în baza de date, după o criptare prealabilă AES-ECB.
- Integritatea end-to-end a mesajelor  $m$  transmise în modulul de chat (folosit pentru comunicarea cu reprezentanții de vânzări) este asigurată de o funcție  $CRC(m)$ , a cărei valoare se apendează mesajului transmis.
- Cu excepția paginii de login, câmpurile de introducere date nu sunt sanitizate și validate. Spre exemplu, aplicația permite introducerea unor date în trecut pentru ziua plasării comenzi, adăugarea unor prețuri negative, etc.

Răspundeți la următoarele cerințe:

- Enunțați un principiu de securitate (referiți-vă la *Pages on Security - Principles*) care este satisfăcut. Argumentați. (1 paragraf) **(5p)**
  - Enunțați un principiu de securitate (referiți-vă la *Pages on Security - Principles*) care este NU este satisfăcut. Argumentați. (1 paragraf) **(5p)**
  - Ce puteți spune despre confidențialitatea, respectiv integritatea de la nivelul aplicației? Argumentați. (1 paragraf) **(2x2.5p)**
  - Dați exemplu de un atac activ care vă poate permite logarea prin impersonarea unui alt utilizator. (1 paragraf) **(5p)**
3. Considerați varianta modificată de semnătură RSA (plecând de la standardul PKCS#1) definită astfel:
- Fie  $pk = (N, e)$  și  $sk = (p, q, d)$  cheile RSA publică, respectiv privată. Se semnează un mesaj  $m$ ,  $0 < |m| < |N|/2$ , unde  $|x|$  este dimensiunea în biți a lui  $x$  astfel:
- Pasul 1.** Se realizează paddingul  $m' = 0^8||0^71||FF^x||0^8||m$ , unde  $\parallel$  este concatenare,  $0^8$  este un byte de 0,  $0^71$  este un byte cu primii 7 biți 0 și ultimul bit 1,  $FF^x$  este byte-ul  $FF$  repetat de  $x$  ori și  $x > 1$  este aleator ales astfel încât  $0 \leq m' \leq N - 1$ .
- Pasul 2.** Se obține semnătura  $\sigma = sign(m, sk) = m'^d \text{ mod } N$
- Explicați cum funcționează funcția de verificare a semnăturii  $verif()$  în acest caz. Precizați în mod clar input-ul, output-ul și modalitatea de calcul. **(5p)**
  - Primiți  $(m, \sigma)$  cu  $\sigma$  semnătură validă pentru  $m$ . Poate fi  $(2m, 2^d\sigma)$  o semnătură validă? De ce/de ce nu? **(5p)**
  - O extensie a acestei scheme permite semnarea mesajelor  $|\bar{m}| \geq |N|/2$  prin adăugarea unui pas anterior:  
**Pasul 0.**  $\bar{m}$  se transformă în  $0 < |\bar{m}| < |N|/2$  astfel:  $m = lsb_{|N|/2-1}(\bar{m} \text{ mod } N)$ , unde  $lsb$  reprezintă cei mai puțin semnificativi biți.  
Ilustrați un atac asupra schemei astfel definită. **(5p)**
  - Propuneți o modificare simplă asupra schemei extinse pentru ca atacul anterior să nu mai funcționeze. **(5p)**

4. Fie  $(\text{Mac}, \text{Vrfy})$  un MAC sigur definit peste  $(K, M, T)$  unde  $M = \{0, 1\}^n$  și  $T = \{0, 1\}^{128}$ . AND și NOT sunt operațiile logice cunoscute, pe biți. Este MAC-ul de mai jos sigur? Argumentați. **(5p)**

$$\text{Mac}'(k, m) = \text{Mac}(k, m \text{ AND } \text{NOT}(m))$$

$$\text{Vrfy}'(k, m, t) = \text{Vrfy}(k, m \text{ AND } \text{NOT}(m), t)$$

**TOTAL disponibile: 65p**

# Examen

---

## Securitatea Sistemelor Informatice

Theodor-Pierre Moroianu – 334

January 26, 2022

### Exercitiul 1

#### A

Rulam urmatorul script:

```
C = 0x253505ba
K = 0x717056ee
M = C ^ K
mesaj = M.to_bytes(4, "big")

print("\n".join([chr(i) for i in mesaj]))
# > TEST
```

Asadar, afirmatia este FALSA. Folosind OTP, textul criptat reprezinta in clar mesajul **"TEST"**.

