



# Criptografie și Securitate

- Prelegerea 0 -  
Informații administrative

Ruxandra F. Olimid

(multe slide-uri preluate din cursul comun realizat impreuna cu Adela Georgescu)

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Cadre didactice
2. Ce aştept de la voi?
3. Organizare și evaluare
4. Structura cursului
5. Referințe bibliografice

# Cadre didactice



Ruxandra F. Olimid



ruxandra.olimid@fmi.unibuc.ro



[www.ruxandraolimid.weebly.com](http://www.ruxandraolimid.weebly.com)



Seminar, Laborator:

Cosmin Obretin, Ionut-Daniel Dobos

Alexandru Cristian Matei, Daniel Popescu

# Ce aştept de la voi?



Sunt prezent pentru că mă interesează!



Studiez individual



Întreb pentru că vreau să ştiu!



Promovez (cu o notă bună)

# Ce așteptați voi de la mine?

## Feedback - students

This form is anonymous. This form does not replace the official feedback form (received from the faculty), please fill that in whenever asked by the decanate!

\* Required

Course \*

Cryptography  
 Network Security

Feedback

Your answer

Submit

<https://forms.gle/hX5YDSpL16jPssgi7>

# Organizare și evaluare

## 1. Organizare:

- ▶ 2h curs / săpt
- ▶ 2h seminar / 2 săpt
- ▶ 2h laborator / 2 săpt

## 2. Evaluare:

- ▶ 60 % examen (cu materiale)
- ▶ 10 % seminar
- ▶ 10 % laborator
- ▶ 10 % proiect (deadlines!)
- ▶ 10 % temă (deadlines!)
- ▶ +10% bonus :)

## 3. Condiții de promovare:

- ▶  $\geq 45\%$  din total

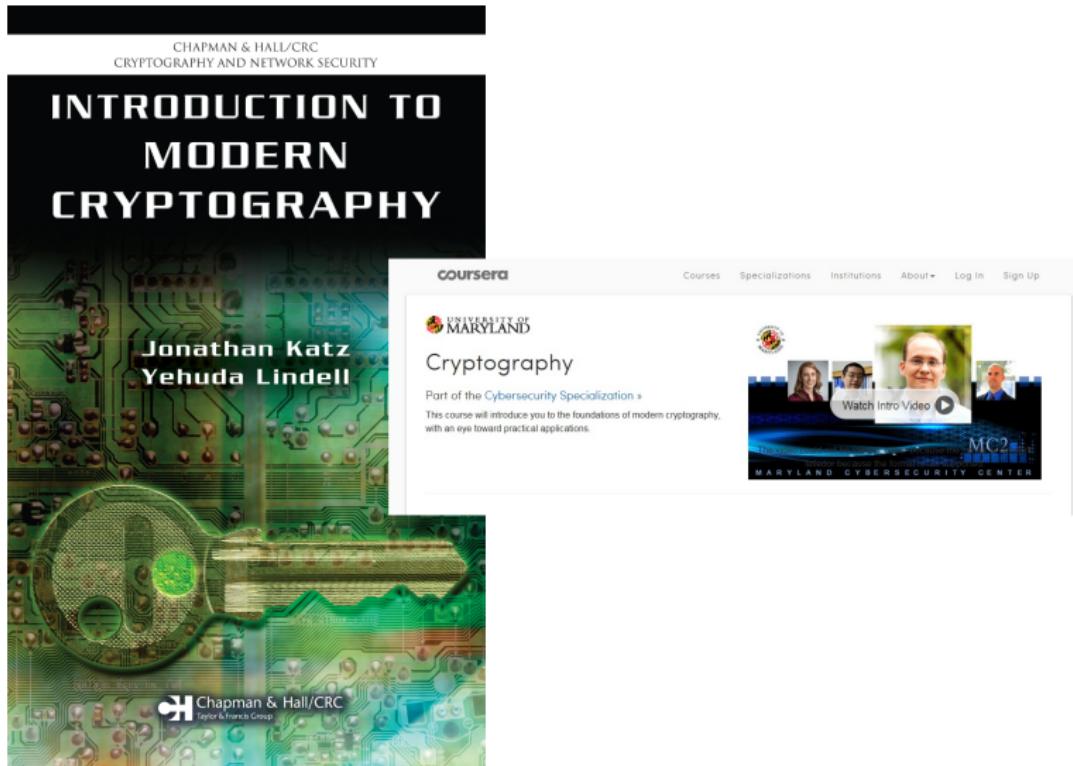


Moodle: Crypto

# Structura cursului

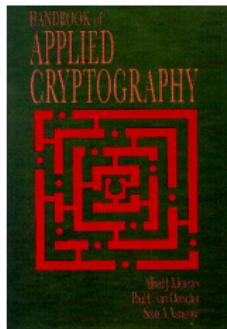
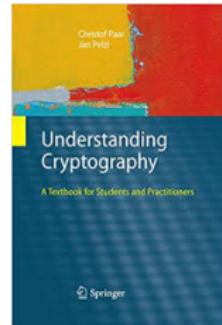
1. Introducere. Motivație. Principii.
2. Sisteme istorice de criptare
3. Securitate perfectă. One time pad.
4. Criptografia computațională. Pseudoaleatorismul.
5. Sisteme de criptare fluide.
6. Sisteme de criptare bloc.
7. Integritatea mesajelor (MAC). Funcții Hash.
8. Noțiuni de teoria numerelor. Probleme dificile în criptografie.
9. Criptografia cu cheie publică.
10. Criptografia pe curbe eliptice.
11. Alte topici de criptografie.

# Referințe bibliografice



<http://www.cs.umd.edu/~jkatz/imc.html>  
Criptografie și Securitate

# Referințe bibliografice



A screenshot of the Coursera website. The header shows 'COURSERA | Global Partners'. Below it, a box for the 'Stanford Cryptography I' course is displayed. The course title is 'Stanford Cryptography I'. Below it, a description reads: 'Learn about the inner workings of cryptographic primitives and how to apply this knowledge in real-world applications!'. A 'Previous Lecture' button is shown. To the right, there is a video player window titled 'Watch intro-video' with a play button. At the bottom, there is a 'About the Course' section with a brief description and a link to 'View Syllabus', and an 'About the Instructor' section with a photo of the instructor and their name.

<http://cacr.uwaterloo.ca/hac/>  
Criptografie și Securitate

<https://www.coursera.org/course/crypto>  
9/9



# Criptografie și Securitate

- Prelegherea 1 -

Introducere. Motivație. Principii

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

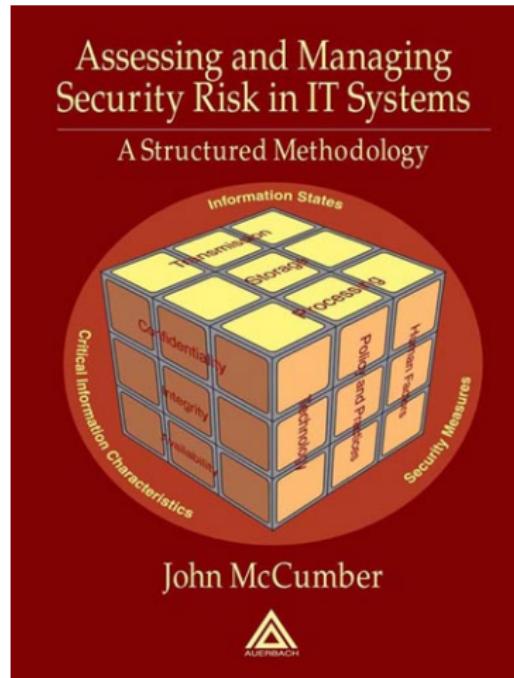
1. Ce este criptografia?
2. Motivație
3. Principiile lui Kerckhoffs

# Ce este criptografia?

cripto + graphe (scriere) = "scriere secretă"

- ▶ *scriere secretă cu ajutorul unui cod de semne convenționale*  
DEX (1998), Dicționar Enciclopedic (1993)
- ▶ *the art or writing or solving codes*, The Concise Oxford Dictionary (2006)
- ▶ *the scientific study of techniques for securing digital information, transactions, and distributed computations*, J.Katz, Y.Lindell, *Introduction to Modern Cryptography* (2008)

# Cubul McCumber (1991)



# Obiectivele criptografiei

**Confidențialitate:** păstrarea secretului informației, accesul la informația sensibilă fiind disponibilă doar persoanelor autorizate.

**Integritate (a datelor):** eliminarea posibilității de modificare (schimbare, inserare, ștergere) neautorizată a informației.

**Disponibilitate:** permiterea entităților autorizate să acceseze în timp util și fiabil informația.

**Autentificare:** identifică o entitate sau atestă sursa datelor.

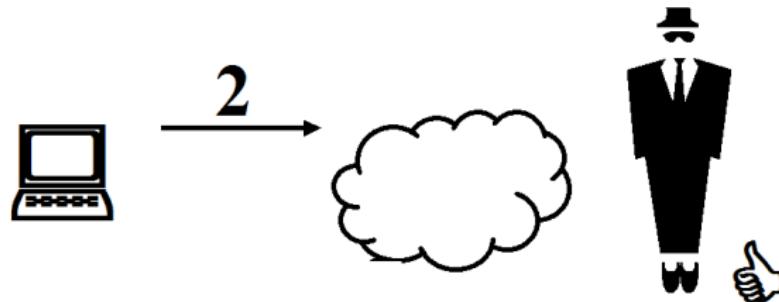
**Non-repudiere:** previne negarea unor evenimente anterioare.

## Studiu de caz: Cumpărarea online a unui bilet



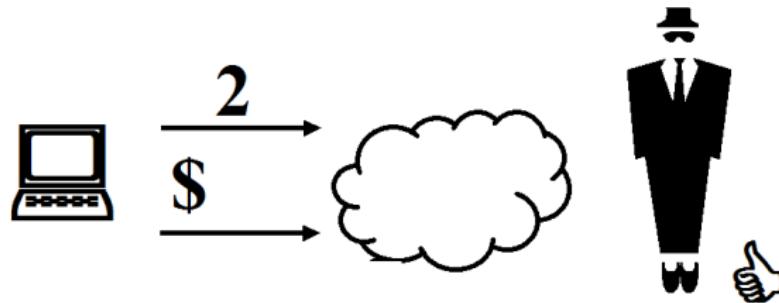
1. Un artist bun anunță un concert. Biletele sunt puse în vânzare online.

## Studiu de caz: Cumpărarea online a unui bilet



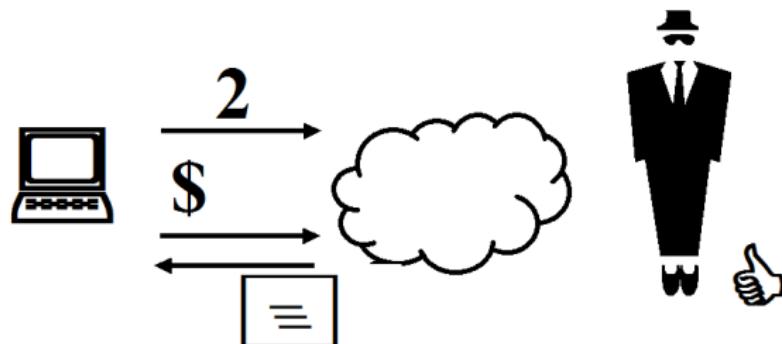
2. Vreau să cumpăr 2 bilete online.

## Studiu de caz: Cumpărarea online a unui bilet



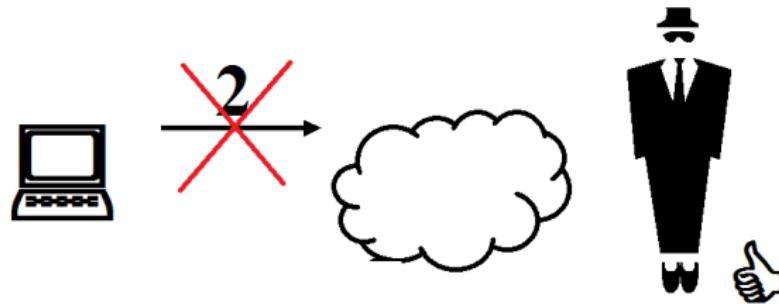
3. Fac plata electronic.

## Studiu de caz: Cumpărarea online a unui bilet



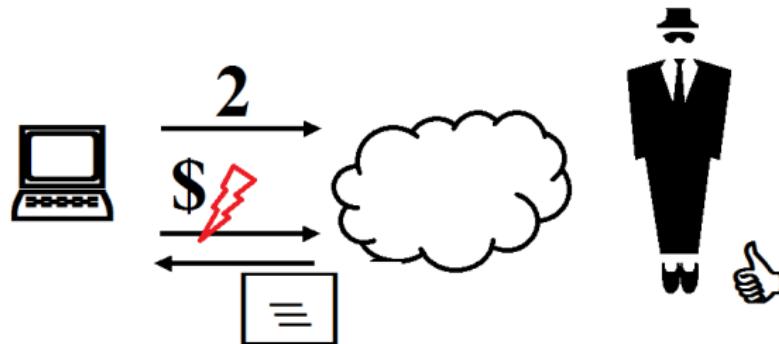
4. Primesc biletele.

## Studiu de caz: Cumpărarea online a unui bilet



**Disponibilitate:** Nu se poate accesa pagina web!

## Studiu de caz: Cumpărarea online a unui bilet



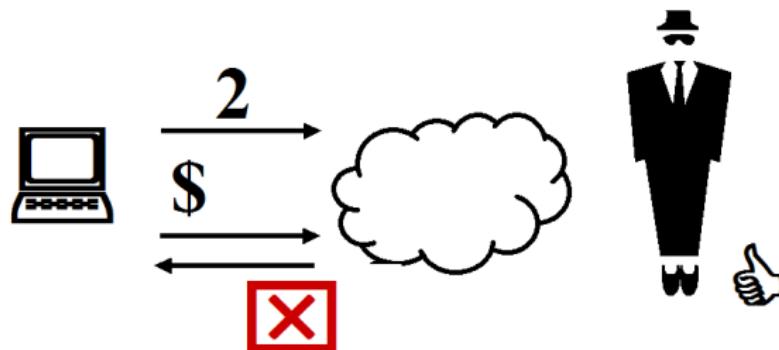
**Confidențialitate:** Se află CCV (Card Code Verification)!

## Studiu de caz: Cumpărarea online a unui bilet



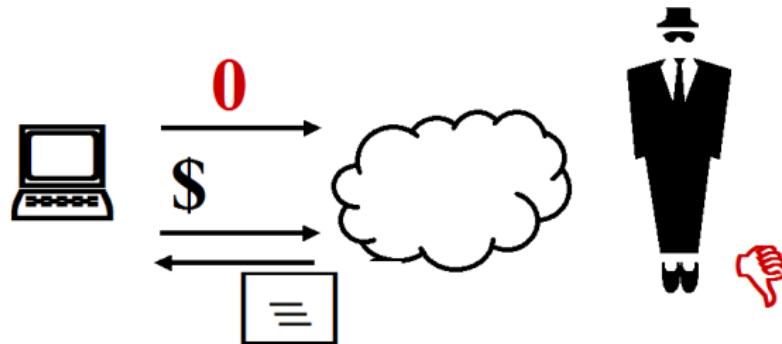
**Integritate:** Se modifică cererea!

## Studiu de caz: Cumpărarea online a unui bilet



**Autentificare:** Biletul este invalid!

## Studiu de caz: Cumpărarea online a unui bilet



**Non-repudiere:** Afirm că ca nu am solicitat bilete!

# Ce este criptografia?

## Definitie

*Criptografia este studiul tehniciilor matematice relate la aspecte ale securității informației precum confidențialitatea, integritatea datelor, autentificarea entităților sau a originii datelor. [HAC6].*

# Utilizarea criptografiei

- ▶ comunicare securizată (criptată)
- ▶ criptarea fișierelor, a bazelor de date
- ▶ autentificarea utilizatorilor
- ▶ securizarea tranzacțiilor bancare (e-cash)
- ▶ vot electronic
- ▶ ...

# Să ne cunoaștem

## Alice & Bob

A History of The World's Most Famous Cryptographic Couple



Alice and Bob are the world's most famous cryptographic couple. Since their invention in 1978, they have at once been called "inseparable," and have been the subject of numerous divorces, travels, and torments. In the ensuing years, other characters have joined their cryptographic family. There's Eve, the passive and submissive eavesdropper, Mallory the malicious attacker, and Trent, trusted by all, just to name a few.

### Alice and Bob are Born

Five years after public key cryptography was invented at GCHQ, two years after public key cryptography was re-invented by Diffie and Hellman, and a year and two articles after a practical cryptosystem was developed by Ron Rivest, Adi Shamir, and Leonard Adleman, Alice and Bob are finally born.

[Read more](#)



February, 1978



1985-88



### Eve is Born

As was customary for cryptology literature by this point, Charles Bennett, Giles Brassard, and Jean-Marc Roberts opened their 1985 abstract "How to Reduce Your Enemy's Information" with a story about Alice and Bob. This time, however, a new character was introduced: Eve.

[Read More](#)

<http://cryptocouple.com>

## Criptarea simetrică (cu cheie privată)



Alice

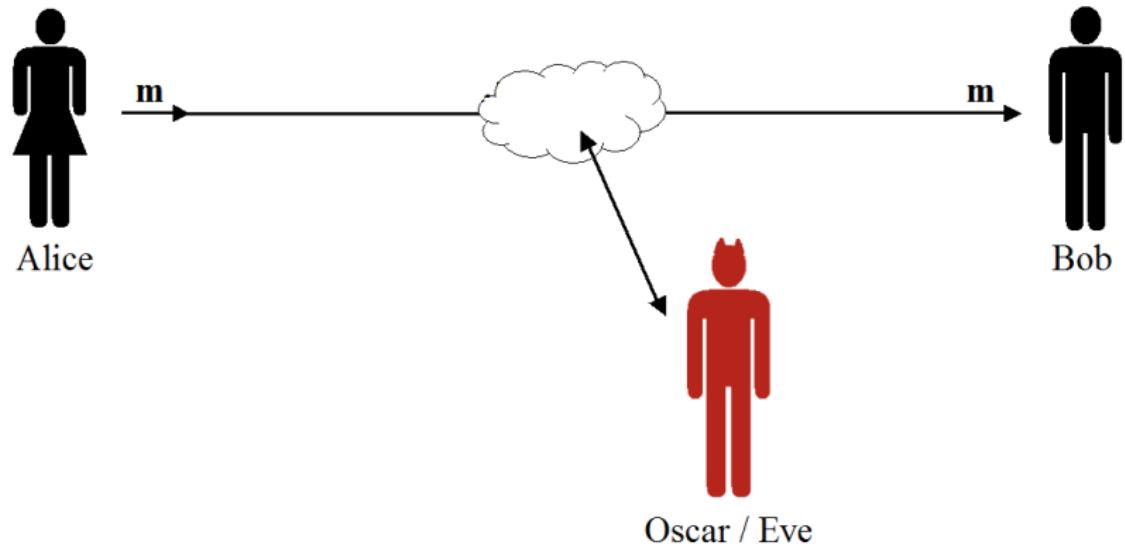


Bob

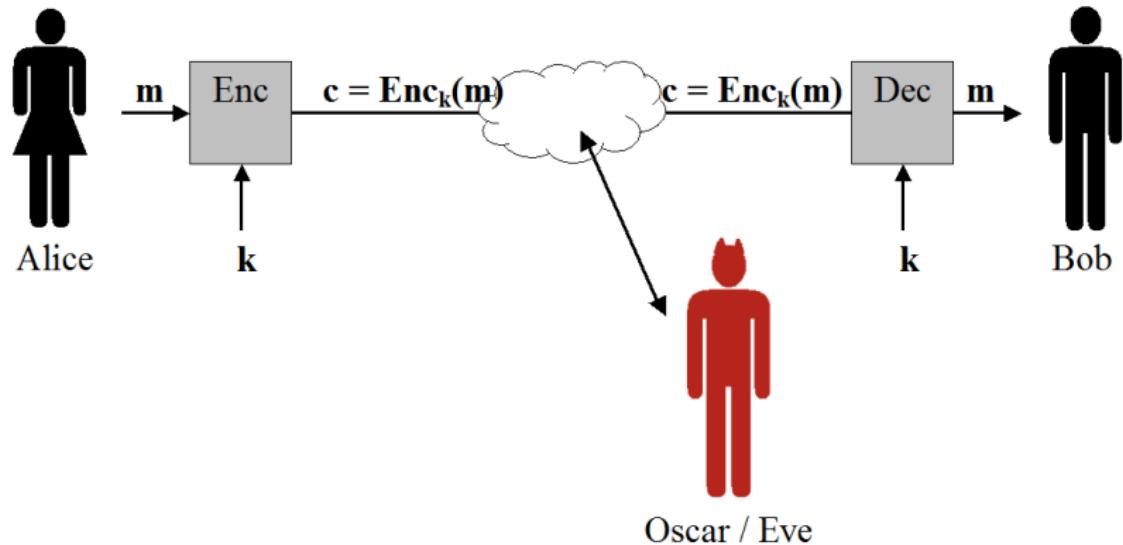


Oscar / Eve

## Criptarea simetrică (cu cheie privată)



## Criptarea simetrică (cu cheie privată)



# Criptarea simetrică (cu cheie privată)

## Definitie

Un *sistem de criptare simetrică* 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 dublet  $(\text{Enc}, \text{Dec})$ , unde:

1.  $\text{Enc} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$
  2.  $\text{Dec} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$
- a.i.  $\forall m \in \mathcal{M}, k \in \mathcal{K} : \text{Dec}_k(\text{Enc}_k(m)) = m.$

## Terminologie

- ▶ Mesajul în forma originară se numește **text clar**;
- ▶ Expeditorul rescrie mesajul folosind un sistem de criptare, adică îl **criptează** și obține un **text criptat**;
- ▶ Destinatarul îl **decriptează** cunoscând metoda folosită pentru **criptare**;
- ▶ Procesul de determinare a cheii aferente unui sistem de criptare, **cunoscând doar textul criptat** (eventual și alte informații auxiliare) se numește **criptanaliză**;
- ▶ Decriptarea și criptanaliza au același scop: găsirea textului clar; diferența constă în faptul că la criptanaliză nu se cunoaște cheia de decriptare.

## Scenarii de atac

- ▶ **Atac cu text criptat:** Atacatorul știe doar *textul criptat* - poate încerca un **atac prin forță brută** prin care se parcurg toate cheile până se găsește cea corectă;
- ▶ **Atac cu text clar:** Atacatorul cunoaște una sau mai multe perechi (*text clar, text criptat*);
- ▶ **Atac cu text clar ales:** Atacatorul poate obține criptarea unor texte clare alese de el;
- ▶ **Atac cu text criptat ales:** Atacatorul are posibilitatea să obțină decriptarea unor texte criptate alese de el.

# Dimensiunea cheii

Estimarea timpului de succes pentru un atac de tip **forță brută** asupra unui sistem de **criptare simetrică**:

---

Lungime cheie (biți)	Securitate estimată
56 - 64	termen scurt (ore sau zile)
112 - 128	termen lung (în absența calculatoarelor cuantice)
256	termen lung (în prezenta calculatoarelor cuantice, folosind algoritmii cunoscuți)

---

## Principiile lui Kerckhoffs

A. Kerckhoffs (1835 - 1903): *La Cryptographie Militaire* în *Journal des sciences militaires* (ian.1883)

1. Sistemul trebuie să fie practic, dacă nu matematic, indescifrabil.
2. **Principiul lui Kerckhoffs:** Sistemul nu trebuie să fie secret, poate să cadă ușor în mâinile adversarului (i.e. securitatea unui sistem de criptare nu constă decât în menținerea secretă a cheii).
3. Cheia trebuie să fie comunicată și menținută fără a fi notată, schimbată sau modificată la cererea corespondenților.
4. Sistemul trebuie să fie compatibil cu comunicarea telegrafică.
5. Sistemul trebuie să fie portabil și să nu necesite mai mult de o persoană.
6. Având în vedere circumstanțele în care este utilizat, sistemul trebuie să fie ușor de utilizat, fără să necesite aplicarea multor reguli.

<http://www.petitcolas.net/fabien/kerckhoffs/index.html>

# Principiile lui Kerckhoffs

**Întrebare:** Care principii rămân valabile?

## Principiile lui Kerckhoffs

*Principiul lui Kerckhoffs. Sistemul nu trebuie să fie secret, poate să cadă ușor în mâinile adversarului (i.e. securitatea unui sistem de criptare nu constă decât în menținerea secretă a cheii).*

- ▶ este mai ușor de păstrat o *cheie secretă* decât un *algoritm secret*;
- ▶ este mai ușor de schimbat o *cheie compromisă* decât un *algoritm compromis*;
- ▶ permite standardizarea algoritmilor de criptare.

## Cursul își propune:

- ▶ familiarizarea cu concetele de bază și principiile criptografiei
- ▶ cunoașterea primitivelor criptografice
- ▶ defininirea unor modele de securitate
- ▶ utilizarea corectă a primitivelor și sistemelor criptografice
- ▶ analizarea securității unor sisteme (gandește ca un atacator!)
- ▶ studiul unor sisteme criptografice folosite în practică : AES, RSA, ...

## Cursul NU își propune:

- ▶ conceperea unor sisteme sigure:

NU există un sistem informațional 100% sigur!

*"...the mathematics is impeccable, the computers are vincible, the networks are lousy, and the people are abysmal."*

(Bruce Schneier, Secrets and Lies: Digital Security in a Networked World)

- ▶ spargerea sistemelor informaționale (hacking)
- ▶ studiul malware (MALicious softWARE): viruși, troieni, ...
- ▶ studiul unor atacuri practice (side channel attacks)

## Important de reținut!

- ▶ Terminologia (criptografie, criptanaliză, mesaj clar, mesaj criptat, criptare, decriptare, ...)
- ▶ Personajele (Alice, Bob, Oscar / Eve)
- ▶ Principiul lui Kerckhoffs
- ▶ Definiția sistemelor de criptare (simetrice)



# Criptografie și Securitate

- Prelegerea 2 -  
Sisteme istorice de criptare

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Cifruri de permutare / transpoziție

2. Cifruri de substitutie

# Cifruri de permutare / transpoziție

## Definitie

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

## Cifruri de permutare / transpoziție

- ▶ sistemul Rail Fence >>> curs
- ▶ cifruri generale de transpoziție >>> seminar

## Rail Fence

A diagram illustrating the Rail Fence cipher with 3 rails. The letters M, A, R, T are arranged in three rows. An arrow points from the top row to the right. A red downward arrow is positioned between the first and second rows.

				→
M	A	R	T	
↓	E	J	I	A
S	C	P	T	

Text clar: mesaj criptat

Cheia:  $k = 3$

Text criptat: MARTEJIASCPT

# Cifruri de substituție

## Definitie

Un *cifru de substituție* presupune înlocuirea unui caracter (set de caractere) cu un alt caracter (set de caractere).

## Clasificare:

- ▶ **monoalfabetice:** pentru o cheie dată, un caracter este întotdeauna înlocuit în textul cifrat de **același caracter**
- ▶ **polialfabetice:** pentru o cheie dată, un caracter este înlocuit în textul cifrat de **caractere diferite**

# Cifruri de substituție monoalfabetice

- ▶ cîfrul lui Cezar >>> curs, seminar
- ▶ substituție simplă >>> curs, seminar
- ▶ sistemul Cavalerilor de Malta >>> seminar

# 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

# Cifrul lui Cezar

- ▶  $\mathcal{K} = \{0, 1, \dots, 25\}$
- ▶  $\mathcal{M} = \{a, b, \dots, z\}^*$
- ▶  $\mathcal{C} = \{A, B, \dots, Z\}^*$
- ▶  $\text{Enc} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$

$$\text{Enc}_k(m) = m + k \pmod{26}$$

- ▶  $\text{Dec} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

$$\text{Dec}_k(c) = c - k \pmod{26}$$

## Criptanaliză - Atac prin forță brută

- ▶  $|\mathcal{K}| = 26$
- ▶ 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.

# Substituția 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

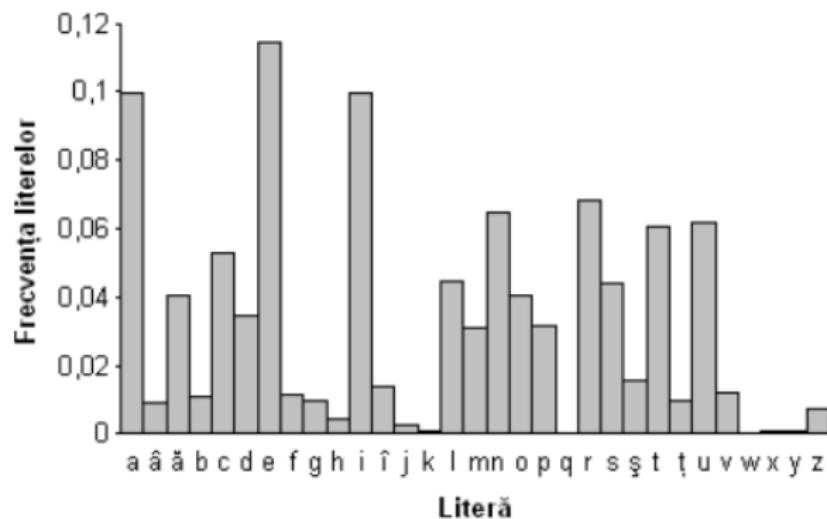
Text clar: mesaj criptat

Text criptat: PRHFG LEDYKFK

## Criptanaliză - Analiza de frecvență

- ▶  $|\mathcal{K}| = 26!$
- ▶ 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**

## Criptanaliză - Analiza de frecvență



[Wikipedia]

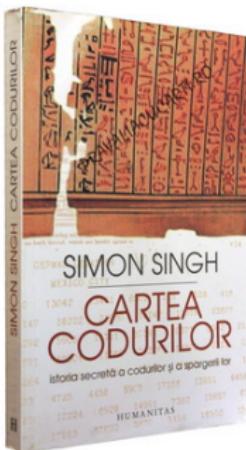
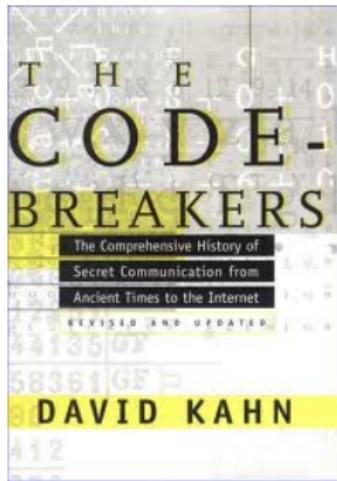
## Cifruri de substituție polialfabetice / poligrafice

- ▶ sistemul Playfair >>> seminar
- ▶ sistemul Vigenére >>> laborator

# Important de reținut!

- ▶ Tipuri de cifruri: transpoziție, substituție
- ▶ Astfel de sisteme sunt total nesigure!

# Referințe bibliografice



<http://simonsingh.net/books/the-code-book/>



# Criptografie și Securitate

- Prelegherea 3 -

## Sisteme mecanice de criptare

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Sisteme mecanice de criptare

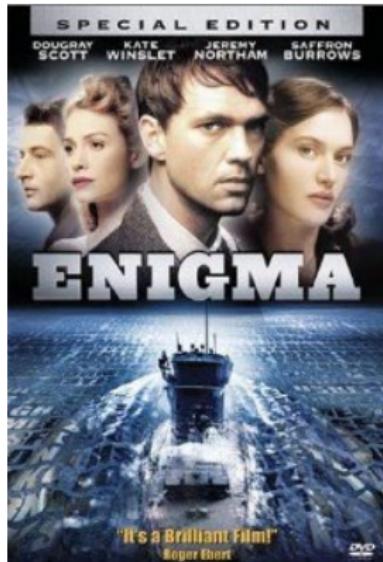
2. Enigma

# Sisteme mecanice

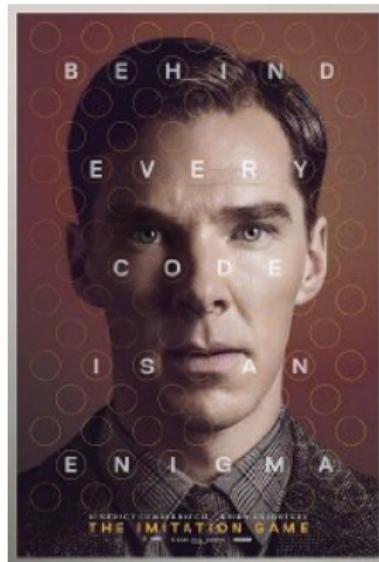


[Bletchley Park 2012]

# Enigma vs. The Imitation Game

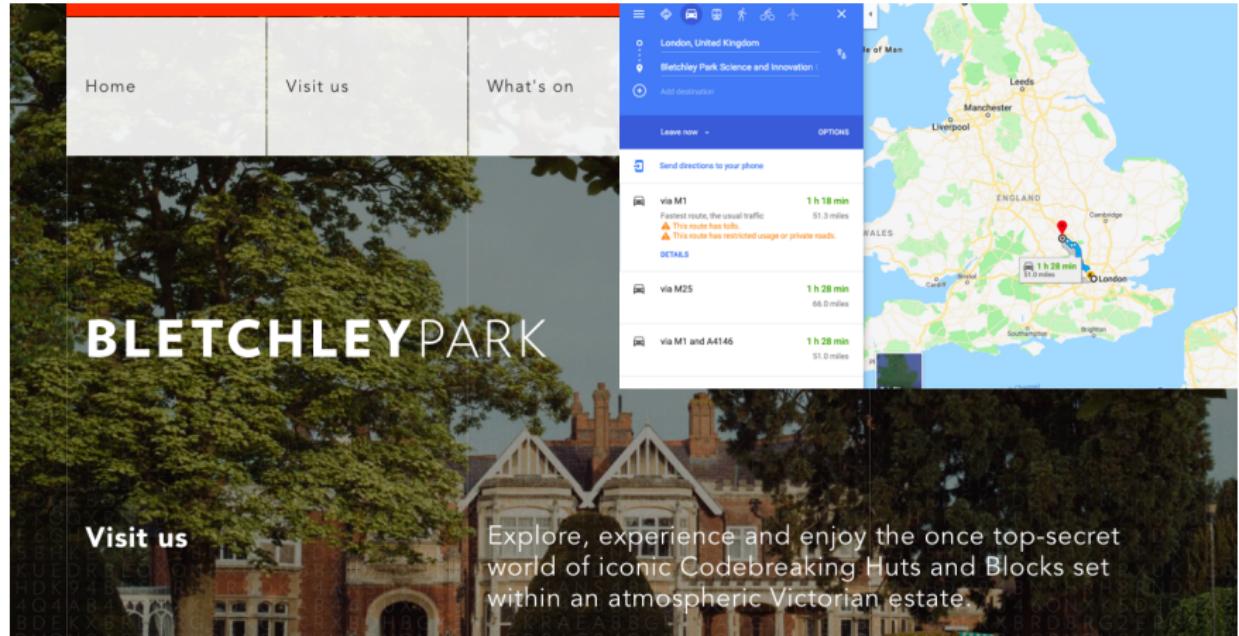


Enigma (2001)



The Imitation Game (2014)

# Bletchley Park



<https://bletchleypark.org.uk/visit-us>

# Enigma



THE TURING BOMBE REBUILD PROJECT



[Bletchley Park 2012]

# Enigma

The Enigma encryption machine | Journey into cryptography | Computer Science | Khan Academy



of each rotor before communication began.

<https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/case-study-ww2-encryption-machines>

# Enigma



[https://www.youtube.com/watch?v=ASfAPOiq\\_eQ](https://www.youtube.com/watch?v=ASfAPOiq_eQ)

## Important de reținut!

- ▶ Cum funcționează mașina Enigma.



# Criptografie și Securitate

- Prelegherea 4 -

Principiile de baza ale criptografiei moderne

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Principiul 1 - Formularea riguroasă a definițiilor de securitate
2. Principiile 2 & 3 - Prezumptions și demonstrații de securitate

# Criptografia modernă

- ▶ reprezintă o trecere de la criptografia (istorică) ca *artă* (până în anii '75) la criptografia ca *știință* (după anii '75)
- ▶ trei principii importante pe care se bazează **criptografia modernă** spre deosebire de **criptografia clasică** (istorică)
  - ▶ **Principiul 1** - orice problemă criptografică necesită o **definiție clară și riguroasă**
  - ▶ **Principiul 2** - securitatea primitivelor criptografice se bazează pe **rezumări clare de securitate** (de regulă, probleme dificile)
  - ▶ **Principiul 3** - orice construcție criptografică trebuie să fie însoțită de o **demonstrație de securitate conform principiilor anterioare**

# Principiul 1 - Formularea riguroasă a definițiilor de securitate

Necesitatea definițiilor exacte:

- ▶ vrem să construim un sistem de criptare sigur; dacă nu știm exact ce vrem să obținem, cum ne putem da seama dacă sau când ne-am atins scopul?
- ▶ vrem să folosim o schema de criptare într-un sistem mai mare; cum știm ce schema să alegem sau care ni se potrivește?
- ▶ fiind date două scheme, cum le putem compara? eficiența nu este un criteriu suficient;
- ▶ NU ne putem baza pe o idee intuitivă a ceea ce înseamnă securitatea;
- ▶ **Atenție!** Formalizarea definițiilor **NU** este o sarcină trivială.

## Un exemplu: criptarea sigură

- ▶ Cum ați defini noțiunea de **schemă de criptare sigură**?

Posibile răspunsuri:(sunt corecte?)

1. Nici un adversar nu poate găsi **cheia secretă** fiind dat un text criptat.
2. Nici un adversar nu poate găsi **textul clar** corespunzător unui text criptat.
3. Nici un adversar nu poate determina nici măcar un **caracter** (o literă) din textul clar corespunzător unui text criptat.
4. Nici un adversar nu poate determina **informații cu sens** din textul clar corespunzător unui text criptat.

*Răspuns corect:*

- ▶ **Nici un adversar nu poate calcula nici o funcție de textul clar pornind de la textul criptat.**

- ▶ Definiția corectă de securitate → prezentare matematică și formală
- ▶ Trebuie să cuprindă:
  - (a) ce inseamnă a sparge o schemă de criptare - vezi slide-ul anterior
  - (b) de ce putere dispune adversarul:  
ce acțiuni are voie să intreprindă + putere computațională

## Definitie

*O schemă de criptare este sigură dacă nici un adversar având puterea specificată nu poate sparge schema în modul specificat.*

## Principiul 2 - prezumptions de securitate și Principiul 3 - demonstrații de securitate

- ▶ majoritatea construcțiilor criptografice moderne nu pot fi demonstrează ca fiind sigure necondiționat
- ▶ fară o demonstrație riguroasă, intuiția că o schemă este corectă poate avea consecințe dezastruoase
- ▶ majoritatea demonstrațiilor folosesc o abordare reductionistă

### Teorema

*Constructia Y este sigură conform definiției dacă presupția X este adevarată.*

- ▶ demonstrația va arăta cum un adversar care sparge schema Y poate încalca presupția X.

## Important de reținut!

- ▶ Criptografia clasică → abordare ad-hoc
- ▶ Criptografia modernă → abordare riguroasă



# Criptografie și Securitate

- Prelegherea 5 -  
Securitate perfectă

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definiție

2. One Time Pad

# Securitate perfectă

- ▶ Primul curs: Sisteme de criptare istorice (substitutie, transpoziție, etc.) care pot fi sparte cu **efort computațional foarte mic**
- ▶ Cursul de azi: **Scheme perfect sigure care rezistă împotriva unui adversar cu putere computațională nelimitată**
- ▶ Însă...limitările sunt inevitabile

# Securitate perfectă (Shannon 1949)

## Definiție

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

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

- ▶  $Pr[M = m]$  - 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 află textul criptat nu are nici un fel de informație în plus decât dacă nu l-ar fi aflat.

# Securitate perfectă (Shannon 1949)

## Definiție echivalentă

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

## Un exemplu de cifru sigur - One Time Pad (OTP)

- ▶ Patentat în 1917 de Vernam (mai poartă denumirea de Cifrul Vernam)
- ▶ Algoritmul:
  1. Fie  $l > 0$  iar  $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0, 1\}^l$
  2. Cheia  $k$  se alege cu distribuție uniformă din spațiul cheilor  $\mathcal{K}$
  3. **Enc:** dată o cheie  $k \in \{0, 1\}^l$  și un mesaj  $m \in \{0, 1\}^l$ , întoarce  $c = k \oplus m$ .
  4. **Dec:** dată o cheie  $k \in \{0, 1\}^l$  și un mesaj criptat  $c \in \{0, 1\}^l$ , întoarce  $m = k \oplus c$ .

## Un exemplu de cifru sigur - 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)
- ▶ Este OTP sigur?

## Un exemplu de cifru sigur - 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	
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!

## Teoremă

*Schema de criptare OTP este perfect sigură.*

- ▶ securitatea perfectă nu este imposibilă dar...
- ▶ cheia trebuie să fie la fel de lungă precum mesajul
- ▶ inconveniente practice (stocare, transmitere)
- ▶ cheia trebuie să fie folosită o singură dată - **one time pad** - de ce?

**Exercițiu** Ce se întâmplă dacă folosim o aceeași cheie de două ori cu sistemul OTP ?

# Limitările securității perfecte

## Teoremă

*Fie  $(Enc, Dec)$  o schemă 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:

## Teoremă

*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.*

## Important de reținut!

- ▶ Schema OTP are securitate perfectă, dar este nepractică pentru majoritatea aplicațiilor;
- ▶ Securitate perfectă  $\Rightarrow$  lungimea cheii  $\geq$  lungimea mesajului.



# Criptografie și Securitate

- Preleghere 6 -

## Criptografie computațională

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Securitate perfectă vs. Criptografie Computațională
2. Criptografie computațională

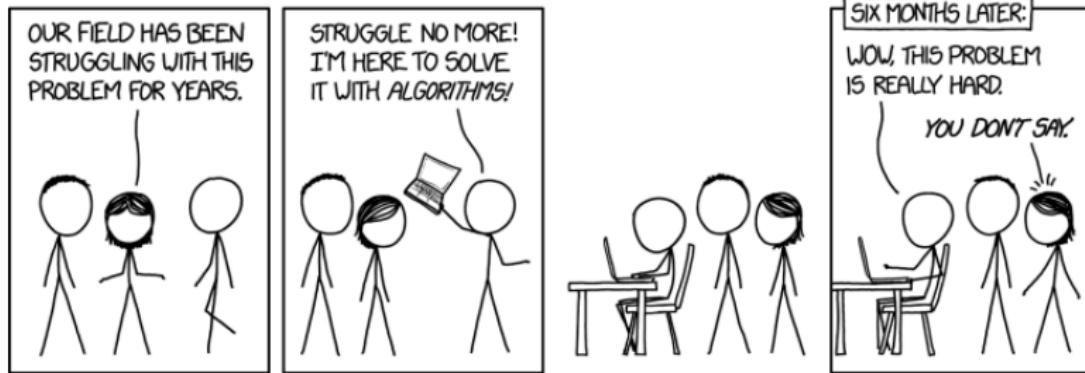
## Securitate perfectă vs. Criptografie computațională

- ▶ Am vazut scheme de criptare care pot fi demonstrate ca fiind sigure în prezența unui adversar cu putere computațională nelimitată;
- ▶ Se mai numesc și **informational-teoretic sigure**;
- ▶ Adversarul nu are suficientă informație pentru a efectua un atac;
- ▶ Majoritatea construcțiilor criptografice moderne → **securitate computațională**;
- ▶ Schemele moderne *pot fi sparte* dacă un atacator are la dispoziție suficient spațiu și putere de calcul.

## Securitate perfectă vs. Criptografie computațională

- ▶ Securitatea computațională mai slabă decât securitatea informațional-teoretică;
- ▶ Prima se bazează pe prezumptii de securitate; a doua este necondiționată;
- ▶ **Întrebare:** de ce renunțăm la securitatea perfectă?
- ▶ **Raspuns:** datorită limitărilor practice!
- ▶ Preferăm un compromis de securitate pentru a obține construcții practice.

# Securitate computațională



<https://xkcd.com/>

# Securitate computațională

- Ideea de bază: principiul 1 al lui Kerckhoffs

*Un cifru trebuie să fie practic, dacă nu matematic, indescifrabil.*

- Sunt de interes mai mare schemele care **practic nu pot fi sparte** deși nu beneficiază de securitate perfectă;
  1. Sunt sigure în fața adversarilor **eficienți** care execută atacul într-un interval de timp realizabil/fezabil;
  2. Adversarii pot efectua un atac cu succes cu o **probabilitate foarte mică**;
  3. Se impune un nouă modalitate de a defini securitatea:

## Definiție

*O schemă este **sigură** dacă orice adversar care dispune de timp polinomial în  $n$  (parametrul de securitate) efectuează un atac cu succes numai cu o probabilitate neglijabilă.*

## Neglijabil și ne-neglijabil

- ▶ **Întrebare:** de ce nu cerem ca probabilitatea de succes a adversarului să fie 0 (ci cerem să fie neglijabilă)?
- ▶ **Raspuns:** pentru că adversarul poate să ghicească (cheia, mesajul clar, etc.)

# Neglijabil și ne-neglijabil

- ▶ În practică:  $\epsilon$  este scalar și
  - ▶  $\epsilon$  ne-neglijabil dacă  $\epsilon \geq 1/2^{30}$
  - ▶  $\epsilon$  neglijabil dacă  $\epsilon \leq 1/2^{128}$
- ▶ 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  $\exists p(n) : \epsilon(n) \geq 1/p(n)$
  - ▶  $\epsilon$  neglijabilă în  $n$  dacă  $\forall p(n), \exists n_d$  a.î.  $\forall n \geq n_d : \epsilon(n) < 1/p(n)$

# Neglijabil și ne-neglijabil

► Întrebare: de ce această definiție și nu alta?

$$\epsilon(n) \text{ negl. în } n \Leftrightarrow \forall p(n), \exists n_d \text{ a.î. } \forall n \geq n_d : \epsilon(n) < 1/p(n)$$

► Răspuns:

- Atacul are loc cu probabilitate  $\epsilon(n)$  ...
- ... deci trebuie repetat de aprox.  $1/\epsilon(n)$  ori ca să reușească
- Dar din definiție  $1/\epsilon(n) > p(n)$  ...
- ... deci necesită un timp super-polynomial în  $n$

Definiția semnifică faptul că sistemul rămâne sigur pentru un adversar **PPT** (Probabilistic Polinomial în Timp)

## Important de reținut!

- ▶ Securitate perfectă vs. securitate computațională
- ▶ Neglijabil vs. ne-neglijabil



# Criptografie și Securitate

- Prelegerea 7 -  
Sisteme fluide

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definiție
2. Securitate
3. Moduri de utilizare
4. Exemple

## Sisteme fluide

- ▶ Am văzut că securitatea perfectă există, dar nu este practic accesibilă - **OTP**;
- ▶ Facem un compromis de securitate, dar obținem o soluție utilizabilă în practică - **sisteme de criptare fluide**;
- ▶ Sistemele fluide sunt similare OTP, cu diferența că secvența **perfect aleatoare** de biți cu care se XOR-ează mesajul clar este înlocuită de o secvență **pseudoaleatoare** de biți.

## Pseudoaleatorismul

- ▶ Un sir **pseudoaleator** "arată" similar unui sir uniform aleator din punct de vedere al oricărui algoritm **polinomial**;
- ▶ Altfel spus: un algoritm **polinomial** nu poate face diferență între o secvență **perfect aleatoare** și una **pseudoaleatoare** (decât cu probabilitate neglijabilă);
- ▶ Sau: o distribuție a secvențelor de lungime  $l$  este **pseudoaleatoare** dacă este **nedistinctibilă** de distribuția uniformă a secvențelor de lungime  $l$ ;
- ▶ Mai exact: nici un algoritm polinomial nu poate spune dacă o secvență de lungime  $l$  este eșantionarea unei distribuții pseudoaleatoare sau este o secvență total aleatoare de lungime  $l$ .

## Pseudoaleatorismul

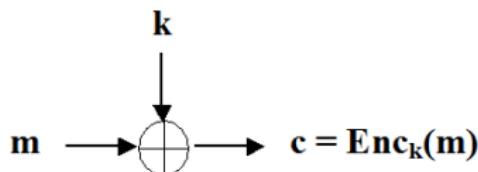
- ▶ În analogie cu ce știm deja:
  - ▶ **pseudoaleatorismul** este o relaxare a **aleatorismului perfect**  
*asa cum*
  - ▶ **securitatea computațională** este o relaxare a **securității perfecte**

# Sisteme fluide

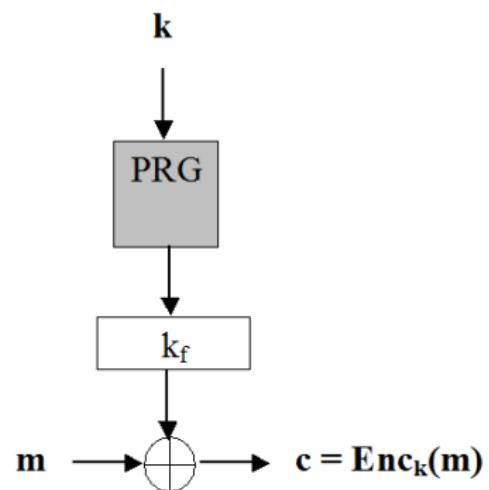
- ▶ Revenind la criptarea fluidă...
- ▶ ... aceasta presupune 2 faze:
  - ▶ **Faza 1:** se generează o secvență pseudoaleatoare de biți, folosind un **generator de numere pseudoaleatoare (PRG)**
  - ▶ **Faza 2:** secvența obținută se XOR-ează cu mesajul clar
- ▶ **Atenție!** De multe ori când ne referim la un sistem de criptare fluid considerăm doar Faza 1

# Sisteme fluide

## OTP (One Time Pad)



## Sisteme fluide



## PRG

- ▶ Ramâne să definim noțiunea de generator de numere aleatoare sau PRG (*PseudoRandom Generator*);
- ▶ Aceasta este un algoritm **determinist** care primește o "sămânță" relativ scurtă  $s$  (*seed*) și generează o secvență *pseudoaleatoare* de biți;
- ▶ Notăm  $|s| = n$ ,  $|PRG(s)| = l(n)$
- ▶ PRG prezintă interes dacă:

$$l(n) \geq n$$

(altfel NU "generează aleatorism" )

## Definitie

Fie  $I(\cdot)$  un polinom și  $G$  un algoritm polinomial determinist a.î.

$\forall s \in \{0, 1\}^n$ ,  $G$  generează o secvență de lungime  $I(n)$ .

$G$  se numește **generator de numere pseudoaleatoare (PRG)** dacă se satisfac **2 proprietăți**:

1. **Expansiune**:  $\forall n, I(n) \geq n$

2. **Pseudoaleatorism**:  $\forall$  algoritm PPT  $\mathcal{D}$ ,  $\exists$  o funcție neglijabilă  $\text{negl}$  a.î.:

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \text{negl}(n)$$

unde  $r \leftarrow^R \{0, 1\}^{I(n)}$ ,  $s \leftarrow^R \{0, 1\}^n$

$I(n)$  se numește **factorul de expansiune** al lui  $G$

## Notății

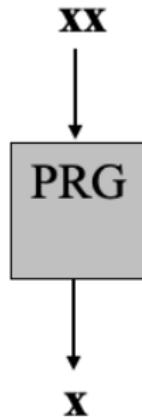
- ▶  $\mathcal{D} = \text{Distinguisher}$
- ▶ PPT = Probabilistic Polynomial Time
- ▶  $x \leftarrow^R X = x$  este ales uniform aleator din  $X$
- ▶  $\text{negl}(n) =$  o funcție neglijabilă în (parametrul de securitate)  $n$

În plus:

- ▶ Vom nota  $\mathcal{A}$  un adversar (Oscar / Eve), care (în general) are putere polinomială de calcul

# DA sau NU

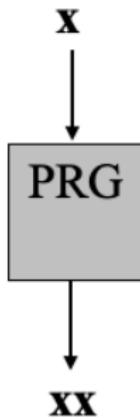
Este acesta un PRG?



**NU** - secvența de ieșire este mai scurtă decât cea de intrare

# DA sau NU

Este acesta un PRG?



**NU** - secvența de ieșire este clar non-pseudoaleatoare

## DA sau NU

Este acesta un PRG?  $|y| \geq |x|$ ,  $y$  pseudoaleator, fixat pentru  $\forall x$



**NU** - secvența de ieșire este aceeași indiferent de secvența de intrare

# Sisteme fluide

## Definitie

Un sistem de criptare  $(Enc, Dec)$  definit peste  $(\mathcal{K}, \mathcal{M}, \mathcal{C})$  se numește **sistem de criptare fluid** 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  $l$ ,  $k \in \{0, 1\}^n$ ,  $m \in \{0, 1\}^{l(n)}$

## Securitate - interceptare unică

### Teorema

Dacă  $G$  este PRG, atunci sistemul fluid definit anterior este un sistem de criptare simetric de lungime fixă computațional sigur pentru un atacator pasiv care poate intercepta un mesaj.

## Demonstrație intuitivă

- ▶ OTP este perfect sigur;
- ▶ Criptarea fluidă se obține din OTP prin *înlocuirea pad cu  $G(k)$* ;
- ▶ Dacă  $G$  este PRG, atunci *pad și  $G(k)$  sunt indistinctibile* pentru orice  $\mathcal{A}$  adversar PPT;
- ▶ În concluzie, OTP și sistemul de criptare fluid sunt indistinctibile pentru  $\mathcal{A}$ .

## Securitate - interceptare multiplă

- ▶ 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.**

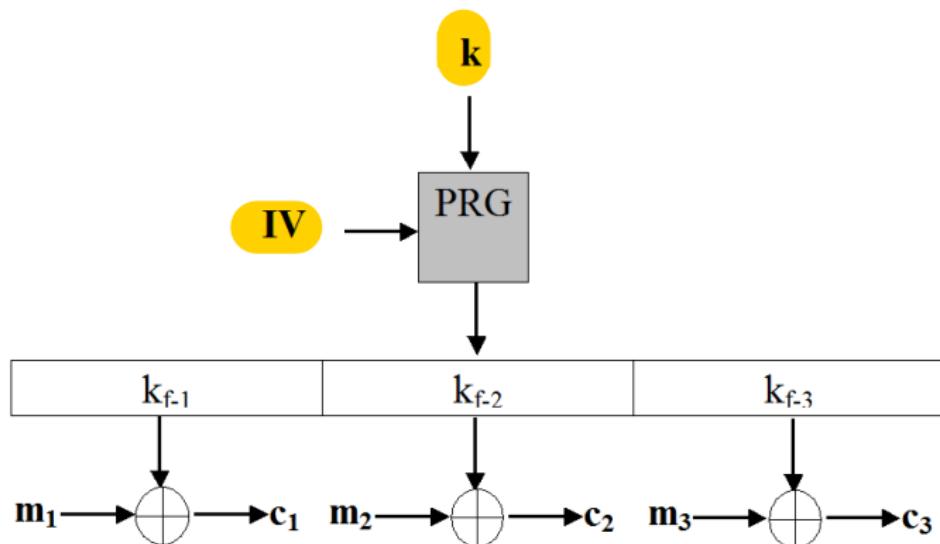
# Moduri de utilizare

- ▶ **modul sincronizat**: partenerii de comunicație folosesc pentru criptarea mesajelor *părți successive* ale secvenței pseudoaleatoare generate;
- ▶ **modul nesincronizat**: partenerii de comunicație folosesc pentru criptarea mesajelor secvențe pseudoaleatoare *diferite*.

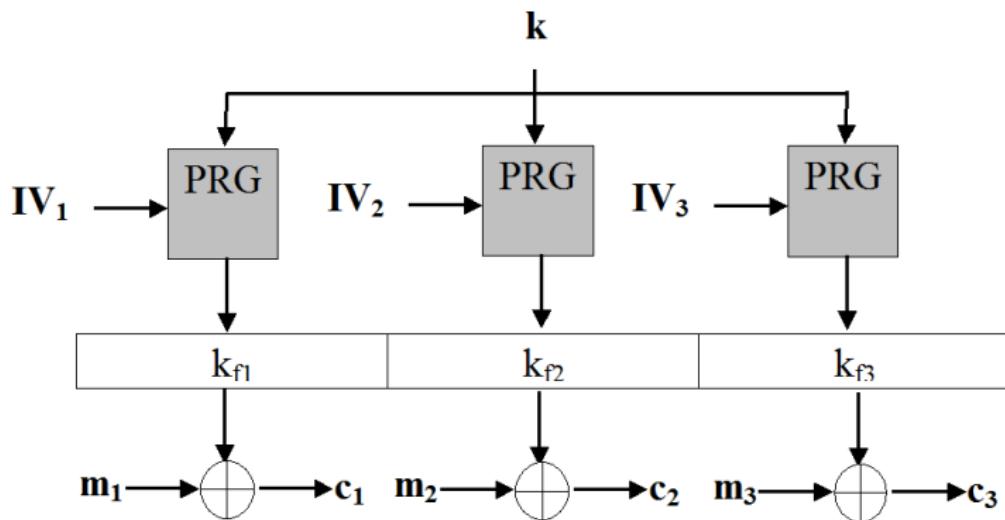
**Atenție!**

PRG va necesita 2 intrări: *cheia k* și un *vector de inițializare IV*.

## Modul sincronizat



## Modul nesincronizat



# Moduri de utilizare

## Modul sincronizat

- ▶ mesajele sunt criptate în mod **succesiv** (participanții trebuie să știe care părți au fost deja folosite)
- ▶ necesită **păstrarea stării**
- ▶ mesajele successive pot fi percepute ca un *singur mesaj clar lung*, obținut prin concatenarea mesajelor succesive
- ▶ 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  $/V$  în modul sincronizat) fac parte din mesajul criptat (sunt necesare pentru decriptare)

## Proprietăți necesare ale PRG în modul nesincronizat

Fie  $G(s, IV)$  un PRG cu 2 intrări:

- ▶  $s = seed$
- ▶  $IV = Initialization Vector$

PRG trebuie să se satisfacă (cel puțin):

1.  $G(s, IV)$  este o secvență pseudoaleatoare chiar dacă  $IV$  este public (i.e. securitatea lui  $G$  constă în securitatea lui  $s$ );
2. dacă  $IV_1$  și  $IV_2$  sunt valori uniform aleatoare, atunci  $G(s, IV_1)$  și  $G(s, IV_2)$  sunt indistinctibile.

## Exemple

- ▶ **RC4** (Ron's Cipher 4):
  - ▶ definit de R.Rivest, în 1987
  - ▶ utilizat în WEP
  - ▶ inițial secret !
  
- ▶ **WEP** (Wired Equivalent Privacy):
  - ▶ standard IEEE 802.11, 1999 (rețele fără fir)
  - ▶ înlocuit în 2003 de WPA (Wi-Fi Protected Access), 2004  
WPA2 - IEEE 802.11i

## Exemple

- ▶ A5/1:
  - ▶ definit în 1987 pentru Europa și SUA
  - ▶ A5/2 definit în 1989 ca o variantă mai slabă pentru alte zone geografice
  - ▶ utilizat în rețelele de telefonie mobilă GSM
  - ▶ inițial secret !
- ▶ SEAL (Software-Optimized Encryption Algorithm)
  - ▶ definit de D.Coppersmith și P.Rogaway, în 1993
  - ▶ prezintă o implementare foarte eficientă pe procesoarele pe 32 de biți
  - ▶ versiunea curentă (SEAL 3.0) este patentată IBM

## Important de reținut!

- ▶ Noțiunile de pseudoaleatorism, PRG
- ▶ OTP vs. Sisteme fluide
- ▶ Transpunerea sistemelor fluide în practică



# Criptografie și Securitate

- Prelegerea 7.1 -  
RC4

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Informații generale

2. Descriere

3. Securitate

## Informații generale

RC4 este:

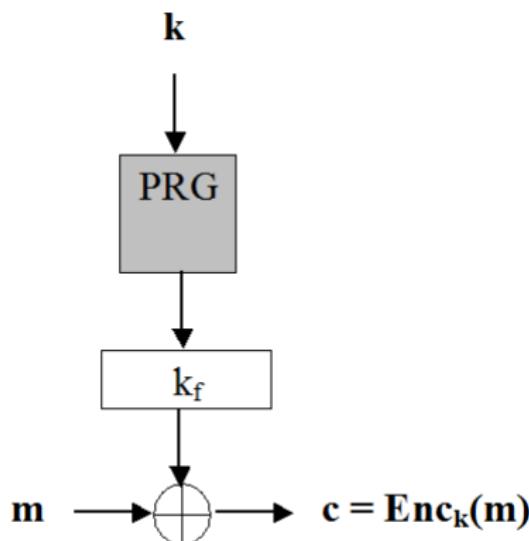
- ▶ introdus de R. Rivest la MIT (1987);
- ▶ înregistrat ca marca a RSA Data Security;
- ▶ păstrat secret până în 1994 când a devenit public;
- ▶ utilizat în WEP, SSL/TLS.

## Descriere

- ▶ RC4 este un sistem de criptare fluid pe octeți:

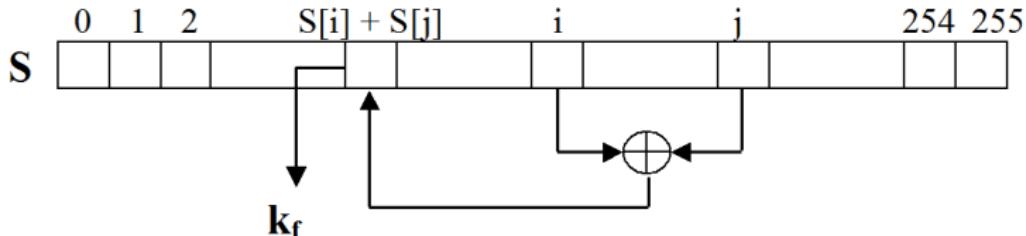
$$m \in \{0, 1\}^8, c \in \{0, 1\}^8$$

- ▶ Ramâne de definit PRG...



## Descriere

- ▶ 2 faze:
  - ▶ inițializare: determină starea internă, fără să producă chei fluide;
  - ▶ generare de chei fluide: modifică starea internă și generează un octet (*cheia fluidă*) care se XOR-ează cu  $m$  pentru a obține  $c$ ;
- ▶ Starea internă:
  - ▶ un tablou  $S$  de 256 octeți:  $S[0], \dots, S[255]$ ;
  - ▶ 2 indici  $i$  și  $j$ ;
- ▶ Toate operațiile se efectuează pe octeți (i.e.  $\pmod{256}$ ).



# Descriere

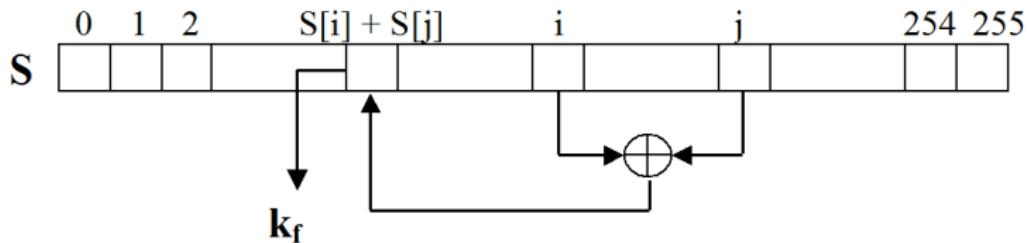
## Faza 1. Inițializare

- ▶  $n = \text{numărul octetilor din cheie}, 1 \leq n \leq 256$
- ▶  $j \leftarrow 0$   
**for**  $i = 0$  **to** 255 **do**  
     $S[i] \leftarrow i$   
**end for**  
**for**  $i = 0$  **to** 255 **do**  
     $j \leftarrow j + S[i] + k[i \pmod n]$   
    swap ( $S[i], S[j]$ )  
**end for**  
     $i \leftarrow 0$   
     $j \leftarrow 0$

# Descriere

## Faza 2. Generarea cheii fluide

- ▶ cheia se obține octet cu octet
- ▶  $i \leftarrow i + 1$
- ▶  $j \leftarrow j + S[i]$
- ▶ swap ( $S[i], S[j]$ )
- ▶ **return**  $S[S[i] + S[j]]$



## Descriere

Detalii de implementare:

- ▶  $5 \leq n \leq 16 \Rightarrow 40 \leq |k| \leq 256$ ;
- ▶ memorie: 256 octeți (pentru  $S$ ) și câteva variabile *byte*;
- ▶ operații simple, rapid de executat.

## Securitate

- ▶ primii octeți generați drept cheie fluidă sunt total ne-aleatori și oferă informații despre cheie (Fluhrer, Mantin and Shamir 2001)
- ▶ RC4 pe 104 biți (utilizat pentru WEP pe 128 biți) a fost spart în aprox. 1 min (algoritm al lui Tews, Weinmann, Pychkine 2001, bazat pe idea lui Klein 2005)
- ▶ un atac recent arată că pot fi determinați primii aprox. 200 octeți din textul clar criptat cu RC4 în TLS cunoscând  $[2^{28} - 2^{32}]$  criptări independente (Royal Holloway, 2013)

## Important de reținut!

- ▶ RC4 - sistem de criptare fluid



# Criptografie și Securitate

- Prelegerea 7.2 -  
WEP

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Informații generale

2. Descriere

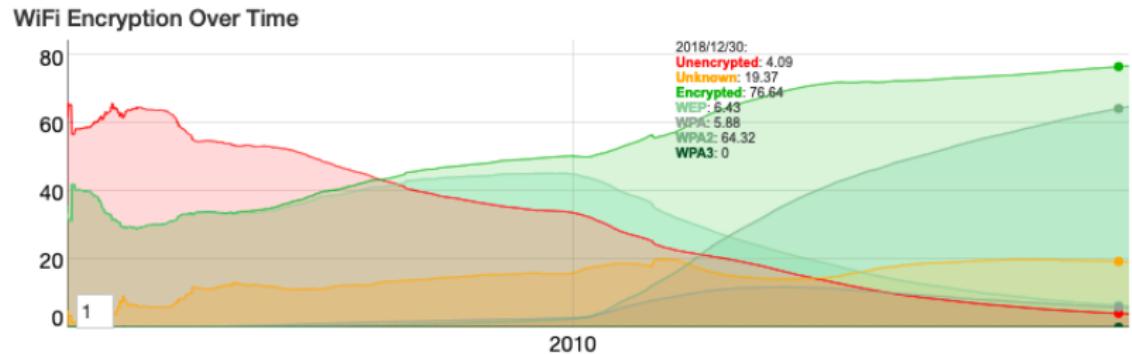
3. Securitate

## Informații generale

WEP este:

- ▶ definit ca standard IEEE 802.11 în 1999;
- ▶ folosit pentru criptare în cadrul rețelor fără fir;
- ▶ înlocuit de WPA, apoi WPA2.

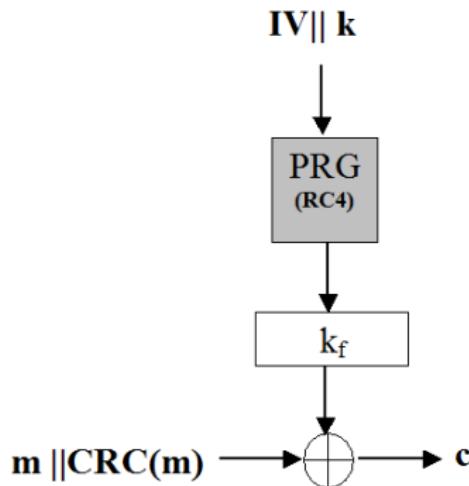
# Informații generale



[<https://wigle.net/stats>]

## Descriere

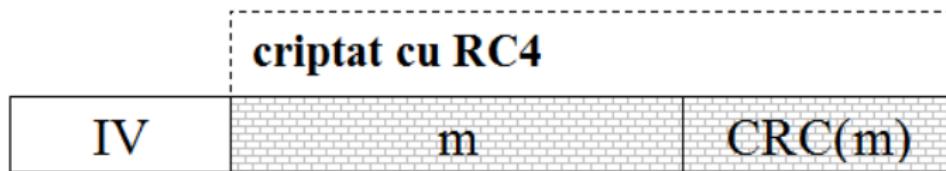
- ▶ WEP utilizează RC4;
- ▶ Introduce în plus (Cyclic Redundancy Check) cu rolul de a detecta eventuale erori de transmisie;
- ▶  $\text{IV}$  este transmis către destinatar, împreună cu  $c$  ( $\text{IV}||c$ ), fiind necesar pentru decriptare .



## Descriere

Detalii de implementare:

- ▶  $|IV| = 24$  (biți)
- ▶  $IV$  se utilizează în *counter mode* ( $0, 1, 2, \dots$ )



# Securitate - Problema 1

## Problema 1: Utilizarea multiplă a lui IV

- ▶ **Întrebare:** Ce efect imediat are utilizarea multiplă a lui IV pentru o cheie fixată  $k$ ?
- ▶ **Răspuns:** Se obține întotdeauna aceeași cheie fluidă  $k_r$

$$IV_1 = IV_2 = IV \Rightarrow k_{f1} = k_{f2} = PRG(IV||k)$$

- ▶ **Întrebare:** Sistemul rămâne sigur în aceste condiții?
- ▶ **Răspuns:** NU!

$$c_1 = m_1 \oplus k_{f1}, c_2 = m_2 \oplus k_{f2} \Rightarrow c_1 \oplus c_2 = m_1 \oplus m_2$$

Exemplu:

- ▶ dacă  $\mathcal{A}$  cunoaște  $m_1$ , atunci poate determina  $m_2$
- ▶  $\mathcal{A}$  poate determina  $m_1$  și  $m_2$  folosind analiza statistică

## Securitate - Problema 1

- ▶ **Întrebare:** Câte valori posibile poate lua IV?
- ▶ **Răspuns:**  $2^{24}$
- ▶ Cum un AP (Access Point) trimite aproximativ 1000 pachete/s, valoarea lui IV se repetă la cel mult la câteva ore!
- ▶ Mai mult, există echipamente care resetează IV la fiecare repornire!

# Securitate - Problema 2

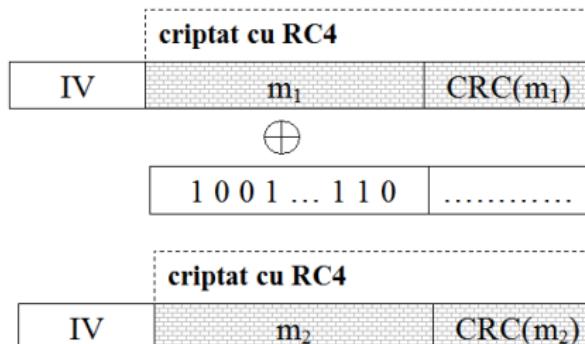
## Problema 2: Liniaritatea

- **Întrebare:** Făcând abstracție de CRC,  $\mathcal{A}$  poate modifica mesajul criptat transmis după bunul său plac?
- **Răspuns:** DA!

$$(m_1 \oplus m_2) \oplus k_f = (m_1 \oplus k_f) \oplus m_2$$

Exemplu:

- $\mathcal{A}$  interceptează  $c_1$  și îl transformă în  $c_2$



## Securitate - Problema 2

- ▶ CRC poate detecta erorile de transmisiune (**neintenționate**) dar nu și modificările premeditate (**intenționate**)
- ▶ În aceste condiții,  $\mathcal{A}$  poate modifica mesajele transmise:
  - ▶ prin XOR-are cu secvențe (convenabile) de biți
  - ▶ prin amestecarea biților din mesajul interceptat

## Important de reținut!

- ▶ O proastă implementare / utilizare poate diminua considerabil securitatea unui sistem!



# Criptografie și Securitate

- Prelegherea 8 -  
Securitate semantică

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Securitate - interceptare simplă
2. Securitate - interceptare multiplă

## Securitate semantică - interceptare simplă

- ▶ Reamintim:  $(Enc, Dec)$  peste spațiul  $(\mathcal{K}, \mathcal{M}, \mathcal{C})$  este perfect sigură dacă

$\forall m_0, m_1 \in \mathcal{M}$  cu  $|m_0| = |m_1|$  are loc egalitatea

$$\{Enc_k(m_0)\} = \{Enc_k(m_1)\}$$

(distribuțiile sunt identice)

pentru  $k \leftarrow^R \mathcal{K}$ ;

- ▶ Incercăm o relaxare:

$\forall m_0, m_1 \in \mathcal{M}$  cu  $|m_0| = |m_1|$  are loc

$$\{Enc_k(m_0)\} \approx \{Enc_k(m_1)\}$$

(distribuțiile sunt indistinctibile computațional)

pentru  $k \leftarrow^R \mathcal{K}$ ;

Însă adversarul trebuie să aleagă  $m_0$  și  $m_1$  explicit.

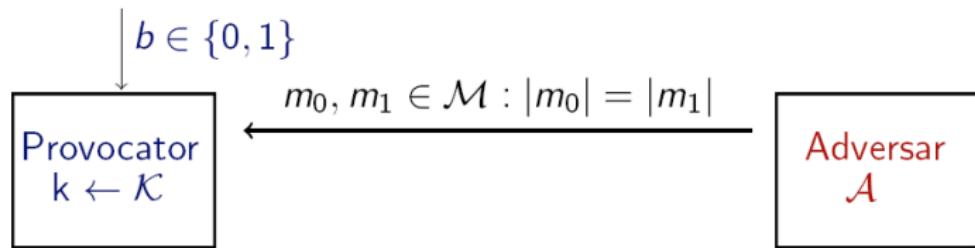
## Securitate semantică - interceptare simplă

- ▶ Vom defini securitatea semantică pe baza unui experiment de indistinctibilitate  $\text{Priv}_{\mathcal{A}, \pi}^{\text{eav}}(n)$  unde  $\pi = (\text{Enc}, \text{Dec})$  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 **provocator (challenger)**.
- ▶ Trebuie să definim capabilitățile adversarului: în contextul sistemelor de criptare fluide, el poate vedea **un singur text criptat cu o anume cheie**, fiind un adversar *pasiv* care poate rula atacuri în timp polinomial.

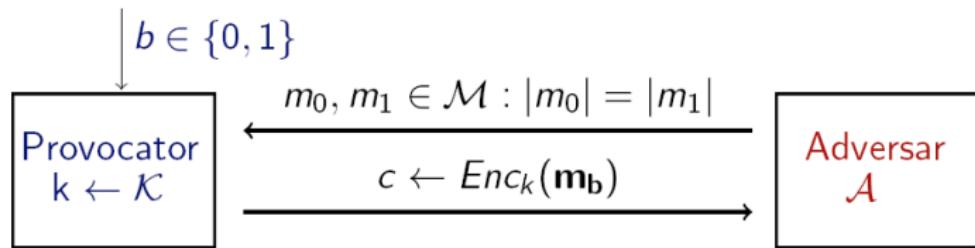
# Experimentul $Priv_{\mathcal{A}, \pi}^{eav}(n)$



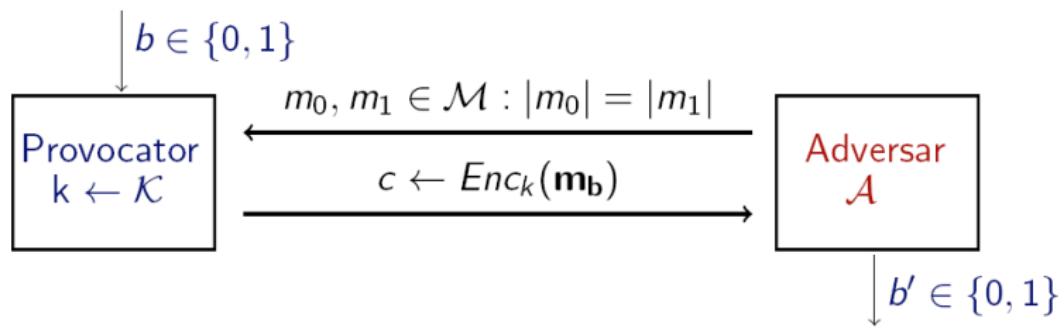
# Experimentul $\text{Priv}_{\mathcal{A}, \pi}^{\text{eav}}(n)$



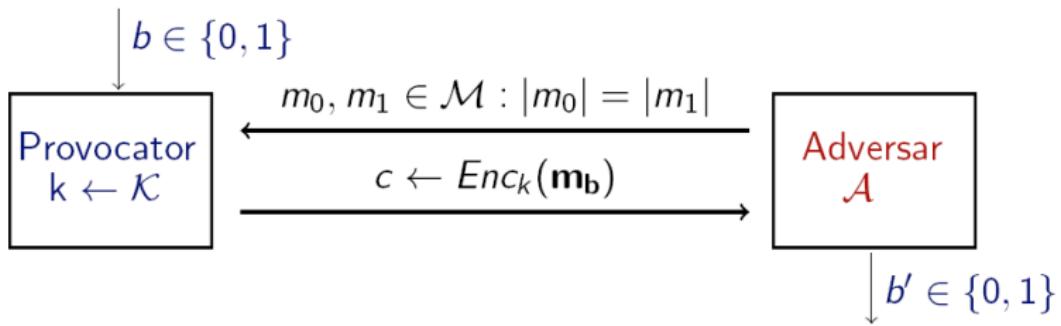
# Experimentul $Priv_{\mathcal{A}, \pi}^{eav}(n)$



# Experimentalul $Priv_{\mathcal{A}, \pi}^{eav}(n)$



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



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

## Securitate semantică - interceptare simplă

### 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 aşa încât

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

- ▶ Un adversar pasiv 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).

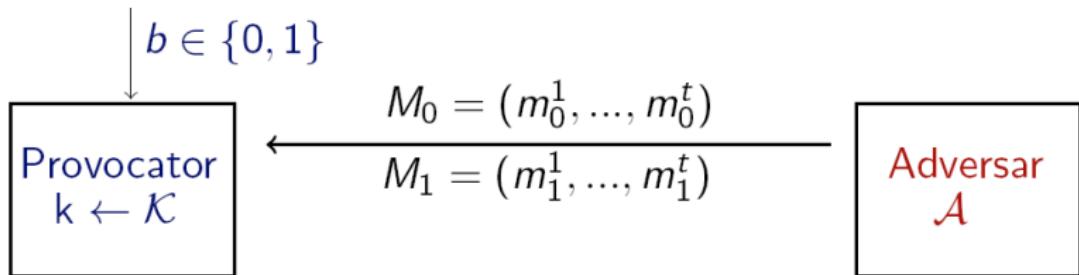
## Securitate pentru interceptare multiplă

- ▶ În definiția precedentă am considerat cazul unui adversar care primește **un singur** text criptat;
- ▶ În realitate, în cadrul unei comunicații se trimit **mai multe mesaje** pe care adversarul le poate intercepta;
- ▶ Definim ce înseamnă o schemă sigură chiar și în aceste condiții.

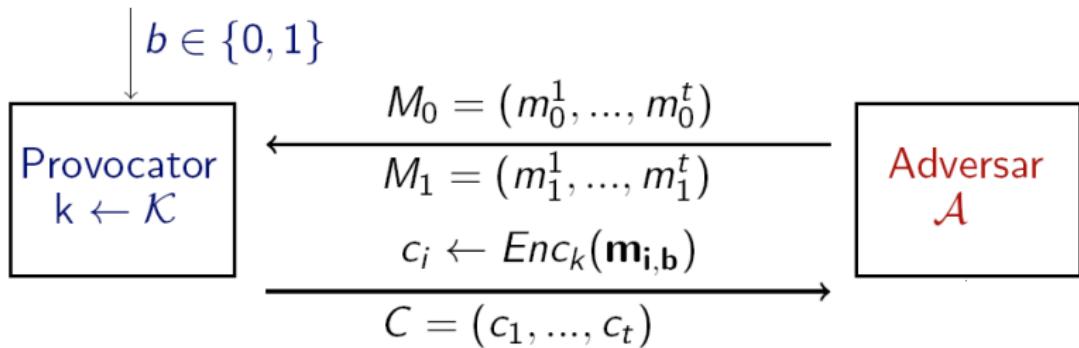
# Experimentul $Priv_{\mathcal{A}, \pi}^{mult}(n)$



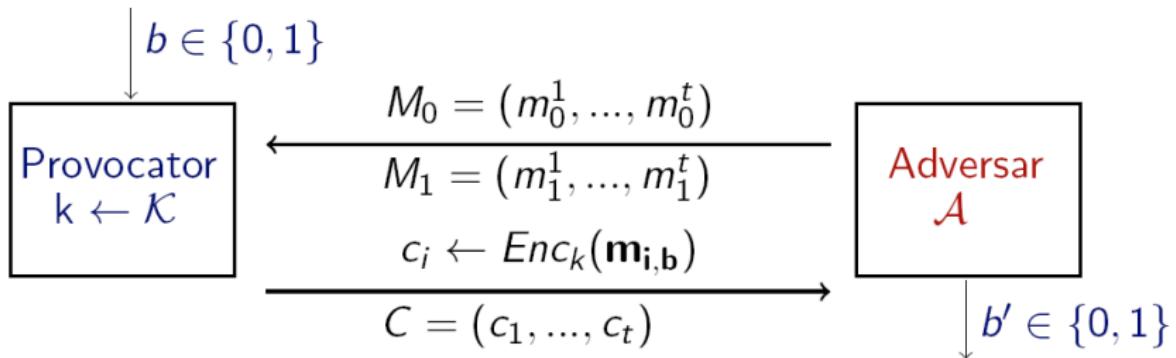
## Experimentul $Priv_{\mathcal{A}, \pi}^{mult}(n)$



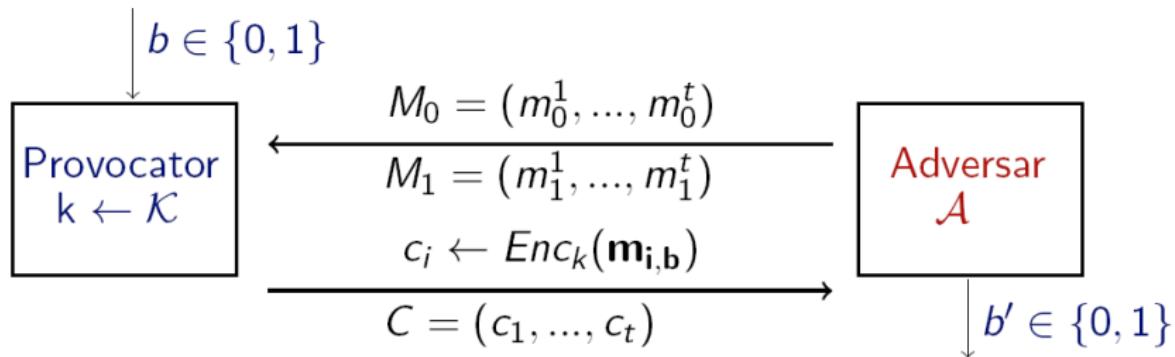
## Experimentalul $Priv_{\mathcal{A}, \pi}^{mult}(n)$



## Experimentalul $Priv_{\mathcal{A}, \pi}^{mult}(n)$



## Experimentul $Priv_{\mathcal{A}, \pi}^{mult}(n)$



- ▶ Output-ul experimentului este 1 dacă  $b' = b$  și 0 altfel;
- ▶ Definiția de securitate este aceeași, doar că se referă la experimentul de mai sus.
- ▶ Securitatea pentru interceptare **simplă** nu implică securitate pentru interceptare **multiplă**!

# Securitate pentru interceptare multiplă

## 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 aşa încât

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

## Teoremă

O schemă de criptare  $(Enc, Dec)$  unde funcția  $Enc$  este deterministă nu are proprietatea de securitate la interceptare multiplă conform cu definiția de mai sus.

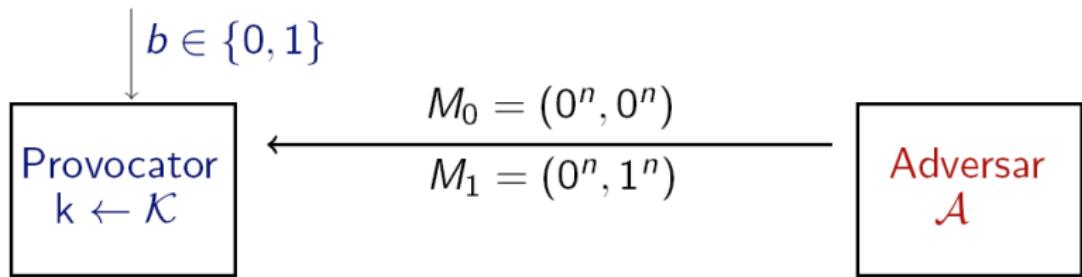
## Demonstrație

- ▶ Intuitiv, am vazut că schema OTP este sigură doar când o cheie este folosită o singură dată;
- ▶ La sistemele fluide se întâmplă același lucru;
- ▶ Vom considera un adversar  $\mathcal{A}$  care atacă schema (în sensul experimentului  $Priv_{\mathcal{A}, \pi}^{mult}(n)$ )

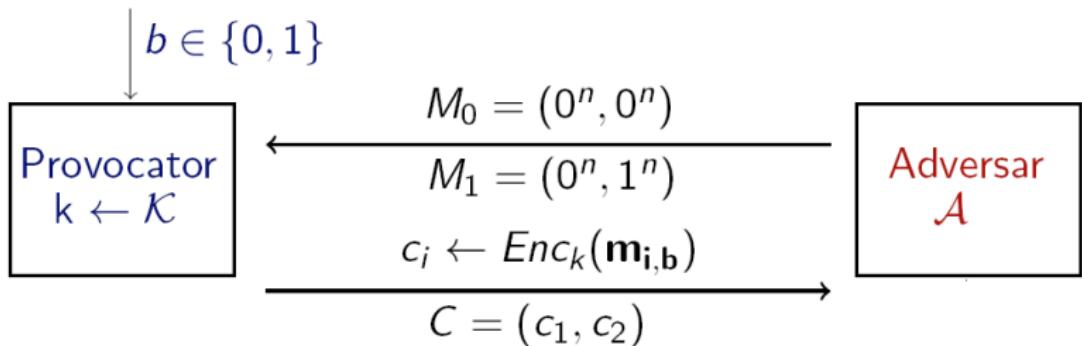
# Demonstrație



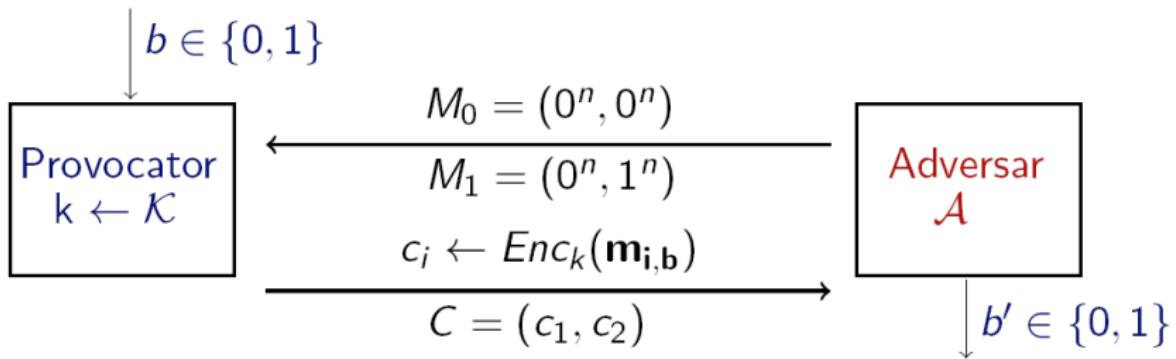
## Demonstrație



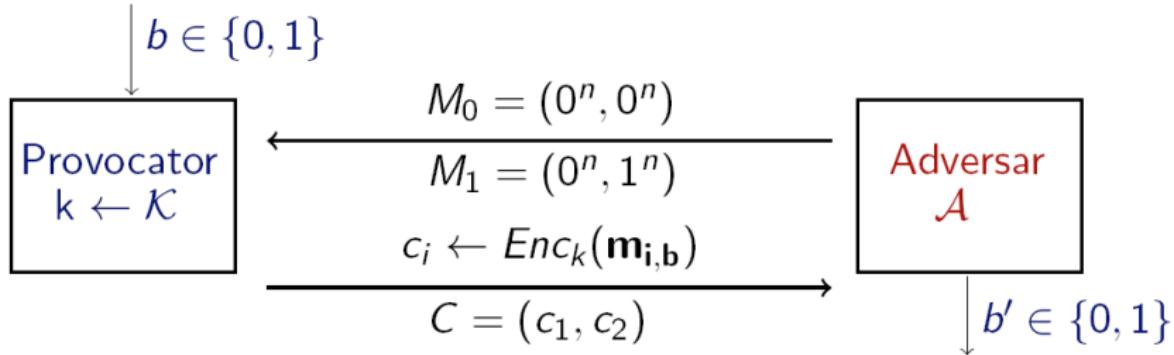
## Demonstrație



# Demonstrație



## Demonstrație



- Dacă  $c_1 = c_2$ , atunci  $\mathcal{A}$  întoarce 0, altfel  $\mathcal{A}$  întoarce 1.
- Analizăm probabilitatea ca  $\mathcal{A}$  să ghicească  $b$ : dacă  $b = 0$ , același mesaj este criptat mereu ( $m_0^1 = m_0^2$ ) iar  $c_1 = c_2$  și deci  $\mathcal{A}$  întoarce mereu 0;
- Dacă  $b = 1$ , atunci ( $m_1^1 \neq m_1^2$ ) iar  $c_1 \neq c_2$  și deci  $\mathcal{A}$  întoarce mereu 1.

## Important de reținut!

- ▶ Securitate - interceptare simplă  $\not\Rightarrow$  securitate - interceptare multiplă
- ▶ Schemele deterministe nu sunt sigure la interceptare multiplă



# Criptografie și Securitate

- Prelegherea 9 -

## Sisteme de criptare bloc

Adela Georgescu, Ruxandra F. Olimid

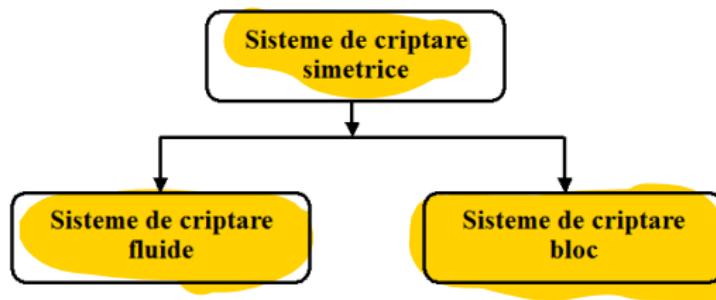
Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

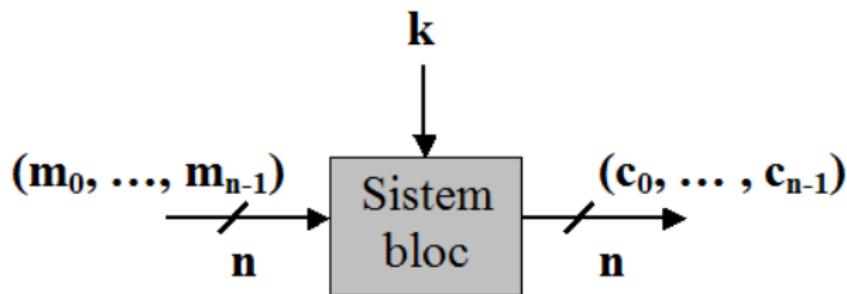
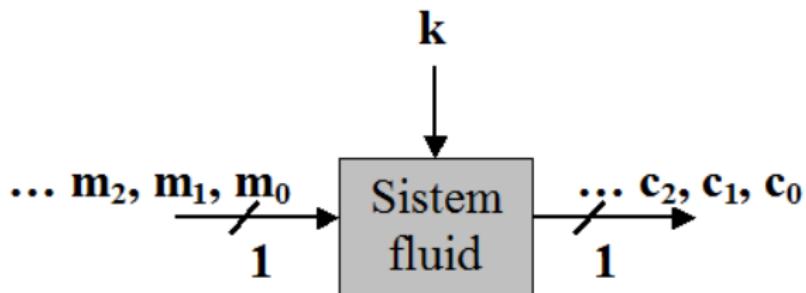
1. Definiție
2. Construcție
3. Moduri de utilizare
4. Exemple

# Criptografia simetrică

- ▶ 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**;



## Sisteme bloc vs. sisteme fluide



# Sisteme bloc vs. sisteme fluide

... 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 bloc vs. sisteme fluide

... d.p.d.v. *tradițional*, în practică:

## Sisteme fluide

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

## Sisteme bloc

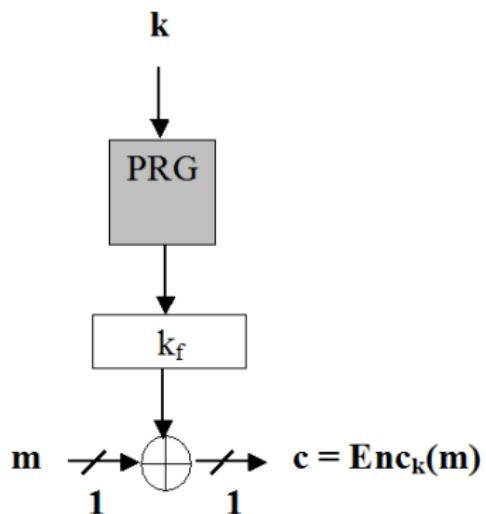
- ▶ necesități computaționale mai avansate
- ▶ utilizare: internet
- ▶ par să fie mai sigure, prezintă încredere mai mare

# Sisteme bloc

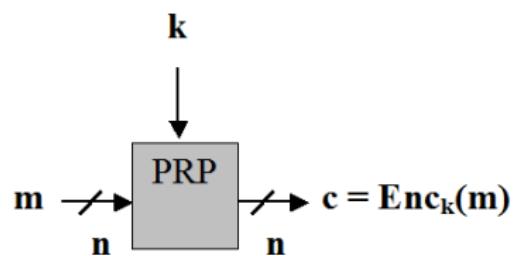
- ▶ Introducem noțiunea de **permutare pseudoaleatoare** sau **PRP** (*PseudoRandom Permutation*)
- ▶ În analogie cu ce știm deja:
  - ▶ **PRP** sunt necesare pentru construcția **sistemelor bloc**  
*asa cum*
  - ▶ **PRG** sunt necesare pentru construcția **sistemelor fluide**

## Sisteme bloc

### Sisteme fluide



### Sisteme bloc



- ▶ Ramâne să definim noțiunea de **permutare pseudoaleatoare** sau **PRP** (*PseudoRandom Permutation*);
- ▶ Aceasta este o funcție **deterministă și bijectivă** care pentru o cheie fixată produce la ieșire o **permutare** a intrării ...
- ▶ ... **indistinctibilă** față de o permutare aleatoare;
- ▶ În plus, atât funcția cât și inversa sa sunt **eficient calculabile**.