#### B

Stim ca daca un sistem este CCA-sigur, atunci este automat CPA sigur.

Orice sistem determinist nu este CPA-sigur (putem sa cerem oracolului de decriptare sa il decripteze pe  $m_1$  si  $m_2$ ), deci afirmatia este ADEVARATA.

#### C

Un atac activ este un atac in care atacatorul intervine fizic (spre diferenta unui atac pasiv, in care atacatorul doar asculta ce se intampla). Asadar, un atac Man-in-the-Middle este activ, si afirmatia este ADEVARATA.

#### D

Afirmatia este FALSA. Prin permutare random, se refera la faptul ca functia este o auto-bijectie peste multimea  $\{0, 1\}^4$ , nu ca permute bitii unui numar (in acelasi mod in care o permutare a numerelor de la 0 la 99 nu reprezinta o permutare a celor doua cifre a fiecarui numar).

Un PRP presupune ca pentru fiecare intrare iesirea pare aleatoare pentru oricine nu stie cheia de encriptare, si ia valori distincte (este bijectiva).

## E

RSA este un algoritm foarte lent, folosit in principal pentru schimb de chei sau semnaturi digitale. Asadar, nu este recomandata transmiterea de fisiere doar cu RSA. Este recomandata schimbarea unei chei publice cu RSA si dupa aceea folosirea unui algoritm simetric cum ar fi AES. Afirmatia este FALSA.

Este recomandat sa se foloseasca **RSA combinat cu o metoda de encodare simetrica, intr-o schema hibrida** pentru transmiterea fisierelor in mod criptat.

## F

Salvarea unui hash pe langa fiecare fisier personal ofera integritatea datelor impotriva unor modificari datorite unor defectiuni hardware, dar nu ofera nicio securitate impotriva unui atacator activ, care poate altera un fisier si inlocui hash-ul acestuia cu noul hash. Asadar, afirmatia este FALSA.

Pentru a asigura integriatea unor fisiere personale, este suficient sa stocati pe calculatorul propriu fisierele si pe un dispozitiv securizat – **ideal mai multe dispozitive, neconectate la retea, hashurile fisierelor. Alternativ, se pot salva pe calculator hasurile fisierelor, encodeate cu o cheie care este tinuta pe un dispozitiv securizat / scrisa pe o foaie.**

## G

Hash-ul SHA256 al cuvantului ”PAROLA” este **467b4a3eca61a4e6 2447400d93fc35d4295c08ffa2b04 ae942f4de03fa62f464**. Asadar, afirmatia este FALSA.

Folosind website-ul <https://hashes.com/en/decrypt/hash>, putem verifica ce cuvant ne-a dat hashul dat in enunt: ”PAR123”.

**SHA256(PAROLA)=0x467b4a3eca61a4e6 2447400d93fc35d42 95c08ffa2b04 ae942f4de 03fa62f464.**

## H

Un atac impotriva algoritmului de schimb de chei Diffie-Hellman poate fi efectuat prin rezolvarea problemei logaritmului discret, dar rezolvarea acestei probleme este foarte grea, si nu constituie scopul principal al atacului (este doar una din posibilele metode de-a afla cheia generata prin Diffie-Hellman), asadar afirmatia este FALSA.

Scopul principal al unui adversar impotriva schimbului de chei Diffie-Hellman este **sa afle care este cheia generata de cele doua entitati care urmeaza sa fie folosita in encriptarea simetrica a datelor.**

## I

Afirmatia este ADEVARATA. Folosirea exponentului 65537 nu este o problema. Din punct de vedere teoretic, nu exista nicio restrictie asupra exponentului  $e$ , infara de cerinta ca acesta sa fie prim cu  $p$  si  $q$ , cei doi factori ai modulului, si suficient de mare ca  $m^e$  sa fie considerabil mai mare ca  $N$ . In practica, 65537 este cel mai folosit folosit exponent de RSA, atat din motive istorice cat si de performanta ( $65537 = 2^{16} + 1$ , deci exponentierea poate fi efectuata prin 17 inmultiri).