## Definitie

O *permutare pseudoaleatoare* definită peste  $(\mathcal{K}, \mathcal{X})$  este o funcție bijectivă

$$F : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{X}$$

care satisface următoarele proprietăți:

1. *Eficiență*:  $\forall k \in \mathcal{K}, x \in \mathcal{X}, \exists$  algoritmi determiniști PPT care calculează  $F_k(x)$  și  $F_k^{-1}(x)$
2. *Pseudoaleatorism*:  $\forall$  algoritm PPT  $\mathcal{D}$ ,  $\exists$  o funcție neglijabilă negl a.î.:

$$|\Pr[D(r) = 1] - \Pr[D(F_k(\cdot)) = 1]| \leq \text{negl}(n)$$

unde  $r \leftarrow^R \text{Perm}(\mathcal{X}), k \leftarrow^R \mathcal{K}$

## Notății

- ▶  $F_k(x) = F(k, x)$   
o cheie este în general (aleator) aleasă și apoi fixată
- ▶  $\text{Perm}(X) =$  mulțimea tuturor funcțiilor bijective de la  $\mathcal{X}$  la  $\mathcal{X}$
- ▶  $\mathcal{X} = \{0, 1\}^n$
- ▶  $\mathcal{D} = \text{Distinguisher}$  care are acces la *oracolul* de evaluare a funcției

# PRF

- ▶ Introducem noțiunea de **funcție pseudoaleatoare** sau **PRF** (*PseudoRandom Function*)...
- ▶ ... ca o generalizare noțiunii de **permutare pseudoaleatoare**;
- ▶ Aceasta este o funcție **cu cheie** care este **indistinctibilă** față de o funcție aleatoare (cu același domeniu și mulțime de valori).

## Definitie

O *funcție pseudoaleatoare* definită peste  $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$  este o funcție bijectivă

$$F : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{Y}$$

care satisface următoarele proprietăți:

1. *Eficiență*:  $\forall k \in \mathcal{K}, x \in \mathcal{X}, \exists$  algoritm PPT care calculează  $F_k(x)$
2. *Pseudoaleatorism*:  $\forall$  algoritm PPT  $\mathcal{D}$ ,  $\exists$  o funcție neglijabilă negl a.î.:

$$|\Pr[D(r) = 1] - \Pr[D(F_k(\cdot)) = 1]| \leq \text{negl}(n)$$

unde  $r \leftarrow^R \text{Func}(\mathcal{X}, \mathcal{Y}), k \leftarrow^R \mathcal{K}$

## Notății

- ▶  $F_k(x) = F(k, x)$   
o cheie este în general (aleator) aleasă și apoi fixată
- ▶  $\text{Func}(X, Y) = \text{mulțimea funcțiilor de la } X \text{ la } Y$
- ▶  $X = \{0, 1\}^n, Y = \{0, 1\}^n$   
considerăm în general că *PRF păstrează lungimea*
- ▶  $\mathcal{D} = \text{Distinguisher}$  care are acces la *oracolul* de evaluare a funcției

## $PRP \subseteq PRF$

- ▶ **Întrebare:** De ce  $PRF$  poate fi privită ca o generalizare a  $PRP$ ?
- ▶ **Răspuns:**  $PRP$  este  $PRF$  care satisfac:
  1.  $\mathcal{X} = \mathcal{Y}$
  2. este inversabilă
  3. calculul funcției inverse este eficient

# Construcții

- ▶ **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

**Întrebare:** Care dintre aceste construcții este trivială?

# Construcții

Răspuns: **PRP  $\Rightarrow$  PRF**

*PRP* este o particularizare a *PRF* :  $\mathcal{X} \times \mathcal{K} \rightarrow \mathcal{Y}$  care satisface:

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

# Construcții

- ▶ **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

## *PRF* $\Rightarrow$ *PRG*

- ▶ Considerăm  $F : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  PRF;
- ▶ Construim  $G : \mathcal{K} \rightarrow \{0, 1\}^{nI}$  PRG sigur:

$$G(k) = F_k(0) || F_k(1) || \dots || F_k(I - 1)$$

- ▶ **Întrebare:** De ce este  $G$  sigur?
- ▶ **Răspuns:**  $F_k(\cdot)$  este *indistinctibilă* față de o funcție aleatoare  
 $\Rightarrow G(k)$  este *indistinctibilă* față de o secvență aleatoare de lungime  $In$ .
- ▶ **Avantaj:** Construcția este *paralelizabilă*

# Construcții

- ▶ **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

## *PRG* $\Rightarrow$ *PRF*

- ▶ Considerăm  $G : \mathcal{K} \rightarrow \mathcal{K}^2$  PRG cu  $|\mathcal{K}| = n$ :

$$G(k) = (G_0(k), G_1(k))$$

$G_0(k)$  și  $G_1(k)$  sunt prima și a doua jumătate a ieșirii din PRG

- ▶ Construim  $F : \mathcal{K} \times \{0, 1\} \rightarrow \mathcal{K}$  PRF:

$$F_k(0) = G_0(k), F_k(1) = G_1(k)$$

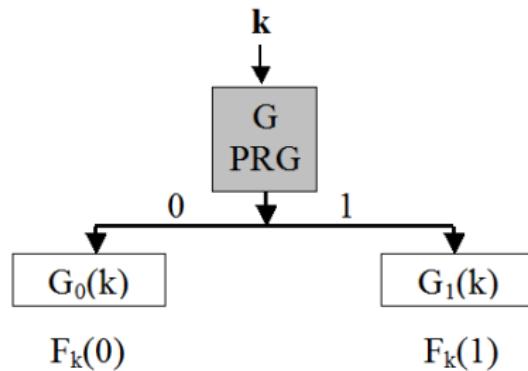
$F_k(0)$  întoarce prima jumătate a secvenței pseudoaleatoare  $G(k)$

$F_k(1)$  întoarce a doua jumătate a secvenței pseudoaleatoare  $G(k)$

- ▶ **Întrebare:** De ce este  $F$  sigură?
- ▶ **Răspuns:**  $G(k)$  este *indistinctibilă* față de o secvență aleatoare de lungime  $2n$ 
  - $\Rightarrow G_0(k)$  și  $G_1(k)$  sunt *indistinctibile* față de o secvență aleatoare de lungime  $n$
  - $\Rightarrow$  pentru  $k \leftarrow^R \{0, 1\}$ ,  $F_k(\cdot)$  este *indistinctibilă* față de o funcție aleatoare.

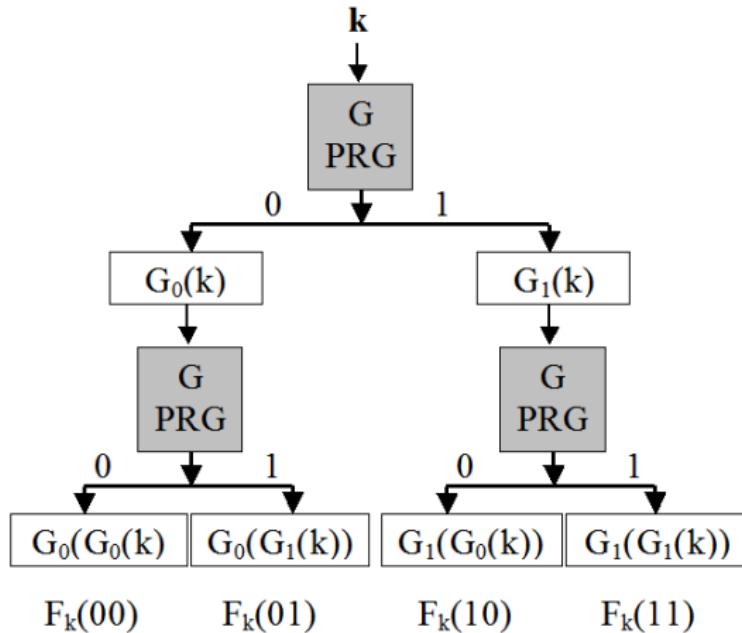
## *PRG $\Rightarrow$ PRF*

- ▶ Construcția pentru un singur bit de intrare...



## *PRG $\Rightarrow$ PRF*

- ...se poate generaliza pentru un număr oarecare de biți



## *PRG* $\Rightarrow$ *PRF*

- ▶ Construcția poate fi reprezentată ca un arbore binar cu cheia  $k$  rădăcină;
- ▶ Pentru un nod de valoare  $k'$ , copilul stâng ia valoarea  $G_0(k')$  și copilul drept ia valoare  $G_1(k')$ ;
- ▶ Valoarea funcției  $F_k(x) = F_k(x_0, \dots, x_{n-1})$  este obținută prin parcurgerea arborelui în funcție de  $x$  ;
- ▶ Adâncimea arborelui este *liniară* în  $n$  ( $n$ );
- ▶ Dimensiunea arborelui este *exponențială* în  $n$  ( $2^n$ );
- ▶ NU se utilizează în practică din cauza performanței scăzute.

# Construcții

- ▶ **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

*PRF*  $\Rightarrow$  *PRP*

### Teorema (Luby-Rackoff '85)

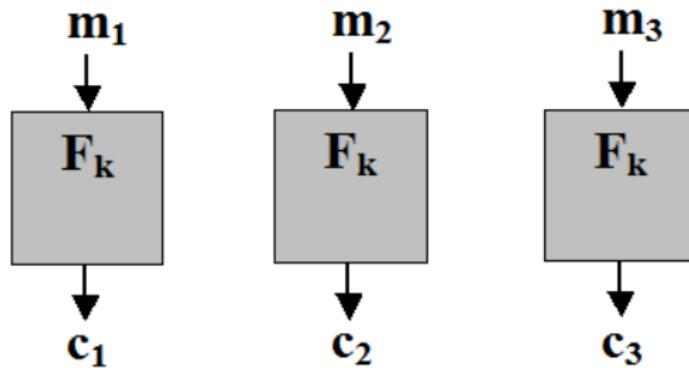
Dacă  $F : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  este PRF, se poate construi  $F' : \mathcal{K} \times \{0, 1\}^2 \rightarrow \{0, 1\}^2$  PRP.

- ▶ Construcția folosește runde **Feistel**, pe care le vom prezenta într-un curs ulterior.

## Moduri de utilizare

- ▶ Să continuam cu ceva mai practic...
- ▶ **Întrebare:** Ce se întâmplă dacă lungimea mesajului clar este **mai mică** decât dimensiunea unui bloc?
- ▶ **Răspuns:** Se completează cu biți: **1 0 ... 0**;
- ▶ **Întrebare:** Ce se întâmplă dacă lungimea mesajului clar este **mai mare** decât lungimea unui bloc?
- ▶ **Răspuns:** Se utilizează un **mod de operare** (ECB, CBC, OFB, CTR);
- ▶ Notăm cu  $F_k$  un sistem de criptare bloc (i.e. PRP) cu cheia  $k$  fixată.

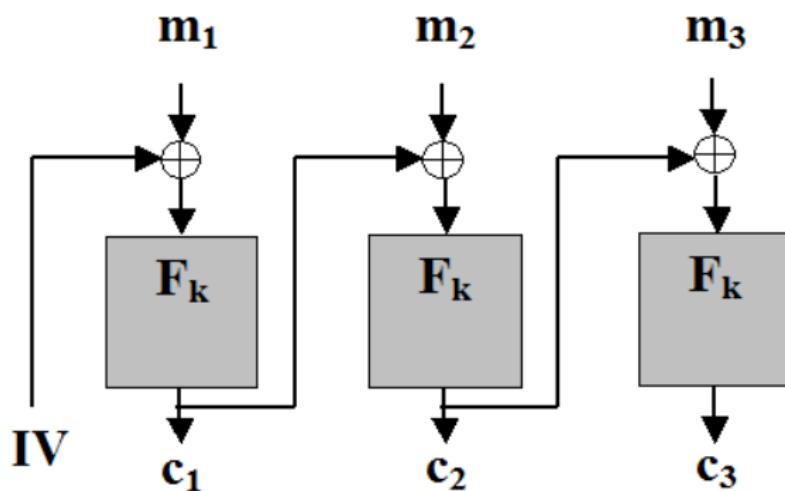
# Modul ECB (Electronic Code Book)



## Modul ECB (Electronic Code Book)

- ▶ Pare modul cel mai **natural** de a cripta mai multe blocuri;
- ▶ Pentru decriptare,  $F_k$  trebuie să fie **inversabiliă**;
- ▶ Este **parallelizabil**;
- ▶ Este **determinist**, deci este **nesigur**;
- ▶ **Întrebare:** Ce informații poate să ofere modul de criptare ECB unui adversar pasiv?
- ▶ **Răspuns:** Un adversar pasiv detectează repetarea unui bloc de text clar pentru că se repetă blocul criptat corespunzător;
- ▶ Modul **ECB NU** trebuie utilizat în practică!

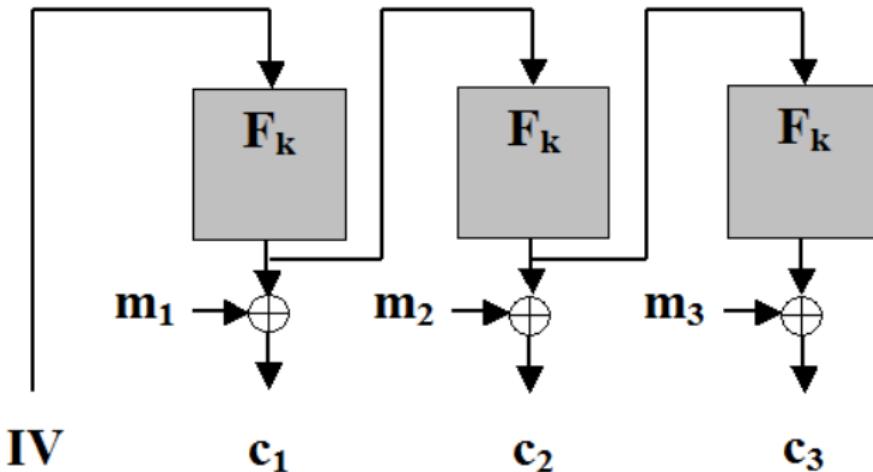
## Modul **CBC** (Cipher Block Chaining)



## Modul CBC (Cipher Block Chaining)

- ▶  $\text{IV}$  este o aleas în mod aleator la criptare;
- ▶  $\text{IV}$  se transmite în clar pentru ca este necesar la decriptare;
- ▶ Pentru decriptare,  $F_k$  trebuie să fie **inversabiliă**;
- ▶ Este **secvențial**, un dezavantaj major dacă se poate utiliza procesarea paralelă.

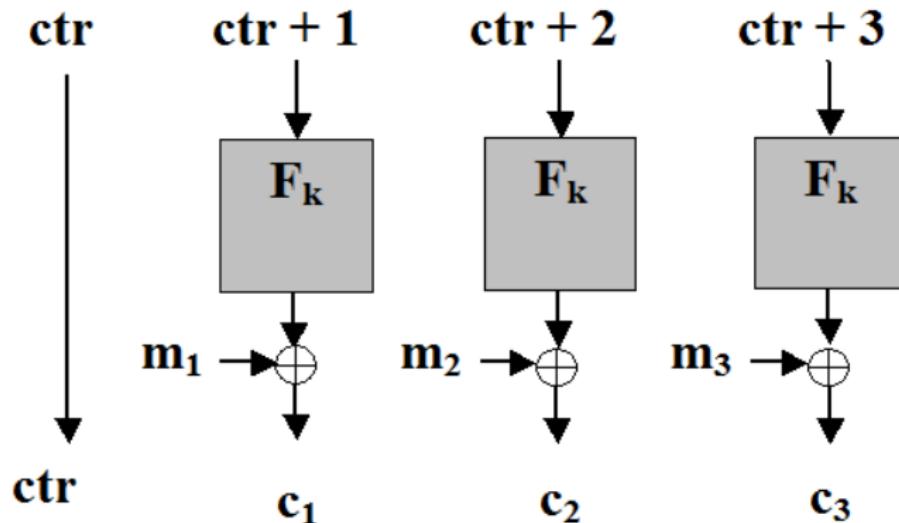
## Modul OFB (Output FeedBack)



## Modul OFB (Output FeedBack)

- ▶ Generează o secvență pseudoaleatoare care se XOR-ează mesajului clar;
- ▶  $IV$  este ales în mod aleator la criptare;
- ▶  $IV$  se transmite în clar pentru ca este necesar la decriptare;
- ▶  $F_k$  nu trebuie neapărat să fie inversabiliă;
- ▶ Este secvențial, însă secvența pseudoaleatoare poate fi pre-procesată anterior decriptării.

## Modul CTR (Counter)



## Modul CTR (Counter)

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

## Exemple

- ▶ **DES** (Data Encryption Standard):
  - ▶ propus de IBM
  - ▶ adoptat ca standard NIST în 1976  
(lungime cheie = 64 biți, lungime bloc = 64 biți)
  - ▶ spart prin căutare exhaustivă în 1997
- ▶ **AES** (Advanced Encryption Standard):
  - ▶ algoritmul *Rijndael* propus de J. Daemen și V. Rijman
  - ▶ adoptat ca standard NIST în 2001  
(lungime cheie = 128, 192, 256 biți, lungime bloc = 128 biți)

## Important de reținut!

- ▶ Sisteme bloc vs. sisteme fluide
- ▶ Noțiunile de PRP, PRF
- ▶ Construcții specifice PRG, PRF, PRP
- ▶ Transpunerea sistemelor bloc în practică - moduri de operare: ECB, CBC, OFB, CTR



# Criptografie și Securitate

- Prelegherea 10 -  
Construcții practice pentru PRP

Adela Georgescu, Ruxandra F. Olimid

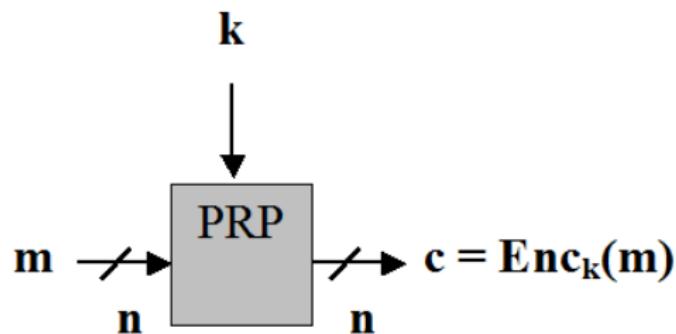
Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Sisteme bloc ca PRP
2. Rețele de substituție - permutare
3. Rețele Feistel

## Sisteme bloc ca PRP

- ▶ Am văzut că sistemele de criptare **bloc** folosesc **PRP**;



## Sisteme bloc ca PRP

- În criteriile de evaluare pentru adoptarea AES s-a menționat:  
*The security provovided by an algorithm is the most important factor... Algorithms will be judged on the following factors...*  
  
***The extent to which the algorithm output is indistinguishable from a random permutation on the input block.***
- Întrebare: Cum se obțin *PRP* în practică?

## Sisteme bloc ca PRP

- ▶ Ideal ar fi să se utilizeze o permutare aleatoare pe  $n$  biți;
- ▶ Ar fi necesari  $\approx n \cdot 2^n$  biți pentru stocare;
- ▶ Pentru  $n = 50$  este necesară o capacitate de stocare de  $\approx 200\,TB$  !
- ▶ Sistemele bloc au în general  $n \geq 64$  sau  $n \geq 128$  biți;
- ▶ În consecință, NU se poate utiliza o (singură) permutare aleatoare!

# Claude Elwood Shannon (1916 - 2001)

- ▶ Shannon propune utilizarea mai multor permutări de dimensiune mai mică;
- ▶ Acesta este al doilea rezultat al lui Shannon pe care îl studiem, după *securitatea perfectă*.



[Wikipedia]

## Rețele de substituție - permutare

- ▶ Se construiește permutarea  $F$ , pe baza mai multor permutări aleatoare  $f_i$  de dimensiune mai mică;
- ▶ Considerăm  $F$  pe 128 biți și 16 permutări 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  $F$ .

## Rețele de substituție - permutare

- ▶ **Întrebare:** Considerând capacitatea necesară de stocare, este  $F$  fesabilă?
- ▶ **Răspuns:** DA.
- ▶ Fiecare  $f_i$  necesită  $\approx 8 \cdot 2^8$  biți pentru stocare;
- ▶ Sunt 16 funcții  $\{f_i\}$ , deci în total  $F$  necesită pentru stocare  $\approx 16 \cdot 8 \cdot 2^8 \approx 32KB$ .

## Rețele de substituție - permutare

- ▶ Întrebare: Este  $F$  PRP?
- ▶ Răspuns: NU.
- ▶ Fie  $x$  și  $x'$  2 intrări care diferă numai prin primul bit:

$$msb(x) \neq msb(x')$$

- ▶ Atunci  $F_k(x)$  și  $F_k(x')$  diferă numai prin primul byte;
- ▶ În cazul unei permutări aleatoare, acest lucru nu se întâmplă.

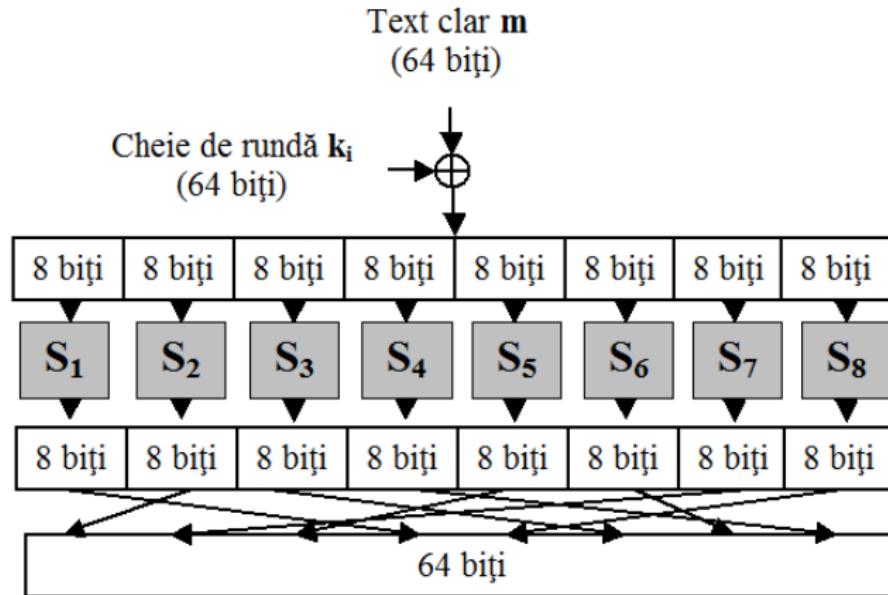
## Rețele de substituție - permutare

- ▶  $F$  se transformă în PRP în 2 pași:
  - ▶ **Pasul 1:** se introduce **difuzie** prin amestecarea (permutarea) biților de ieșire;
  - ▶ **Pasul 2:** se repetă o **rundă** (care presupune *confuzie* și *difuzie*) de mai multe ori;
- ▶ Repetarea *confuziei* și *difuziei* face ca o modificarea unui singur bit de intrare să fie propagată asupra tuturor biților de ieșire;
- ▶ Un exemplu pentru 2 runde și intrarea  $x$ :
  - ▶  $x' = F_k(x)$ ;
  - ▶  $x_1$  se obține prin reordonarea biților din  $x'$ ;
  - ▶  $x'_1 = F_k(x_1)$ ;
  - ▶  $x_2$  se obține prin reordonarea biților din  $x'_1$ .

## Rețele de substituție - permutare

- ▶ 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);
- ▶  $\{f_i\}$  se numesc **S-boxes** (Substitution-boxes);
- ▶ S-box-urile rămân în continuare permutări;
- ▶ 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.

## Rețele de substituție - permutare



## Rețele de substituție - permutare

- ▶ Există 2 principii de bază în proiectarea rețelelor de substituție - permutare:
  - ▶ **Principiul 1:** Inversabilitatea S-box-urilor;
  - ▶ **Principiul 2:** Efectul de avalanșă.

## Principul 1: Inversabilitatea S-box

- ▶ O rețea de substituție-permutare trebuie să fie inversabilă;
- ▶ Dacă fiecare rundă este inversabilă, atunci rețeaua este inversabilă;
- ▶ Într-o rundă:
  - ▶ XOR este inversabil;
  - ▶ Permutarea finală a biților de ieșire este inversabilă;
- ▶ În concluzie, dacă toate S-box-urile sunt inversabile, atunci rețeaua este inversabilă;
- ▶ Principiul 1 - necesitate funcțională (pentru decriptare).

## Principul 2: Efectul de avalanșă

- ▶ Un singur bit modificat la intrare **trebuie** să afecteze toți biții din secvența de ieșire;
- ▶ Efectul de avalanșă apare într-o rețea de substituție-permutare dacă:
  1. S-box-urile sunt proiectate a.î. un singur bit schimbat la intrare să schimbe cel puțin 2 biți de la ieșire;
  2. Permutarea este proiectată a.î. biții de la ieșirea unui S-box să fie împărțiți între intrările în S-box-uri diferite la runda următoare.
- ▶ Principiul 2 - necesitate de securitate.

## Principul 2: Efectul de avalanșă

- ▶ **Întrabare:** De ce sunt suficiente cele 2 proprietăți pentru obținerea efectului de avalanșă?
- ▶ Presupunem 2 intrări care diferă printr-un singur bit;
- ▶ După Runda 1, ieșirile diferă prin 2 biți (*Proprietatea 1*);
- ▶ La intrarea în Runda 2, biții care diferă sunt intrări în S-box-uri diferite (*Proprietatea 2*);
- ▶ După Runda 2, ieșirile diferă prin 4 biți (*Proprietatea 1*);
- ▶ Urmând acest raționament, după Runda  $r$ , ieșirile diferă prin aproximativ  $2^r$  biți;
- ▶ În concluzie, un sistem bloc cu intrarea de  $n = 2^r$  biți obține efectul de avalanșă după cel puțin  $r$  runde.

## Rețele 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*.

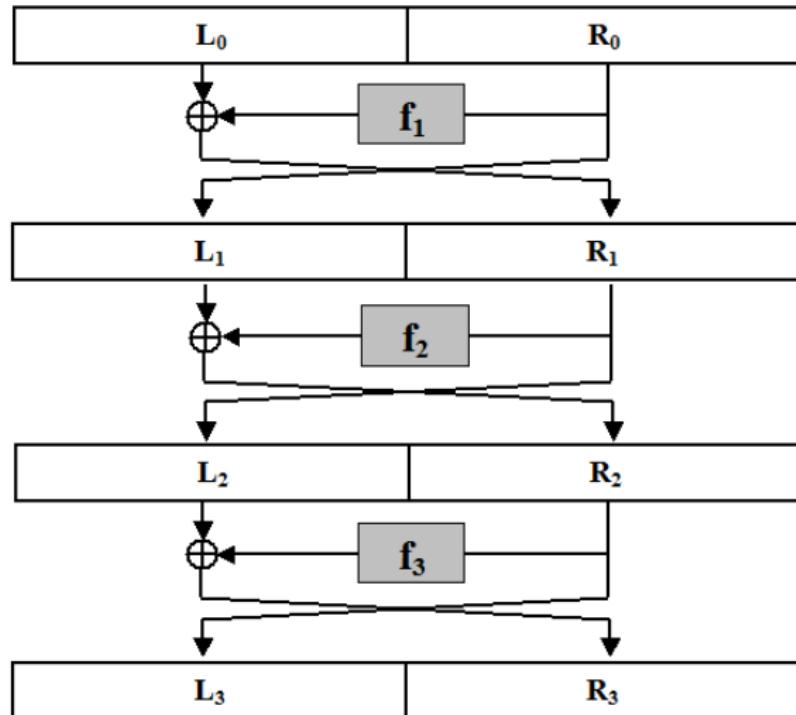
# Horst Feistel (1915 - 1990)



[Wikipedia]

- ▶ Structurile simetrice utilizate în construcția sistemelor bloc poartă numele lui Feistel;
- ▶ Munca sa de cercetare la IBM a condus la sistemul de criptare Lucifer și mai târziu la DES.

# Rețele Feistel



## Rețele Feistel

- ▶ 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  $\widehat{f}_i$ :

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

# Rețele Feistel

- ▶ 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:

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

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

# Construcții

Am omis în cursurile anterioare ultima construcție:

- ▶ **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** ✓

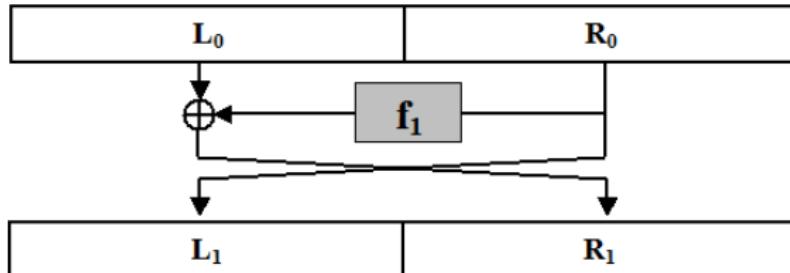
Pornind de la PRP se poate construi PRF

- ▶ **PRF  $\Rightarrow$  PRP**

Pornind de la PRF se poate construi PRP

## *PRF $\Rightarrow$ PRP*

- ▶ Revenim acum asupra construcției, când cunoaștem noțiunea de rundă Feistel;
- ▶ Considerăm o singură rundă Feistel pentru care  $f_1(x) = F_k(x)$ , cu  $F$  PRF;



- ▶ **Întrebare:** Este această construcție PRP?

## *PRF $\Rightarrow$ PRP*

- ▶ **Răspuns:** NU.
- ▶ Construcția este o permutare pe  $\{0, 1\}^n, \dots$
- ▶ ... este inversabilă, ...
- ▶ ... dar ieșirea nu este o secvență pseudoaleatoare: primii  $n/2$  biți de ieșire sunt egali cu ultimii  $n/2$  biți de intrare;
- ▶ Se mărește numărul de runde Feistel, păstrând  $f_i(x) = F_{k_i}(x)$ , unde  $k_i$  sunt chei independente;
- ▶ Pentru  $i = 3$  se obține PRP;
- ▶ Altfel spus, o rețea Feistel de 3 runde construită folosind 3 PRF  $f_1(x) = F_{k_1}(x)$ ,  $f_2(x) = F_{k_2}(x)$  și  $f_3(x) = F_{k_3}(x)$  cu  $k_1, k_2$  și  $k_3$  independente este PRP.

## Important de reținut!

- ▶ Rețele de substituție-permutare
- ▶ Rețele Feistel



# Criptografie și Securitate

- Prelegerea 10.1 -

Data Encryption Standard - DES

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

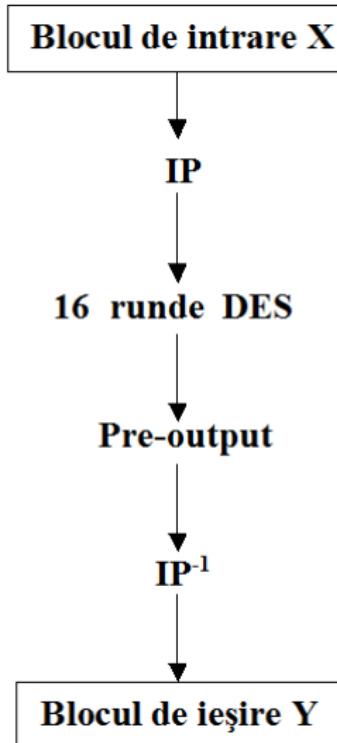
# Cuprins

1. Scurt istoric
2. DES cu număr redus de runde
3. Securitatea sistemului DES

## DES - Data Encryption Standard

- ▶ 1970 - Horst Feistel proiectează Lucifer, o familie de cifruri bloc, la IBM, cu lungime cheie = 128 biți și lungime bloc = 128 biți;
- ▶ 1973 - NIST inițiază o cerere pentru propuneri în vederea standardizării cifrurilor bloc în SUA; IBM trimite o variantă de Lucifer.
- ▶ 1976 - NIST adoptă Lucifer modificat ca standard DES cu lungime cheie = 56 biți și lungime bloc = 64 biți.
- ▶ 1997 - DES este spart prin căutare exhaustivă (forță brută).
- ▶ 2001 - NIST adoptă Rijndael ca noul standard AES în locul lui DES.

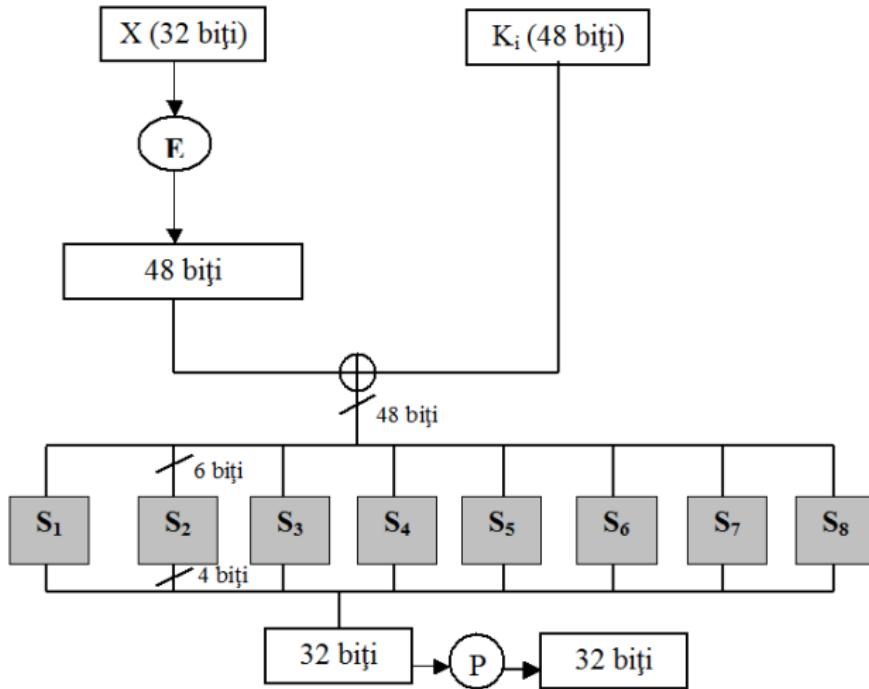
# DES



## Descriere 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 derive 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 $f(k_i, R)$



## Funcția $f(k_i, R)$

- ▶ Funcția  $\hat{f}$  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.

S <sub>5</sub>		Cei 4 biți din mijloc															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Biții 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 **două biți** de la ieșire.

## Efectul de avalanșă

- ▶ DES are un puternic efect de avalanșă generat de ultima proprietate menționată mai sus și de permutările folosite;
- ▶ Pentru a vedea aceasta, vom urmări diferența între valorile intermediare dintr-un calcul DES a două valori de intrare care diferă printr-un singur bit;
- ▶ Notăm cele două valori cu  $(L_0, R_0)$  și  $(L_0', R_0')$  unde  $R_0 = R_0'$ ;
- ▶ După prima rundă, valorile  $(L_1, R_1)$  și  $(L_1', R_1')$  încă diferă printr-un singur bit, deși acum diferența e în partea dreaptă;
- ▶ În a doua rundă DES,  $R_1$  și  $R_1'$  trec prin funcția  $\hat{f}$ ;

## Efectul de avalanșă

- ▶ Presupunând că bitul în care  $R_1$  și  $R_1'$  diferă nu este duplicat în pasul de expandare, ele încă diferă printr-un bit înainte de aplicarea SBOX-ului;
- ▶ După SBOX, valorile intermediare diferă în cel puțin *doi* biți;
- ▶  $(L_2, R_2)$  și  $(L_2', R_2')$  diferă în *trei* biți: 1-bit diferență între  $L_2$  și  $L_2'$  și 2-biți diferență între  $R_2$  și  $R_2'$ ;
- ▶ Permutarea din  $\hat{f}$  împărătie diferența dintre  $R_2$  și  $R_2'$  în diferite regiuni ale lor;
- ▶ La următoarea rundă fiecare bit care diferă va fi folosit ca intrare într-un SBOX diferit, rezultând o diferență de 4 biți între jumătățile drepte ale valorilor intermediare;
- ▶ Efectul este exponențial și după 7 runde toți 32 biți vor fi modificați.

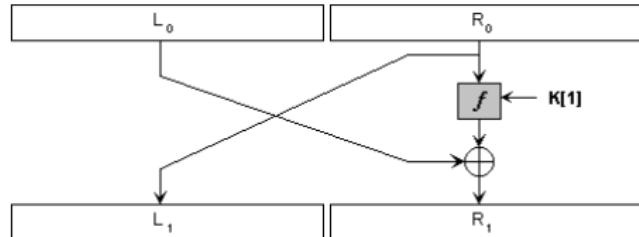
## Efectul de avalanșă

- ▶ După 8 runde toți biții din jumătatea stângă vor fi modificați;
- ▶ DES are 16 runde deci efectul de avalanșă este atins foarte repede;
- ▶ Deci DES aplicat pe două intrări similare întoarce ieșiri complet diferite și independente;
- ▶ Efectul se datorează și permutărilor alese cu grijă (este verificat faptul că permutări aleatoare nu oferă același efect puternic de avalanșă).

## Atacuri pe DES cu număr redus de runde

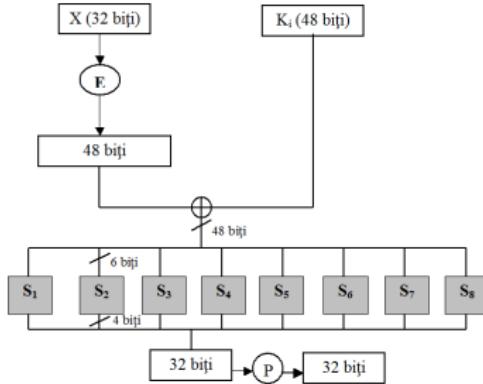
- ▶ Pentru a înțelege securitatea DES vom studia mai întâi comportamentul lui DES pe un număr redus de runde (maxim 3);
- ▶ Sigur, DES cu 3 runde nu este o funcție pseudoleatoare pentru că efectul de avalanșă încă nu e complet;
- ▶ Vom arăta câteva atacuri cu text clar care găsesc cheia  $k$ ;
- ▶ Adversarul are deci acces la perechi de forma  $\{(x_i, y_i)\}$  cu  $y_i = DES_k(x_i)$ ;
- ▶ În descrierea atacurilor, ne vom concentra asupra unei singure perechi  $(x, y)$ .

## DES cu o singură rundă



- ▶ Cunoaștem  $x = (L_0, R_0)$  și  $y = (L_1, R_1)$  unde
  - ▶  $L_1 = R_0$
  - ▶  $R_1 = L_0 \oplus f_1(R_0)$
- ▶ De asemenea,  $f_1(R_0) = R_1 \oplus L_0$

# DES cu o singură rundă

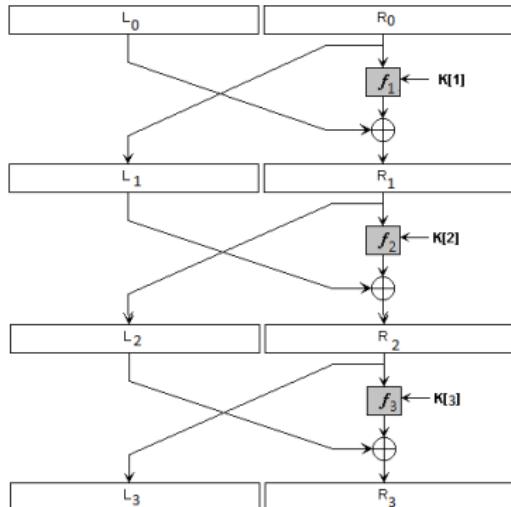


- ▶ Aplicând  $P^{-1}(R_1 \oplus L_0)$  obținem o valoare intermediară care reprezintă ieșirea din cele 8 SBOX-uri;
- ▶ Intrarea în SBOX-uri este  $E(R_0) \oplus k_1$ ;  $R_0$  este cunoscut, la fel ieșirea din SBOX-uri;
- ▶ Pentru fiecare ieșire din SBOX, există 4 valori posibile ale porțiunii corespunzătoare de 6 biți din cheia  $k_1$  care ar conduce la acea valoare.

## DES cu una sau două runde

- ▶ Atac DES cu o singură rundă
  - ▶ Am redus numărul cheilor posibile de la  $2^{48}$  la  $4^8 = 2^{16}$ ;
  - ▶ Acum se pot verifica pe rând toate variantele și recuperă complet cheia.
- ▶ Atac DES cu două runde
  - ▶ Atacul găsește cheile  $k_1$  și  $k_2$  în timp  $2 \cdot 2^{16}$  atunci când se cunoaște o pereche text clar/text criptat.

## DES cu trei runde



- ▶ Nu se cunosc  $R_1$ ,  $L_2$  și deci nici toate intrările/ieșirile din fiecare funcție  $f$ ;
- ▶ Atacul anterior nu funcționează; un nou atac va explora relațiile dintre ieșirile respectiv intrările în  $f_1$  și  $f_3$ ;
- ▶ Atacul traversează spațiul cheilor pentru fiecare jumătate a cheii master și, în timp  $2 \cdot 2^{28}$  va produce câte  $2^{12}$  variante pentru fiecare jumătate a cheii;
- ▶ Complexitatea timp totală este  $2 \cdot 2^{28} + 2^{24} < 2^{30}$ .

# 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\$;

## Securitatea sistemului DES

- ▶ Primul atac practic a fost demonstrat în 1997 când un număr de provocări 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\$

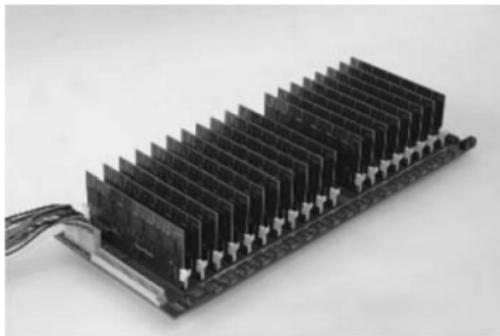
# Securitatea sistemului DES



Figure: Deep Crack-construită pentru căutare exhaustivă DES în 1998

- ▶ Ultima provocare a fost spartă în 22 de ore (efort combinat de la ultimele două provocări);
- ▶ Atacurile prin forță brută pe DES au devenit un studiu de caz în încercarea de a micșora costurile;

## Securitatea sistemului DES



**Figure:** COPACABANA-Cost Optimized Parallel Code-Breaker

- ▶ În 2006, o echipă de cercetători de la universitățile Bochum și Kiel din Germania a construit mașina COPACABANA (*Cost Optimized Parallel Code-Breaker*) care permite găsirea cheii DES cu un timp de căutare mediu de mai puțin de 7 zile, la un cost de 10.000\$.

## Securitatea sistemului DES

- ▶ O altă problemă a sistemului DES, mai puțin importantă, este lungimea blocului relativ scurtă (64 biți);
- ▶ 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 compromisă cu probabilitate mare;
- ▶ Concluzionând, putem spune că insecuritatea sistemului DES nu are a face cu structura internă sau construcția in sine (care este remarcabilă), ci se datorează numai lungimii cheii prea mici.

## Criptanaliză avansată

- ▶ La sfârșitul anilor '80, Biham și Shamir au dezvoltat o tehnică numită **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 mulțime de  $2^{47}$  texte clare alese;
- ▶ Din punct de vedere teoretic, atacul a fost o inovație, dar practic e aproape imposibil de realizat;
- ▶ La inceputul anilor '90, Matsui a dezvoltat **criptanaliza liniară** aplicată cu succes pe DES;
- ▶ Deși necesită  $2^{43}$  texte criptate, avantajul este că textele clare nu trebuie să fie alese de atacator, ci doar cunoscute de el;
- ▶ Problema însă rămâne aceeași: atacul e 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 e mai mare decât  $2^{-n}$ ;
- ▶ La un cifru bloc slab, ea apare cu o probabilitate mult mai mare;

## Criptanaliza diferențială

- ▶ Dacă o diferențială există cu probabilitate  $p \gg 2^{-n}$ , cîfrul bloc nu mai este permutare pseudoaleatoare;
- ▶ Ideea este de a folosi multe diferențiale cu  $p$  ușor mai mare decât  $2^{-n}$  pentru a găsi cheia secretă folosind un atac cu text clar ales;
- ▶ Criptanaliza diferențială a fost folosită cu succes pentru a ataca cifruri bloc (altele decât DES și AES), de pildă FEAL-8;

## 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  $k$

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

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.

## Creșterea lungimii cheii

- ▶ Singura vulnerabilitate practică DES este cheia scurtă;
- ▶ S-au propus diverse metode de a construi un sistem bazat pe DES care să folosească o cheie mai lungă;
- ▶ Nu se recomandă schimbarea structurii interne încărcând securitatea sistemului ar putea fi afectată;
- ▶ Solutie alternativă: considerăm DES o cutie neagră care implementează un cifru bloc "perfect" cu o cheie pe 56 biți.

## 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

- ▶ lată 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  $O(2^n)$ .

## Criptare triplă

- Există două variante:

1. Trei chei independente -  $k_1, k_2$  și  $k_3$  iar

$$F'_{k_1, k_2, k_3} = F_{k_3}(F_{k_2}^{-1}(F_{k_1}(x)))$$

2. Două chei independente -  $k_1$  și  $k_2$  iar

$$F'_{k_1, k_2} = F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$$

- $F'$  este ales astfel pentru a fi compatibil cu  $F$  atunci când cheile sunt alese  $k_1 = k_2 = k_3$ ;
- Prima variantă are lungimea cheii  $3n$  dar cel mai bun atac necesită timp  $2^{2n}$  (funcționează atacul meet-in-the-middle);
- A doua variantă are lungimea cheii  $2n$  și cel mai bun atac necesită timp  $2^{2n}$ .

## Triplu-DES (3DES)

- ▶ Se bazează pe tripla invocare a lui DES folosind două sau trei chei;
- ▶ Este considerat sigur și în 1999 l-a înlocuit pe DES ca standard;
- ▶ 3DES este foarte eficient în implementările hardware (la fel ca și DES) dar totuși lent în implementări software;
- ▶ Este încă folosit la scară largă, fiind considerat un cifru bloc puternic;
- ▶ Este popular în aplicațiile financiare și în protejarea informațiilor biometrice din pașapoartele electronice;

## Triplu-DES (3DES)

- ▶ Singurele dezavantaje ar fi lungimea mică a blocurilor și faptul că este destul de lent fiindcă aplică DES de trei ori;
- ▶ Acestea au dus la înlocuirea lui ca standard cu AES;
- ▶ DES-X este o variantă DES care rezistă mai bine (decât DES) la forța brută;
- ▶ DES-X folosește două chei suplimentare  $k_1, k_2$ :

$$DESX_{k,k_1,k_2} = k_2 \oplus DES_k(x \oplus k_1)$$

## Important de reținut!

- ▶ DES a fost sistemul simetric dominant de la mijlocul anilor '70 până la mijlocul anilor '90;
- ▶ DES cu cheia pe 56 biți poate fi spart relativ ușor astăzi prin forță brută;
- ▶ Însă, este foarte greu de spart folosind criptanaliza diferențială sau liniară;
- ▶ Pentru 3DES nu se cunoaste nici un atac practic.



# Criptografie și Securitate

- Prelegerea 10.2 -  
Advanced Encryption Standard - AES

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Scurt istoric
2. Construcție
3. Securitatea sistemului AES

## AES - Advanced Encryption Standard

- ▶ ianuarie 1997 - NIST anunță competiția pentru selecția unui nou sistem de criptare bloc care să înlocuiască DES;
- ▶ septembrie 1997 - 15 propuneri: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, and Twofish;
- ▶ 1998, 1999 - au loc 2 workshop-uri în urma cărora rămân 5 finaliști: MARS, RC6, Rijndael, Serpent, Twofish;
- ▶ octombrie 2000 - după un al treilea workshop se anunță câștigătorul: **Rijndael**.

# AES - Advanced Encryption Standard



[Google Scholar - User profiles]



[<http://keccak.noekeon.org/team.html>]

Rijndael = Rijmen + Daemen

## Descriere AES

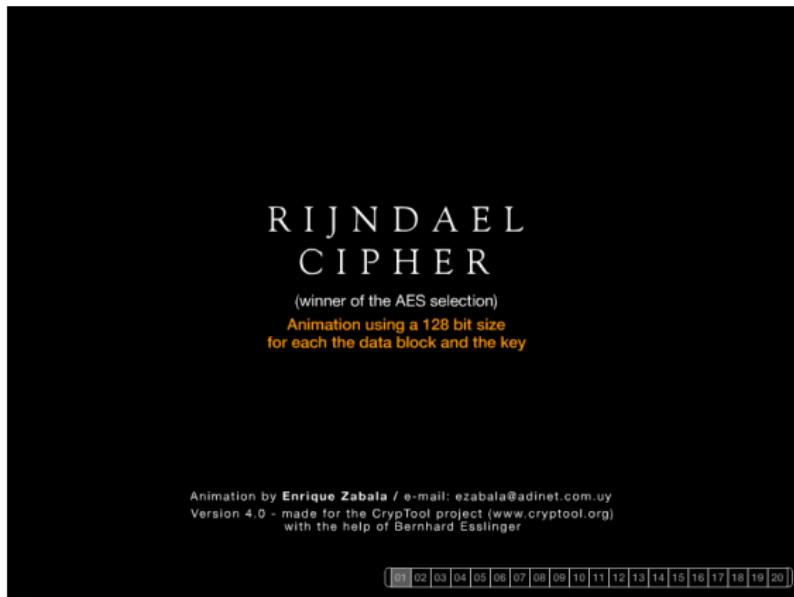
- ▶ AES este o rețea de substituție - permutare pe 128 biți care poate folosi chei de 128, 192 sau 256 biți;
- ▶ Lungimea cheii determină numărul de runde:

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.

# Descriere AES

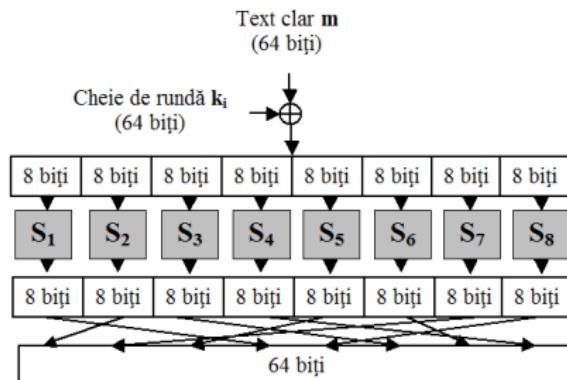
- ▶ Rijndael Animation - CrypTool Project:



[<http://www.cryptool.org/en/>; <https://youtu.be/gP4PqVGudtg>]

# Descriere AES

- ▶ Să ne reamintim exemplul de rețea de substituție - permutare prezentat în cursurile anterioare:
  1. XOR cu cheia de rundă;
  2. aplicarea S-box-urilor pentru a obține *confuzie*;
  3. amestecarea bițiilor pentru a obține *difuzie*.



## Descriere AES

- ▶ AES este o rețea de substituție - permutare:
  - ▶ **AddRoundKey**: XOR cu cheia de rundă;
  - ▶ **SubBytes**: fiecare octet este înlocuit de un alt octet conform tabelei de substituție S-box (unică pentru AES!);
  - ▶ **ShiftRows** și **MixColumns**: amestecarea bițiilor presupune mai mult decât o simplă permutare, folosind o transformare liniară pe biți.

## Securitatea sistemului AES

- ▶ 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.

**"It is free, standardized, efficient, and highly secure."**

(J.Katz, Y.Lindell, *Introduction to Modern Cryptography*)

## Important de reținut!

- ▶ AES este standard actual NIST;
- ▶ AES are la bază algoritmul Rijndael, fiind o rețea de substituție-permutare;
- ▶ Pentru AES cu număr complet de runde nu se cunoaște nici un atac mai eficient decât căutarea exhaustivă.



# Criptografie și Securitate

- Prelegherea 11 -  
Securitate CPA și CCA

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Scenarii de atac

2. Securitate CPA

3. Securitate CCA

## Scenarii de atac

- ▶ Reamintim câteva dintre scenariile de atac pe care le-am mai întâlnit:
  - ▶ **Atac cu text criptat:** Atacatorul știe doar *textul criptat* - poate încerca un **atac prin forță brută** prin care se parcurg toate cheile până se găsește cea corectă;
  - ▶ **Atac cu text clar:** Atacatorul cunoaște una sau mai multe perechi (*text clar, text criptat*);
  - ▶ **Atac cu text clar aleas:** Atacatorul poate obține criptarea unor texte clare alese de el;
  - ▶ **Atac cu text criptat aleas:** Atacatorul are posibilitatea să obțină decriptarea unor texte criptate alese de el.

## Scenarii de atac

- ▶ Ultimele 2 scenarii de atac oferă adversarului putere crescută;
- ▶ Acesta devine un adversar **activ**, care primește abilitatea de a obține criptarea și / sau decriptarea unor mesaje, respectiv texte criptate alese de el;
- ▶ În plus, adversarul poate alege mesajele sau textele criptate în mod **adaptiv** în funcție de răspunsurile primite precedent.

# Noțiuni de securitate

- ▶ Definim astfel 2 noțiuni de securitate:
  - ▶ **CPA (Chosen-Plaintext Attack)**: adversarul poate să obțină criptarea unor mesaje alese de el;
  - ▶ **CCA (Chosen-Ciphertext Attack)**: adversarul poate să obțină criptarea unor mesaje alese de el și decriptarea unor texte criptate alese de el.

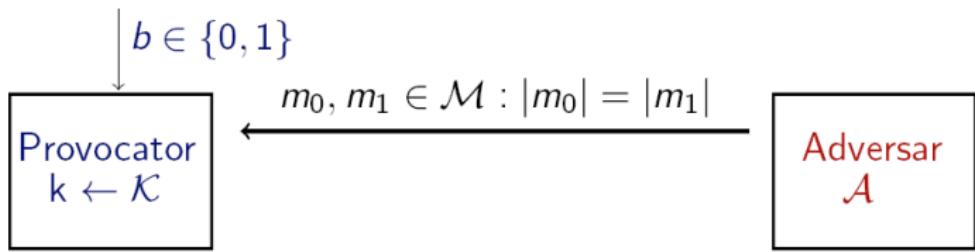
- ▶ Capabilitățile adversarului: el poate interacționa cu un **oracol de criptare**, fiind un adversar *activ* care poate rula atacuri în timp polinomial;
- ▶ Adversarul poate transmite către oracol orice mesaj  $m$  și primește înapoi textul criptat corespunzător;
- ▶ Dacă sistemul de criptare este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

- ▶ Considerăm că securitatea este impactată dacă adversarul poate să distingă între criptările a două mesaje aleatoare;
- ▶ Vom defini securitatea CPA pe baza unui experiment de indistinctibilitate  $\text{Priv}_{\mathcal{A}, \pi}^{\text{cpa}}(n)$  unde  $\pi = (\text{Enc}, \text{Dec})$  este schema de criptare iar  $n$  este parametrul de securitate al schemei  $\pi$ ;
- ▶ Personajele participante: **adversarul  $\mathcal{A}$**  care încearcă să spargă schema și un **provocator (challenger)**;

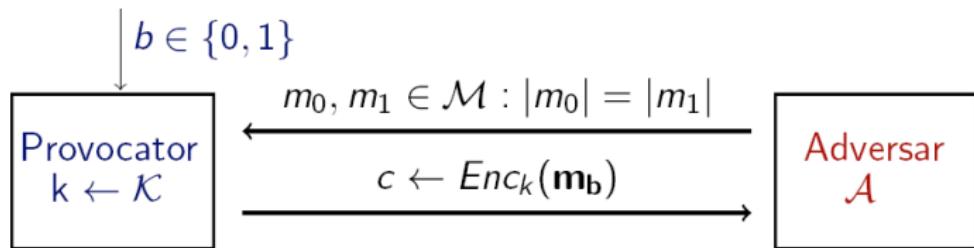
# Experimentul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$



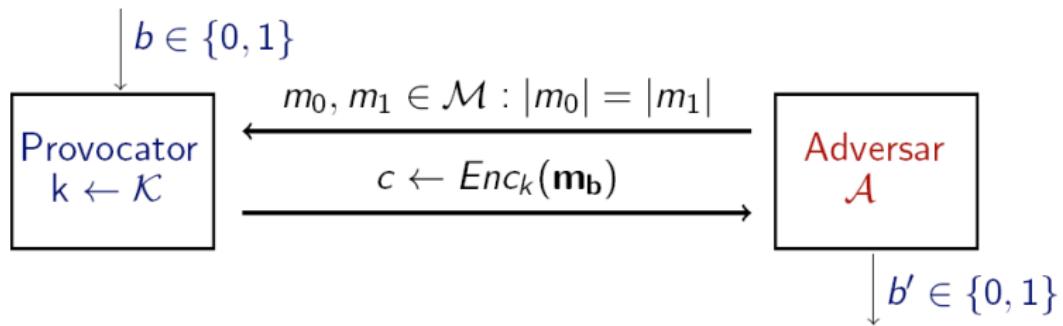
# Experimentul $\text{Priv}_{\mathcal{A}, \pi}^{cpa}(n)$



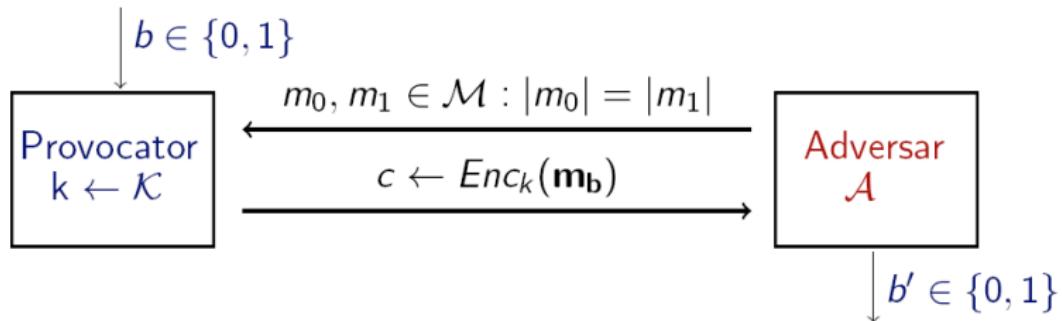
# Experimentalul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$



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

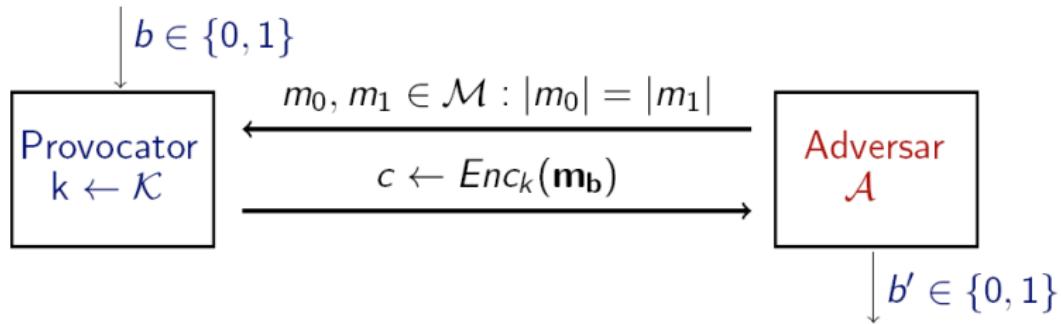


## Experimentul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$



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

## Experimentul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$



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

## Experimentul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$

### 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

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

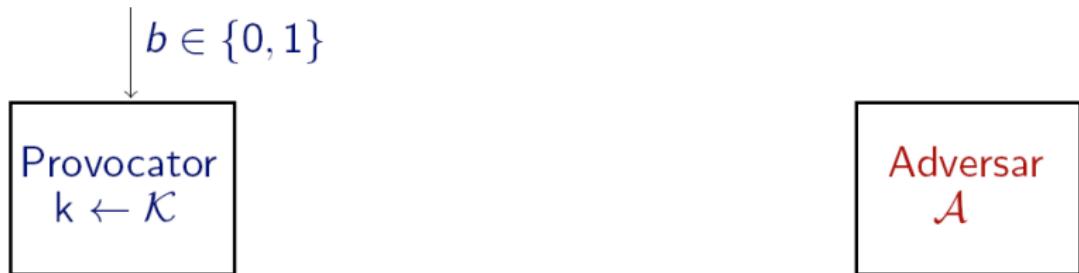
- ▶ Un adversar nu poate determina care text clar a fost criptat cu o probabilitate semnificativ mai mare decât dacă ar fi ghicit (în sens aleator, dat cu banul), chiar dacă are acces la oracolul de criptare.

- ▶ **Întrebare:** Un sistem de criptare CPA-sigur este întotdeauna semantic sigur?
- ▶ **Răspuns:** DA! Experimentul  $Priv_{\mathcal{A}, \pi}^{eav}(n)$  este  $Priv_{\mathcal{A}, \pi}^{cpa}(n)$  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.

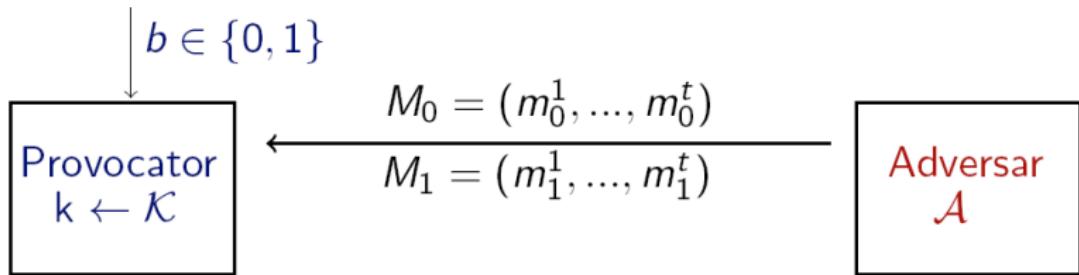
## Securitate CPA - Criptare multiplă

- ▶ În definiția precedentă am considerat cazul unui adversar care primește **un singur** text criptat;
- ▶ În realitate, în cadrul unei comunicații se trimit **mai multe mesaje** pe care adversarul le poate intercepta;
- ▶ Definim ce înseamnă o schemă sigură chiar și în aceste condiții.

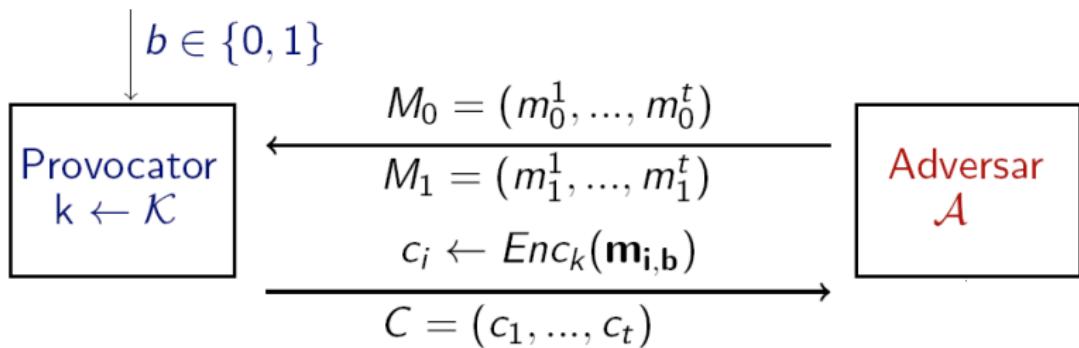
# Experimentul $\text{Priv}_{\mathcal{A}, \pi}^{cpa}(n)$



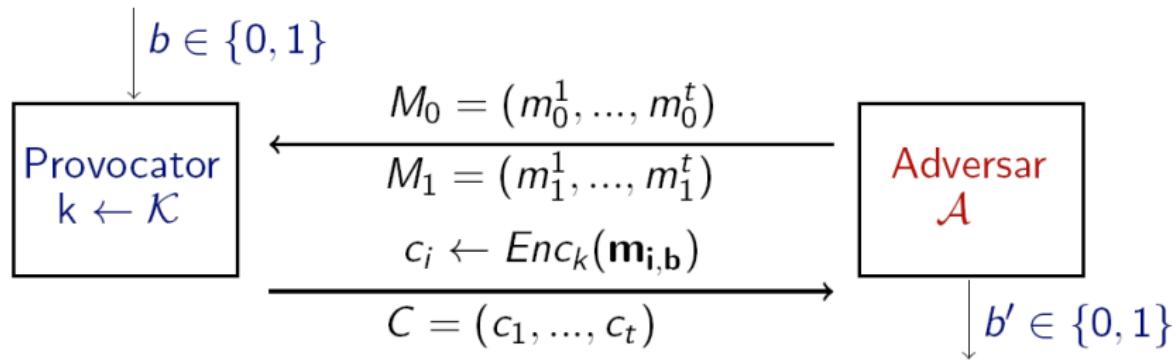
## Experimentul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$



## Experimentalul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$

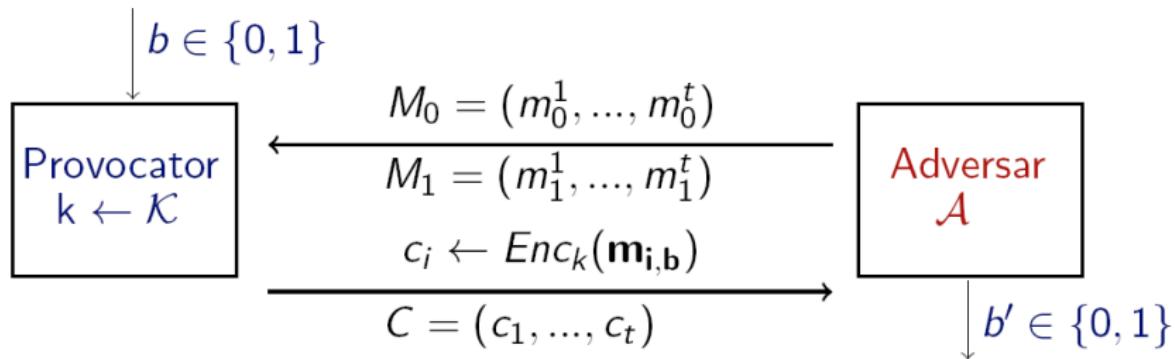


## Experimentalul $\text{Priv}_{\mathcal{A}, \pi}^{cpa}(n)$



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

## Experimentul $Priv_{\mathcal{A}, \pi}^{cpa}(n)$



- ▶ Output-ul experimentului este 1 dacă  $b' = b$  și 0 altfel;
- ▶ Definiția de securitate este aceeași, doar că se referă la experimentul de mai sus.
- ▶ Securitatea pentru criptare **simplă** implică securitate pentru criptare **multiplă**!

## Securitate CCA

- ▶ Capabilitățile adversarului: el poate interacționa cu un **oracol de criptare** și cu un **oracol de decriptare**, fiind un adversar *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.

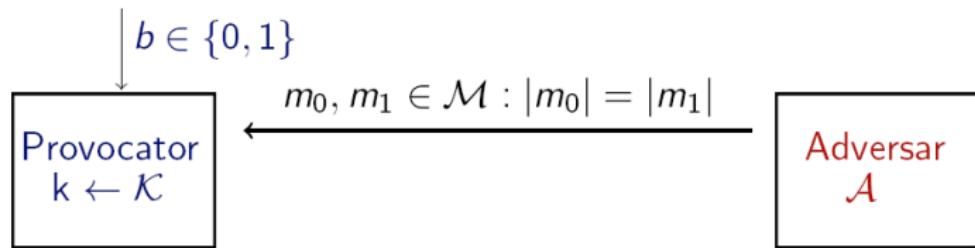
## Securitate CCA

- ▶ Considerăm că securitatea este impactată dacă adversarul poate să distingă între criptările a două mesaje aleatoare;
- ▶ Vom defini securitatea CCA pe baza unui experiment de indistinctibilitate  $\text{Priv}_{\mathcal{A}, \pi}^{cca}(n)$  unde  $\pi = (\text{Enc}, \text{Dec})$  este schema de criptare iar  $n$  este parametrul de securitate al schemei  $\pi$ ;
- ▶ Personajele participante: **adversarul  $\mathcal{A}$**  care încearcă să spargă schema și un **provocator (challenger)**;

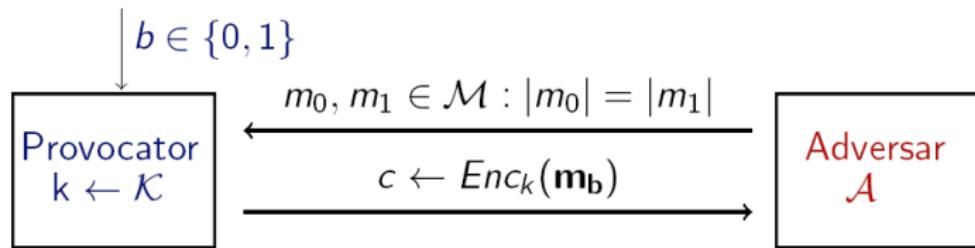
## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



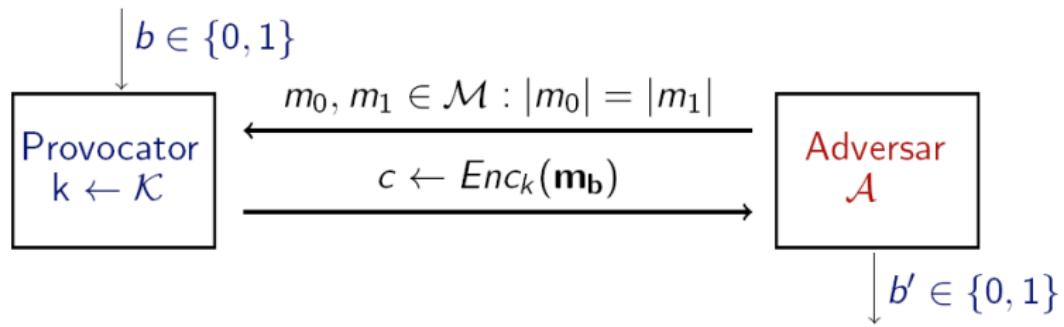
# Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



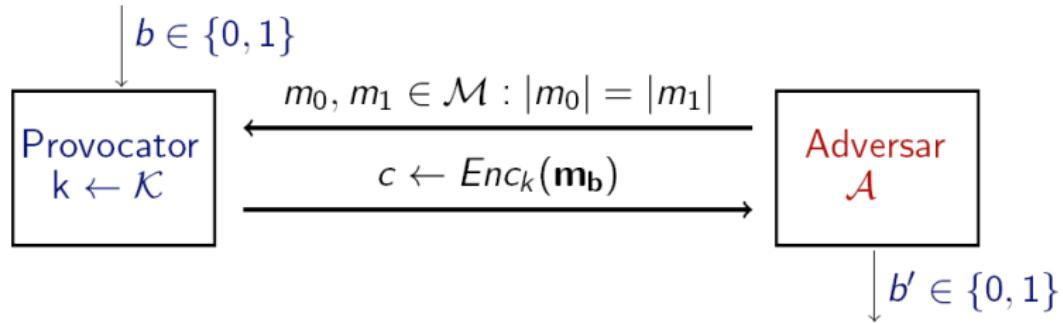
## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



## Experimentalul $Priv_{\mathcal{A}, \pi}^{cca}(n)$

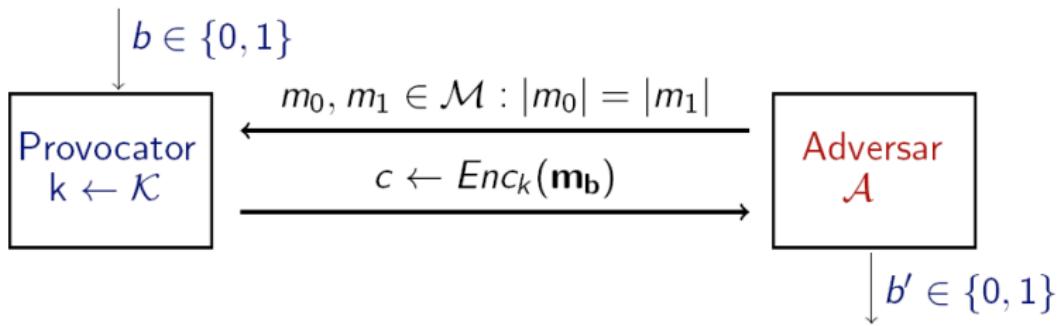


## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



- ▶ Pe toată durata experimentului,  $\mathcal{A}$  are acces la oracolul de criptare  $Enc_k(\cdot)$  și la oracolul de decriptare  $Dec_k(\cdot)$  cu restricția că nu poate decripta  $c$ !

## Experimentul $Priv_{\mathcal{A},\pi}^{cca}(n)$



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

## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$

### Definiție

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

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

- 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 oracolele de criptare și decriptare.

# Securitate CCA

- ▶ **Întrebare:** Un sistem de criptare CCA-sigur este întotdeauna CPA-sigur?
- ▶ **Răspuns:** DA! Experimentul  $Priv_{\mathcal{A}, \pi}^{cpa}(n)$  este  $Priv_{\mathcal{A}, \pi}^{cca}(n)$  în care  $\mathcal{A}$  nu folosește oracolul de decriptare.
- ▶ **Întrebare:** Un sistem de criptare determinist poate fi CCA-sigur?
- ▶ **Răspuns:** NU! Sistemul nu este CPA-sigur, deci nu poate fi CCA-sigur.

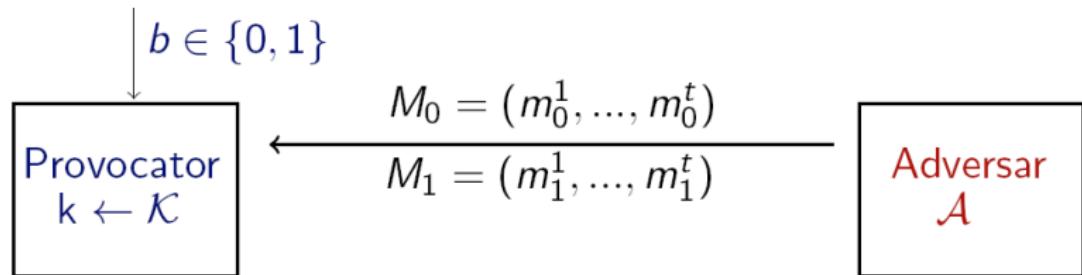
## Securitate CCA - Criptare multiplă

- ▶ În definiția precedentă am considerat cazul unui adversar care primește **un singur** text criptat;
- ▶ În realitate, în cadrul unei comunicații se trimit **mai multe mesaje** pe care adversarul le poate intercepta;
- ▶ Definim ce înseamnă o schemă sigură chiar și în aceste condiții.

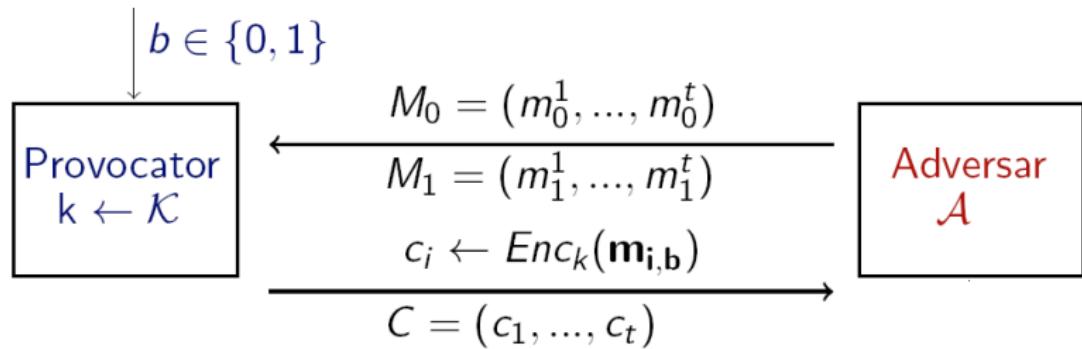
# Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



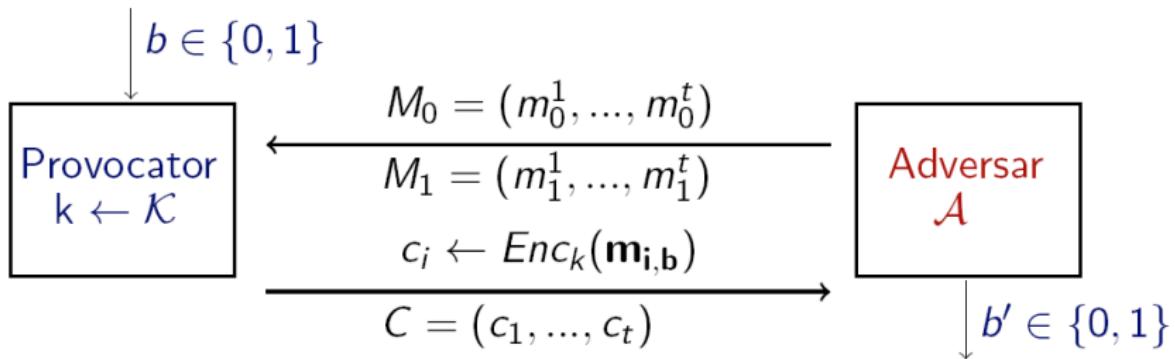
## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



## Experimentalul $Priv_{\mathcal{A}, \pi}^{cca}(n)$

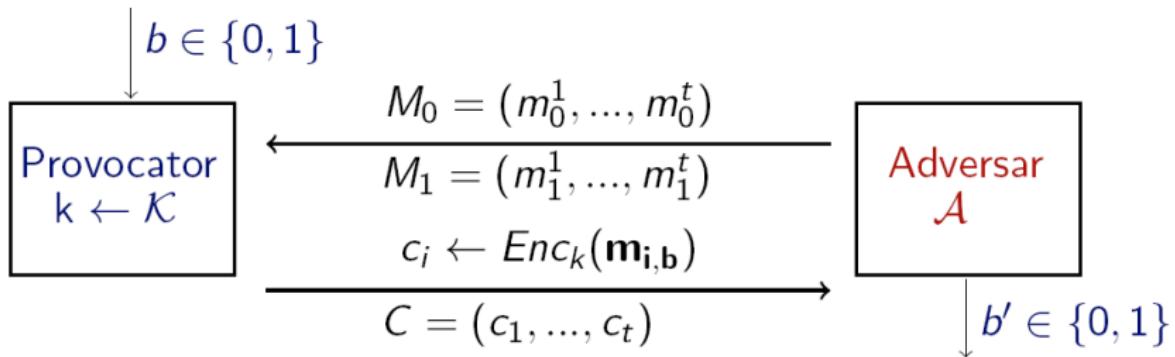


## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



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

## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



- ▶ Output-ul experimentului este 1 dacă  $b' = b$  și 0 altfel;
- ▶ Definiția de securitate este aceeași, doar că se referă la experimentul de mai sus.

## Important de reținut!

- ▶ Securitate CCA  $\Rightarrow$  securitate CPA  $\Rightarrow$  securitate semantică
- ▶ Schemele deterministe nu sunt semantic / CPA / CCA sigure



# Criptografie și Securitate

- Prelegherea 12 -

Coduri de autentificare a mesajelor

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Necesitatea autentificării
2. Definiție MAC
3. Securitate MAC
4. CBC-MAC

## Comunicare sigură și integritatea mesajelor

- ▶ Un scop de bază al criptografiei este să asigure comunicarea sigură de-a lungul unui canal public de comunicare;
- ▶ Am vazut cum putem obține aceasta cu ajutorul **schemelor de criptare**;
- ▶ Însă, nu ne interesează doar ca adversarul să nu aibă acces la mesajele trimise, ci...
- ▶ Vrem să garantăm **integritatea mesajelor** (sau **autentificarea mesajelor**)
- ▶ Aceasta înseamnă că **mesajul primit de Bob este exact mesajul trimis de Alice.**

## Comunicare sigură și integritatea mesajelor

- ▶ Iată un exemplu:
- ▶ Să considerăm cazul în care un mare lanț de supermarket-uri trimite o comandă pe email către un furnizor pentru a achiziționa 10.000 bax-uri de apă minerală;
- ▶ Odată primită comanda, furnizorul trebuie să verifice următoarele:
  1. Comanda este autentică? A fost trimisă cu adevărat de un supermarket sau de către un adversar care a furat contul de email al clientului respectiv ?
  2. Dacă s-a convins de autenticitatea comenzii, trebuie verificat dacă detaliile ei sunt cele originale sau au fost modificate pe parcurs de un adversar.

## Comunicare sigură și integritatea mesajelor

- ▶ În exemplul precedent, problema este doar de integritate a mesajelor, și nu de confidențialitate (comanda nu e secretă);
- ▶ În general nu ne putem baza pe încredere în ceea ce privește integritatea mesajelor transmise, indiferent că ele sunt:
  - ▶ comenzi efectuate online
  - ▶ operațiuni bancare online
  - ▶ email, SMS
- ▶ Vom vedea cum putem folosi tehnici criptografice pentru a preveni modificarea nedectată a mesajelor transmise.

## Criptare vs. autentificarea mesajelor

- ▶ Criptarea, în general, **NU** oferă integritatea mesajelor!
- ▶ Dacă un mesaj este transmis criptat de-a lungul unui canal de comunicare, nu înseamnă că un adversar nu poate modifica/altera mesajul aşa încât modificarea să aiba sens în textul clar;
- ▶ Verificăm, în continuare, că **nici o schemă de criptare studiată** nu oferă integritatea mesajelor;

# Criptare vs. autentificarea mesajelor

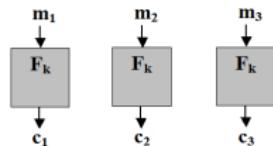
## ► Criptarea folosind sisteme fluide

- ▶  $Enc_k(m) = G(k) \oplus m$ , unde G este un PRG;
- ▶ Dacă modificăm un singur bit din textul criptat  $c$ , modificarea se va reflecta imediat în același bit din textul clar;
- ▶ Consecințele pot fi grave: de pildă, să considerăm transferul unei sume de bani în dolari criptate, reprezentată în binar;
- ▶ Modificarea unui bit poate schimba suma foarte mult (al 11 lsb schimbă suma cu mai mult de 1000\$);
- ▶ Același atac se poate aplica și la OTP.

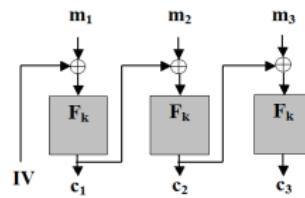
# Criptare vs. autentificarea mesajelor

## ► Criptarea folosind sisteme bloc

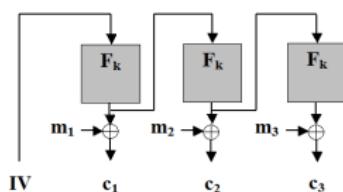
► Intrebare: Atacul de mai sus se poate aplica și pentru sistemele bloc cu modurile de operare studiate?



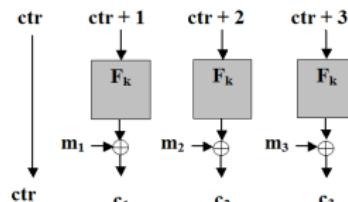
(a) ECB



(b) CBC



(c) OFB



(d) CTR

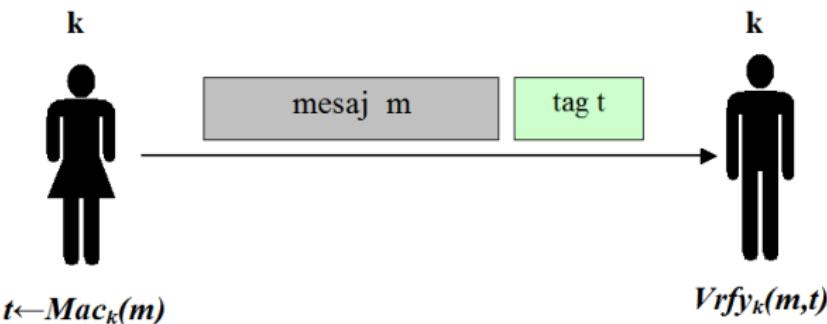
## Criptare vs. autentificarea mesajelor

- ▶ **Răspuns:** Atacul se aplică identic pentru modurile OFB și CTR;
- ▶ Pentru modul ECB, modificarea unui bit din al  $i$ -lea bloc criptat afectează numai al  $i$ -lea bloc clar, dar este foarte greu de prezis efectul exact;
- ▶ Mai mult, ordinea blocurilor la ECB poate fi schimbată;
- ▶ Pentru modul CBC, schimbarea bitului  $j$  din IV va schimba bitul  $j$  din primul bloc;
- ▶ Toate celelalte blocuri de text clar rămân neschimbate ( $m_i = F_k^{-1}(c_i) \oplus c_{i-1}$  iar blocurile  $c_i$  și  $c_{i-1}$  nu au fost modificate).

## Coduri de autentificare a mesajelor - MAC

- ▶ Așa cum am vazut, criptarea nu rezolvă problema autentificării mesajelor;
- ▶ Vom folosi un mecanism diferit, numit **cod de autentificare a mesajelor - MAC (Message Authentication Code)**;
- ▶ Scopul lor este de a împiedica un adversar să modifice un mesaj trimis fără ca părțile care comunică să nu detecteze modificarea;
- ▶ Vom lucra în continuare în contextul criptografiei cu cheie secretă unde părțile trebuie să prestablească de comun acord o cheie secretă.

## MAC - Definiție



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

## MAC - Definiție

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

# MAC - Definiție

## Definiție

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

1.  $\text{Mac} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$  este algoritmul de generare a tag-urilor  
 $t \leftarrow \text{Mac}_k(m);$
2.  $\text{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$   
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.î :  $\forall m \in \mathcal{M}, k \in \mathcal{K} \text{ } \text{Vrfy}_k(m, \text{Mac}_k(m)) = 1.$

## Securitate MAC - discuție

- ▶ Intuiție: nici un adversar polinomial nu ar trebui să poată genera un tag valid pentru nici un mesaj "nou" care nu a fost deja trimis (și autentificat) de părțile care comunică;
- ▶ Trebuie să definim puterea adversarului și ce înseamnă spargerea sau un atac asupra securității;
- ▶ Adversarul lucrează în timp polinomial și are acces la mesajele trimise între părți împreună cu tag-urile aferente.
- ▶ Adversarul poate influența conținutul mesajelor (direct sau indirect), fiind deci un adversar *activ*.

## Securitate MAC - formalizare

- ▶ Formal, îi 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  $t$  așa 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  $m$ .

## Securitate MAC - formalizare

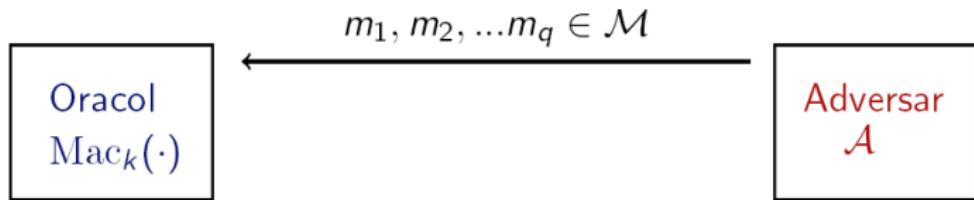
- ▶ Despre un MAC care satisface nivelul de securitate de mai sus spunem că *nu poate fi falsificat printr-un atac cu mesaj ales*;
- ▶ Aceasta înseamnă că un adversar nu este capabil să falsifice un tag valid pentru nici un mesaj ...
- ▶ ... deși poate obține tag-uri pentru orice mesaj ales de el, chiar *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  $n$ ;

# Experimentalul Mac $_{\mathcal{A}, \pi}^{forge}(n)$

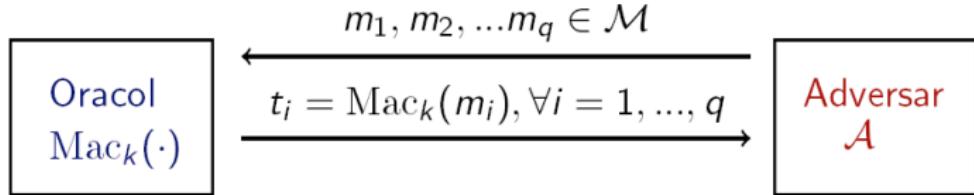
Oracol  
Mac $_k(\cdot)$

Adversar  
 $\mathcal{A}$

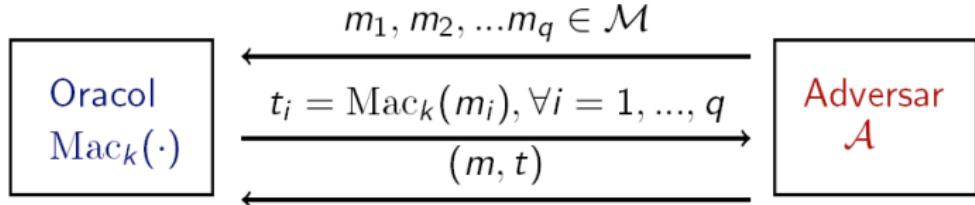
# Experimentalul Mac $_{\mathcal{A}, \pi}^{forge}(n)$



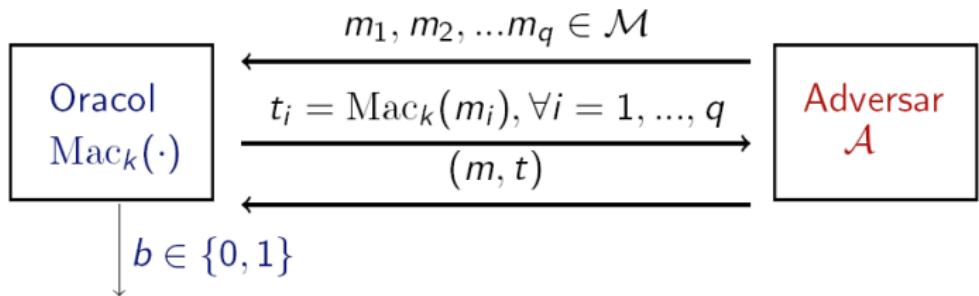
# Experimentalul Mac $_{\mathcal{A}, \pi}^{forge}(n)$



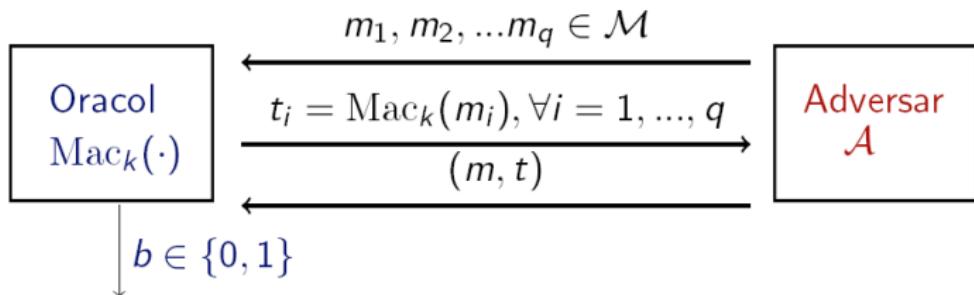
# Experimentalul Mac $_{\mathcal{A}, \pi}^{forge}(n)$



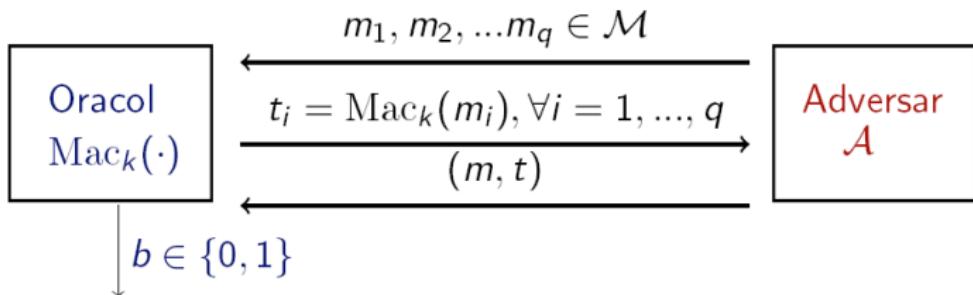
# Experimentalul Mac $_{\mathcal{A}, \pi}^{forge}(n)$



# Experimentalul Mac $_{\mathcal{A}, \pi}^{forge}(n)$



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



- ▶ 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  $\text{Mac}_{\mathcal{A}, \pi}^{forge}(n) = 1$ , spunem că  $\mathcal{A}$  a efectuat experimentul cu succes.