## J

Afirmatia este ADEVARATA. TLS foloseste mai multe protocoale cryptografice, cum ar fi un schimb de chei printr-o comunicare pe baza de chei asimetrice, o encriptare a traficului cu o cheie simetrica si generarea unui digest, pentru a oferi autenticitate, confidentialitate si integritate.

## Exercitiul 2

### A

Un principiu care este satisfacut este principiul diversitatii.

Conform definitiei, principul diversitatii spune ca consta in "Use different types of cryptographic algorithms", pentru a evita un singur atac impotriva intregului sistem.

Putem observa ca aplicatia web foloseste:

- Stocarea parolelor sub forma de hash cu un salt, ceea ce ofera (cel putin in principiu) confidentialitatea parolelor.
- O conexiune TLS intre server si client, care ofera confidentialitatea si integritatea datelor trimise.
- Stocarea locala a fisierelor prin criptarea cu AES-ECB (care din pacate nu este perfect sigur, permite unui atacator sa observe corelatii intre blocurile criptate).
- Foloseste (sau cel putin incerca sa foloseasca) un sistem MAC pentru a oferi integritatea mesajelor in sistemul lor end-to-end.
- Sanitizeaza formul de login, pentru a evita atacuri de tipul SQL-injection.

Asadar, prin multitudinea abordarilor si algoritmilor criptografici folositi, aplicatia respecta principiul diversitatii.

### B

Un principiu care nu este satisfacut de aplicatie este principiul lui Kerckhoff, care afirma ca sigurele informatii ascunse trebuie sa fie cheile folosite.

Cum functia H de hash folosita pentru stocarea parolelor este o functie hash proprietara, aceasta incalca principiul lui Kerckhoff. Aceasta abordare nu este sigura, pentru ca nu poate garanta folosirea unei functii hash sigure (mult mai putine persoane o pot verifica), si leakuirea acestora poate conduce la probleme grave de securitate si de confidentialitate a parolelor.

Alt principiu incalcat este Security By Design, avand in vedere ca website-ul nu ofera validare pentru toate informatiile primite (accepta preturi negative, introducerea unei date din trecut etc).

### C

Atat confidentialitatea cat si integritatea la nivel de aplicatie sunt compromise.

Integritatea este compromisa prin sistemul de integritate end-to-end, care foloseste o functie CRC pentru a asigura integritatea datelor trimise.

CRC este folosit in retelistica pentru a asigura ne-alterarea din defecte hardware a pachetelor trimise, dar nu ofera nicio securitate impotriva unui atac activ. Spre diferența folosirii unui algoritm MAC, CRC-ul nu necesita nicio cheie, si deci atacatorul poate sa schimbe mesajul trimis, si sa recalculeze CRC-ul corespunzator.

Confidentialitatea datelor este compromisa prin folosirea algoritmului de criptare AES-ECB, care nu este sigur din punct de vedere semantic. Acest lucru este explicat atat la laborator cat si pe pagina de documentatie al unei implementari ale algoritmului:

<https://pycryptodomex.readthedocs.io/en/latest/src/cipher/classic.html#ecb-mode>.

Confidentialitatea este si compromisa prin sistemul naiv folosit pentru resetarea parolelor, care permite unui atacator sa imparsoneze cu usurinta alt utilizator.

## D

Un atac foarte simplu impotriva sistemului de recuperare a parolei, care permite unui atacator sa imparsoneze orice utilizator, este urmatorul:

1. Atacatorul afla username-ul persoanei pe care doreste sa o imparsoneze.
2. Atacatorul foloseste functionalitatea website-ului de recuperare a parolei, folosind usernamul gasit, si cere website-ului sa genereze si sa trimita pe email linkul de resetare a parolei.
3. Pentru a gasi linkul de resetare a parolei, atacatorul poate folosi PRNG-ul cunoscut, folosind ca seed username-ul si ziua curenta (ambele cunoscute). Poate asadar initializa acelasi PRNG ca cel folosit pentru generarea linkului de resetare, cu acelasi seed. Obtine deci acelasi link.
4. Dupa ce a generat link-ul cu ajutorul PRNG-ului compromis, atacatorul poate folosi linkul pentru a reseta parola.
5. Atacatorul se logheaza pe contul utilizatorului cu noua parola.