# Securitate MAC

## Definiție

Un cod de autentificare al mesajelor  $\pi = (\text{Mac}, \text{Vrfy})$  este sigur (nu poate fi falsificat printr-un atac cu mesaj ales) dacă pentru orice adversar polinomial  $\mathcal{A}$  există o funcție neglijabilă  $\text{negl}$  așa încât

$$\Pr[\text{Mac}_{\mathcal{A}, \pi}^{\text{forge}}(n) = 1] \leq \text{negl}(n).$$

## Atacuri prin replicare

- ▶ **Întrebare:** De ce este necesară a două condiție de la securitatea MAC (un adversar nu poate întoarce un mesaj pentru care anterior a cerut un tag)?
- ▶ **Răspuns:** Pentru a evita atacurile prin replicare în care un adversar copiază un mesaj împreună cu tag-ul aferent trimise de părțile comunicante;
- ▶ **Întrebare:** Definiția MAC oferă protecție la atacurile prin replicare efectuate chiar de părțile comunicante?
- ▶ **Răspuns:** NU! MAC-urile nu oferă nici un fel de protecție la atacurile prin replicare efectuate de părțile comunicante.

## Atacuri prin replicare

- ▶ **De exemplu:** Alice trimită către banca sa un ordin de transfer a 1.000\$ din contul ei în contul lui Bob;
- ▶ Pentru aceasta, Alice calculează un tag MAC și îl atașază mesajului aşa încât banca știe că mesajul este autentic;
- ▶ Dacă MAC-ul este sigur, Bob nu va putea intercepta mesajul și modifica suma la 10.000\$;
- ▶ Dar Bob poate intercepta mesajul și îl poate replica de zece ori către bancă;
- ▶ Dacă banca îl acceptă, Bob va avea în cont 10.000\$.

## Atacuri prin replicare

- ▶ Un MAC nu protejează împotriva unui atac prin replicare pentru că definiția nu încorporează nici o noțiune de *stare* în algoritmul de verificare;
- ▶ Mai degrabă, protecția împotriva replicării trebuie făcută la nivel înalt de către aplicațiile care folosesc MAC-uri;
- ▶ Două tehnici comune de protejare împotriva atacurilor prin replicare folosesc secvențe de numere sau *stampilă de timp*;
- ▶ Pentru secvențe de numere, fiecare mesaj  $m$  are asignat un număr  $i$  iar tag-ul este calculat pe mesajul  $i||m$ ;
- ▶ **Întrebare:** De ce această tehnică protejează împotriva atacurilor prin replicare?

## Atacuri prin replicare

- ▶ **Răspuns:** Pentru că orice nouă replicare a lui  $m$  trebuie să construiască un tag pentru mesajul  $i' || m$ , unde  $i'$  nu a mai fost folosit niciodată;
- ▶ Dezavantajul este că trebuie stocată o listă cu numerele folosite anterior
- ▶ O alternativă ar fi folosirea stampilelor de timp: la receptia mesajului, destinatarul verifică dacă stampila de timp inclusă se află într-un interval de timp acceptabil;
- ▶ Aceasta presupune ca partile să aibă ceasuri sincronizate;
- ▶ În plus, dacă un atac prin replicare este suficient de rapid, el se poate desfășura cu succes chiar și în aceste condiții.

# Constructia MAC-urilor sigure

- ▶ Funcțiile 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 în felul următor:

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

# Construcția MAC-urilor sigure

## Teoremă

Dacă  $F$  este o funcție aleatoare, construcția de mai sus reprezintă un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales).

## Demonstrație intuitivă

- ▶ Dacă un tag  $t$  este obținut prin aplicarea unei funcții pseudoaleatoare pe un mesaj  $m$ , atunci falsificarea unui tag aferent unui mesaj ne-autentificat anterior presupune ca adversarul să ghicească valoarea funcției într-un "punct nou" (i.e. mesaj);
- ▶ Probabilitatea de a ghici valoarea unei funcții aleatoare într-un punct nou este  $2^{-n}$  (unde  $n$  este lungimea ieșirii funcției);
- ▶ Prin urmare, probabilitatea de a ghici valoarea într-un punct nou pentru o funcție pseudoaleatoare nu poate fi decât neglijabil mai mare.

## MAC-uri pentru mesaje de lungime variabilă

- ▶ Construcția prezentată anterior funcționează doar pe mesaje de lungime fixă;
- ▶ 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:

## MAC-uri pentru mesaje de lungime variabilă

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.î. XOR-ul blocurilor nu se schimbă, el obținând un tag valid pentru un mesaj nou;

## MAC-uri pentru mesaje de lungime variabilă

2. Autentificare separată pentru fiecare bloc:

$$(t_1, \dots, t_d), \text{ unde } t_i = \text{Mac}'_k(m_i)$$

- ▶ **Intrebare:** Este sigură această metodă?
- ▶ **Răspuns:** NU! Un adversar poate schimba ordinea blocurilor în mesajul  $m$ , el obținând un tag valid pentru un mesaj nou;

## MAC-uri pentru mesaje de lungime variabilă

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

$$(t_1, \dots, t_d), \text{ unde } t_i = \text{Mac}'_k(i||m_i)$$

- ▶ **Intrebare:** Este sigură această metodă?
- ▶ **Răspuns:** NU! Un adversar poate scoate blocuri de la sfârșitul mesajului:  $(t_1, \dots, t_{d-1})$  este un tag valid pentru mesajul  $(m_1, \dots, m_{d-1})$ ;  
Mai mult, dacă  $(t_1, \dots, t_d)$  și  $(t'_1, \dots, t'_d)$  sunt tag-uri valide pentru mesajele  $m = m_1, \dots, m_d$  și  $m' = m'_1, \dots, m'_d$ , atunci  $(t_1, t'_2, t_3, t'_4, \dots)$  este un tag valid pentru mesajul  $m_1, m'_2, m_3, m'_4, \dots$

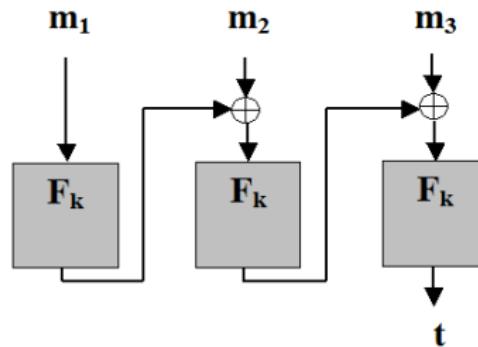
## MAC-uri pentru mesaje de lungime variabilă

- ▶ O soluție pentru atacurile anterioare o reprezintă adăugarea de informație suplimentară în fiecare bloc, în afara numărului de secvență:
  - ▶ un identificator aleator de mesaj - previne combinarea blocurilor din mesaje diferite
  - ▶ lungimea mesajului - previne modificarea lungimii mesajelor

- ▶ Soluția este ineficientă și greu de folosit în practică;
- ▶ Însă, am văzut că putem construi MAC-uri sigure (chiar pentru mesaje de lungime variabilă) pe baza funcțiilor pseudoaleatoare (intrare de lungime fixă);
- ▶ Ceea ce înseamnă că putem construi MAC-uri sigure pornind de la cifruri bloc;
- ▶ Dar, cu construcția de mai sus, rezultatul e foarte ineficient: pentru un tag aferent unui mesaj de lungime  $l \cdot n$ , trebuie să aplicăm sistemul bloc de  $4l$  ori iar tag-ul rezultat are  $(4l + 1)n$  biți;

## 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.



## Definiție

Fie  $F$  o funcție pseudoaleatoare. Un CBC-MAC este format dintr-o pereche de algoritmi polinomiali probabilisti (Mac, Vrfy):

1. Mac: pentru o cheie  $k \in \{0, 1\}^n$  și un mesaj  $m$  de lungime  $l$ :
  - ▶ Sparge  $m$  în  $m = m_1, \dots, m_l$ ,  $|m_i| = n$  și notează  $t_0 = 0^n$ ;
  - ▶ Pentru  $i = 1, \dots, l$ , calculează  $t_i = F_k(t_{i-1} \oplus m_i)$ ;

întoarce  $t_l$  ca tag-ul rezultat;
2. Vrfy : pentru o cheie  $k \in \{0, 1\}^n$ , un mesaj  $m$  de lungime  $l$ , și un tag  $t$  de lungime  $n$ :  
întoarce 1 dacă și numai dacă  $t = \text{Mac}_k(m)$ .

Rămâne valabilă condiția de corectitudine:

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

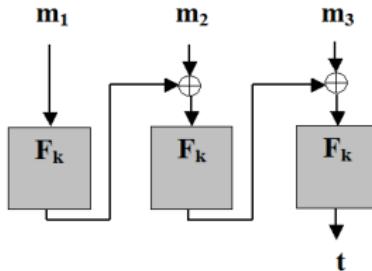
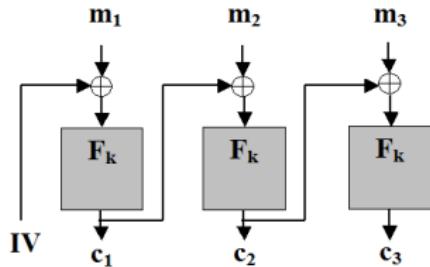
# Securitatea CBC-MAC

## Teoremă

*Dacă  $F$  este o funcție pseudoaleatoare, construcția de mai sus reprezintă un cod de autentificare a mesajelor sigur (nu poate fi falsificat prin atacuri cu mesaj ales) pentru mesaje de lungime  $l \cdot n$ .*

- ▶ Construcția prezentată este sigură numai pentru autentificarea mesajelor de lungime fixă;
- ▶ Avantajul acestei construcții față de cea anterioară este că ea poate autentifica mesaje de lungime mult mai mare;

# CBC-MAC vc. Criptare în mod CBC



## Criptare în mod CBC

- ▶ /IV 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)

## CBC-MAC pentru mesaje de lungime variabilă

Putem modifica construcția anterioară în diverse moduri ca să obținem o versiune de CBC-MAC pentru mesaje de lungime variabilă. Iată trei dintre ele care pot fi demonstreate ca fiind sigure:

1. Calculează  $k_I = F_k(I)$ ; Apoi folosește CBC-MAC cu cheia  $k_I$ ; aceasta asigură faptul că sunt folosite chei diferite pentru a autentifica mesaje de lungimi diferite;
2. Se adaugă un bloc de mesaj (în fața primului bloc) care conține  $|m|$  și se aplică CBC-MAC pe mesajul rezultat.
3. Se poate modifica schema astfel încât să se aleagă două chei  $k_1, k_2 \in \{0, 1\}^n$ ; se autentifică mesajul  $m$  cu CBC-MAC folosind cheia  $k_1$  și se obține  $t$  iar tag-ul rezultat va fi  $t' = F_{k_2}(t)$ .

## Important de reținut!

- ▶ MAC-urile oferă două proprietăți importante de securitate: integritatea mesajelor și autentificarea mesajelor;
- ▶ Pentru construcția lor se folosesc funcții pseudoaleatoare (în practică, sisteme bloc) fiind destul de rapide.



# Criptografie și Securitate

- Prelegherea 13 -

Scheme de criptare CCA sigure

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

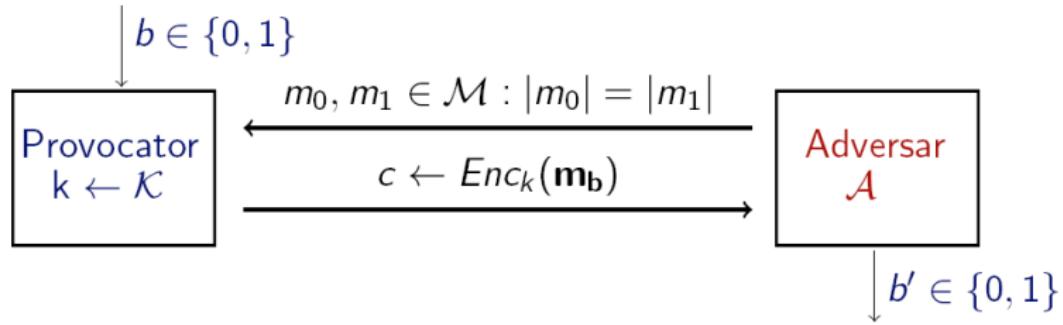
# Cuprins

1. Schemă de criptare CCA sigură - construcție
2. Schemă de criptare CCA sigură - demonstrație

## Securitate CCA

- ▶ În cursul precedent am introdus noțiunile de securitate CPA și securitate CCA;
- ▶ Multe dintre schemele prezentate până acum sunt CPA sigure (sistemele bloc împreună cu modurile de utilizare CBC, OFB și CTR);
- ▶ Însă nici una din schemele prezentate nu este CCA sigură;
- ▶ În acest curs vom folosi MAC-uri împreună cu scheme CPA sigure pentru a construi scheme CCA sigure;
- ▶ Incepem prin a reaminti noțiunea de schemă CCA sigură;

## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$



- ▶ Pe toată durata experimentului,  $\mathcal{A}$  are acces la oracolul de criptare  $Enc_k(\cdot)$  și la oracolul de decriptare  $Dec_k(\cdot)$  cu restricția că nu poate decripta  $c$ !

## Experimentul $Priv_{\mathcal{A}, \pi}^{cca}(n)$

### Definiție

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

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

## Schemă de criptare CCA sigură

- ▶ Cele două părți comunicante partajează două chei secrete, una pentru schema de criptare CPA sigură și încă una pentru un cod de autentificare a mesajelor (MAC).
- ▶ Pentru criptarea unui mesaj  $m$ , Alice procedează astfel:
  - ▶ criptează  $m$  folosind schema CPA sigură, rezultând textul criptat  $c$
  - ▶ calculează un tag MAC  $t$  pe textul criptat  $c$
  - ▶ rezultatul final al criptării este  $\langle c, t \rangle$
- ▶ Pentru un text criptat  $\langle c, t \rangle$ , Bob verifică validitatea tag-ului înainte de a decripta;
- ▶ Un text criptat  $\langle c, t \rangle$  este *valid* dacă  $t$  este un tag valid pentru  $c$ .

# O schemă de criptare CCA sigură

## Construcție

Fie  $\Pi_E = (\text{Enc}, \text{Dec})$  o schemă de criptare cu cheie secretă și

$\Pi_M = (\text{Mac}, \text{Vrfy})$  un cod de autentificare a mesajelor.

Definim schema de criptare  $(\text{Enc}', \text{Dec}')$  astfel:

- ▶ Enc: pentru o cheie  $(k_1, k_2)$  și un mesaj  $m$ , calculează  $c = \text{Enc}_{k_1}(m)$  și  $t = \text{Mac}_{k_2}(c)$  și întoarce textul criptat  $\langle c, t \rangle$ ;
  - ▶ Dec: pentru o cheie  $(k_1, k_2)$  și un text criptat  $\langle c, t \rangle$ , verifică dacă  $\text{Vrfy}_{k_2}(c, t) = 1$ . În caz afirmativ, întoarce  $\text{Dec}_{k_1}(c)$ , altfel întoarce  $\perp$ .
- 
- ▶ Simbolul  $\perp$  indică eșec;
  - ▶ Corectitudinea schemei cere ca  $\text{Dec}_{k_1, k_2}(\text{Enc}_{k_1, k_2}(m)) \neq \perp$ .
  - ▶ Spunem că  $(\text{Mac}, \text{Vrfy})$  are tag-uri unice dacă  $\forall m \forall k \exists$  un unic tag  $t$  a.î.  $\text{Vrfy}_k(m, t) = 1$ .

# O schemă de criptare CCA sigură

## Teoremă

*Dacă schema de criptare  $\Pi_E$  este CPA-sigură și  $\Pi_M$  este un MAC sigur cu tag-uri unice, atunci construcția precedentă reprezintă o schemă de criptare CCA-sigură.*

## Demonstrație intuitivă

- ▶ Un text criptat  $\langle c, t \rangle$  este valid (în raport cu o cheie  $(k_1, k_2)$ ) dacă  $\text{Vrfy}_{k_2}(c, t) = 1$ ;
- ▶ Mesajele pe care adversarul  $\mathcal{A}$  le trimite către oracolul de decriptare sunt de 2 feluri:
  - ▶ texte criptate pe care  $\mathcal{A}$  le-a primit de la oracolul de criptare (știe deja textul clar, deci nu îi sunt de folos);
  - ▶ texte criptate pe care nu le-a primit de la oracolul de criptare;
- ▶ Însă, cum  $\Pi_M$  este un MAC sigur, cu probabilitate foarte mare textele criptate care nu au fost obținute de la oracolul de criptare sunt invalide, iar oracolul de decriptare va întoarce  $\perp$  în acest caz;
- ▶ Cum oracolul de decriptare este inutil, securitatea schemei  $(\text{Enc}', \text{Dec}')$  se reduce la securitatea CPA a schemei  $\Pi_E$ .

## Important de reținut!

- ▶ Schemă de criptare CPA-sigură și MAC sigur aplicat pe textul criptat (*encrypt then MAC*)  $\Rightarrow$  schemă de criptare CCA sigură



# Criptografie și Securitate

- Prelegherea 14 -

## Confidențialitate și autentificare a mesajelor

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Transmitere sigură a mesajelor
2. Abordări diferite pentru a combina criptarea și autentificarea

## Confidențialitate și integritate

- ▶ Am văzut cum putem obține confidențialitate folosind scheme de criptare;
- ▶ Am văzut cum putem garanta integritatea datelor folosind MAC-uri;
- ▶ În practică avem nevoie de ambele proprietăți de securitate: confidențialitate și integritatea datelor;
- ▶ Nu orice combinație de schemă de criptare sigură și MAC sigur oferă cele două proprietăți de securitate!

# Confidențialitate și integritate

- ▶ 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  $\perp$ .

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

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

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

## Confidențialitate și integritate

3. *Criptare-apoi-autentificare*: întâi se criptează mesajul și apoi se calculează tag-ul

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

La recepție, se verifică întâi  $t$  înainte de a decripta  $c$ ; aceasta este chiar construcția pentru schema CCA-sigură.

- ▶ Vom analiza fiecare abordare la instanțierea cu o schemă de criptare CPA-sigură și un MAC sigur (cu tag-uri unice).
- ▶ Ne vor interesa doar acele abordări care oferă confidențialitate și integritate pentru *orice* schemă de criptare sigură și *orice* MAC sigur.

## Securitate

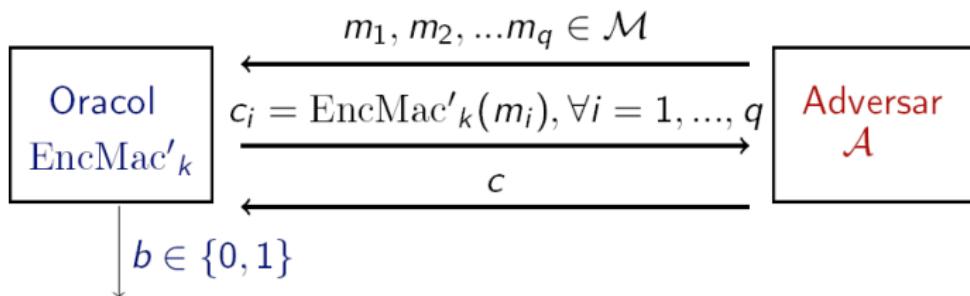
- ▶ Pentru a analiza care combinație de confidențialitate și integritate este sigură, definim ce înseamnă "combinăție sigură";
- ▶ Introducem noțiunea de *schemă de transmitere a mesajelor*
- ▶ Fie  $\Pi_E = (\text{Enc}, \text{Dec})$  o schemă de criptare arbitrară și  $\Pi_M = (\text{Mac}, \text{Vrfy})$  un cod de autentificare a mesajelor. O *schemă de transmitere a mesajelor*  $\Pi' = (\text{EncMac}', \text{Dec}')$  constă din următorii algoritmi:
  - ▶ EncMac - algoritm de transmitere a mesajelor care pentru o cheie  $(k_1, k_2)$  și un mesaj  $m$ , întoarce o valoare  $c$  derivată prin aplicarea unei combinații a algoritmilor  $\text{Enc}_{k_1}$  și  $\text{Mac}_{k_2}$ ;
  - ▶ Dec - algoritm de decriptare care pentru o cheie  $(k_1, k_2)$  și un mesaj transmis  $c$ , aplică o combinație a algoritmilor  $\text{Dec}_{k_1}$  și  $\text{Vrfy}_{k_2}$ , întorcând un text clar  $m$  sau simbolul  $\perp$  de eroare.

# Securitate

- Corectitudinea schemei cere ca  $\forall n \forall (k_1, k_2) \forall m \in \{0, 1\}^*$

$$\text{Dec}'_{k_1, k_2}(\text{EncMac}'_{k_1, k_2}(m)) = m$$

- Pentru a defini securitatea unei astfel de scheme, folosim un experiment  $\text{Auth}_{\mathcal{A}, \Pi}(n)$ :



- Output-ul experimentului este 1 dacă și numai dacă:
  - (1)  $m \neq \perp$  și (2)  $m \notin \{m_1, \dots, m_q\}$  unde  $m = \text{Dec}'_k(c)$ ;

## Definiție

O schemă de transmitere a mesajelor  $\Pi'$  oferă comunicare autentificată dacă pentru orice adversar polinomial  $\mathcal{A}$  există o funcție neglijabilă negl aşa încât

$$\Pr[\text{Auth}_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n).$$

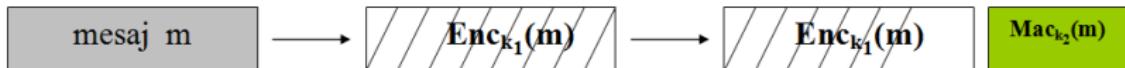
## Definiție

O schemă de transmitere a mesajelor  $\Pi'$  este sigură dacă este o schemă de criptare CCA-sigură și oferă comunicare autentificată.

# Securitate Criptare-și-autentificare

- ▶ Revenim la cele trei combinații de criptare și autentificare:

## 1. Criptare-și-autentificare (*Encrypt AND Authenticate*)



- ▶ Combinarea aceasta **nu** este neapărat sigură; un MAC sigur nu implică nici un fel de confidențialitate;
- ▶ Dacă  $(\text{Mac}, \text{Vrfy})$  este un MAC sigur atunci și schema definită de  $\text{Mac}'_k(m) = (m, \text{Mac}_k(m))$  este un MAC sigur dar dezvăluie mesajul  $m$

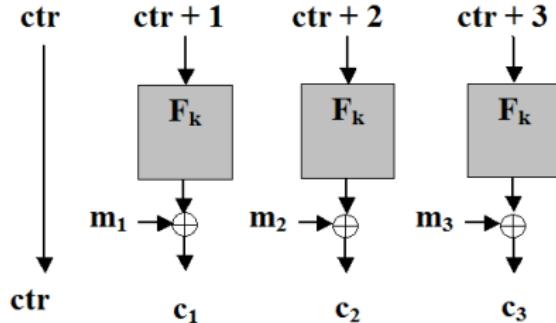
# Securitate Autentificare-apoi-criptare

## 2. Autentificare-apoi-criptare (Authenticate THEN Encrypt)



- ▶ 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;
- ▶ Definim Transform( $m$ ) astfel:
  - ▶ orice 0 din  $m$  se transformă în 00;
  - ▶ orice 1 din  $m$  se transformă arbitrar în 01 sau 10;

# Securitate Autentificare-apoi-criptare



- Definim  $\text{Enc}_k(m) = \text{Enc}'_k(\text{Transform}(m))$  unde  $\text{Enc}'$  reprezintă criptare în modul CTR folosind o funcție pseudoaleatoare;

## Securitate Autentificare-apoi-criptare

- ▶ Arătăm că o combinație de tipul autentificare-apoi-criptare a schemei de criptare de mai sus cu *orice* MAC nu este sigură la un atac de tip CCA.
- ▶ Atacul funcționează atâtă timp cât un adversar poate verifica dacă un text criptat dat este valid;
- ▶ Fiind dată o provocare  $c = \text{Enc}'_{k_1}(\text{Transform}(m||\text{Mac}_{k_2}(m)))$ , atacatorul modifică primii doi biți din al 2-lea bloc al lui  $c$  și verifică dacă rezultatul este valid;
- ▶ Dacă primul bit al mesajului clar  $m$  este 1, atunci  $c$  modificat este valid;
- ▶ **Intrebare:** De ce?

## Securitate Autentificare-apoi-criptare

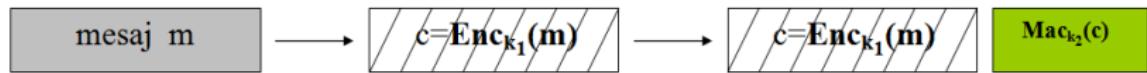
- ▶ Pentru că în acest caz, primii doi biți ai lui  $\text{Transform}(m)$  sunt 01 sau 10 și o modificare a lor oferă o codificare validă a lui  $m$ .
- ▶ Tag-ul rămâne valid pentru că este aplicat pe  $m$ ;
- ▶ Dacă însă primul bit al lui  $m$  este 0, c modificat nu este valid...
- ▶ ... pentru că primii doi biți din  $\text{Transform}(m)$  sunt 00 și prin complementare devin 11;
- ▶ Atacul poate fi aplicat pe fiecare bit din  $m$ , recuperând astfel întreg mesajul  $m$ .

## Securitate Autentificare-apoi-criptare

- ▶ Totusi, anumite instanțieri ale acestei combinații pot fi sigure;
- ▶ O astfel de combinație este folosită și în SSL și CCMP (WPA2).

# Securitate Criptare-apoi-autentificare

## 3. Criptare-apoi-autentificare (Encrypt THEN Authenticate)



- ▶ Combinăția 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;

## Necesitatea de a folosi chei diferite

- ▶ Pentru scopuri diferite de securitate trebuie să folosim întotdeauna chei diferite;
- ▶ Să urmărim ce se întâmplă dacă folosim metoda criptare-apoi-autentificare atunci când folosim aceeași cheie  $k$  atât pentru criptare cât și pentru autentificare;
- ▶ Definim  $\text{Enc}_k(m) = F_k(m||r)$ , pentru  $m \in \{0, 1\}^{n/2}$ ,  
 $r \xleftarrow{R} \{0, 1\}^{n/2}$ , iar  $F_k(\cdot)$  o permutare pseudoaleatoare;
- ▶ Definim  $\text{Mac}_k(c) = F_k^{-1}(c)$ ;
- ▶ Schema de criptare și MAC-ul sunt sigure dar...
  - ▶  $\langle \text{Enc}_k(m), \text{Mac}_k(\text{Enc}_k(m)) \rangle = \langle F_k(m||r), F_k^{-1}(F_k(m||r)) \rangle = \langle F_k(m||r), m||r \rangle$ .

## Important de reținut!

- ▶ Metoda sigură de a combina criptarea și autentificarea este *criptare-apoi-autentificare*;
- ▶ Este important să se folosească chei simetrice diferite pentru a atinge scopuri diferite (criptare și autentificare).



# Criptografie și Securitate

## - Prelegherea 15 - Funcții hash

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definiție
2. Securitatea funcțiilor hash
3. Atacul "zilei de naștere"
4. Transformarea Merkle-Damgård

# Funcții Hash

- ▶ **Întrebare:** Ați auzit vreodată de funcții hash? În ce context?
- ▶ Acestea primesc ca argument o secvențe de *lungime variabilă* pe care o **comprimă** într-o secvențe de *lungime mai mică, fixată*;
- ▶ Utilizarea clasică a funcțiilor hash este în domeniul *structurilor de date*;
- ▶ Să luăm un exemplu...

# Funcții Hash

- ▶ Considerăm o mulțime de elemente de dimensiune mare care trebuie stocată (într-un tablou);

---

Adresă

Batiștei nr.17

Academiei nr.23

Tudor Arghezi nr.103

Nicolae Bălcescu nr.10

C.A.Rosetti nr.7

Hristo Botev nr.35

...

---

- ▶ Ulterior, elementele trebuie să fie ușor accesibile;
- ▶ **Întrebare:** Cum procedăm?

# Funcții Hash

- ▶ **Varianta 1.** Elementele se stochează la rând;

Index	Adresă
1	Batiștei nr.17
2	Academiei nr.23
3	Tudor Arghezi nr.103
4	Nicolae Bălcescu nr.10
5	C.A.Rosetti nr.7
6	Hristo Botev nr.35
...	.....

- ▶ Dar o căutare necesită un algoritm de complexitate ...  $O(n)$  ;

# Funcții Hash

- **Varianta 2.** Tabloul de elemente este sortat.

Index	Adresă
...	.....
17	Academiei nr.23
...	.....
120	Batiștei nr.17
...	.....
223	C.A.Rosetti nr.7
...	.....
401	Hristo Botev nr.35
...	.....
503	Nicolae Bălcescu nr.10
...	.....
696	Tudor Arghezi nr.103
...	.....

- În acest caz o căutare necesită un algoritm de complexitate ...  
 $O(\log n)$  ;

# Funcții Hash

- ▶ **Varianta 3.** Se folosește o **funcție hash**  $H$  și fiecare element  $x$  se stochează la indexul  $H(x)$ ;

Index	Adresă
...	...
14	Tudor Arghezi nr.103
...	...
29	Batiștei nr.17
...	...
113	C.A.Rosetti nr.7
...	...
365	Academiei nr.23
...	...
411	Nicolae Bălcescu nr.10
...	...
703	Hristo Botev nr.35
...	...

- ▶ Căutarea devine optimă pentru că se realizează în ...  $O(1)!$

## Funcții Hash

- ▶ În exemplul precedent:
  - ▶  $H("Tudor Arghezi nr.103") = 14;$
  - ▶  $H("Batiștei nr.17") = 29;$
  - ▶ ...
- ▶ Analizăm, pe rând, cele 3 operații: **căutare, adăugare, stergere**;
- ▶ Pentru **căutarea** adresei *Edgar Quinet nr.35*, se calculează  $H("Edgar Quinet nr.35")$ ;
- ▶ Presupunând că  $H("Edgar Quinet nr.35") = 79$ , atunci se verifică ce este stocat pe poziția 79;

## Funcții Hash

- ▶ Pentru **introducerea** adresei *Nicolae Filipescu nr.31*, se calculează  $H("Nicolae Filipescu nr.31")$ ;
- ▶ Presupunând că  $H("Nicolae Filipescu nr.31") = 153$ , atunci se introduce valoarea *Nicolae Filipescu nr.31* la indexul 153;
- ▶ Pentru **ștergerea** adresei *Hristo Botev nr.35*, se calculează  $H("Hristo Botev nr.35")$ ;
- ▶ Se obține  $H("Hristo Botev nr.35") = 401$  și se eliberează această celulă;

## Funcții Hash

- ▶ În general, funcția hash poate fi aplicată pe orice valoare  $x'$  (în legătură cu  $x$ );
- ▶ De exemplu, se poate stoca adresa la indexul corespunzător numelui;
- ▶ În acest caz, dacă Alice locuiește la *Nicolae Filipescu nr.31* și  $H("Alice") = 254$ , atunci *Nicolae Filipescu nr.31* se stochează la indicele 254;
- ▶ Dimensiunea tabloului este întotdeauna dată de mulțimea de valori posibile ale funcției hash;
- ▶ **Întrebare:** Ce probleme identificați în exemplele date?

## Funcții Hash

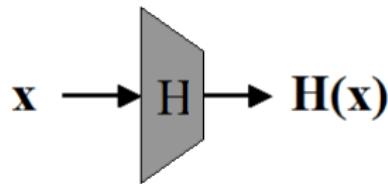
- ▶ **Răspuns:** Probleme sunt multe: adresele nu sunt mereu identic introduse, numele nu sunt unice, ...
- ▶ ... dar pe noi ne interesează următorul aspect:

## Funcții Hash

- ▶ În criptografie, o funcție hash rămâne o funcție care comprimă secvențe de lungime variabilă în secvențe de lungime fixă, însă:
- ▶ Dacă în contextul structurilor de date este *preferabil* să se minimizeze coliziunile, în criptografie acest lucru este **impus**;
- ▶ Dacă în contextul structurilor de date coliziunile apar *arbitrar* (valorile sunt alese independent de funcția hash), în criptografie coliziunile trebuie evitate chiar dacă sunt căutate **în mod voit** (de către adversar).

# Funcții Hash

- **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;



# Funcții Hash

- ▶ În aceste condiții, o funcție hash impune ca determinarea unei coliziuni să devină dificilă;
- ▶ Considerăm funcțiile hash de domeniu infinit și ieșire de lungime fixată  $l(n)$ , unde  $l(n)$  este un polinom în parametrul de securitate  $n$ :

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

## Experimentul $\text{Hash}_{\mathcal{A}, H}^{\text{coll}}(n)$

- ▶ Considerăm experimentul  $\text{Hash}_{\mathcal{A}, H}^{\text{coll}}(n)$ ;
- ▶ Adversarul  $\mathcal{A}$  indică 2 valori  $x, x' \in \{0, 1\}^*$ ;
- ▶ Se definește valoarea experimentului  $\text{Hash}_{\mathcal{A}, H}^{\text{coll}}(n) = \mathbf{1}$  dacă  $x \neq x'$  și  $H(x) = H(x')$ ;
- ▶ Altfel,  $\text{Hash}_{\mathcal{A}, H}^{\text{coll}}(n) = \mathbf{0}$ .

# Rezistență la coliziuni

## Definiție

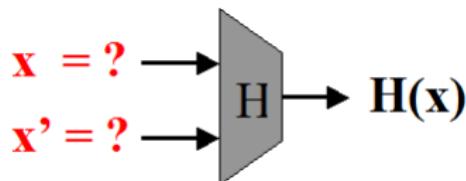
$H : \{0, 1\}^* \rightarrow \{0, 1\}^{l(n)}$  se numește **rezistentă la coliziuni** (*collision-resistant*) dacă pentru orice adversar PPT  $\mathcal{A}$  există o funcție neglijabilă  $\text{negl}$  așa încât

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

## Securitatea funcțiilor hash

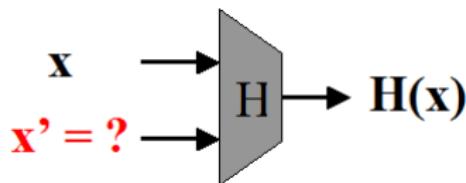
- ▶ În practică, rezistența la coliziuni poate fi dificil de obținut;
- ▶ Pentru anumite aplicații sunt utile noțiuni mai relaxate de securitate;
- ▶ Există **3 nivele de securitate**:
  1. **Rezistența la coliziuni**: este cea mai puternică noțiune de securitate și deja am definit-o formal;
  2. **Rezistența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

## Rezistență la coliziuni



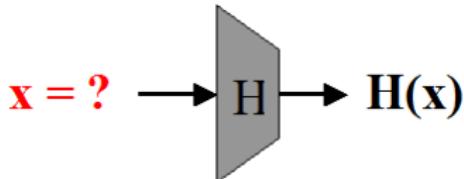
- ▶ **Provocare:** Se cer 2 valori  $x \neq x'$  a.î.  $H(x) = H(x')$ ;
- ▶ **Atac:** "Atacul zilei de naștere" necesită  $\approx 2^{l(n)/2}$  evaluări pentru  $H$ .

## Rezistență la a doua preimagine



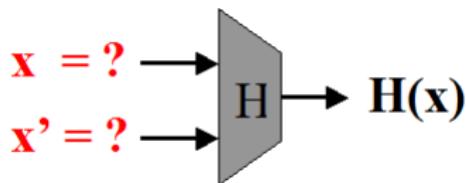
- ▶ **Provocare:** Fiind dat  $x$ , se cere  $x'$  a.î.  $H(x) = H(x')$ ;
- ▶ **Atac:** Un atac generic necesită  $\approx 2^n$  evaluări pentru  $H$ .

## Rezistență la prima preimagine



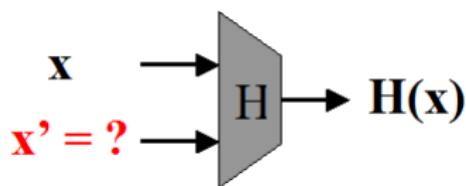
- ▶ **Provocare:** Fiind dat  $y = H(x)$ , se cere  $x$  a.î.  $H(x) = y$ ;
- ▶ O astfel de funcție se numește și *calculabilă într-un singur sens (one-way function)*;
- ▶ **Atac:** Un atac generic necesită  $\approx 2^n$  evaluări pentru  $H$ .

## Atacuri în practică



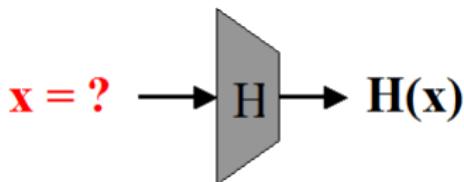
$\{x\}$  = documente originale  
 $\{x'\}$  = documente false  
cineva este de acord să semneze electronic documentul original, însă semnătura devine valabilă și pentru un document fals

---



documentul  $x$  care este semnat electronic poate fi înlocuit de documentul  $x'$

---



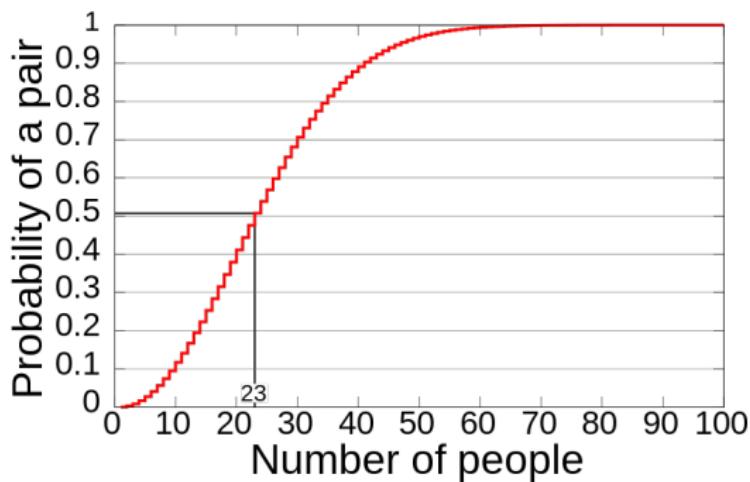
dacă se cunoaște cheia de sesiune  $k$ ; calculată din cheia master  $k$   
 $x = H(k||i)$ , atunci se determină cheia  $k$

## Securitatea funcțiilor hash

- ▶ **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 naștere"

- ▶ Analizăm posibilitatea de a determina o coliziune pornind de la un exemplu clasic: *paradoxul nașterilor*;
- ▶ Întrebare: Care este dimensiunea unui grup de oameni pentru ca 2 dintre ei să fie născuți în aceeași zi cu probabilitate  $1/2$  ?
- ▶ Răspuns: 23!



## Atacul "zilei de naștere"

- ▶ 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  $\geq 1/2$ .
- ▶ Aceast 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  $\approx 2$ .

## Atacul "zilei de naștere"

- ▶ Atacul necesită timp de rulare  $O(2^{n/2})$  și capacitate de stocare  $O(2^{n/2})$ ;
- ▶ O variantă îmbunătățită păstrează timpul necesar pentru rulare, dar micșorează spațiul de stocare la  $O(1)$ :
  - ▶ Adversarul alege  $x_0$  uniform aleator;
  - ▶ Calculează pentru fiecare  $x_i = H(x_{i-1})$  și  $x_{2i} = H(H(x_{2(i-1)}))$ ;
  - ▶ Dacă  $x_i = x_{2i}$  a găsit o coliziune ( $x_{i-1} = H(x_{2(i-1)})$  cu probabilitate neglijabilă).

## Transformarea Merkle-Damgård

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

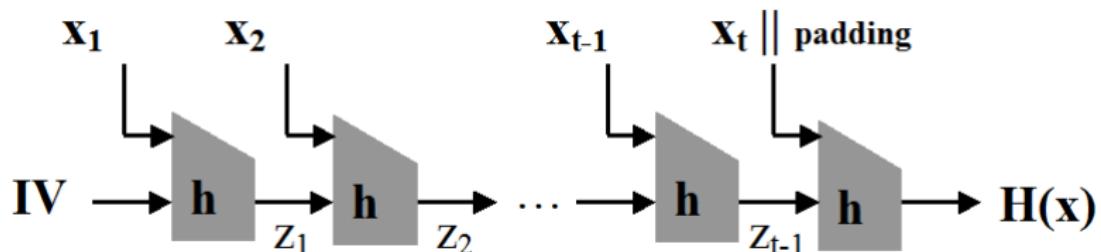
$$H_1 : \{0, 1\}^{m_1} \rightarrow \{0, 1\}^{n_1}, m_1 > n_1, m_1 \approx n_1$$

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

- ▶ **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).

## Transformarea Merkle-Damgård

- ▶ 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;



## Notății

- ▶  $h$  = o funcție de compresie de dimensiune fixă
- ▶  $x = x_1 || x_2 || \dots || x_{t-1} || x_t$  = valoarea de intrare
- ▶  $IV$  = vector de inițializare
- ▶ padding = 100...0|| lungimea mesajului

# Transformarea Merkle-Damgård

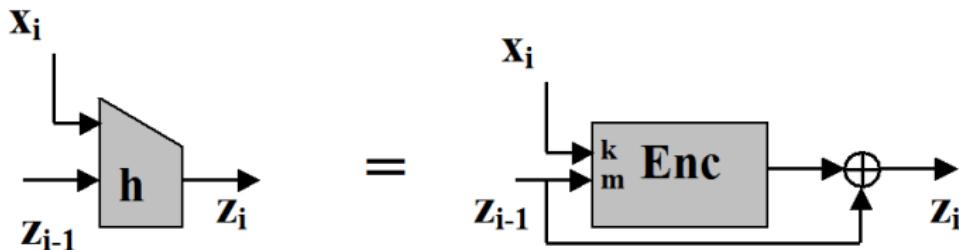
## Teoremă

Dacă  $h$  prezintă rezistență la coliziuni, atunci și  $H$  prezintă rezistență la coliziuni.

- ▶ Intuitiv, dacă există  $x \neq x'$  a.î.  $H(x) = H(x')$ , atunci trebuie să existe  $\langle z_{i-1}, x_i \rangle \neq \langle z'_{i-1}, x'_i \rangle$  a.î.  
 $h(z_{i-1} || x_i) = h(z'_{i-1} || x'_i);$
- ▶ Altfel spus, dacă se găsește o coliziune pentru  $H$  se găsește o coliziune și pentru  $h$ .

## Transformarea Merkle-Damgård

- ▶ Rămâne să prezentăm o construcție pentru  $h$ ;
- ▶ Construcția Davies-Meyer:



- ▶  $Enc$  este un sistem bloc care cripteză  $z_{i-1}$  cu cheia  $x_i$ :

$$h(z_{i-1} || x_i) = Enc_{x_i}(z_{i-1}) \oplus z_{i-1}$$

## Exemple

- ▶ **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

## Important de reținut!

- ▶ Definiția funcției hash și nivelele de securitate
- ▶ Din 23 de oameni, 2 sunt născuți în aceeași zi cu probabilitate de peste 50%
- ▶ Transformarea Merkle-Damgård



# Criptografie și Securitate

- Prelegerea 15.1 -  
Stocarea parolelor

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Stocarea parolelor
2. Atacuri
3. Salting
4. Parole de unică folosință

## Stocarea parolelor

- ▶ Cea mai utilizată metodă de autentificare este bazată pe nume de utilizator și parolă;
- ▶ Metoda presupune o etapă de înregistrare, în care utilizatorul alege un *username* și o *parolă* (*pwd*);
- ▶ Ulterior, la fiecare logare, utilizatorul introduce cele 2 valori;
- ▶ Dacă *username* se regăsește în lista de utilizatori înregistrați și parola introdusă este corectă atunci autentificarea se realizează cu succes;
- ▶ În caz contrar, autentificarea eșuează.

## Stocarea parolelor

- ▶ O greșală frecventă de implementare o reprezintă afișarea unor mesaje de tipul:

*Nume de utilizator inexistent*

*Parolă greșită!*

- ▶ **Întrebare:** De ce **nu** este indicată utilizarea unor astfel de mesaje de eroare?
- ▶ **Răspuns:** Pentru că oferă informații suplimentare adversarului!
- ▶ Corect este să se întoarcă un mesaj de eroare generic de tipul:

*Nume de utilizator sau parolă incorecte!*

## Stocarea parolelor

- ▶ Pentru autentificare, utilizatorul introduce numele de utilizator *username* și parola *pwd*;
- ▶ Sistemul de autentificare calculează  $H(pwd)$  și se verifică dacă valoarea obținută este stocată în fișierul de parole pentru utilizatorul indicat prin *username*;
- ▶ Dacă da, atunci autentificarea se realizează cu succes; în caz contrar, autentificarea eșuează;
- ▶ Funcțiile hash sunt funcții **one-way**: cunoscând  $H(pwd)$  nu se poate determina *pwd*;
- ▶ Această metodă de stocare a parolelor introduce deci avantajul că nu oferă adversarului acces direct la parole, chiar dacă acesta deține fișierul de parole.

## Atac de tip dicționar

- ▶ Adversarul poate însă să verifice, pe rând, 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 şansele unor astfel de atacuri:
  - ▶ se blochează procesul de autentificare după un anumit număr de încercări nereușite;
  - ▶ se obligă utilizatorul să folosească o parolă care satisfac anumite criterii : o lungime minimă, utilizarea a cel puțin 3 tipuri de simboluri (litere mici, litere mari, cifre, caractere speciale);
  - ▶ În caz de succes, adversarul determină parola unui singur utilizator;

## Atac folosind tabele hash precalculate

- ▶ 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 parole, 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 parolă din dicționar;

## Rainbow tables

- ▶ **Rainbow tables** introduc un compromis între capacitatea de stocare și timp;
- ▶ Se compun lanțuri de lungime  $t$ :  
$$\begin{aligned} \textit{pwd}_1 &\rightarrow^H H(\textit{pwd}_1) \rightarrow^f \textit{pwd}_2 \rightarrow^H H(\textit{pwd}_2) \rightarrow^f \dots \rightarrow^f \\ &\textit{pwd}_{t-1} \rightarrow^H H(\textit{pwd}_{t-1}) \rightarrow^f \textit{pwd}_t \rightarrow^H H(\textit{pwd}_t) \end{aligned}$$
- ▶  $f$  este o funcție de mapare a valorilor hash în parole;
- ▶ 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;

# Rainbow tables

► Algoritmul de determinare a unei parole:

1. Se caută valoarea hash a parolei 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  $f$  încr-o parolă căreia i 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, **distincă** 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.

## Salting

- ▶ 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 deși 2 utilizatori folosesc aceeași parolă, 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 parolă.

## Parole de unică folosință

- ▶ Cazul extrem de schimbare periodică a parolei îl reprezintă parolele de unică folosință;

## Parole de unică folosință

- ▶ Cazul extrem de schimbare periodică a parolei îl reprezintă parolele de unică folosință;
- ▶ Utilizatorul folosește o listă de parole, la fiecare logare utilizând următoarea parolă din listă;

## Parole de unică folosință

- ▶ La înregistrare, în fișierul de parole se păstrează tripletul:  
 $(username, 1, P_0 = H(H(H(H(x)))))$
- ▶ 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 continuă 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.

## Important de reținut!

- ▶ Nu păstrați parolele în clar!
- ▶ Utilizați mecanisme de tip salting!



# Criptografie și Securitate

- Prelegerea 15.2 -

## Message Digest 5 - MD5

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Informații generale

2. Descriere

3. Securitate

## Informații generale

MD5 este:

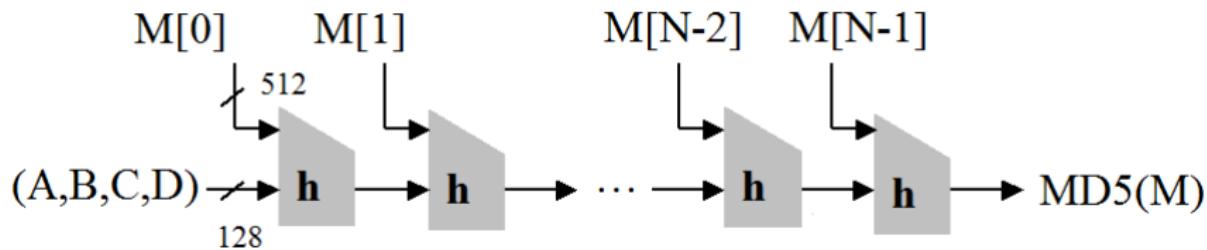
- ▶ definit în 1991 de R.Rivest ca înlocuitor pentru MD4
- ▶ publicat ca standard internet RFC1321 în 1992
- ▶ face parte dintr-o familie de funcții hash: MD2, MD4, MD6 (finalist SHA-3)
- ▶ realizat pentru calculatoarele pe 32-biți
- ▶ utilizat pentru stocarea parolelor în versiuni mai vechi de Moodle sau pentru a asigura integritatea fișierelor la download sau transfer

## Descriere

- ▶ MD5 este o funcție hash cu ieșirea pe 128 biți:

$$MD5 : \{0, 1\}^* \rightarrow \{0, 1\}^{128}$$

- ▶ Folosește construcția Merkle-Damgård pentru blocuri de 512 biți și vector de inițializare de 128 biți:



## Descriere

- ▶ Presupune 5 pași:
  - ▶ **Pas 1:** Padding
  - ▶ **Pas 2:** Concatenarea lungimii mesajului
  - ▶ **Pas 3:** Inițializarea buffer-ului MD
  - ▶ **Pas 4:** Procesarea mesajului
  - ▶ **Pas 5:** Ieșirea

## Descriere

### Pas 1. Padding

- ▶ Mesajul de intrare este spart în blocuri de 512 biți :

$$M[0], M[1], \dots, M[N - 2]$$

- ▶ Ultimul bloc  $M[N - 1]$  este completat până la 448 biți cu secvența :

100...0

- ▶ Padding-ul se realizează întotdeauna, chiar dacă (din întâmplare) ultimul bloc are exact 448 biți.

## Descriere

### Pas 2. Concatenarea lungimii mesajului

- ▶ Lungimea mesajului (în biți) se reprezintă pe 64 de biți;
- ▶ În cazul (foarte puțin probabil!) că lungimea  $\geq 2^{64}$ , se folosesc numai ultimii 64 biți din reprezentarea binară;
- ▶ Rezultatul se concatenează la ultimul bloc  $M[N - 1]$ , care devine complet (512 biți);
- ▶ Mesajul rezultat conține numai blocuri de 512 biți:

$$M[0], M[1], \dots, M[N - 1]$$

## Descriere

### Pas 3. Inițializarea buffer-ului MD

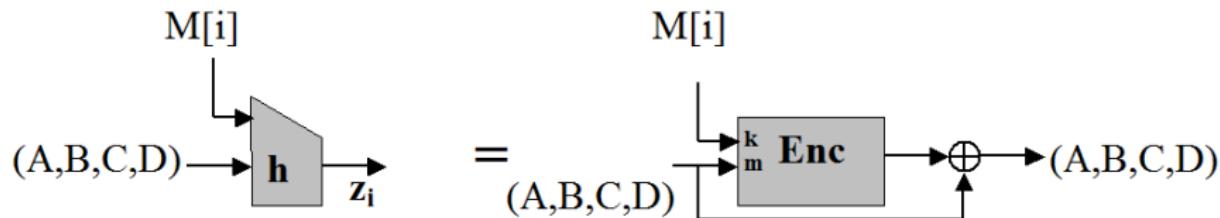
- ▶ Inițializarea construcției Merkle-Damgård necesită un vector de inițializare pe 128 biți;
- ▶ Aceasta este sub forma unui buffer de 4 word-uri ( $A, B, C, D$ ), care va prelua valori intermediare după fiecare transformare a unui bloc:
- ▶ Inițializarea se face la valorile indicate:

A	=	01	23	45	67
B	=	89	ab	cd	ef
C	=	fe	dc	ba	98
D	=	76	54	32	10

# Descriere

## Pas 4. Procesarea mesajului

- ▶ Procesarea mesajului se realizează în blocuri de câte 16 word-uri;
- ▶ Compresia este de fapt o construcție Davies-Meyer unde adunarea se face modulo  $2^{32}$ :



# Descriere

## Pas 4. Procesarea mesajului

- ▶ Se definesc 4 funcții auxiliare:

$$F(X,Y,Z) = \text{dacă } X, \text{ atunci } Y; \text{ altfel } Z$$

$$G(X,Y,Z) = \text{dacă } Z, \text{ atunci } X; \text{ altfel } Y$$

$$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$$

$$I(X,Y,Z) = Y \text{ xor } (X \text{ and } (\text{not } Z))$$

- ▶ Se definește o tabelă calculată folosind funcția *sin*:

$$T[i] = 4294967296 \cdot \text{abs}(\sin(i))$$

# Descriere

## Pas 4. Procesarea mesajului

- ▶ Funcția de criptare din construcția Davies-Meyer este constituită din 4 runde;
- ▶ Fiecare rundă folosește una dintre funcțiile auxiliare  $F, G, H, I$ ;
- ▶ Pentru exemplificare, introducem prima rundă:

```
/* Round 1. */
/* Let [abcd k s i] denote the operation
   a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
[ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]
```

[RFC1321]

## Descriere

### Pas 5. Ieșirea

- ▶ Valoarea funcției hash este dată de valoarea finală a bufferului (după aplicarea transformării Merkle-Damgård pe întreg mesajul M):

$$MD5(M) = ABCD$$

# Securitate

- ▶ "Atacul zilei de naștere" asupra MD5 necesită numai  $\approx 2^{128/2} = 2^{64}$  evaluări;
- ▶ În 1996, au fost determinate coliziuni în MD5:

*"The presented attack does not yet threaten practical applications of MD5, but it comes rather close. In view of the flexibility of the new analytic techniques it would be unwise to assume that the attack could not be improved."*

(Hans Dobbertin: The Status of MD5 After a Recent Attack, RSA Laboratories' CryptoBytes vol.2, no.2, 1996)

## Securitate

- ▶ Şi într-adevăr, MD5 devine inutilizabil în practică...
- ▶ MD5 era folosit pentru a asigura integritatea fişierelor la descărcare;
- ▶ Serverul pune la dispoziţia utilizatorului o valoare MD5 precalculată corespunzătoare fişierelor descărcate (*md5sum*);
- ▶ Utilizatorul primeşte fişierele, calculează valoarea hash MD5 şi verifică dacă este identică celei iniţiale;
- ▶ Această utilizare nu mai este sigură în practică:

*"Two researchers from the Institute for Cryptology and IT-Security have generated PostScript files with identical MD5-sums but entirely different (but meaningful!) content."*

(Bruce Schneier: Schneier on Security, 10 iunie 2005)

# Tool-uri



**hashcat**  
advanced password recovery

**Download**

Name	Version	Date	Download	Signature
hashcat binaries	v5.1.0	2018.12.02	Download	PEF
hashcat sources	v5.1.0	2018.12.02	Download	PDF

Signing key on PKG keyservers: RSA, 2048-bit. Key ID: 2048R/BA16544F. Fingerprint: A708 3322 9D04 0B41 99CC 0052 3C17 DA8B BA16 544F

Check out our [GitHub Repository](#) for the latest development version

**GPU Driver requirements:**

- AMD GPUs on Linux require "RadeonOpenCompute (ROCm)" Software Platform (1.6.180 or later)
- AMD GPUs on Windows require "AMD Radeon Software Crimson Edition" (15.12 or later)
- Intel GPUs require "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.1.1 or later)
- Intel GPUs on Linux require "OCL-ICD" (16.1.1 or later)
- Intel GPUs on Windows require "OCL-ICD" (16.1.1 or later)
- NVIDIA GPUs require "NVIDIA CUDA" (8.0 or later)

Forum      Wiki      Tools      Events      Converter      Contact

**Features**

- World's fastest password cracking
- World's first and only integrated GPU support
- Free
- Open-Source (MIT License)
- Multi-OS (Linux, Windows and more)
- Multi-Platform (CPU, GPU, DSP, etc.)
- Multi-Hash (Cracking multiple hashes at once)
- Multi-Devices (Utilizing multiple devices simultaneously)

Please list your hashes below...

Please input the MD5 hashes that you would like to look up. NOTE that the space character is replaced with [space].

Your Hashes:

```
Put your hash/text list in here! (Max: 64)
e.g.
$hashcat$0$4f0c$421f7a3b017caad8c
250cf#8fb#01c77f1f#fd#0b#0e#08#7xa#02
3921e#436ea#027f1d#4d#08#33#09#08#12
001a#404#0#7f5#0#3a#4d#5c#0#01#727#1#MF#X
60391#51#2#7#8#e#2#4#1#e#d#8#c#3#0#8#2#3#0#(:(#h#:#))#e#K#n#1#4#y#3#m#d#t#t,c#M#o#--#
4#0#d#0#1#5#4#0#8#5#e#3#5#1#5#3#2#9#e#1#f#f#i#e#h#d#b#0#e#f#
7#3#1#4#4#5#e#n#3#3#5#7#7#4#1#0#7#7#e#d#4#4#4#6#6#3#
e#3#6#d#4#4#2#9#f#e#1#4#9#f#t#4#8#9#4#d#9#2#4#2#7#e#4#1#e#4#4#4#9#b#9#4#e#9#5#9#3#1#7#6#5#2#8#5#5#
```

Crack my hashes

HashKiller.co.uk is a hash lookup service. This allows you to input an MD5 hash and search for its corresponding plaintext ("found") in our database of already-cracked hashes.

It's like having your own massive password-cracking cluster - but with immediate results!

We have been building our hash database since August 2007.

Note that we do not use terms like "decrypted", "dehashed", or "reversed" - hashes can only be looked up quickly after they've been cracked the hard way.

In other words, we are not cracking your hash in realtime - we're just caching the hard work of many cracking enthusiasts over the years.

**Output Formats:**

- Found: `Hash[:$salt] $type $pass`
- Not Found: `Hash:$salt` [No Match]
- Invalid: `Hash [Invalid]`

<https://hashcat.net/hashcat/>

<https://hashkiller.co.uk/Cracker/MD5>

## Important de reținut!

- ▶ MD5 NU este o funcție hash sigură!



# Criptografie și Securitate

- Prelegerea 15.3 -  
SHA - 3

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Competiția SHA-3

2. Keccak

# Competiția SHA-3

- ▶ Atacurile asupra MD5, SHA-0, SHA-1 au impus necesitatea unei noi funcții hash;

## Google Security Blog

The latest news and insights from Google on security and safety on the Internet

---

### Announcing the first SHA1 collision

February 23, 2017

Posted by Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam),  
Ange Albertini (Google), Yarik Markov (Google), Alex Petit Bianco (Google), Clement Baisse (Google)

Cryptographic hash functions like SHA-1 are a cryptographer's swiss army knife. You'll find that hashes play a role in browser security, managing code repositories, or even just detecting duplicate files in storage. Hash functions compress large amounts of data into a small message digest. As a cryptographic requirement for wide-spread use, finding two messages that lead to the same digest should be computationally infeasible. Over time however, this requirement can fail due to [attacks on the mathematical underpinnings](#) of hash functions or to increases in computational power.

[<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>]

## Competiția SHA-3

- ▶ 2 noiembrie 2007 - NIST anunță competiția publică SHA-3;
- ▶ 31 octombrie 2008 - se primesc 64 de propuneri din toată lumea;
- ▶ decembrie 2008 - rămân 51 de candidați pentru prima rundă (restul sunt eliminați din cauza dosarelor incomplete);

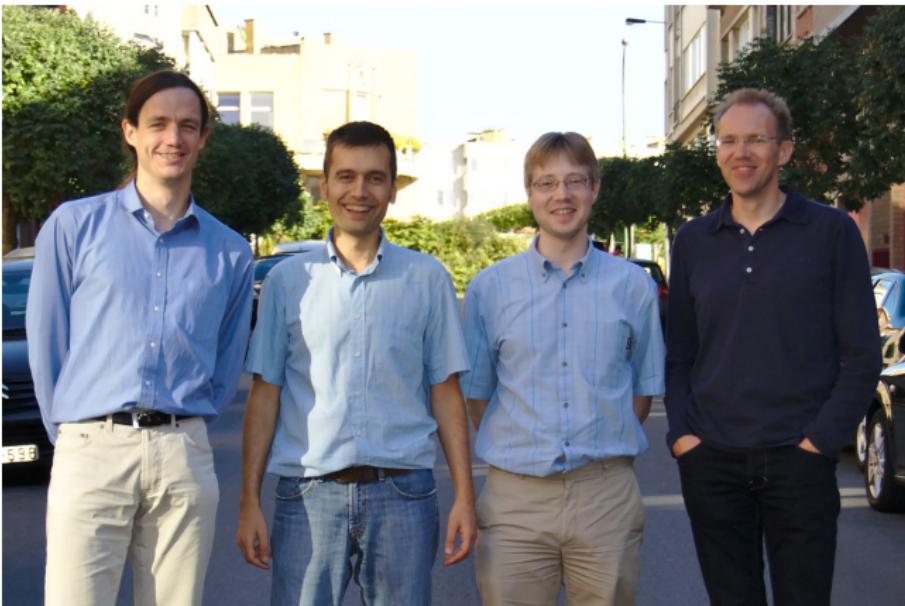
## Competiția SHA-3

- ▶ februarie 2009 - are loc prima conferință la care sunt prezentate 37 de propuneri (dintr-un total de 41, 10 fiind retrase într-o perioadă din cauza unor atacuri);
- ▶ iulie 2009 - rămân 14 candidați în runda a 2-a;
- ▶ decembrie 2010 - cei 5 candidați în runda finală: BLAKE, Grøstl, JH, Keccak and Skein;
- ▶ 2 octombrie 2012 - NIST anunță câștigătorul: **Keccak**.

## Cei 5 finaliști

BLAKE	Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan
Grøstl	Lars Ramkilde Knudsen, Praveen Gauravaram, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, Søren S. Thomsen
JH	Hongjun Wu
Keccak	Joan Daemen, Guido Bertoni, Michaël Peeters, Gilles Van Assche
Skein	Bruce Schneier, Niels Ferguson, Stefan Lucks, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jesse Walker, Jon Callas

# Echipa Keccak



[<http://keccak.noekeon.org/team.html>]

## Criterii de selecție

- ▶ Lungimea secvenței de ieșire:  $n = 224, 256, 384$  și  $512$  biți;
- ▶ Alte dimensiuni ale secvenței de ieșire sunt opționale;
- ▶ Eficiență crescută față de SHA-2;
- ▶ Utilizare în HMAC;
- ▶ Rezistență la coliziuni, prima și a doua preimagine (conform cu atacurile generice, tradiționale);
- ▶ Demonstrație de securitate;
- ▶ Parametrizabilă, număr de runde variabil;
- ▶ Simplicitate, claritate.

## Motivație

*"The NIST team praised the Keccak algorithm for its many admirable qualities, including its elegant design and its ability to run well on many different computing devices. The clarity of Keccak's construction lends itself to easy analysis (during the competition all submitted algorithms were made available for public examination and criticism), and Keccak has higher performance in hardware implementations than SHA-2 or any of the other finalists."*

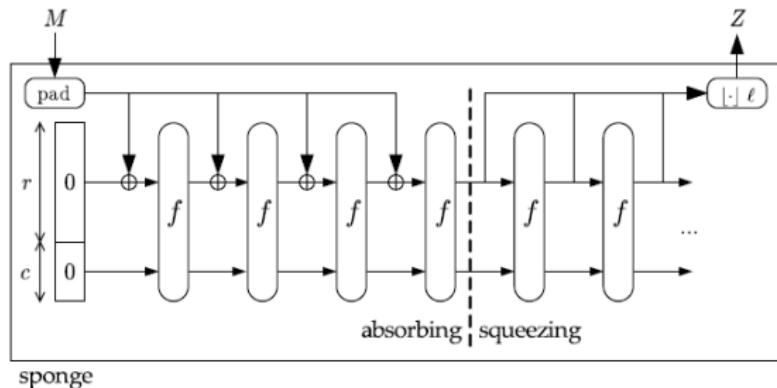
(NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition -  
<http://www.nist.gov/itl/csd/sha-100212.cfm>)

*"One benefit that KECCAK offers as the SHA-3 winner is its difference in design and implementation properties from that of SHA-2. It seems very unlikely that a single new cryptanalytic attack or approach could threaten both algorithms."*

(SHA-3 Selection Announcement - [http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3\\_selection\\_announcement.pdf](http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf))

# Keccak

- ▶ A fost gândit să difere de construcțiile existente (AES, SHA-2);
- ▶ Folosește **sponge functions**:



[Cryptographic Sponge Functions -  
<http://sponge.noekeon.org/CSF-0.1.pdf>]

# Keccak

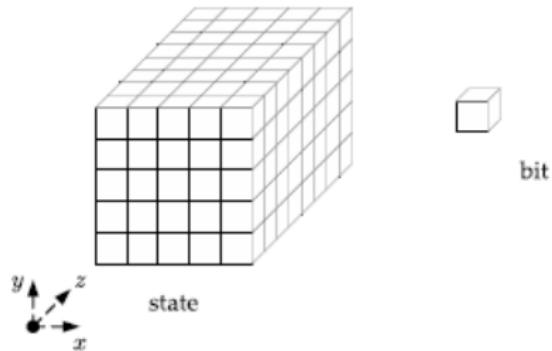
- ▶ Notații:
  - ▶  $r = \text{bitrate}$
  - ▶  $c = \text{capacity}$
  - ▶  $b = c + r = \text{width}$
  - ▶  $f = \text{o permutare}$
- ▶ Folosește o **stare** de  $b$  biți inițializată la 0;
- ▶ Presupune 2 etape:
  1. **Absorbing phase**: mesajul de intrare se sparge în blocuri de lungime  $r$  care se XOR-ează la prima parte a stării, alternând cu aplicarea funcției  $f$ ;
  2. **Squeezing phase**: partea superioară a stării este returnată la ieșire, alternând cu aplicarea funcției  $f$ ; numărul de iterații depinde de numărul de biți / necesari la ieșire.

# Keccak

- ▶ Pentru implementarea Keccak:
  - ▶  $c \leq 512$
  - ▶  $600 \leq b \leq 2400$
  - ▶  $r = b - c$
  - ▶  $f$  = o funcție non-liniară cu bune proprietăți de difuzie;
- ▶ SHA-3:
  - ▶  $c = 512$
  - ▶  $b = 1600$
  - ▶  $r = b - c = 1088$

# Keccak

- ▶ Starea este considerată o structură 3D  $5 \times 5 \times 2^l$  ( $l \in \{1, 2, 4, 8, 16, 32, 64\}$ ):



[The Keccak Reference -

<http://keccak.noekeon.org/Keccak-reference-3.0.pdf>]

## Keccak

- ▶ O rundă presupune trecerea stării prin 5 transformări menite să introducă proprietățile necesare (non-liniaritate, difuzie, non-simetrie, etc.):

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- ▶ Numărul de runde depinde de parametrul  $l$ :  $12 + 2l$ ;
- ▶ Pentru  $b = 1600 = 5 \times 5 \times 2^6 \Rightarrow 12 + 2 * 6 = 24$  runde.

## Important de reținut!

- ▶ Keccak este câștigătorul competiției SHA-3
- ▶ SHA-2 rămâne în continuare sigură



# Criptografie și Securitate

## - Prelegherea 16 - HMAC

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Revizuire - MAC

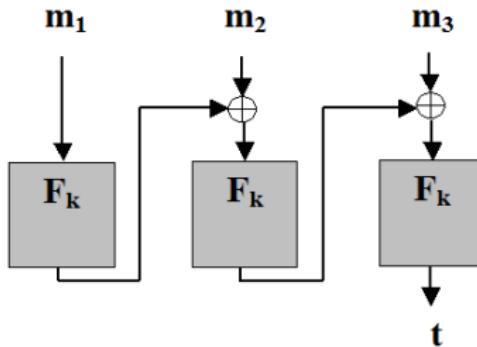
2. Hash MAC

## Revizuire: MAC-uri

- ▶ În prelegerile anterioare am vazut că putem construi MAC-uri sigure pe baza funcțiilor pseudoaleatoare (PRF);
- ▶ Pentru  $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  o PRF, defineam un MAC astfel
  - ▶  $\text{Mac}(k, m) : t = F_k(m)$ ;
  - ▶  $\text{Vrfy}(k, m, t) = 1$  dacă și numai dacă  $t = F_k(m)$  (altfel întoarce 0).
- ▶ Această construcție este bună pentru mesaje de lungime mică, dar avem nevoie de construcții de MAC-uri pentru mesaje mult mai mari;

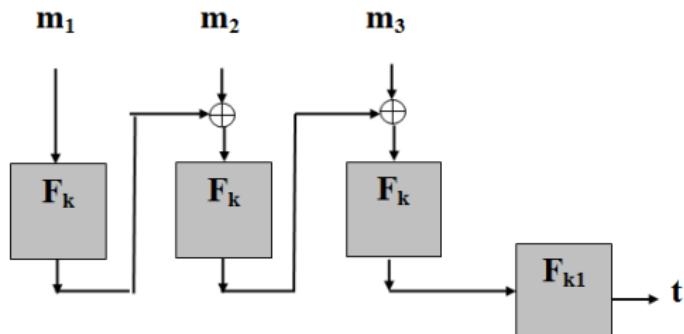
## Revizuire: MAC-uri

- ▶ Există două construcții de bază care se folosesc în practică:
  - ▶ **CBC-MAC** - folosit pe larg în industria bancară, WPA2, ...
  - ▶ **HMAC** - pentru protocoale pe Internet: SSL, IPsec, SSH...
- ▶ Reamintim construcția CBC-MAC sigură pentru mesaje de lungime fixă:



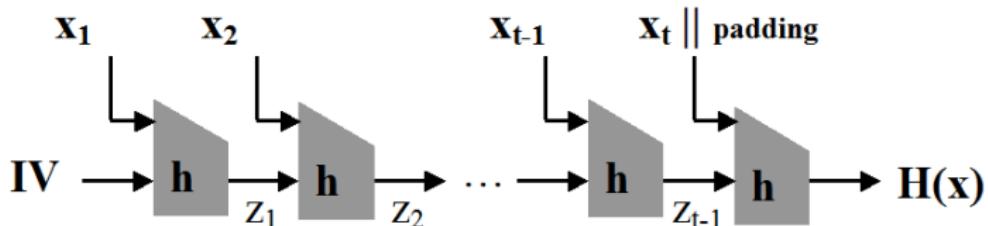
## Revizuire: MAC-uri

- ▶ Însă pentru mesaje de lungime variabilă, putem reține această construcție ca fiind sigură:



# HMAC

- ▶ Am studiat deja funcții hash și transformarea Merkle-Damgård pentru a obține funcții hash (rezistente la coliziuni) cu intrarea de lungime variabilă pornind de la funcții de compresie cu intrarea de lungime fixă;
- ▶ Reamintim construcția Merkle-Damgård:

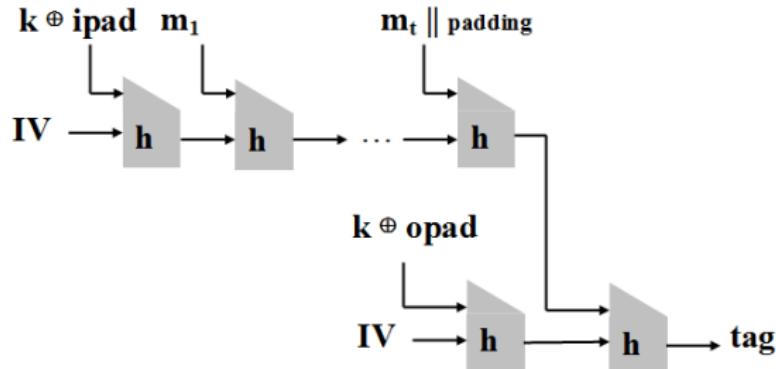


## HMAC

- ▶ Încercăm să construim un MAC direct pornind de la  $H(\cdot)$ , unde  $H(\cdot)$  este funcția hash obținută cu transformarea Merkle-Damgård;
- ▶ Definim  $\text{Mac}(k, m)$  astfel:  $t = H(k||m)$ .
- ▶ **Întrebare:** Este acesta un MAC sigur (nu poate fi falsificat printr-un atac cu mesaj clar ales)?
- ▶ **Răspuns:** NU! Un advesar poate calcula un tag  $t'$  pentru un mesaj nou care nu a mai fost autentificat: extinde mesajul anterior cu încă un bloc  $d$ , și calculează:  
$$H(k||m||d) = h(d||H(k||m))$$

# HMAC

- ▶ Folosim metoda standardizată HMAC (Hash MAC):



- ▶ HMAC se definește astfel:
- ▶  $\text{Mac}(k, m)$ :  $t = H((k \oplus \text{opad}) \parallel H((k \oplus \text{ipad}) \parallel m))$ ;
- ▶  $\text{Vrfy}(k, m, t) = 1 \iff t = \text{Mac}(k, m)$ .

## Notății

- ▶ ipad și opad sunt două constante de lungimea unui bloc  $m$ ;
- ▶ ipad constă din byte-ul 0x5C repetat de atâtea ori cât e nevoie;
- ▶ opad constă din byte-ul 0x36 repetat de atâtea ori cât e nevoie;
- ▶ IV este o constantă fixată.

## Securitate HMAC

- ▶ Definim  $G(k) = h(IV \parallel (k \oplus \text{opad})) \parallel h(IV \parallel (k \oplus \text{ipad}))$
- ▶ Privind secvența ca  $G(k) = k_1 \parallel k_2$ , dacă  $G$  este PRG și  $k \leftarrow^R \{0,1\}^n$ , deși  $k_1, k_2$  sunt dependente, acestea par alese în mod uniform și independent;
- ▶ Dacă  $G$  este PRG, atunci dăm următorul rezultat de securitate pentru HMAC :

### Teoremă

Dacă  $G$  este PRG,  $h$  prezintă rezistență la coliziuni și MAC-ul  $h(k \parallel m)$  construit pe baza ei este sigur (pentru mesaje de lungime fixă), atunci HMAC este sigur (pentru mesaje de lungime arbitrară) - nu poate fi falsificat printr-un atac cu mesaj ales.

## Important de reținut!

- ▶ HMAC este un MAC foarte popular folosit în multe protocoale practice precum TLS;
- ▶ Construcția lui se bazează pe funcții hash, de exemplu SHA-2 (SHA-256).



# Criptografie și Securitate

- Prelegherea 17 -  
Criptografia asimetrică

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Limitările criptografiei simetrice
2. Criptografia 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 părților 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.

## Problema 1 - Distribuirea cheilor

- ▶ **Varianta 2.** Se transmit printr-un canal de comunicație sigur care presupune un serviciu de mesagerie de încredere;
- ▶ Opțiunea poate fi 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 2 - Stocarea secretă a cheilor

- ▶ Sistemele informaticе sunt deseori infectate de programe malițioase care fură cheile secrete și le transmit prin internet către atacator;
- ▶ Totuși dacă numărul de chei este mic, există soluții de stocare cu securitate crescută;
- ▶ Un exemplu îl reprezintă dispozitivele de tip *smartcard*;
- ▶ Acestea realizează calculele criptografice folosind cheia stocată, asigurând faptul că niciodată cheia secretă nu ajunge în calculator;
- ▶ Capacitatea de stocare a unui smartcard este însă limitată, neputând memora, de exemplu mii de chei criptografice.

## Problema 3 - Medii de comunicare deschise

- ▶ Deși 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;

*"Solutions that are based on private-key cryptography are not sufficient to deal with the problem of secure communication in open systems where parties cannot physically meet, or where parties have transient interactions."*

(J.Katz, Y.Lindell: Introduction to Modern Cryptography)

## Problema 4 - Imposibilitatea non-repudierii

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

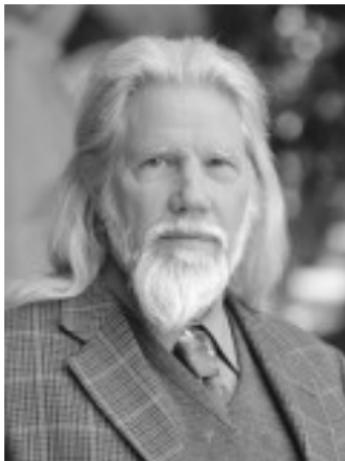
# Criptografia asimetrică

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

*"Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of written signature. This paper suggests ways to solve these currently open problems."*

(W.Diffie, M.Hellman: New Directions in Cryptography - abstract)

# Criptografia asimetrică



[[http://cisac.stanford.edu/people/whitfield\\_diffie/](http://cisac.stanford.edu/people/whitfield_diffie/)]



[<http://www-ee.stanford.edu/~hellman/>]

# Criptografia asimetrică



Home > Awards > CRYPTOGRAPHY PIONEERS RECEIVE ACM A.M. TURING AWARD



## CRYPTOGRAPHY PIONEERS RECEIVE ACM A.M. TURING AWARD

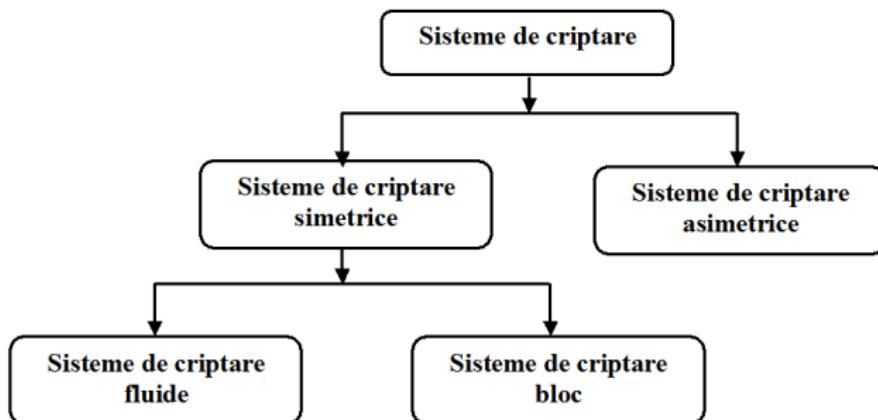
### Diffie and Hellman's Invention of Public-Key Cryptography and Digital Signatures Revolutionized Computer Security

ACM, the Association for Computing Machinery, today named Whitfield Diffie, former Chief Security Officer of Sun Microsystems and Martin E. Hellman, Professor Emeritus of Electrical Engineering at Stanford University, recipients of the 2015 ACM A.M. Turing Award for critical contributions to modern cryptography. The ability for two parties to use encryption to communicate privately over an otherwise insecure channel is fundamental for billions of people around the world. On a daily basis, individuals establish secure online connections with banks, e-commerce sites, email servers and the cloud. Diffie and Hellman's

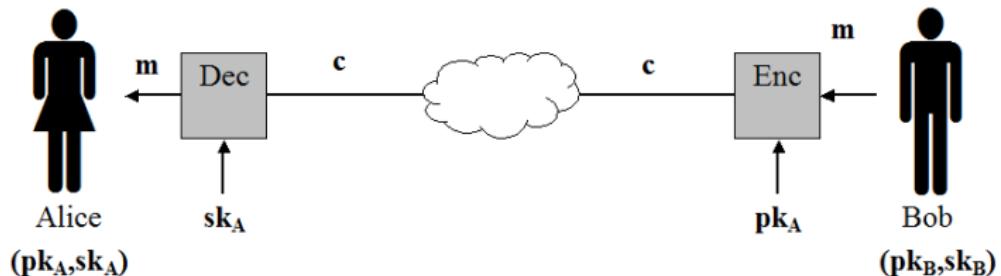
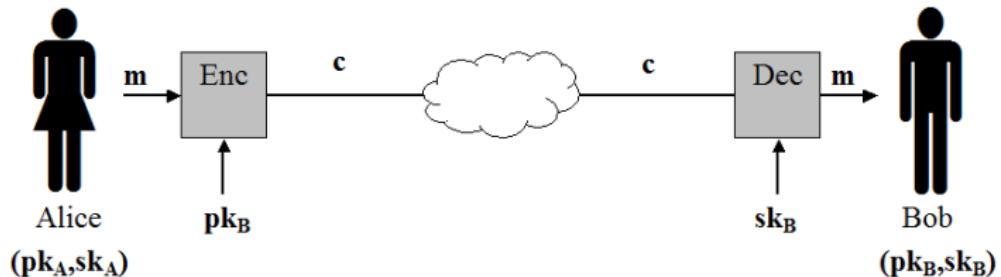
[<https://www.acm.org/awards/2015-turing>]

# 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**;



## Criptarea asimetrică (cu cheie publică)



# Criptarea asimetrică (cu cheie publică)

## Definiție

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

- ▶  $\mathcal{K} = \mathcal{K}_{pk} \times \mathcal{K}_{sk} =$  spațiul cheilor, de forma unor perechi  $(pk, sk)$ , unde  $pk$  este **cheia publică** și  $sk$  este **cheia secretă**
- ▶  $\mathcal{M} =$  spațiul textelor clare (mesaje)
- ▶  $\mathcal{C} =$  spațiul textelor criptate

este un dublet  $(\text{Enc}, \text{Dec})$ , unde:

1.  $\text{Enc} : \mathcal{K}_{pk} \times \mathcal{M} \rightarrow \mathcal{C}$
  2.  $\text{Dec} : \mathcal{K}_{sk} \times \mathcal{C} \rightarrow \mathcal{M}$
- a.î.  $\forall m \in \mathcal{M}, (pk, sk) \in \mathcal{K} : \text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m.$

## Terminologie

- ▶ Spre deosebire de criptarea simetrică, criptarea asimetrică folosește o pereche de chei:
- ▶ Cheia publică **pk** este folosită pentru criptare;
- ▶ Cheia secretă **sk** este folosită pentru decriptare;
- ▶ Cheia publică este larg răspândită pentru a asigura posibilitatea de criptare oricui dorește să transmită un mesaj către entitatea căreia îi corespunde;
- ▶ Cheia secretă este privată și nu se cunoaște decât de entitatea căreia îi corespunde;
- ▶ Considerăm (pentru simplitate) că ambele chei au lungime cel puțin  $n$  biți.

# Criptografia asimetrică vs. Criptografia simetrică

## 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

## 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

# Criptografia asimetrică

## 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

## 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)

## Important de reținut!

- ▶ Criptografia simetrică NU rezolvă toate problemele criptografiei
- ▶ Criptografia asimetrică apare în completarea criptografiei simetrice



# Criptografie și Securitate

- Prelegherea 18 -

Prezumptions criptografice dificile

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

## 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 prezumptie e nenaturală și foarte puternică;
- ▶ În practică, PRF și PRP 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 demonstrezi existența permutărilor pseudoaleatoare pe baza unei prezumptii mult mai slabe, cea a existenței funcțiilor one-way;

## Prezumptii criptografice dificile

- ▶ În continuare vom introduce câteva probleme considerate "dificile" și vom prezenta funcții conjecturate ca fiind one-way bazate pe aceste probleme;
- ▶ Tot materialul ce urmează se bazează pe noțiuni de teoria numerelor;

## Primalitate și factorizare

- ▶ 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 aşa încât  $N = x_1 \cdot x_2$ ;
- ▶ Cele mai greu de factorizat sunt numerele cele care au factori primi foarte mari.

# Primalitate și factorizare

*"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length... The dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated."*

(C.F.Gauss 1777 – 1855)

## Generarea numerelor prime

- ▶ Pentru a putea folosi problema în criptografie, trebuie să generăm numere prime aleatoare *în mod eficient*;
- ▶ Putem genera un numar prim aleator pe  $n$  biți prin alegerea repetată de numere aleatoare pe  $n$  biți până când găsim unul prim;
- ▶ Pentru eficiență, ne interesează două aspecte:
  1. probabilitatea ca un număr aleator de  $n$  biți să fie prim;
  2. cum testăm eficient că un număr dat  $p$  este prim.

## Generarea numerelor prime

- ▶ Pentru distribuția numerelor prime, se cunoaște următorul rezultat matematic:

### Teoremă

*Există o constantă  $c$  aşa încât, pentru orice  $n > 1$  numărul de numere prime pe  $n$  biți este cel puțin  $c \cdot 2^{n-1}/n$*

- ▶ Rezultă imediat că probabilitatea ca un numar ales aleator pe  $n$  biți să fie prim este cel putin  $c/n$ ;
- ▶ Iar dacă testăm  $t = n^2/c$  numere, probabilitatea ca un număr prim să nu fie ales este  $(1 - c/n)^t$ , adică cel mult  $e^{-n}$ , deci neglijabilă.

## Testarea primalității

- ▶ 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 și să se fi produs o eroare;
- ▶ Un algoritm determinist polinomial a fost propus în 2002, dar este mai lent decât algoritmii probabilisti;
- ▶ Prezentăm un algoritm probabilist foarte răspândit: Miller-Rabin.

# Testarea primalității

---

## Algorithm 1 Algoritmul Miller-Rabin

---

**Input:**  $N$ ,  $t$

**Output:** o decizie dacă  $N$  este compus sau nu

```
1: if  $N$  este par sau  $N = N_1^2$  then
2:   return "compus"
3: end if
4: compute  $r \geq 1$  și  $u$  impar a.î.  $N - 1 = 2^r u$ 
5: for  $j = 1$  to  $t$  do
6:    $a \leftarrow \{1, \dots, N - 1\}$ 
7:   if  $(\gcd(a, N) \neq 1)$  or  $(a^u \neq \pm 1 \bmod N \text{ and } a^{2^i u} \neq -1 \bmod N, \text{ pentru orice } i \in \{1, \dots, r - 1\})$  then
8:     return "compus"
9:   end if
10: end for
11: return "prim"
```

---

# Algoritmul Miller-Rabin

- ▶ Acceptă la intrare un numar  $N$  și un parametru  $t$  care determină *probabilitatea de eroare*;
- ▶ Rulează în timp **polinomial** în  $|N|$  și  $t$  și satisface:

## Teoremă

Dacă  $N$  este prim, atunci testul Miller-Rabin întoarce mereu "prim". Dacă  $N$  este compus, algoritmul întoarce "prim" cu probabilitate cel mult  $2^{-t}$  (i.e. întoarce rezultatul corect "compus" cu probabilitate  $1 - 2^{-t}$ )

## Algoritmul Miller-Rabin

- ▶ Ne întoarcem la problema generării *eficiente* a numerelor prime;
- ▶ Folosim algoritmul Miller-Rabin pentru a descrie un algoritm **polinomial** de generare a numerelor prime;
- ▶ Pentru o intrare  $n$ , întoarce un număr aleator pe  $n$  biți care este prim cu excepția unei probabilități neglijabile în  $n$ .

# Algoritmul de generare a numerelor prime

---

## Algorithm 2 Algoritmul de generare a numerelor prime

---

**Input:**  $n$

**Output:** Un număr aleator prim pe  $n$  biți

```
1: for  $i = 1$  to  $n^2/c$  do
2:    $p' \leftarrow \{0, 1\}^{n-1}$ 
3:    $p = 1||p'$ 
4:   execută testul Miller-Rabin pentru  $p$  cu parametrul  $n$ 
5:   if output = "prim" then
6:     return  $p$ 
7:   end if
8: end for
9: return eșec
```

---

# Algoritmi de factorizare

- ▶ Deocamdată nu se cunosc *algoritmi polinomiali* pentru problema factorizării;
- ▶ Dar există algoritmi mult mai buni decât forța brută;
- ▶ Prezentăm în continuare câțiva algoritmi de factorizare.

# 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;
- ▶ Pentru  $N < 10^{12}$  metoda este destul de eficientă.

# Algoritmi de factorizare

- ▶ 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  $O(N^{1/4} \cdot (\log N)^c)$ ;
  - ▶ **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  $2^{O(n^{1/3} \cdot (\log n)^{2/3})}$ .

## Algoritmul sitei pătratice

- Un element  $y \in \mathbb{Z}_p^*$  este *rest pătratic modulo p* dacă  $\exists x \in \mathbb{Z}_p^*$  a.î.

$$x^2 = y \text{ mod } p$$

- $x$  se numește *rădăcina pătratică* a lui  $y$ ;
- Algoritmul se bazează pe două observații simple:
  1. Dacă  $N = pq$  cu  $p, q$  prime distințe, atunci fiecare rest pătratic modulo  $N$  are exact 4 rădăcini pătratice diferite;
  2. Dacă se pot afla  $x, y$  cu  $x^2 = y^2 \text{ mod } N$  și  $x \neq \pm y \text{ mod } N$ , atunci  $\gcd(x - y, N)$  este un factor prim al lui  $N$  calculabil în timp polinomial;

## Algoritmul sitei pătratice

- ▶ Algoritmul încearcă să genereze o pereche de valori  $x, y$  a.î.  $x^2 = y^2 \text{ mod } N$ , în speranța că  $x \neq \pm y \text{ mod } N$  cu probabilitate constantă; căutarea se desfășoară astfel:
- ▶ Se fixează o bază  $B = \{p_1, \dots, p_k\}$  de numere prime mici;
- ▶ Se caută  $l > k$  numere distincte  $x_1, \dots, x_l \in \mathbb{Z}_N^*$  pentru care  $q_i = [x_i^2 \text{ mod } N]$  este "mic" aşa încât toți factorii primi ai lui  $q_i$  se găsesc în  $B$ ;
- ▶ Vor rezulta relații de forma

$$x_j^2 = p_1^{e_{j,1}} \cdot p_2^{e_{j,2}} \cdots \cdots p_k^{e_{j,k}} \text{ mod } N$$

unde  $1 \leq j \leq k$ .

## Algoritmul sitei pătratice

- ▶ Pentru fiecare  $j$  se consideră vectorul:

$$e_j = (e_{j,1} \bmod 2, \dots, e_{j,k} \bmod 2)$$

- ▶ Dacă se poate determina o submulțime formată din astfel de vectori, a căror suma modulo 2 să fie  $(0,0,\dots,0)$ , atunci pătratul produsului elementelor  $x_j$  corespunzătoare va avea în  $B$  toți divizorii reprezentați de un număr par de ori.
- ▶ Se poate arăta că algoritmul optimizat rulează în timp  $2^{O(\sqrt{n} \cdot \log n)}$  pentru factorizarea unui număr  $N$  de lungime  $O(n)$ , deci **sub-exponențial** în lungimea lui  $N$ .

## Exemplu

- ▶ Fie  $N = 377753$ . Se știe că  $6647 = [620^2 \bmod N]$  și putem factoriza

$$6647 = 17^2 \cdot 23$$

- ▶ Deci:

$$620^2 = 17^2 \cdot 23 \bmod N$$

- ▶ Similar:

$$621^2 = 2^4 \cdot 17 \cdot 29 \bmod N$$

$$645^2 = 2^7 \cdot 13 \cdot 23 \bmod N$$

$$655^2 = 2^3 \cdot 13 \cdot 27 \cdot 29 \bmod N$$

- ▶ Baza de numere prime mici este:

$$B = \{2, 13, 17, 23, 29\}$$

## Exemplu

- ▶ Obținem:

$$620^2 \cdot 621^2 \cdot 645^2 \cdot 655^2 = 2^{14} \cdot 13^2 \cdot 17^4 \cdot 23^2 \cdot 29^2 \pmod{N}$$

$$[620 \cdot 621 \cdot 645 \cdot 655 \pmod{N}]^2 = [2^7 \cdot 13 \cdot 17^2 \cdot 23 \cdot 29 \pmod{N}]^2 \pmod{N}$$

- ▶ Toți exponentii sunt pari!
- ▶ După efectuarea calculelor:

$$\Rightarrow 127194^2 = 45335^2 \pmod{N}$$

- ▶ Cum  $127194 \neq 45335 \pmod{N}$ , se calculează un factor prim al lui N:

$$\gcd(127194 - 45335, 377753) = 751$$

## 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, câte 1 pentru fiecare dimensiune (în biți) 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:

# RSA Challenge

## ► RSA-1024

13506641086599522334960321627880596993888147560566  
70275244851438515265106048595338339402871505719094  
41798207282164471551373680419703964191743046496589  
27425623934102086438320211037295872576235850964311  
05640735015081875106765946292055636855294752135008  
52879416377328533906109750544334999811150056977236  
890927563

[[http://www.emc.com/emc-plus/rsa-labs/historical/  
the-rsa-challenge-numbers.htm](http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm)]

# RSA Challenge

## ► RSA-2048

25195908475657893494027183240048398571429282126204  
03202777713783604366202070759555626401852588078440  
69182906412495150821892985591491761845028084891200  
72844992687392807287776735971418347270261896375014  
97182469116507761337985909570009733045974880842840  
17974291006424586918171951187461215151726546322822  
16869987549182422433637259085141865462043576798423  
38718477444792073993423658482382428119816381501067  
48104516603773060562016196762561338441436038339044  
14952634432190114657544454178424020924616515723350  
77870774981712577246796292638635637328991215483143  
81678998850404453640235273819513786365643912120103  
97122822120720357

[[http://www.emc.com/emc-plus/rsa-labs/historical/  
the-rsa-challenge-numbers.htm](http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm)]

## Prezumپia logaritmului discret

- ▶ O altă prezumپie dificilă este DLP (Discrete Logarithm Problem) (sau PLD (Problema Logaritmului Discret));

## Prezumția logaritmului discret

- ▶ O altă prezumție dificilă este DLP (Discrete Logarithm Problem) (sau PLD (Problema Logaritmului Discret));
- ▶ Considerăm  $\mathbb{G}$  un grup ciclic de ordin  $q$ ;

## Prezumția 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.  $\mathbb{G} = \{g^0, g^1, \dots, g^{q-1}\}$ ;

## Prezumptia 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 generatorul 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 astfel 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 dificilă;
- ▶ Una dintre ele este clasa grupurilor ciclice de *ordin prim* (în aceste grupuri, problema este "cea mai dificilă");
- ▶ DLP nu poate fi rezolvată în timp polinomial în grupurile care nu sunt de ordin prim, ci doar este *mai ușoară*;
- ▶ În aceste grupuri căutarea unui generator și verificarea că un număr dat este generator sunt triviale.