## Exercitiul 3

### A

Algoritmul RSA functioneaza pe baza urmatoarei ecuatii:

$$(a^d)^e \equiv (a^e)^d \equiv a \pmod{N}$$

Asadar, pentru a obtine pe  $m'$  inapoi din  $\sigma$ , este suficient sa folosim egalitatea  $m' \equiv \sigma^e$ .

Dupa ce a fost gasit  $m'$ , dorim sa verificam daca poate fi scris conform regulii de mai sus:

$$m' = 0^8 || 0^7 1 || FF^x || 0^8 || m$$

Asadar, algoritmul de verificare a unei semnaturi este  $Verif(N, e, m, \sigma)$ :

1. Calculam  $m' \equiv \sigma^e \pmod{N}$ .
2. Verificam daca scrierea in baza 2 al lui  $m'$  este de forma urmatoare:
  - Cei  $|m|$  cei mai nesemnificativi biti sunt egali cu  $m$ .
  - Urmatorii 8 cei mai nesemnificativi biti sunt 0.
  - Urmatorii  $8k + 1$  biti sunt 1, pentru un  $k$  intreg pozitiv.
  - Toti ceilalti biti sunt 0.
3. Daca pasul 2. a functionat, atunci acceptam semnatura, daca nu o refuzam.

Functia primeste ca input cheia publica, mesajul plain text, si semnatura acestuia. Concert, primeste:

- Cheia publica ( $N$  si  $e$ ).
- Mesajul pe care dorim sa il verificam ( $m$ ).
- Mesajul semnat cu cheia privata ( $\sigma$ ).

Outputul verificatorului este un boolean DA/NU.

### B

Fie  $m_2 = 2 * m$ , si  $m'_2$  mesajul cu padding corespunzator lui  $m_2$ .

Uitandu-ne la reprezentarea in baza 2,  $m_2$  arata identic ca  $m$ , cu exceptia unui bit de 0 adaugat la sfarsit. Asadar, si  $m'_2$  va fi identic cu  $m'$  cu exceptia unui bit de 0 in plus la sfarsit.

Asadar, avem:

$$m'_2 = 2 * m'$$

Fie  $\sigma_2$  semnatura lui  $m_2$ . Conform definitiei,  $\sigma_2$  va fi:

$$\sigma_2 \equiv m'_2^d \equiv (2 * m')^d \equiv 2^d * m'^d \equiv 2^d * \sigma$$

Asadar,  $sign(m', sk) = 2^d * \sigma$ , deci perechea  $(2m, 2^d\sigma)$  este o semnatura valida.

Desi poate parca un risc de securitate, generarea a astfel de semnaturi, plecand de la o alta semnatura valida, necesita cunoasterea cheii private  $sk$ , mai precis de  $d$ . Cum acesta nu este facut public, aceasta metoda nu poate fi folosita pentru a semna mesaje fara a cunoaste cheia privata.

## C

Vulnerabilitatea acestei extensii este ca mesajele  $m, m + N, \dots, m + k * N$  se vor reduce toate la aceeasi semnatura, toate numerele fiind reduse prin **PAS 0.** la  $m \bmod N$ . Concret, adaugarea sau scaderea din  $m$  al lui  $N$  nu schimba validitatea semnatului.

Un atac posibil este:

1. Atacatorul intercepteaza un mesaj  $(m, \sigma)$  cu o semnatura valida.
2. Atacatorul modifica  $m$ , scazand sau adaugand multiplii de  $N$ , obtinand  $m_{bad} = m + k * N$ .
3. Atacatorul transmite mai departe mesajul  $(m_{bad}, \sigma)$ , care prezinta in continuare o semnatura valida, pierzand astfel integritatea mesajului.

Concret, orice mesaj  $m_{bad}$  cu  $m_{bad} \equiv m \pmod{N}$  va avea aceeasi semnatura ca  $m$ . Putem asadar sa luam orice fisier / program / date, sa le apenduim  $\log_2(N)$  biti la sfarsit ca sa dea restul bun la impartirea la  $N$ , si acesta va pastra semnatura valida.

Putem astfel sa interceptam de exemplu un program normal, pe care il inlocuim pe un program malitos. Pentru a pastra valoarea modulo  $N$ , putem sa adaugam bitii necesari la sfarsitul programului malitos, care nu influenteaza rularea acestuia.