## Lucrul în $\mathbb{Z}_p^*$

- ▶ 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  $\mathbb{Z}_p^*$ ;
- ▶ **Reamintim:** Un element  $y \in \mathbb{Z}_p^*$  este *rest pătratic modulo p* dacă  $\exists x \in \mathbb{Z}_p^*$  a.î.  $x^2 = y \text{ mod } p$ ;
- ▶ Multimea resturilor pătratice modulo  $p$  formează un subgrup al lui  $\mathbb{Z}_p^*$ ;
- ▶ Dacă  $p$  este număr *prim tare* (i.e.  $p = 2q + 1$  cu  $q$  prim), subgrupul resturilor pătratice modulo  $p$  are ordin  $q$ , deci este ciclic și toate elementele sunt generatori.

# Criptografia bazată pe latici

- ▶ Criptografia bazată pe latici:

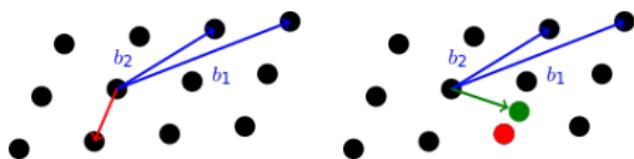


$$\left\{ \sum_{i=1}^n a_i v_i \mid a_i \in \mathbb{Z}, v_i \text{ bază} \right\}$$

- ▶ Probleme dificile: SVP (Shortest Vector Problem), CVP (Closest Vector Problem), ...

# Criptografia bazată pe latici

SVP (Shortest Vector Problem) și CVP (Closest Vector Problem):



[Images: [https://en.wikipedia.org/wiki/Lattice\\_problem#Shortest\\_vector\\_problem\\_\(SVP\)](https://en.wikipedia.org/wiki/Lattice_problem#Shortest_vector_problem_(SVP))]

## Important de reținut!

- ▶ Cel mai rapid algoritm de factorizare necesită timp sub-exponențial;
- ▶ Problema logaritmului discret este dificilă.



# Criptografie și Securitate

- Prelegherea 19 -

Noțiuni de securitate în criptografia asimetrică

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Securitate perfectă
2. Securitate semantică = Securitate CPA

## Securitate perfectă

- ▶ Începem studiul securității 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;

### Definiție

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

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

## Securitate perfectă

- ▶ **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.

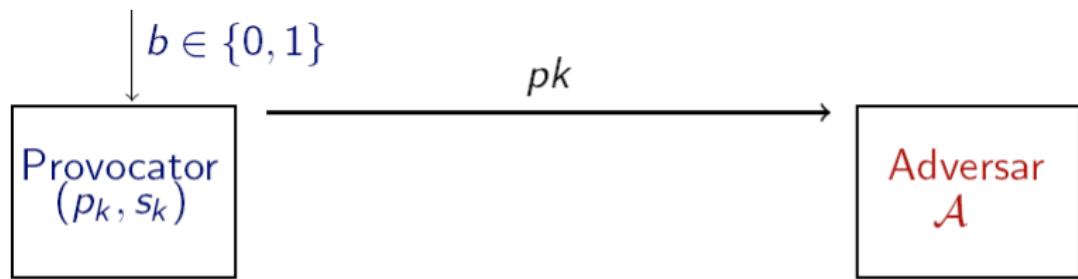
## Securitate semantică

- ▶ Securitatea semantică în criptografia cu cheie publică este corespondenta noțiunii similare din criptografia cu cheie secretă;
- ▶ Vom defini securitatea semantică pe baza unui experiment de indistinctibilitate  $\text{PubK}_{\mathcal{A}, \pi}^{\text{eav}}(n)$  unde  $\pi = (\text{Enc}, \text{Dec})$  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 **provocator (challenger)**.

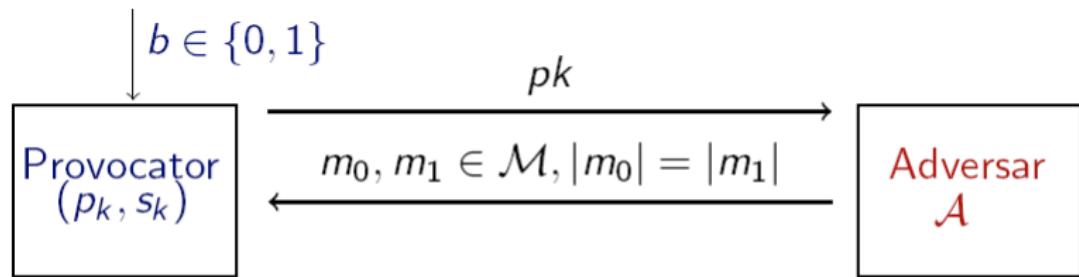
# Experimentul $\text{PubK}_{\mathcal{A}, \pi}^{\text{eav}}(n)$



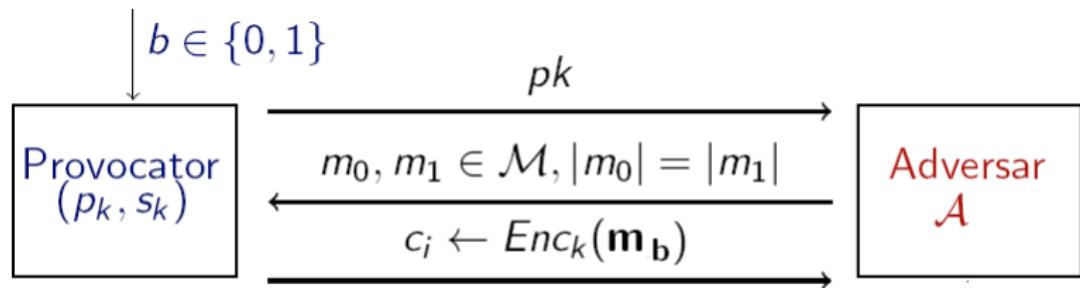
## Experimentul $\text{PubK}_{\mathcal{A}, \pi}^{\text{eav}}(n)$



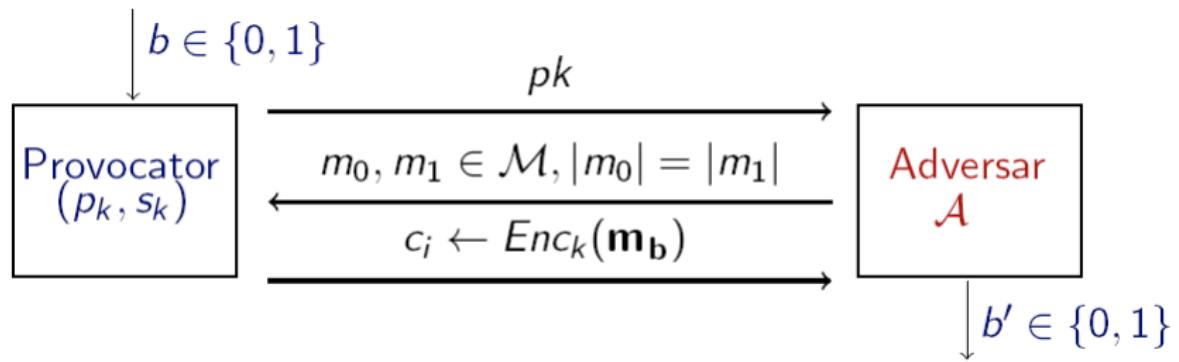
## Experimentul $\text{PubK}_{\mathcal{A}, \pi}^{\text{eav}}(n)$



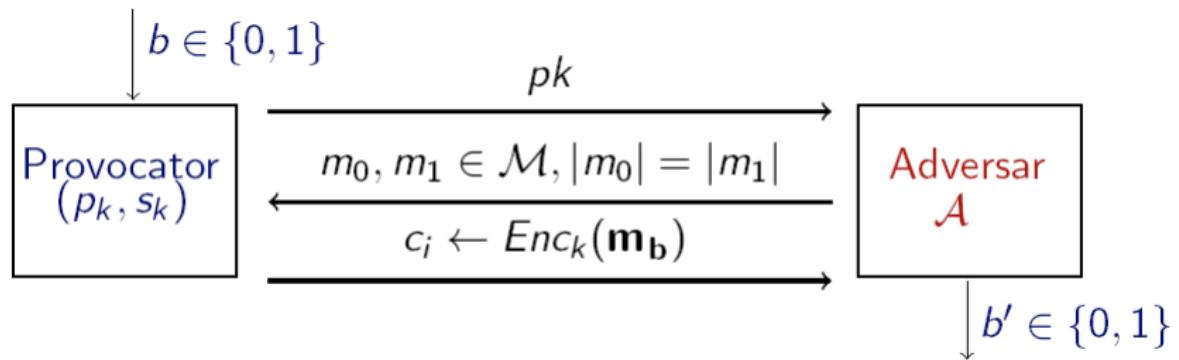
## Experimentalul $\text{PubK}_{\mathcal{A}, \pi}^{\text{eav}}(n)$



## Experimentalul $\text{PubK}_{\mathcal{A}, \pi}^{\text{eav}}(n)$



## Experimentul $PubK_{\mathcal{A}, \pi}^{eav}(n)$



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

# Securitate pentru interceptare simplă

## 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 aşa 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 sigure la securitate semantică dar care nu sunt CPA-sigure .

## Insecuritatea schemelor deterministe

- ▶ După cum am văzut la criptografia simetrică, nici o schemă deterministă nu poate fi CPA sigură;
- ▶ Datorită echivalenței între noțiunile de securitate CPA și securitate semantică pentru interceptare simplă (în criptografia asimetrică) concluzionăm că:

### Teoremă

*Nici o schemă de criptare cu cheie publică deterministă nu poate fi semantic sigură pentru interceptarea simplă.*

## Insecuritatea schemelor deterministe

- ▶ În realitate, schemele de criptare cu cheie publică deterministe sunt vulnerabile la atacuri practice;
- ▶ Acestea permit unui adversar să determine când un mesaj este trimis de două ori;
- ▶ Mai mult, îi permit să găsească mesajul  $m$  cu probabilitate 1, dacă spațiul mesajelor este mic.

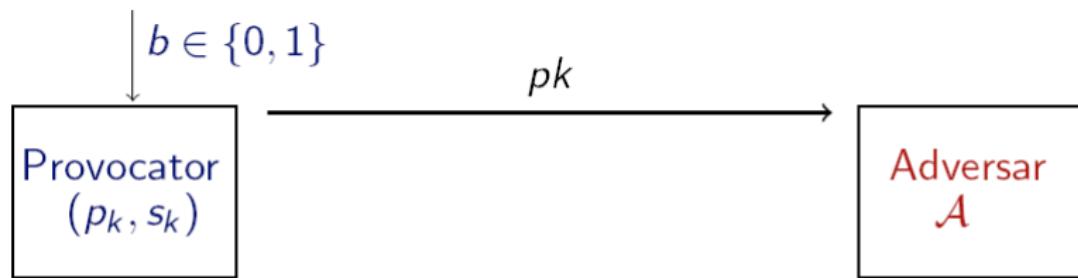
## Criptare multiplă

- ▶ Definim noțiunea de securitate la interceptare multiplă analog cu definiția similară din criptografia simetrică, pe baza unui experiment;
- ▶ Este clar că ea e echivalentă cu o definiție în care sunt considerate atacuri CPA;
- ▶ De remarcat că securitatea la interceptare simplă implică securitate la interceptare multiplă;

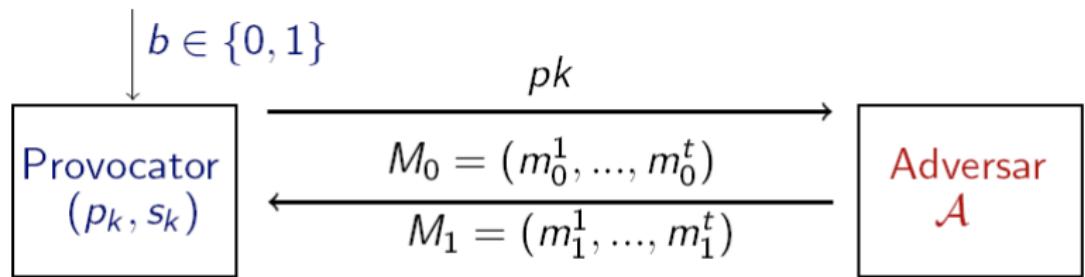
# Experimentul $\text{PubK}_{\mathcal{A}, \pi}^{\text{mult}}(n)$



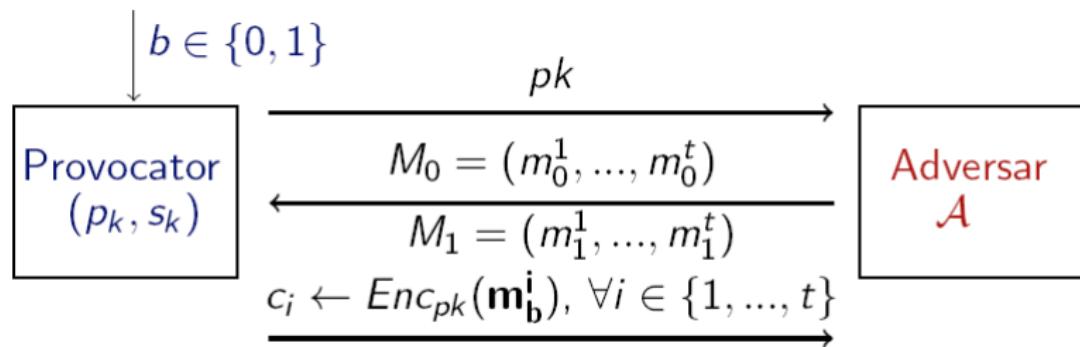
# Experimentul $\text{PubK}_{\mathcal{A}, \pi}^{\text{mult}}(n)$



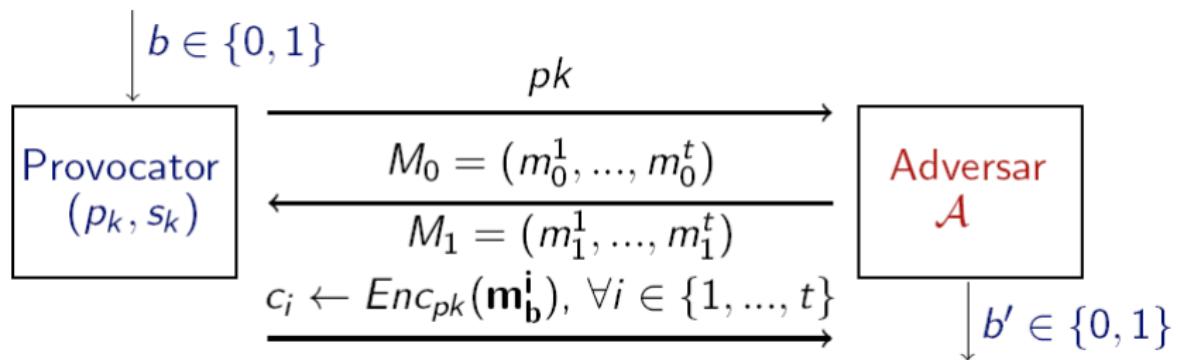
## Experimentalul $\text{PubK}_{\mathcal{A}, \pi}^{\text{mult}}(n)$



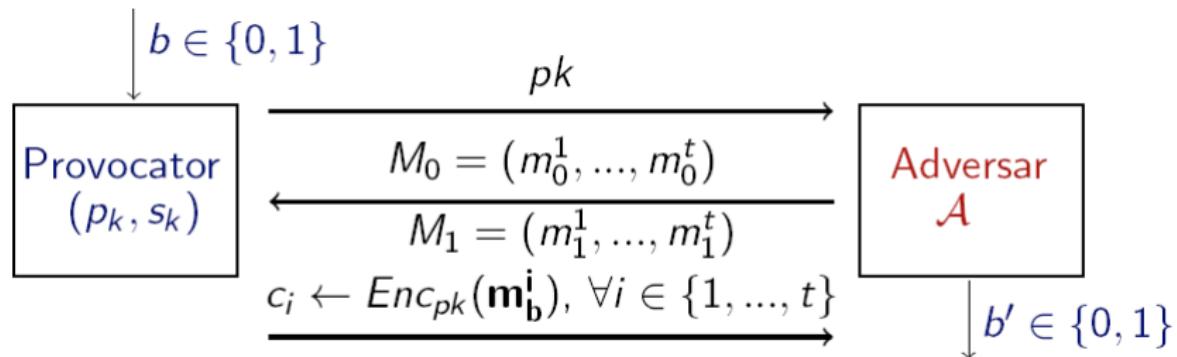
## Experimentalul $\text{PubK}_{\mathcal{A}, \pi}^{\text{mult}}(n)$



## Experimentalul $\text{PubK}_{\mathcal{A}, \pi}^{\text{mult}}(n)$



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



- ▶ Output-ul experimentului este 1 dacă  $b' = b$  și 0 altfel;
- ▶ Definiția de securitate este aceeași, doar că se referă la experimentul de mai sus.

# Securitate pentru interceptare multiplă

## Definiție

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

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

## Teoremă

Dacă o schemă de criptare cu cheie publică este sigură la interceptare simplă, atunci ea este sigură și la interceptare multiplă.

## Criptarea mesajelor de lungime arbitrară

- ▶ Consecință a rezultatului anterior: o schemă de criptare sigură pentru mesaje de *lungime fixă* este sigură și pentru mesaje de *lungime arbitrară*;
- ▶ Dacă  $\pi = (\text{Enc}, \text{Dec})$  este o schemă de criptare cu spațiul mesajelor  $\mathcal{M} = \{0, 1\}$ , putem construi o schemă sigură peste spațiul mesajelor  $\mathcal{M} = \{0, 1\}^*$  definind  $\text{Enc}'$ :

$$\text{Enc}'_{pk}(m) = \text{Enc}_{pk}(m_1), \dots, \text{Enc}_{pk}(m_t)$$

unde  $m = m_1 \dots m_t$

- ▶ Rezultatul este adevărat pentru atacuri CPA dar nu este adevărat pentru atacuri CCA.

## Securitate CCA

- ▶ Noțiunea de securitate CCA rămâne identică cu cea de la sistemele simetrice;
- ▶ Capabilitățile adversarului: el poate interacționa cu un **oracol de decriptare**, fiind un adversar *activ* care poate rula atacuri în timp polinomial;
- ▶ Adversarul poate transmite către oracolul de decriptare *anumite* mesaje  $c$  și primește înapoi mesajul clar corespunzător;
- ▶ Ca și în cazul securității CPA, adversarul nu mai necesită acces la oracolul de criptare pentru că detine cheia publică  $pk$  și poate realiza singur criptarea oricărui mesaj  $m$ .

## Important de reținut!

- ▶ În criptografia cu cheie publică:
  - ▶ NU există securitate perfectă
  - ▶ securitate semantică = securitate CPA



# Criptografie și Securitate

- Prelegherea 20 -  
Criptarea hibridă

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definiție

2. Securitate

## Criptarea hibridă

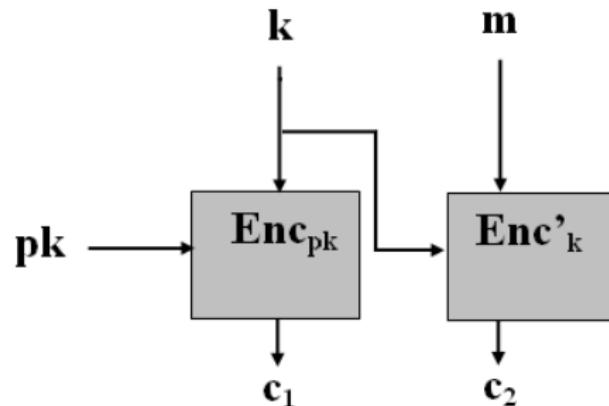
- ▶ Conform cu ce am văzut în anterior, criptarea unui mesaj de  $t$  biți necesită  $t$  apelări ale schemei de criptare originale;

## Criptarea hibridă

- ▶ Conform cu ce am văzut în anterior, criptarea unui mesaj de  $t$  biți necesită  $t$  apelări ale schemei de criptare originale;
- ▶ Aceasta înseamnă că și calculele dar și lungimea textului criptat cresc cu un factor multiplicativ în raport cu  $t$ ;

## Criptarea hibridă

- Rezultatul acestei combinații se numește **criptare hibridă** și este folosită extensiv în practică;



## Criptare hibridă

- ▶ 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  $c = (c_1, c_2)$ ;
- ▶ Construcția este o schemă de criptare asimetrică (cele două părți nu partajează o cheie secretă în avans).

## Criptare hibridă

- ▶ Când  $|m| \gg n = |k|$ , criptarea hibridă oferă o îmbunătățire substanțială a eficienței față de criptarea bit cu bit sau bloc cu bloc;
- ▶ Deci, pentru mesaje suficient de lungi, ea îmbină funcționalitatea criptării cu cheie publică cu eficiența criptării cu cheie secretă.

## 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 hibride.

## Important de reținut!

- ▶ Pentru criptarea mesajelor lungi, în practică se folosește criptarea hibridă
- ▶ Aceasta îmbină avantajele criptării simetrice și criptării asimetrice



# Criptografie și Securitate

- Prelegherea 21 -

Permutări cu trapă secretă

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definiție
2. Problema rucsacului
3. Construcția sistemelor de criptare asimetrice

## Permutări cu trapă secretă

- ▶ Reamintim noțiunea de funcție **one-way**;
- ▶ Aceasta este o funcție pentru care este **ușor** de calculat valoarea funcției...
- ▶ ... dar este **dificil** de calculat valoarea funcției inverse;
- ▶ Am întâlnit noțiunea când am studiat funcțiile hash;
- ▶ Dacă  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$  este o funcție hash (rezistentă la prima preimagine), atunci:
  - ▶ Fiind dat  $x$ , este *eficient* de calculat  $H(x)$ ;
  - ▶ Cunoscând  $H(x)$  este (*computațional*) **dificil** de calculat  $x$ .

## Permutări cu trapă secretă

- ▶ Definim noțiunea de permutare cu trapă secretă sau TDP (TrapDoor Permutation);
- ▶ Aceasta este o permutare one-way...
- ▶ ... care permite calculul eficient al inversului dacă se cunoaște o informație adițională, numită cheie secretă;
- ▶ Utilizarea cheii secrete permite deținătorului să folosească o trapă secretă, de unde provine și denumirea construcției.

# Permutări cu trapă secretă

## Definiție

O permutare cu trapă secretă sau **TDP** (TrapDoor Permutation) este un triplet  $(\text{Gen}, F, F^{-1})$  unde:

1. *Gen este un algoritm nedeterminist PPT care generează o pereche de chei  $(pk, sk)$ ;*
2.  *$F(pk, \cdot) : \mathcal{X} \rightarrow \mathcal{X}$  este o funcție one-way;*
3.  *$F^{-1}(sk, \cdot) : \mathcal{X} \rightarrow \mathcal{X}$  este o funcție eficient calculabilă;*

$$\forall x \in \mathcal{X}, F^{-1}(sk, F(pk, x)) = x$$

- ▶  $F$  este sigură dacă poate fi eficient evaluată, dar nu poate fi inversată fără cunoașterea cheii secrete  $sk$  (decât cu probabilitate neglijabilă);
- ▶ Notații:  $F(pk, \cdot) = F_{pk}(\cdot)$ ,  $F^{-1}(sk, \cdot) = F_{sk}^{-1}(\cdot)$ .

## Problema rucsacului

- ▶ Un exemplu de funcție *one-way* este **problema rucsacului**;
- ▶ Se dă un vector  $A = (a_1, a_2, \dots, a_n)$  de  $n$  elemente distincte  $a_i \in \mathbb{Z}_+$  și o valoare  $k \in \mathbb{Z}_+$ ;
- ▶ Se cere să se determine elementele vectorului a căror sumă este  $k$ ;
- ▶ Pentru un vector de  $n$  elemente, problema se poate rezolva verificând pe rând toate submulțimile lui  $A$ ;
- ▶ Cum numărul submulțimilor este de în  $2^n - 1$ , această modalitate de rezolvare este imposibilă pentru  $n$  mare;
- ▶ Problema este (în general) dificilă.

## Problema rucsacului

- ▶ Există însă clase ușoare ale problemei rucsacului;
- ▶ Una dintre acestea o reprezintă vectorii **super-crescători**;
- ▶ Un vector  $A = (a_1, a_2, \dots, a_n)$  este *super-crescător* dacă satisface:

$$\forall j \geq 2, a_j > \sum_{i=1}^{j-1} a_i$$

- ▶ Un exemplu este vectorul:

$$A = \{1, 3, 5, 11, 21, 44, 87\}$$

$$3 > 1$$

$$5 > 1 + 3$$

$$11 > 1 + 3 + 5$$

$$21 > 1 + 3 + 5 + 11$$

$$44 > 1 + 3 + 5 + 11 + 21$$

$$87 > 1 + 3 + 5 + 11 + 21 + 44$$

## Problema rucsacului

- ▶ Dăm un algoritm de rezolvare a problemei rucsacului pentru vectori super-crescători;
- ▶ Cunoscând  $k$ , se parcurge vectorul de la dreapta spre stânga;
- ▶ Dacă  $k \geq a_i$ , atunci  $a_i$  face parte din sumă (suma tuturor celorlalte elemente este mai mică decât  $a_i$ );
- ▶ Dacă  $a_i$  face parte din sumă, atunci valoarea  $k$  se actualizează cu  $k - a_i$ ;
- ▶ Se repetă procedeul până se parcurge întreg vectorul sau  $k$  devine 0.

## Problema rucsacului

- ▶ Pentru exemplul anterior  $A = \{1, 3, 5, 11, 21, 44, 87\}$ , fie  $k = 58$ ;
- ▶ Se obține:
  - $k = 58 < 87 \Rightarrow 87$  nu apare în sumă
  - $k = 58 > 44 \Rightarrow 44$  apare în sumă și  $k = 58 - 44 = 14$
  - $k = 14 < 21 \Rightarrow 21$  nu apare în sumă
  - $k = 14 > 11 \Rightarrow 11$  apare în sumă și  $k = 14 - 11 = 3$
  - $k = 3 < 5 \Rightarrow 5$  nu apare în sumă
  - $k = 3 = 3 \Rightarrow 3$  apare în sumă și  $k = 3 - 3 = 0$
- ▶ S-a obținut deci  $k = 44 + 11 + 3$ .

## Problema rucsacului

- ▶ Transformăm o problemă **simplă** a rucsacului într-o problemă **dificilă** pe baza unei informații secrete și obținem astfel o **funcție cu trapă secretă**;
- ▶ Fie un vector supercrescător  $A = (a_1, a_2, \dots, a_n)$ ;
- ▶ Se aleg un **modul**  $m$  și un **multiplicator**  $t$  a.î.  $\gcd(c, m) = 1$ ;
- ▶ Se calculează  $B = (b_1, b_2, \dots, b_n)$ , unde  $b_i = a_i \cdot t \pmod{m}$ ;
- ▶ Cunoscând  $A$  problema este simplă, dar cunoscând  $B$  problema este dificilă.

## Problema rucsacului

- ▶ Pentru exemplul anterior:  $A = \{1, 3, 5, 11, 21, 44, 87\}$ , fie  $t = 43$  și  $m = 1590$ ;
- ▶ Se obține  $B = \{43, 129, 215, 473, 903, 302, 561\}$ ;
- ▶ Se cere rezolvarea problemei rucsac pentru  $k = 904$  și  $B$ , care este dificilă (facem abstracție de dimensiunea lui  $n$ );
- ▶ Pentru deținătorul trapei secrete  $(t, m) = (43, 1590)$  problema devine ușoară;
- ▶ Aceasta se rezumă la rezolvarea problemei pentru  $k = 904 \cdot 43^{-1} \pmod{1590} = 58$  și  $A$  pe care am rezolvat-o anterior.

## Algoritmul lui Euclid extins

- ▶ Pentru calculul  $43^{-1} \pmod{1590}$  am folosit **algoritmul lui Euclid extins**;

- ▶ Se fac împărțiri cu rest repetitive (împărțitorul se împarte la rest) până se obține restul 1:

$$1590 = 43 \cdot 36 + 42$$

$$43 = 42 \cdot 1 + 1$$

- ▶ Se înlocuiesc valorile restului în sens invers:

$$1 = 43 - 42 \pmod{1590}$$

$$1 = 43 - (1590 - 43 \cdot 36) \pmod{1590} = 43 \cdot 37 \pmod{1590}$$

- ▶ Cum  $43 \cdot 37 \pmod{1590} = 1 \Rightarrow 43^{-1} \pmod{1590} = 37$ .

# Construcția sistemelor de criptare asimetrice

- ▶ Folosim TDP pentru construcția sistemelor de criptare asimetrice;

## Construcție

*Fie  $(Gen, F, F^{-1})$  TDP cu  $F : \mathcal{X} \rightarrow \mathcal{Y}$ ,  $(Enc, Dec)$  un sistem de criptare simetric sigur cu autentificarea mesajelor definit peste  $(\mathcal{X}, \mathcal{Y})$  și  $H : \mathcal{X} \rightarrow \mathcal{K}$  o funcție hash. Definim un sistem de criptare asimetrică peste  $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$  în felul următor:*

- ▶  $\text{Enc}_{pk}(m) = (y, c) = (F_{pk}(x), Enc_k(m))$ , unde  $k = H(x)$  și  $x \leftarrow^R \mathcal{X}$ ;
- ▶  $\text{Dec}_{sk}(y, c) = Dec_k(c)$ , unde  $k = H(x)$  și  $x = F_{sk}^{-1}(y)$ ;

# Exemple

## ► Merkle-Hellman

- definit în 1978 de R.Merkle și M.Hellman
- bazat pe problema rucsacului
- spart la numai câțiva ani de la publicare

## ► RSA

- definit în 1977 de R.Rivest, A.Shamir și L.Adleman
- bazat pe problema RSA și indirect a factorizării numerelor mari
- cel mai cunoscut sistem de criptare cu cheie publică

## Important de reținut!

- ▶ Noțiunea de permutare cu trapă secretă (TDP)
- ▶ Construcția sistemelor de criptare asimetrice folosind TDP



# Criptografie și Securitate

- Prelegerea 21.1 -  
RSA

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Scurt istoric
2. Problema RSA
3. Textbook RSA

# RSA

- ▶ 1976 - Diffie și Hellman definesc conceptul de criptografie asimetrică;
- ▶ 1977 - R.Rivest, A.Shamir și Leonard Adleman introduc sistemul RSA;
- ▶ RSA este cel mai cunoscut și mai utilizat algoritm cu cheie publică.

# RSA

*The era of electronic mail may soon be upon us; we must ensure that two important properties of the current paper mail system are preserved:*

*(a) messages are private, and (b) messages can be signed. We demonstrate in this paper how to build these capabilities into an electronic mail system.*

*At the heart of our proposal is a new encryption method. This method provides an implementation of a public-key cryptosystem, an elegant concept invented by Diffie and Hellman. Their article motivated our research, since they presented the concept but not any practical implementation of such a system..."*

(R.Rivest, A.Shamir, L.Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems - introduction)

# RSA



[<http://people.csail.mit.edu/rivest/>]

**RSA = Rivest + Shamir + Adleman**

## 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  $\phi(N) = (p - 1)(q - 1)$ ;

## Problema RSA

*Fiind dat  $N$ , un întreg  $e > 0$  care satisfacă  $(e, \phi(N)) = 1$ , și un element  $y \in \mathbb{Z}_N^*$ , se cere să se găsească  $x$  a.î.  $x^e = y \text{ mod } N$ .*

- ▶ Dacă se cunoaște factorizarea lui  $N$ , atunci  $\phi(N)$  este ușor de calculat și problema RSA este ușor de rezolvat;
- ▶ Dacă nu se cunoacște  $\phi(N)$ , problema RSA este dificilă.

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

- ▶ 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*

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

## GenRSA

- ▶ Prezumptia RSA este ca exista un algoritm GenRSA pentru care problema RSA este dificila;
- ▶ Un algoritm GenRSA poate fi construit pe baza unui numar compus impreun cu factorizarea lui;

---

### Algorithm 3 GenRSA

---

**Input:**  $n$

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

- 1: **print**  $N$  cu factorii  $p$  si  $q$
  - 2:  $\phi(N) = (p - 1)(q - 1)$
  - 3: **gaseste**  $e$  a.i.  $gcd(e, \phi(N)) = 1$
  - 4: **calculeaza**  $d := e^{-1} \text{ mod } \phi(N)$
  - 5: **return**  $N, e, d$
-

## GenRSA

- ▶ Pentru ca problema RSA să fie dificilă, trebuie ca N-ul ales în GenRSA să fie dificil de factorizat în produs de două numere prime;
- ▶ Deci problema RSA nu este mai dificilă decât problema factorizării;
- ▶ Dacă se cunosc  $N$ ,  $e$  și  $d$  cu  $ed = 1 \text{ mod } \phi(N)$ , se poate calcula factorizarea lui  $N$  în timp probabilist polinomial;
- ▶ 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  $d$ .

# GenRSA

- ▶ Definim sistemul de criptare *Textbook RSA* pe baza problemei prezentată anterior;
  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  $m \in \mathbb{Z}_N$ , întoarce  $c = m^e \pmod{N}$ ;
  3. **Dec:** dată o cheie secretă  $(N, d)$  și un mesaj criptat  $c \in \mathbb{Z}_N$ , întoarce  $m = c^d \pmod{N}$ .

## Implementarea RSA

- ▶ Pentru ca GenRSA să fie dificilă trebuie ca  $N$  să fie produs de 2 numere prime mari;
- ▶ Sistemul necesită exponențieri modulare de tip  $x^c \pmod{N}$ ;
- ▶ Efectuarea a  $c - 1$  înmulțiri modulare este foarte ineficientă;
- ▶ Se utilizează algoritmul de exponențiere rapidă.

# Implementarea RSA

---

## Algorithm 4 Exponentiere rapidă

---

**Input:**  $N, x, c$

**Output:**  $x^c \bmod N$

```
1: descompune  $c$  în binar:  $c = \sum_{i=0}^{n-1} c_i 2^i$ 
2:  $z \leftarrow 1$ 
3: for  $i = n - 1$  to 0 do
4:    $z \leftarrow z^2 \bmod N$ 
5:   if  $c_i = 1$  then
6:      $z \leftarrow z \cdot x \bmod N$ 
7:   end if
8: end for
9: return  $z$ 
```

---

# Securitate - Problema 1

## Problema 1: Determinismul

- ▶ **Întrebare:** Este Textbook RSA CPA-sigur sau CCA-sigur?
- ▶ **Răspuns:** NU! Sistemul este determinist, deci nu poate rezista definițiilor de securitate!

## Securitate - Problema 2

### Problema 2: Coeficient de criptare mic

- ▶ **Întrebare:** Este o valoare mică o alegere corectă pentru coeficientul de criptare  $e$ ?
- ▶ **Răspuns:** NU! Orice mesaj  $m < N^{1/e}$  nu folosește reducerea modulară la criptare:

$$c = m^e \mod N = m^e$$

- ▶ Fiind dat  $c$ , se determină imediat  $m$  ca:

$$m = c^{1/e} \mod N$$

## Securitate - Problema 3

### Problema 3: Atac cu text criptat ales

- ▶ Fie  $m_1, m_2$  2 texte clare și  $c_1, c_2$  textele criptate corespunzătoare;
- ▶ Atunci:  
 $c_1 \cdot c_2 \mod N = m_1^e \cdot m_2^e \mod N = (m_1 \cdot m_2)^e \mod N$
- ▶ **Întrebare:** Cum poate un adversar să determine mesajul clar  $m_1$ , cunoscând textul criptat corespunzător  $c_1$  printr-un atac cu text criptat ales?
- ▶ **Răspuns:** Adversarul alege un  $m_2 \in \mathbb{Z}_n$  oarecare și cere decriptarea textului criptat  $m_2^e \cdot c_1 \mod N$ ;
- ▶ Adversarul primește un mesaj clar  $m_3 = m_1 \cdot m_2 \mod N$ , apoi determină  $m_1 = m_3 \cdot m_2^{-1} \mod N$ .

## Securitate - Problema 4

### Problema 4: 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:

$$\begin{aligned}pk_1 &= (N, e_1); sk_1 = (N, d_1) \\pk_2 &= (N, e_2); sk_2 = (N, d_2)\end{aligned}$$

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

# Sistemul de criptare RSA

- ▶ Folosim construcția prezentată în prelegherea anterioară, utilizând RSA ca TDP:

## Construcție

*Fie  $(Enc, Dec)$  un sistem de criptare simetric sigur cu autentificarea mesajelor definit peste  $\mathbb{Z}_N$  și  $H : \mathbb{Z}_N \rightarrow \mathcal{K}$  o funcție hash. Definim un sistem de criptare asimetrică peste în felul următor:*

- ▶  $Enc_{pk}(m) = (y, c) = (RSA_{pk}(x), Enc_k(m))$ , unde  $k = H(x)$  și  $x \leftarrow^R \mathbb{Z}_N$ ;
- ▶  $Dec_{sk}(y, c) = Dec_k(c)$ , unde  $k = H(x)$  și  $x = RSA_{sk}(y)$ ;

## Important de reținut!

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



# Criptografie și Securitate

- Prelegerea 21.2 -  
PKCS

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Padded RSA
2. PKCS #1 v1.5
3. PKCS #1 v2.0 (OAEP)

## Padded RSA

- ▶ Am văzut că Textbook RSA este nesigur;
- ▶ Eliminăm 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;
- ▶ Notăm în continuare cu  $n$  parametrul de securitate (conform GenRSA).

## Padded RSA

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.

## Padded RSA

- ▶ 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:

### Teoremă

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

## PKCS #1 v1.5

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

## PKCS #1 v1.5

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

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

- ▶  $r$  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  $m$ .

# OAEPEP

- ▶ 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 OAEPEP = Optimal Asymmetric Encryption Standard;

## OAEP

- ▶ 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 criptează  $m \in \{0, 1\}^{n/2}$  folosind cheia publică  $(N, e)$  ca:

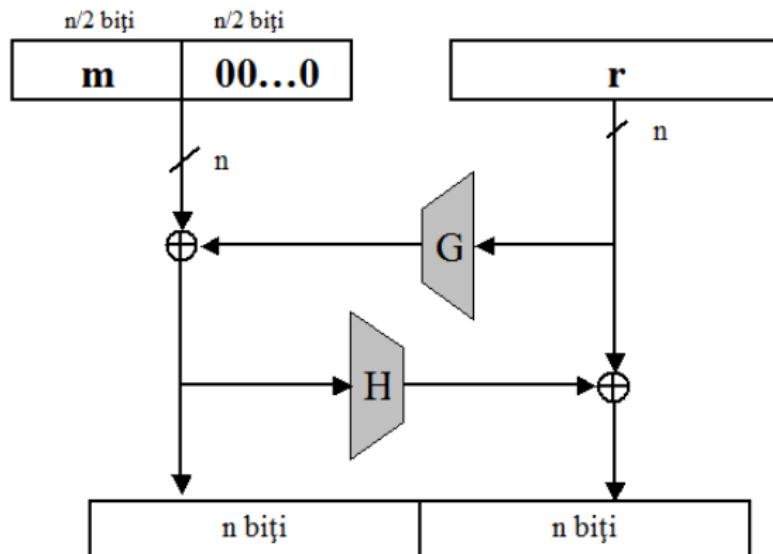
$$(OAEP(m, r))^e \mod N$$

- ▶ RSA-OAEP decriptează  $c$  folosind cheia secretă  $(N, d)$  ca:

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

# OAEP

- ▶ OAEP este definit astfel:



## Notății

- ▶  $G, H =$  funcții hash
- ▶  $m \in \{0, 1\}^{n/2}$  mesajul
- ▶  $r \leftarrow^R \{0, 1\}^n$
- ▶ se obține  $OAEPE(m, r)$  pe  $2n$  biți

# Important de reținut!

- ▶ Utilizarea RSA în practică:  
PKCS #1 v1.5 și PKCS #1 v2.0 (OAEP)



# Criptografie și Securitate

- Prelegherea 22 -

Despre problema logaritmului discret

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Algoritmi pentru DLP
2. Funcții hash rezistente la coliziuni bazate pe PLD

# Algoritmi pentru calculul logaritmului discret

- ▶ 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  $\log_g h = [x' \bmod q]$

## Algoritmi pentru calculul logaritmului discret

- ▶ 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.

## Algoritmi pentru calculul logaritmului discret

- ▶ 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  $\mathcal{O}(\sqrt{q} \cdot (\log q)^c)$ ;
  - ▶ Algoritmul **Pohlig-Hellman** poate fi folosit atunci când se cunoaște factorizarea ordinului  $q$  al grupului;

## Algoritmi generici pentru calculul logaritmului discret

- ▶ Metoda Baby-Step/Giant-Step este optimă ca timp de rulare, însă există alți algoritmi mai eficienți d.p.d.v. al complexității spațiu;
- ▶ În cazul algoritmului Pohlig-Hellman, timpul de rulare depinde 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-expoential în lungimea lui  $p$ .
- ▶ Această metodă seamănă cu algoritmul sitei pătratice pentru factorizare;
- ▶ Metoda funcționează în 2 etape; prima etapă este de pre-procesare și necesită cunoașterea modulului  $p$  și a bazei  $g$ ;

## Metoda de calcul a indicelui

- ▶ **Pasul 1.** Fie  $q = p - 1$ , ordinul lui  $\mathbb{Z}_p^*$  și  $B = \{p_1, \dots, p_k\}$  o bază de numere prime mici;
- ▶ Se caută  $l \geq k$  numere distincte  $x_1, \dots, x_l \in \mathbb{Z}_q$  pentru care  $g_i = [g^{x_i} \text{ mod } p]$  este "mic" aşa încât toți factorii primi ai lui  $g_i$  se găsesc în  $B$ ;
- ▶ Vor rezulta relații de forma

$$g^{x_i} = p_1^{e_{i,1}} \cdot p_2^{e_{i,2}} \cdots \cdots p_k^{e_{i,k}} \text{ mod } p$$

unde  $1 \leq i \leq k$ .

- ▶ sau

$$x_i = e_{i,1} \log_g p_1 + e_{i,2} \log_g p_2 + \cdots + e_{i,k} \log_g p_k \text{ mod } p - 1$$

## Metoda de calcul a indicelui

- ▶ În ecuațiile de mai sus, necunoscutele sunt valorile  $\{\log_g p_i\}$
- ▶ **Pasul 2.** Se dă  $y$  pentru care se caută  $\log_g y$ ;
- ▶ Se găsește o valoare  $s \in \mathbb{Z}_q$  pentru care  $g^s \cdot y \text{ mod } p$  este "mic" și poate fi factorizat peste baza  $B$ , obținându-se o relație de forma

$$g^s \cdot y = p_1^{s_1} \cdot p_2^{s_2} \cdot \dots \cdot p_k^{s_k} \text{ mod } p$$

- ▶ sau

$$s + \log_g y = s_1 \log_g p_1 + s_2 \log_g p_2 + \dots + s_k \log_g p_k \text{ mod } p - 1$$

unde  $s$  și  $s_i$  se cunosc;

## Metoda de calcul a indicelui

- ▶ În combinație cu ecuațiile din slide-ul anterior, sunt în total  $l + 1 \geq k + 1$  ecuații liniare în  $k + 1$  necunoscute  $\log_g p_i$ , pentru  $i = 1, \dots, k$  și  $\log_g y$ .
- ▶ O variantă optimizată a acestei metode rulează în timp  $2^{\mathcal{O}(\sqrt{n \cdot \log n})}$  pentru un grup  $\mathbb{Z}_p^*$  cu  $p$  prim de lungime  $n$ .
- ▶ Algoritmul este sub-exponențial în lungimea lui  $p$ .

## Exemplu

- ▶ Fie  $p = 101$ ,  $g = 3$  și  $y = 87$ . Se știe că

$$3^{10} = 65 \text{ mod } 101 \text{ și } 65 = 5 \cdot 13$$

La fel,

$$3^{12} = 2^4 \cdot 5 \text{ mod } 101$$

și

$$3^{14} = 13 \text{ mod } 101$$

- ▶ Prin urmare:

$$10 = \log_3 5 + \log_3 13 \text{ mod } 100$$

$$12 = 4 \cdot \log_3 2 + \log_3 5 \text{ mod } 100$$

$$14 = \log_3 13 \text{ mod } 100$$

- ▶ Baza de numere prime mici este:

$$B = \{2, 5, 13\}$$

## Metoda de calcul a indicelui

- De asemenea,  $3^5 \cdot 87 = 32 = 2^5 \pmod{101}$  sau

$$5 + \log_3 87 = 5 + \log_3 2 \pmod{100}$$

- Combinând primele 3 ecuații, rezultă că  $4 \cdot \log_3 2 = 16 \pmod{100}$ ;
- Această relație nu determină  $\log_3 2$  unic dar se găsesc 4 valori posibile: 4, 29, 54 și 79.
- Prin încercări, se găsește  $\log_3 2 = 39$ , și deci  $\log_3 87 = 40$ .

## Funcții hash rezistente la coliziuni

- ▶ În cadrul criptografiei simetrice, am văzut construcții euristice de funcții hash rezistente la coliziuni care sunt folosite pe larg în practică;
- ▶ În continuare prezentăm o construcție pentru funcții hash rezistente la coliziuni bazată pe PLD (rezumpția logaritmului discret) în grupuri de ordin prim;
- ▶ Construcția este mai puțin eficientă în practică ....
- ▶ ... însă arată că e posibil a obține rezistență la coliziuni pe baza unor rezumări criptografice standard și bine studiate.