## D

Pentru a evita atacul descris mai sus, propun doua modificari posibile:

### Semnam hashul lui $m$

In loc sa il semnam direct pe  $m$ , care poate fi mai mare decat  $N$ , putem sa:

1. Calculam  $h = SHA256(m)$ . Evident, putem folosi orice alta functie de hash, si presupunem ca  $N$  are cel putin 512 de biti (cam toate modulurile de RSA au 2048 de biti).
2. Semnam valoarea  $h$ , care este in limitele necesare algoritmului descris in enunt.

Aceasta abordare evita complet nevoia de-a trata situatii cand numarul de semnat este prea mare, fara sa ofere unui atacator posibilitatea de-a altera mesajul (presupunand ca functia de hash folosita este rezistenta la a doua preimagine).

Pentru a verifica semnatura, este suficient ca functia de verificare sa aplique aceeasi functie de hash asupra lui  $m$ .

### Semnam independent cifrele dintr-o baza mai mica

Ne alegem un numar  $MOD$ , de exemplu  $\sqrt{N}$ . Il scriem pe  $m$  in baza  $MOD$ :

$$m = \sum_{i=0} b_i * MOD^i$$

Observam ca scrierea este unica, si fiecare "cifra"  $b_i$  are cel mult  $|N|/2$  cifre.

Putem acum sa ne construim vectorul  $\sigma_i$ , prin ecuatia:

$$\sigma_i = sign(b_i, sk)$$

Vectorul  $\sigma_i$  reprezinta asadar semnatura fiecarui bloc de  $|N|/2$  biti, care poate fi transmis cu mesajul. Functia de verificare a semnatului imparte in mod analog  $m$  in baza  $MOD$ , si verifica independent fiecare bucata.

Desi aceasta metoda este mai simpla, are doua probleme:

- Necesa transmiterea unui vector pentru semnatura, in loc de o singura valoare.
- Este vulnerabil la un atac in care atacatorul schimba ordinea blocurilor din  $m$  si din  $\sigma$ . Acest lucru poate fi mitigat prin folosirea altui mod de operatie al block ciphers, cum ar fi modul cu feedback (CFB), care impiedica inversiunea bucatilor din  $m$ .

## Exercitiul 4

Observatia cheie pentru a rezolva exercitiul este urmatoarea:

$$\forall X, \quad X \text{ AND NOT}(X) = 0$$

Asadar, conform definitiei:

$$\text{Mac}'(k, m) = \text{Mac}(k, m \text{ AND NOT}(M)) = \text{Mac}(k, 0)$$

Altfel spus,  $\text{Mac}'(k, m)$  este o functie constanta, care depinde doar de cheia  $k$ .

Asadar, observam ca indiferent ce mesaj  $m$  este pasat, calcularea MAC-ului se traduce in calcularea  $\text{Mac}(k, 0)$ , si verificarea MAC-ului se traduce in calcularea  $\text{Vrfy}(k, 0, t)$ .

In mod evident, MAC-ul acesta nu este sigur de oarece putem modifica mesajul pasat oricum ne dorim, si verificarile de integritate vor valida mesajul.

Nume: \_\_\_\_\_ Grupă: \_\_\_\_\_

**1. Adevărat sau Fals**

Răspundeți cu adevărat sau fals. Dacă afirmația este falsă, transformați-o într-o afirmație adevărată printr-o schimbare minimală (i.e., păstrați contextul, dar nu negați). Subliniați modificarea adusă.

*Exemplu: RSA este un sistem de criptare simetric.*

*Răspuns corect: Fals. RSA este un sistem de criptare asimetric.*

- (a) Sistemul OTP sigur permite folosirea unei singure chei de criptare pentru mai multe mesaje diferite atâtă timp cât lungimea cheii este cel puțin la fel de mare ca lungimea mesajelor. **(2p)**
  - (b) Este recomandat să se folosească AES pentru transmiterea fișierelor de dimensiuni mari. **(2p)**
  - (c) Pentru schimbul de chei Diffie-Hellman este esențial ca cele două părți să partajeze în avans un secret. **(2p)**
  - (d) Pentru a asigura integritatea mesajelor, este suficient să le trimitem criptate cu un sistem de criptare public și sigur. **(2p)**
  - (e) La crearea semnăturilor digitale, are mai mult sens mai întâi să semnăm mesajul și apoi să îl comprimăm cu o funcție hash.
  - (f) Timpul necesar unui atacator pentru a găsi coliziuni pentru funcția hash SHA-256 este de  $2^{256}$ . **(2p)**
  - (g) Nerepudierea îi permite lui Bob să prezinte spre verificare unei terțe părți un document semnat de Alice și această proprietate este asigurată numai de codurile de autentificare a mesajelor (MAC). **(2p)**
  - (h) Combinarea autentifică-apoi-cripteză este întotdeauna sigură indiferent de cum sunt instanțiate componentele ei. **(2p)**
  - (i) Sistemele de criptare post-cuantice sunt sisteme de criptare care folosesc metode cuantice pentru asigurarea securității în fața unui adversar clasic. **(2p)**
  - (j) Protocolul TLS este folosit de către browser-ul web de fiecare dată când realizează o conexiune sigură cu un site web folosind *http*. **(2p)**
2. Sunteți angajat să verificați securitatea în cadrul unei companii. Observați că se folosesc următoarele:
    - Sistemul de criptare AES pentru stocarea criptată a fișierelor unde cheia pentru criptare/decriptare este generată cu un PRNG cunoscut care primește ca seed ziua și numele companiei.
    - Funcția hash SHA-2 pentru stocarea parolelor clientilor (de 6 caractere) cu un *salt* pe 8 biți.
    - Protocolul de schimb de chei Diffie-Hellman autentificat (deci rezistent la un atac de tip man-in-the-middle) pentru generarea cheilor necesare securizării comunicației interne (i.e., între angajații firmei) într-un grup  $\mathbb{G}$  în care problema logaritmului discret este ușoară. O cheie astfel generată este apoi utilizată ca și cheie secretă a sistemului de criptare 3DES pentru transmiterea criptată a mesajelor.

- Site-ul web al companiei este securizat folosind certificate digitale cu modulul RSA N pe 128 biți.
- Integritatea end-to-end a mesajelor  $m$  transmise în modulul de chat (folosit pentru comunicarea în cadrul companiei) este asigurată de algoritmul simetric de criptare DES aplicat mesajului, a cărui valoare se apendează mesajului transmis.

Răspundeți la următoarele cerințe:

- Sunt parolele clientilor stocate în mod sigur? Dar fișierele criptate? Argumentați. (1 paragraf) **(10p)**
  - Ce puteți spune despre securitatea sistemului RSA folosit în cadrul certificatelor digitale? Argumentați (1 paragraf) **(5p)**
  - Există alte probleme de securitate (confidențialitate, integritate) la nivelul aplicației? Argumentați. (1 paragraf) **(5p)**
3. Se consideră modul de operare definit mai jos pentru criptarea unei secvențe de blocuri de text clar  $m_1||m_2||m_3||\dots$  într-o secvență de blocuri  $c_1||c_2||c_3||\dots$ :
- $$c_i = F_{k_2}(c_{i-1} \oplus F_{k_1}(m_{i-1} \oplus m_i)), i \geq 1$$
- unde  $m_0$  și  $c_0$  sunt vectori de inițializare publici și fixați.
- Ce reprezintă notația  $F_{k_i}$ , pentru  $i \in \{1, 2\}$ ? Ce proprietate esențială trebuie să satisfacă funcția  $F_{k_i}$  pentru ca sistemul să fie corect? **(2 × 2.5p)**
  - Indicați cum se realizează decriptarea. **(7.5p)**
  - Câte valori posibile pot lua  $m_0$  și  $c_0$  dacă sunt reprezentate fiecare pe 16 biți? **(2.5p)**
  - Presupunând că un bloc  $c_i$  suferă erori de transmisie, care blocuri de text clar sunt impacitate? **(5p)**
4. Fie  $(Mac, Vrfy)$  un MAC sigur definit peste  $(K, M, T)$  unde  $M = \{0, 1\}^n$  și  $T = \{0, 1\}^{128}$ . Este MAC-ul de mai jos sigur? Argumentați răspunsul.

$$Mac'(k, (m_1, m_2)) = Mac(k, m_1 \oplus m_2)$$

$$Vrfy'(k, (m_1, m_2), t) = Vrfy(k, (m_1, m_2), t)$$

**(5p)**