## Funcții hash rezistente la coliziuni

- ▶ Fie  $\mathbb{G}$  un grup ciclic de ordin prim  $q$  (cu  $|q| = n$ ) și  $g$  un generator al său iar  $h$  un element aleator din  $\mathbb{G}$ ;
- ▶ Definim o funcție hash  $H$  cu intrarea de lungime fixă după cum urmează:
- ▶  $H$ : pentru intrarea  $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$

$$H(x_1, x_2) = g^{x_1} h^{x_2}$$

# Funcții hash rezistente la coliziuni

## Teoremă

*Dacă problema logaritmului discret este dificilă în grupul  $\mathbb{G}$ , atunci construcția de mai sus este o funcție hash de intrare fixă rezistentă la coliziuni.*

- ▶ Pentru demonstrație vom folosi abordarea reducționistă:
- ▶ Arătăm că o construcție criptografică e sigură atâtă timp cât problema pe care se bazează e dificilă, în felul următor:
- ▶ Prezentăm o reducție explicită arătând cum putem transforma un adversar eficient  $\mathcal{A}$  care atacă securitatea construcției cu probabilitate ne-neglijabilă într-un algoritm eficient  $\mathcal{A}'$  care rezolvă problema dificilă;

## Demonstrație

- ▶ Fie  $H$  o funcție hash precum cea din construcția de mai sus și  $\mathcal{A}$  un adversar PPT; notăm cu

$$\epsilon(n) = \Pr[\text{Hash - coll}_{\mathcal{A}, H} = 1]$$

probabilitatea ca  $\mathcal{A}$  să găsească coliziuni în funcția  $H$ ;

- ▶ Aratăm că  $\mathcal{A}$  poate fi folosit de  $\mathcal{A}'$  pentru a rezolva DLP cu probabilitate de succes  $\epsilon(n)$ ;

## Demonstrație

- ▶ **Algoritmul  $\mathcal{A}'$**  primește la intrare  $s = (\mathbb{G}, q, g, h)$ .
  1. Execută  $\mathcal{A}(s)$  și obține  $x$  și  $x'$ ;
  2. Dacă  $x \neq x'$  și  $H(x) = H(x')$  atunci
    - 2.1 Dacă  $h = 1$  întoarce 0;
    - 2.2 Altfel ( $h \neq 1$ ), notează  $x = (x_1, x_2)$  și  $x' = (x'_1, x'_2)$ . Întoarce  $[(x_1 - x'_1) \cdot (x'_2 - x_2)^{-1} \bmod q]$ .
  - ▶ Clar,  $\mathcal{A}'$  rulează în timp polinomial;
  - ▶ Verificăm faptul că dacă  $\mathcal{A}$  găsește o coliziune,  $\mathcal{A}'$  întoarce răspunsul corect  $\log_g h$ :

## Demonstrație

- Dacă  $h = 1$ , atunci răspunsul lui  $\mathcal{A}'$  este corect pentru că  $\log_g h = 0$ ;
- Altfel, existența unei coliziuni implică:

$$\begin{aligned} H(x_1, x_2) = H(x'_1, x'_2) &\Rightarrow g^{x_1} h^{x_2} = g^{x'_1} h^{x'_2} \\ &\Rightarrow g^{x_1 - x'_1} = h^{x'_2 - x_2} \end{aligned}$$

- Notăm  $\Delta = x'_2 - x_2$ .
- Observăm că  $\Delta \neq 0 \bmod q$ . **De ce?**
- Pentru că ar însemna că  $[(x_1 - x'_1) \bmod q] = 0$  și deci  $x = (x_1, x_2) = (x'_1, x'_2) = x'$ , contradicție cu  $x \neq x'$ ;

## Demonstrație

- ▶ Cum  $q$ -prim și  $\Delta \neq 0 \text{ mod } q$  atunci inversul  $[\Delta^{-1} \text{ mod } q]$  există;
- ▶ Deci  $g^{(x_1 - x'_1) \cdot \Delta^{-1}} = (h^{x'_2 - x_2})^{\Delta^{-1} \text{ mod } q} = h^{\Delta \cdot \Delta^{-1} \text{ mod } q} = h$
- ▶ rezultă că  $\log_g h = [(x_1 - x'_1) \cdot \Delta^{-1} \text{ mod } q] = [(x_1 - x'_1) \cdot (x_2 - x'_2)^{-1} \text{ mod } q]$
- ▶ Observăm că  $\mathcal{A}'$  rezolvă DLP corect cu probabilitate exact  $\epsilon(n)$
- ▶ Cum DLP este dificilă din ipoteză, concluzionăm că  $\epsilon(n)$  este neglijabilă.

## Important de reținut!

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



# Criptografie și Securitate

- Prelegerea 22.1 -  
Schimbul de chei Diffie-Hellman

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definiție
2. Schimbul de chei Diffie-Hellman
3. Securitate

## 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. semnături digitale
  3. schimb de chei
- ▶ Deși au introdus 3 concepte diferite, Diffie și Hellman au introdus o singură construcție, pentru schimbul de chei.

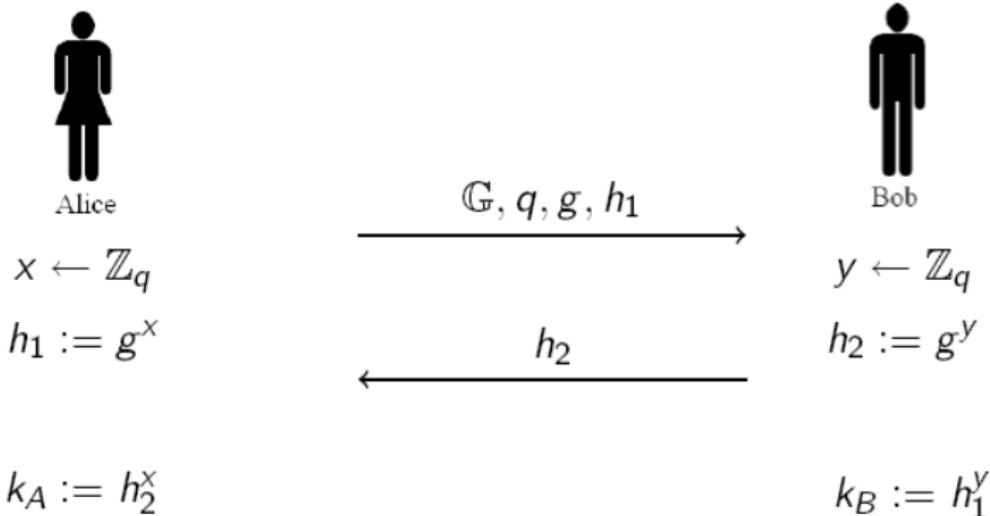
## Schimb de chei

- ▶ **Sistemele de criptare cu cheie publică** le-am studiat și le vom mai studia în detaliu;
- ▶ **Semnăturile digitale** sunt analogul MAC-urilor din criptografia simetrică (sau corespondentul digital unei semnături 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 partajează încă prealabil nici un secret pot genera o cheie comună, secretă;
- ▶ Comunicarea necesară pentru stabilirea cheii se realizează printr-un canal public!
- ▶ Deci un adversar poate intercepta toate mesajele transmise pe canalul de comunicație, dar NU trebuie să afle nimic despre cheia secretă obținută în urma protocolului;
- ▶ Protocolele de schimb de chei reprezintă o primitivă fundamentală în criptografie;
- ▶ În continuare, ne vom rezuma strict la **schimbul de chei Diffie-Hellman**.

# Schimbul de chei Diffie-Hellman



## 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 trimitе 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 trimitе  $h_2$  lui Alice și întoarce cheia  $k_B := h_1^y$ ;
- ▶ Alice primește  $h_2$  și întoarce cheia  $k_A = h_2^x$ .

## Schimbul de chei Diffie-Hellman

- ▶ 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

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

## Securitate

- ▶ 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 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 (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 așa 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 până acum securitatea față de atacatori pasivi;
- ▶ Arătăm acum că 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 comunicație;
- ▶ Un astfel de adversar se poate interpune între Alice și Bob, dând naștere unui atac de tip **Man-in-the-Middle**.

# Atacul Man-in-the-Middle



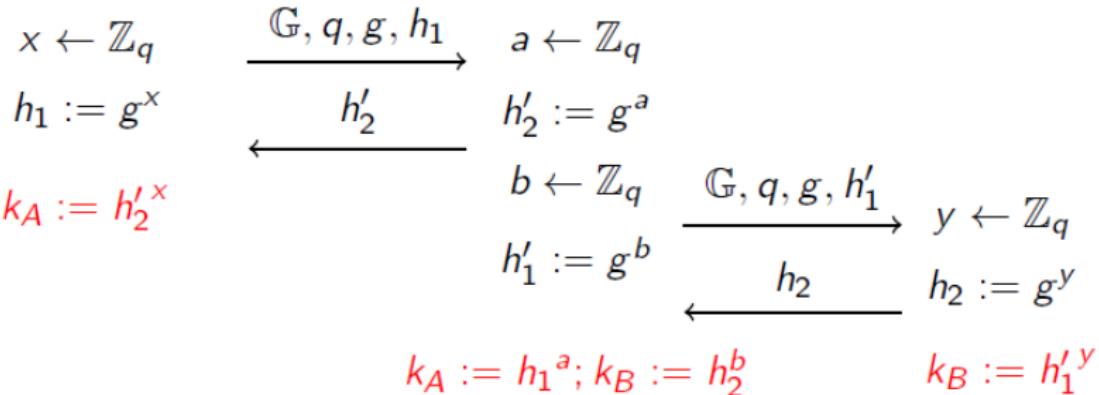
Alice



Oscar



Bob



## Atacul 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 Man-in-the-Middle

- ▶ Atacul este posibil pentru că poate **impersona** pe Alice și pe Bob;
- ▶ De fiecare dată când Alice va transmite un mesaj criptat către Bob, Oscar îl interceptează și îl previne să ajungă la Bob;
- ▶ Oscar îl decriptează folosind  $k_A$ , apoi îl recriptează folosind  $k_B$  și îl transmite către Bob;
- ▶ Alice și Bob comunică fără să fie conștienți de existența lui Oscar.

## Important de reținut!

- ▶ Schimbul de chei - o primitivă criografică importantă
- ▶ Prezumții criptografice: CDH, DDH
- ▶ Schimbul de chei Diffie-Hellman nu rezistă la atacuri active



# Criptografie și Securitate

- Prelegerea 22.2 -  
Sistemul de criptare ElGamal

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Scurt istoric
2. Sistemul de criptare ElGamal
3. Securitate

# 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

- ▶ Se bazează pe DLP ...
- ▶ ... sau mai exact pe dificultatea problemei DDH...
- ▶ ... și pe următoarea observație simplă:

### Observație

*Fie  $\mathbb{G}$  un grup finit și  $m \leftarrow^R \mathbb{G}$ . Dacă  $g \leftarrow^R \mathbb{G}$ , atunci  $g' = m \cdot g$  rămâne aleator în  $\mathbb{G}$ :*

$$\Pr[m \cdot g = g'] = 1/|\mathbb{G}|$$

*unde probabilitatea este dată de alegerea aleatoare a lui  $g$ .*

## Sistemul de criptare ElGamal

- Dacă emitătorul și receptorul folosesc  $g$  drept cheie secretă, atunci un mesaj  $m \in \mathbb{G}$  se criptează ca:

$$g' = m \cdot g$$

- Receptorul decriptează:

$$m = g' \cdot g^{-1}$$

- Abordarea este asemănătoare cu OTP, unde se folosea grupul secvențelor de lungime fixată împreună cu operația XOR;
- O astfel de construcție este deci perfect sigură (dacă  $g$  este total aleator!).

## Sistemul de criptare ElGamal

- ▶ În criptografia cu cheie publică, se folosește  $g$  **pseudoaleator**, deci se pierde securitatea perfectă;
- ▶ Ideea de bază este alegerea lui  $g$  astfel încât la recepție să poată fi *calculat* pe baza cheii secrete...
- ▶ ... dar  $g$  să *pară aleator* pentru un adversar;
- ▶ Pentru aceasta se folosește prezumția DDH, construcția fiind imediată din schimbul de chei Diffie-Hellman.

# Sistemul de criptare ElGamal

- ▶ 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  $(\mathbb{G}, q, g, x)$ ;
  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  $m = c_2 \cdot c_1^{-x}$ .

# Securitate - Problema 1

## 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  $y \neq y'$ :

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

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

- ▶ În caz contrar, sistemul NU ar putea fi CPA-sigur.

## Securitate - Problema 2

### 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.

## Securitate - Problema 3

### 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)

## Securitate - Problema 4

### 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.

# Securitate - teoremă

## Teoremă

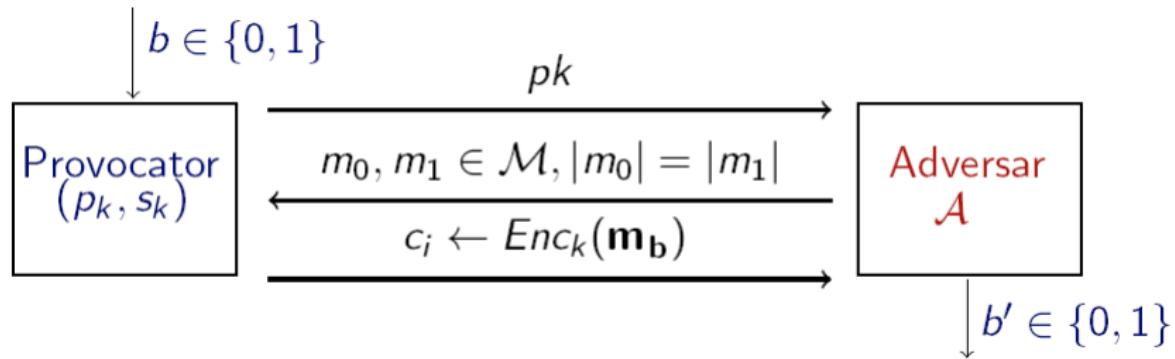
Dacă problema decizională Diffie-Hellman (DDH) este dificilă în grupul  $\mathbb{G}$ , atunci schema de criptare ElGamal este CPA-sigură.

- ▶ Notăm cu  $\Pi$  schema de criptare ElGamal. E suficient să arătăm că schema este sigură la interceptare simplă;
- ▶ Fie  $\mathcal{A}$  un adversar PPT; notăm cu

$$\epsilon(n) = \Pr[\text{PubK}_{\mathcal{A}, \Pi}^{eav}(n) = 1]$$

probabilitatea ca  $\mathcal{A}$  să câștige experimentul de mai jos folosit pentru a defini securitatea la interceptare simplă.

## Demonstrație



- ▶ Considerăm schema modificată  $\tilde{\Pi}$  care diferă de schema  $\Pi$  prin faptul că algoritmul de criptare alege aleator  $y, z \leftarrow \mathbb{Z}_q$  și întoarce textul criptat

$$(g^y, g^z \cdot m)$$

## Demonstrație

- ▶ A doua componentă a textului criptat din  $\tilde{\Pi}$  este un element uniform distribuit din  $\mathbb{G}$  și independent de  $m$ ;
- ▶ Prima componentă este și ea independentă de  $m$ ; rezultă că

$$\Pr_{\mathcal{A}, \tilde{\Pi}}[PubK_{\mathcal{A}, \tilde{\Pi}}^{eav}(n) = 1] = \frac{1}{2}$$

- ▶ Deși  $\tilde{\Pi}$  nu e o schemă de criptare (nu se poate decripta), experimentul  $PubK_{\mathcal{A}, \tilde{\Pi}}^{eav}(n)$  este bine-definit pentru că folosește doar algoritmul de criptare;
- ▶ Aratăm că  $\mathcal{A}$  poate fi folosit de un algoritm  $\mathcal{D}$  ca o subrutină pentru a rezolva problema DDH cu probabilitate  $\epsilon(n)$ ;

## Demonstrație

- ▶ Algoritmul  $\mathcal{D}$  primește la intrare tuplul  $(\mathbb{G}, q, g, g_1, g_2, g_3)$ , unde  $g_1 = g^x$ ,  $g_2 = g^y$  și  $g_3 = g^{xy}$  sau  $g_3 = g^z$  pentru  $x, y, z$  aleatoare, după care:
  1. Alege  $pk = (\mathbb{G}, q, g, g_1)$  și execută  $\mathcal{A}(pk)$  și obține două mesaje  $m_0$  și  $m_1$ ;
  2. Alege un bit aleator  $b$  și notează  $c_1 = g_2$  și  $c_2 = g_3 \cdot m_b$ ;
  3. Îl dă textul criptat  $(c_1, c_2)$  lui  $\mathcal{A}$  și obține de la el un bit  $b'$ . Dacă  $b' = b$ ,  $\mathcal{D}$  întoarce 1, altfel întoarce 0.
- ▶ În continuare, analizăm comportamentul lui  $\mathcal{D}$  considerând două cazuri:

## Demonstrație

- ▶ **Cazul 1:** Să presupunem că tuplul pe care  $\mathcal{D}$  îl primește la intrare este generat alegând aleator  $x, y, z \leftarrow \mathbb{Z}_q$  și calculând  $g_1 = g^x, g_2 = g^y$  și  $g_3 = g^z$ .
- ▶ Atunci  $\mathcal{D}$  execută  $\mathcal{A}$  cu cheia publică  $pk = (\mathbb{G}, q, g, g^x)$  și textul criptat construit  $(c_1, c_2) = (g^y, g^z \cdot m_b)$ ;
- ▶ În acest caz,  $\mathcal{A}$  nu poate distinge între cele două situații: atunci când este executat ca o subrutină a lui  $\mathcal{D}$  interacționând cu el sau atunci când efectuează experimentul  $PubK_{\mathcal{A}, \tilde{\Pi}}^{eav}(n)$
- ▶ Cum  $\mathcal{D}$  întoarce 1 exact atunci când output-ul  $b'$  al lui  $\mathcal{A}$  este egal cu  $b$ , rezultă că:

## Demonstrație

$$\Pr[D(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] = \Pr[PubK_{\mathcal{A}, \tilde{\Pi}}^{eav}(n) = 1] = \frac{1}{2}$$

- ▶ **Cazul 2:** Să presupunem că tuplul pe care  $\mathcal{D}$  îl primește la intrare este generat alegând aleator  $x, y \leftarrow \mathbb{Z}_q$  și calculând  $g_1 = g^x, g_2 = g^y$  și  $g_3 = g^{xy}$ .
- ▶ Atunci  $\mathcal{D}$  execută  $\mathcal{A}$  cu cheia publică  $pk = (\mathbb{G}, q, g, g^x)$  și textul criptat construit

$$(c_1, c_2) = (g^y, g^{xy} \cdot m_b) = (g^y, (g^x)^y \cdot m_b)$$

- ▶ În acest caz,  $\mathcal{A}$  nu poate distinge între următoarele două situații: atunci când este executat ca o subrutină a lui  $\mathcal{D}$  sau atunci când efectuează experimentul  $PubK_{\mathcal{A}, \Pi}^{eav}(n)$

## Demonstrație

- ▶ Cum  $\mathcal{D}$  întoarce 1 exact atunci când output-ul  $b'$  al lui  $\mathcal{A}$  este egal cu  $b$ , rezultă că:

$$\Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] = \Pr[PubK_{\mathcal{A}, \Pi}^{eav}(n) = 1] = \epsilon(n)$$

- ▶ Dar cum problema DDH este dificilă, rezultă că există o funcție neglijabilă negl a.î.

$$\text{negl}(n) \geq$$

$$|\Pr[D(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]|$$

$$= \left| \frac{1}{2} - \epsilon(n) \right|$$

- ▶ Adică  $\epsilon(n) \leq \frac{1}{2} + \text{negl}(n)$

□.

## Important de reținut!

- ▶ Sistemul de criptare ElGamal
- ▶ Proprietatea de homomorfism



# Criptografie și Securitate

- Prelegherea 23 -

## Criptografia bazată pe curbe eliptice

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Definirea curbelor eliptice
2. ECDLP - Problema logaritmului discret pe curbe eliptice
3. ECC- Criptografia bazată pe curbe eliptice

## Grupuri ciclice pentru uz criptografic

- ▶ În cursul precedent am discutat despre grupuri ciclice și am subliniat faptul că sunt de preferat, pentru criptografie, grupurile ciclice de ordin prim;
- ▶ Am menționat că, de obicei, se lucrează în grupul  $\mathbb{Z}_p^*$  cu  $p$  prim iar un subgrup de ordin prim al lui este format din mulțimea resturilor pătratice modulo  $p$ ;
- ▶ În continuare introducem o altă clasă de grupuri care constă din **punctele unei curbe eliptice**;
- ▶ Aceste grupuri sunt folosite în criptografie pentru că, spre deosebire de  $\mathbb{Z}_p^*$ , nu se cunoaște deocamdată nici un algoritm sub-exponențial pentru rezolvarea DLP în aceste grupuri.

# 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

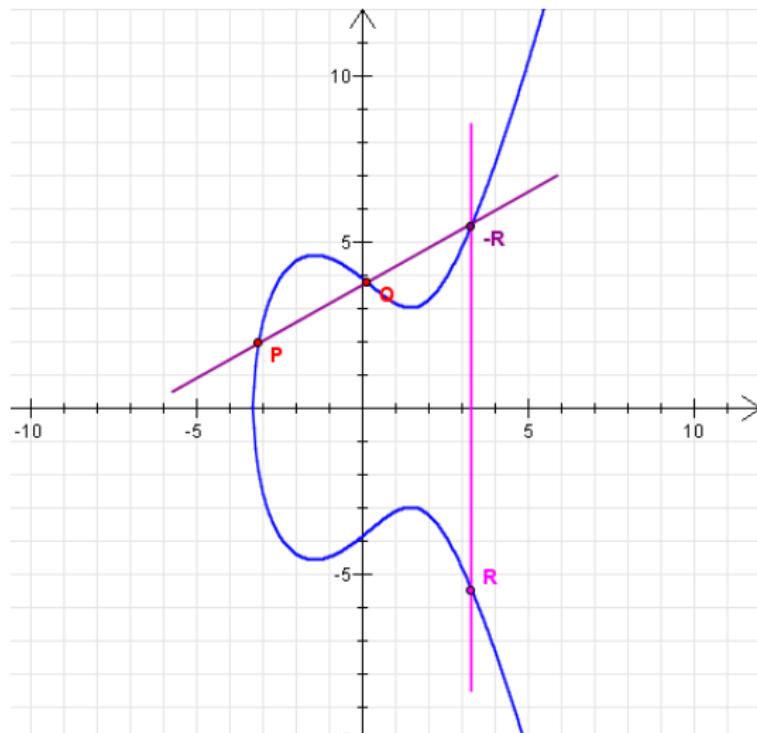
$A, B \in \mathbb{Z}_p$  sunt constante care respectă  $4A^3 + 27B^2 \neq 0 \text{ mod } p$

- Vom nota cu  $E(\mathbb{Z}_p)$  o curbă eliptică definită peste  $\mathbb{Z}_p$

# Curbe eliptice

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  $y_2 = -y_1$ ,  $P + Q = \mathcal{O}$

## Grupul punctelor de pe o curba eliptică

- ▶ altfel,  $P + Q = R$  de coordonate  $(x_3, y_3)$  care se calculează astfel:

$$x_3 = [m^2 - x_1 - x_2 \bmod p]$$
$$y_3 = [m(x_1 - x_3) - y_1 \bmod p]$$

- ▶ iar  $m$  se calculează astfel:

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \bmod p & \text{dacă } P \neq Q \\ \frac{3x_1 + A}{2y_1} \bmod p & \text{dacă } P = Q \end{cases}$$

## Grupul punctelor de pe o curba eliptică

- ▶ Geometric, suma a două puncte  $P$  și  $Q$  se obține trasând o linie prin cele două puncte și găsind cel de-al treilea punct  $R$  de intersecție al liniei cu  $E$ ;
- ▶ În această situație,  $m$  reprezintă panta dreptei care trece prin  $P$  și  $Q$ ;
- ▶ Se poate arăta că mulțimea de puncte  $E(\mathbb{Z}_p)$  împreună cu operația aditivă definită formează un grup abelian;
- ▶ Există o teoremă de structură pentru  $E(\mathbb{Z}_p)$  care exprimă condițiile în care grupul este ciclic.

# Grupul punctelor de pe o curba eliptică

## Teoremă

Fie  $E$  o curbă eliptică peste  $\mathbb{Z}_p$  cu  $p > 3$  număr prim. Atunci există două numere întregi  $n_1$  și  $n_2$  aşa încât

$$E(\mathbb{Z}_p) \approx \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$$

unde  $n_2|n_1$  și  $n_2|(p - 1)$

- ▶ Consecință: dacă  $n_2 = 1$ , atunci  $E(\mathbb{Z}_p)$  este ciclic;
- ▶ În practică, sunt căutate acele curbe eliptice pentru care ordinul grupului ciclic generat este prim;
- ▶ O curbă eliptică definită peste  $\mathbb{Z}_p$  are aproximativ  $p$  puncte.  
Mai precis [Teorema lui Hasse]:

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

## 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:  $\underbrace{P + P + \dots + P}_{s \text{ ori}} = sP$ .

## ECDLP - Securitate

- ▶ 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  $2^{160}$ .

## ECDLP - Securitate

- ▶ 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.

# Criptografia pe curbe eliptice - ECC (Elliptic Curve Cryptography)

- ▶ A fost inventată independent în 1987 de Neal Koblitz și în 1986 de Victor Miller;
- ▶ La începutul anilor 1990 se făceau foarte multe speculații despre securitatea și practicalitatea ECC, mai ales comparativ cu RSA;
- ▶ După cercetări intensive, ECC pare foarte sigură, la fel de sigură precum RSA sau schemele bazate pe DLP;
- ▶ Încrederea a crescut după ce în 1999 și 2001 au fost standardizate, pentru domeniul bancar, semnături digitale și schimburi de chei bazate pe curbe eliptice.

# Criptografia pe curbe eliptice - ECC (Elliptic Curve Cryptography)

- ▶ Curbele eliptice sunt folosite pe larg și în standardele comerciale precum IPsec sau TLS;
- ▶ ECC este adesea preferată în fața criptografiei cu cheie publică pentru sistemele încorporate precum dispozitivele mobile...
- ▶ ...din motive de performanță;
- ▶ implementările pentru ECC sunt considerabil mai mici și mai rapide decât cele pentru RSA;
- ▶ ECC cu chei pe 160-250 biți oferă cam același nivel de securitate precum RSA sau sistemele bazate pe DLP cu chei pe 1024-3072 biți.

# Comparație între ECC, criptografia simetrică și asimetrică

Chei criptografia simetrică	Chei RSA	Chei ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

**Tabel:** Dimensiunile cheilor recomandate de NIST

## Important de reținut!

- ▶ Curbele eliptice oferă un suport bun pentru criptografie;
- ▶ ECDLP este dificilă.



# Criptografie și Securitate

- Prelegerea 23.1 -

Schimbul de chei Diffie-Hellman pe curbe eliptice

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

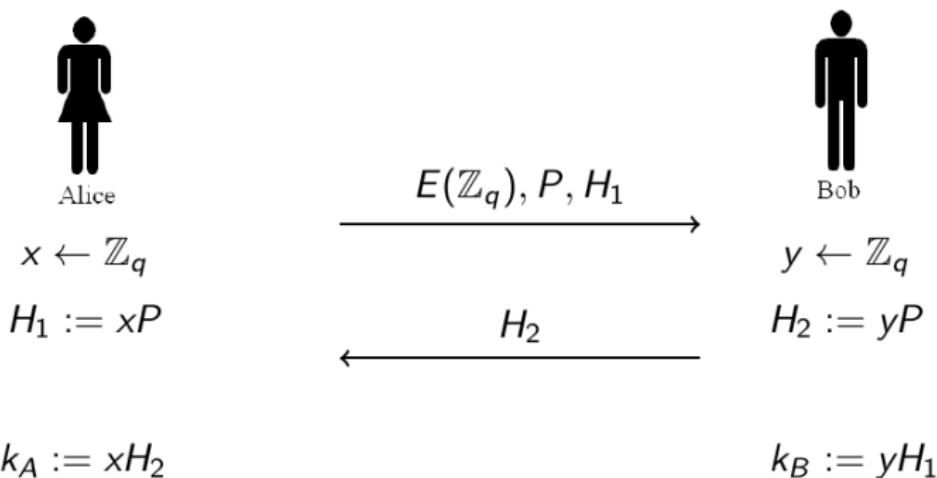
1. Schimbul de chei Diffie-Hellman pe curbe eliptice
2. Securitate

## Schimbul de chei Diffie-Hellman pe curbe eliptice

- ▶ Am studiat schimbul de chei Diffie-Hellman peste un grup ciclic  $\mathbb{G}$ , de ordin  $q$ ;
- ▶ Transpunem construcția pe curbe eliptice:

$$(\mathbb{G}, \cdot) \rightarrow (E(\mathbb{Z}_q), +)$$

# Schimbul de chei Diffie-Hellman pe curbe eliptice



## 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 trimitе 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 trimitе  $H_2$  lui Alice și întoarce cheia  $k_B := yH_1$ ;
- ▶ Alice primește  $H_2$  și întoarce cheia  $k_A = xH_2$ .

## Schimbul de chei Diffie-Hellman pe curbe eliptice

- ▶ 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

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

## Securitate

- ▶ 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  $H_1, H_2 \xleftarrow{R} E(\mathbb{Z}_q)$ , să se determine:

$$ECCDH(H_1, H_2) = (ECDLP(P, H_1)ECDLP(P, H_2))P$$

- ▶ 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).

## ECDDH (Elliptic Curve 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 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

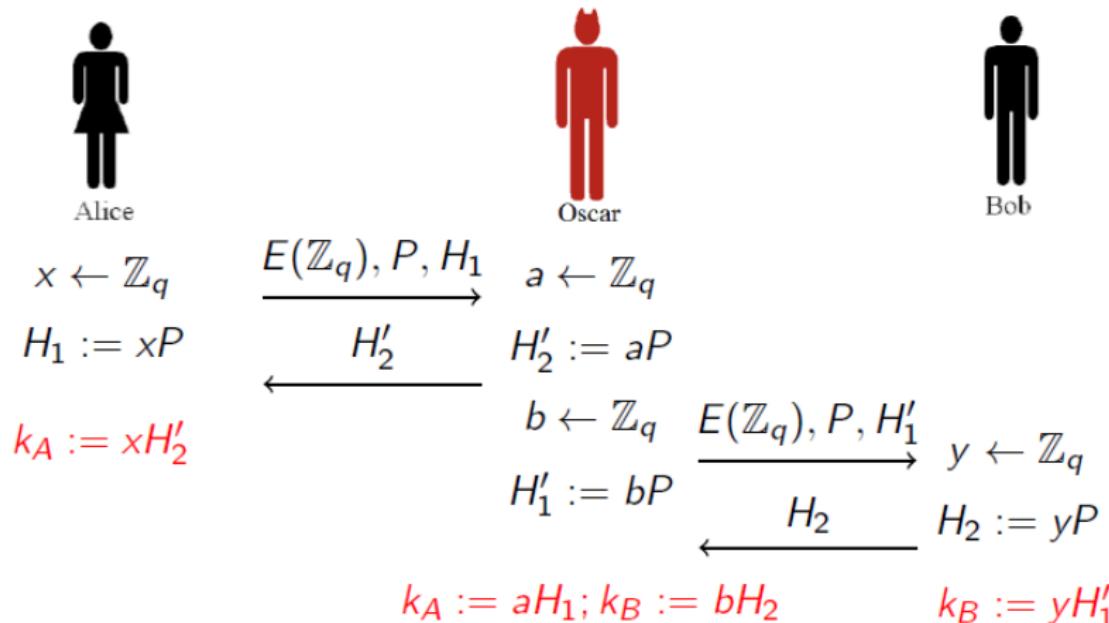
*Spunem că problema decizională Diffie-Hellman (ECDDH) este dificilă (relativ la curba eliptică  $E(\mathbb{Z}_q)$ ), dacă pentru orice algoritm PPT  $\mathcal{A}$  există o funcție neglijabilă negl aşa încât:*

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

## Atacul Man-in-the-Middle

- ▶ Am analizat până acum securitatea față de atacatori pasivi;
- ▶ Arătăm acum că 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 comunicație;
- ▶ Un astfel de adversar se poate interpune între Alice și Bob, dând naștere unui atac de tip **Man-in-the-Middle**.

# Atacul Man-in-the-Middle



## Atacul Man-in-the-Middle

- ▶ Alice generează o curbă eliptică  $E(\mathbb{Z}_q)$  și  $P$  un punct pe curbă;
- ▶ Alice alege  $x \leftarrow^R \mathbb{Z}_q$  și calculează  $H_1 := xP$ ;
- ▶ Alice îi trimită lui Bob mesajul  $(E(\mathbb{Z}_q), P, H_1)$ ;
- ▶ Oscar interceptează mesajul și răspunde lui Alice în locul lui Bob: alege  $a \leftarrow^R \mathbb{Z}_q$  și calculează  $H'_2 := aP$ ;
- ▶ Oscar și Alice dețin acum cheia comună  $k_A = axP$ ;
- ▶ Oscar inițiază, în locul lui Alice, o nouă sesiune cu Bob: alege  $b \leftarrow^R \mathbb{Z}_q$  și calculează  $H'_1 := bP$ ;
- ▶ Bob alege  $y \leftarrow^R \mathbb{Z}_q$  și calculează  $H_2 := yP$ ;
- ▶ Oscar și Bob dețin acum cheia comună  $k_B = ybP$ .

## Atacul Man-in-the-Middle

- ▶ Atacul este posibil pentru că poate **impersona** pe Alice și pe Bob;
- ▶ De fiecare dată când Alice va transmite un mesaj criptat către Bob, Oscar îl interceptează și îl previne să ajungă la Bob;
- ▶ Oscar îl decriptează folosind  $k_A$ , apoi îl recriptează folosind  $k_B$  și îl transmite către Bob;
- ▶ Alice și Bob comunică fără să fie conștienți de existența lui Oscar.

## 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  $q$



# Criptografie și Securitate

- Prelegerea 23.2 -

Sistemul de criptare ElGamal pe curbe eliptice

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Sistemul de criptare ElGamal pe curbe eliptice
2. Securitate

## Sistemul de criptare ElGamal pe curbe eliptice

- ▶ Am studiat sistemul de criptare ElGamal peste un grup ciclic  $\mathbb{G}$ , de ordin  $q$ ;
- ▶ Transpunem construcția pe curbe eliptice:

$$(\mathbb{G}, \cdot) \rightarrow (E(\mathbb{Z}_q), +)$$

# Sistemul de criptare ElGamal pe curbe eliptice

1. Se generează  $E(\mathbb{Z}_q)$  o curbă eliptică și  $P$  un punct pe curbă (generator), se alege  $x \leftarrow^R \mathbb{Z}_q$  și se calculează  $H = xP$ ;
  - ▶ Cheia publică este:  $(E(\mathbb{Z}_q), P, H)$ ;
  - ▶ Cheia privată este  $(E(\mathbb{Z}_q), P, x)$ ;
2. **Enc:** dată o cheie publică  $(E(\mathbb{Z}_q), P, H)$  și un mesaj  $M \in E(\mathbb{Z}_q)$ , alege  $y \leftarrow^R \mathbb{Z}_q$  și întoarce  $C = (C_1, C_2) = (yP, M + yH)$ ;
3. **Dec:** dată o cheie secretă  $(E(\mathbb{Z}_q), P, x)$  și un mesaj criptat  $C = (C_1, C_2)$ , întoarce  $M = C_2 + x(-C_1)$ .

- ▶ Sistemul transpus pe curbe eliptice păstează proprietățile sistemului inițial;
- ▶ Deci curba eliptică trebuie aleasă a.î. ECDLP și ECDDH să fie dificile...
- ▶ ... și sistemul rămâne nedeterminist și homomorfic.

## Important de reținut!

- ▶ Sistemul de criptare ElGamal pe curbe eliptice



# Criptografie și Securitate

- Prelegherea 24 -  
Semnături digitale

Adela Georgescu, Ruxandra F. Olimid

Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Scheme de semnătură digitală
2. Infrastructură cu chei publice - PKI

## Scheme de semnatură digitală

- ▶ Schemele de semnatură digitală reprezintă echivalentul MAC-urilor în criptografia cu cheie publică, deși există câteva diferențe importante între ele;
- ▶ O schemă de semnatură digitală îi permite unui semnatar  $S$  care a stabilit o cheie publică  $pk$  să semneze un mesaj în aşa fel încât oricine care cunoaște cheia  $pk$  poate verifica originea mesajului (ca fiind  $S$ ) și integritatea lui;
- ▶ De pildă, o companie de software vrea să transmită patch-uri de software într-o manieră autentificată, aşa încât orice client să poată recunoaște dacă un patch este autentic;
- ▶ În schimb, o persoană malicioasă nu poate păcăli un client să accepte un patch care a nu a fost realizat de companie respectivă.

## Scheme de semnatură digitală

- ▶ Pentru aceasta, compania generează o cheie publică  $pk$  împreună cu o cheie secretă  $sk$  și distribue cheia  $pk$  clienților săi, păstrând cheia secretă;
- ▶ Atunci când lansează un patch de software  $m$ , compania calculează o semnătură digitală  $\sigma$  pentru  $m$  folosind cheia  $sk$  și trimit fiecarui client perechea  $(m, \sigma)$ ;
- ▶ Fiecare client stabilește autenticitatea lui  $m$  verificând dacă  $\sigma$  este o semnătură legitimă pentru  $m$  cu privire la cheia publică  $pk$ ;
- ▶ Deci compania folosește aceeași cheie publică pentru toți clienții și calculează o singură semnătură pe care o trimite tuturor.

## Avantaje semnături digitale față de MAC-uri

- ▶ MAC-urile și schemele de semnătură digitală sunt folosite pentru asigurarea integrității (autenticității) mesajelor cu următoarele **diferențe**:
- ▶ Schemele de semnătură digitală sunt *public verificabile*...
- ▶ ...ceea ce înseamnă că semnăturile digitale sunt *transferabile* - o terță parte poate verifica legitimitatea unei semnături și poate face o copie pentru a convinge pe altcineva că aceea este o semnătură validă pentru  $m$ ;
- ▶ Schemele de semnătură digitală au proprietatea de *non-repudiere* - un semnatar nu poate nega faptul că a semnat un mesaj;
- ▶ MAC-urile au avantajul că sunt cam de 2-3 ori mai eficiente (mai rapide) decât schemele de semnătură digitală.

## Construcție scheme de semnătură digitală

- ▶ Se pot construi scheme de semnătură digitală pe baza problemei RSA (semnarea se face prin decriptare iar verificarea prin criptare) însă acestea sunt complet nesigure;
- ▶ Paradigma "hash-and-sign" este mai sigură: înainte de semnare, mesajul trece printr-o funcție hash; varianta aceasta se folosește pe larg în practică;
- ▶ Există scheme de semnătură *one-time* care sunt sigure atât timp cât sunt folosite pentru semnarea unui singur mesaj;
- ▶ Un alt exemplu folosit în practică este Digital Signature Algorithm (DSA) bazat pe problema logaritmului discret (a devenit standard US în 1994) dar și ECDSA (varianta DSA bazată pe curbe eliptice devenită standard în 1998), ambele fiind incluse în DSS (Digital Signature Standard).

## 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 semnătura care atașează unei entități o anume cheie publică;

## Certificate și PKI

- ▶ 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 î-o dă lui Bob:

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

- ▶ 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;

## Certificate și PKI

- ▶ Rămân câteva probleme: cum află Alice  $pk_C$ , cum poate fi Charlie sigur că  $pk_B$  este cheia publică a lui Bob, cum decide Alice dacă să aibă încredere în Charlie;
- ▶ Toate aceastea sunt specificate într-o *infrastructură cu chei publice* (PKI-public key infrastructure) care permite distribuirea la scară largă a cheilor publice;
- ▶ Există mai multe modele diferite de PKI, după cum vom vedea în continuare;

## PKI cu o singură autoritate de certificare

- ▶ Aici există o singură autoritate de certificare (CA) în care toată lumea are încredere și care emite certificate pentru toate cheile publice;
- ▶ CA este o companie, sau agenție guvernamentală sau un departament dintr-o organizație;
- ▶ Oricine apelează la serviciile CA trebuie să obțină o copie legitimă a cheii ei publice  $pk_{CA}$ ;
- ▶ Cheia  $pk_{CA}$  se poate obține chiar prin mijloace fizice; deși inconvenient, acest pas este efectuat o singură dată;

## PKI cu mai multe autorități de certificare

- ▶ Modelul cu o singură CA nu este practic;
- ▶ În modelul cu multiple CA, dacă Bob dorește să obțină un certificat pentru cheia lui publică, poate apela la oricare CA dorește, iar Alice, care primește un certificat sau mai multe, poate alege în care CA să aibă încredere;
- ▶ De exemplu, browser-ele web vin preconfigurate cu un număr de chei publice ale unor CA stabilite ca toate fiind de încredere în mod egal (în configurația default a browser-ului);
- ▶ Utilizatorul poate modifica această configurație aşa încât să accepte doar certificate de la CA-uri în care el are încredere;

## Delegare și lanțuri de certificate

- ▶ Charlie este un CA care emite certificate, inclusiv pentru Bob;
- ▶ Dacă  $pk_B$  este o cheie publică pentru semnătură, atunci Bob poate emite certificate pentru alte persoane; un certificat pentru Alice are forma

$$\text{cert}_{B \rightarrow A} = \text{Sign}_{sk_B}("Cheia lui Alice este pk_A")$$

- ▶ Atunci când comunică cu Dan, Alice îi trimite

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

- ▶ De fapt,  $\text{cert}_{C \rightarrow B}$  conține, în afară de  $pk_B$  și afirmația "Bob este de încredere pentru a emite certificate"; astfel, Charlie îl deleagă pe Bob să emită certificate;
- ▶ Totul se poate organiza ca o ierarhie unde există un CA "rădăcină" pe primul nivel și  $n$  CA-uri pe al doilea nivel.

## Modelul "web of trust"

- ▶ Aici oricine poate emite certificate pentru orice altcineva și fiecare utilizator decide cât de multă încredere poate acorda certificatelor emise de alții utilizatori;
- ▶ De exemplu, dacă Alice are cheile publice  $pk_1, pk_2, pk_3$  corespunzătoare lui  $C_1, C_2, C_3\dots$
- ▶ ...iar Bob, care vrea să comunice cu Alice, are certificatele  $\text{cert}_{C_1 \rightarrow B}, \text{cert}_{C_3 \rightarrow B}$  și  $\text{cert}_{C_4 \rightarrow B}$  pe care i le trimit lui Alice;
- ▶ Alice nu are  $pk_4$  și nu poate verifica  $\text{cert}_{C_4 \rightarrow B}$ ; deci ca să accepte  $pk_B$ , Alice trebuie să decidă cât de multă încredere are în  $C_1$  și  $C_3$ ;
- ▶ Modelul este atrăgător pentru că nu necesită încredere într-o autoritate centrală;

## Invalidarea certificatelor

- ▶ Atunci când un angajat părăsește o companie sau își pierde cheia secretă, certificatul lui trebuie invalidat;
- ▶ Există mai multe metode de invalidare între care:
- ▶ **Expirarea.** Se poate include data de expirare ca parte a unui certificat, care trebuie verificată împreună cu validitatea semnăturii;
- ▶ **Revocarea.** CA-ul poate, în mod explicit, revoca un certificat de îndată ce acesta nu mai poate fi folosit;
- ▶ Aceasta se poate realiza prin includerea unui număr serial în certificat; la sfârșitul unei zile CA generează o listă de certificate revocate (care conține numerele seriale) pe care o distribuie sau publică.

## Important de reținut!

- ▶ Semnături electronice
- ▶ Certificate digitale



# Criptografie și Securitate

- Prelegerea finală -

Mai multe despre criptografie

Adela Georgescu, Ruxandra F. Olimid

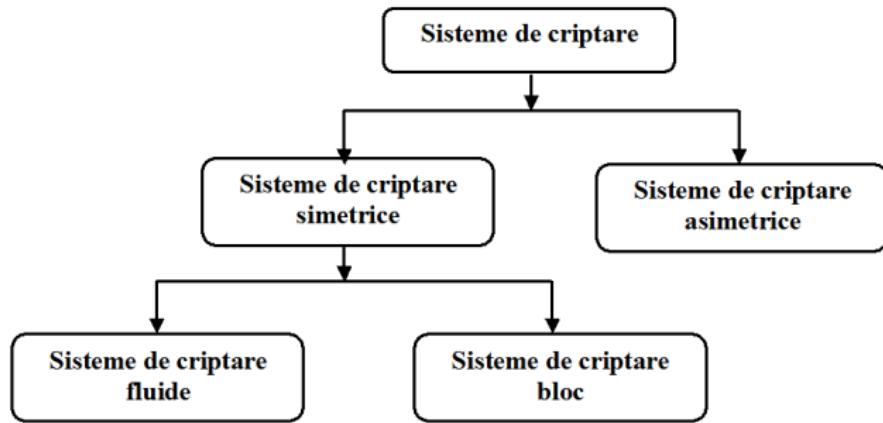
Facultatea de Matematică și Informatică  
Universitatea din București

# Cuprins

1. Scheme de semnătură digitală
2. Infrastructură cu chei publice - PKI

# Primitive criptografice studiate

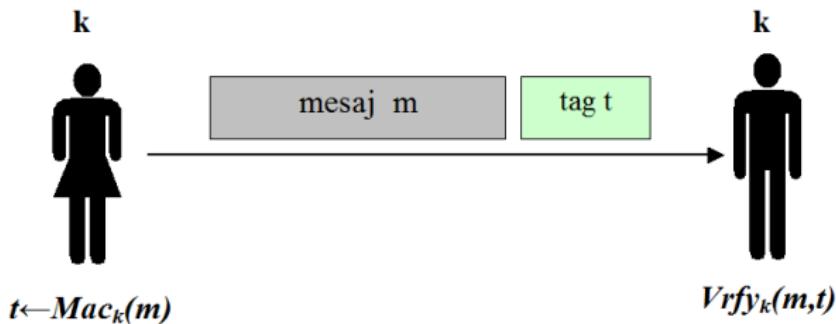
- ▶ Am studiat în timpul cursului **sisteme de criptare**:



- ▶ Acestea au rolul de a asigura **confidențialitatea**.

## Primitive criptografice studiate

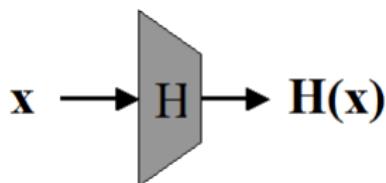
- ▶ Am studiat în timpul cursului construcțiile **MAC**:



- ▶ Acestea au rolul de a asigura **integritatea**.

## Primitive criptografice studiate

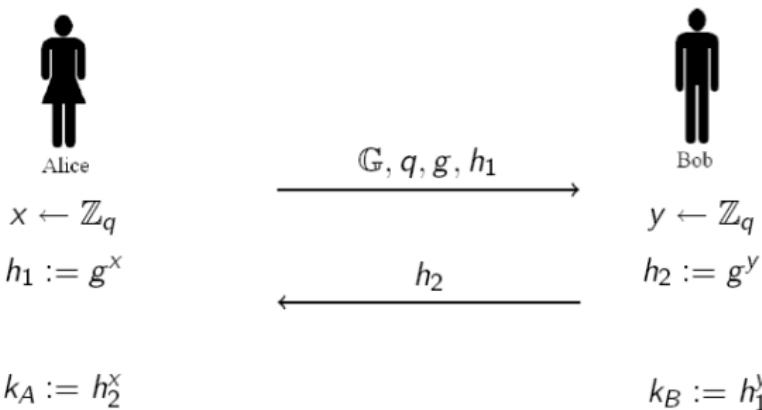
- ▶ Am studiat în timpul cursului funcții **hash**:



- ▶ Acestea asigură **integritatea** și **autentificarea** datelor prin utilizare în MAC-uri, semnături digitale...

## Primitive criptografice studiate

- ▶ Am studiat în timpul cursului, protocoale de **schimb de chei** (Diffie-Hellman):



- ▶ Acestea asigură **stabilirea unei chei comune**, utilizată ulterior în scopuri criptografice (ex. criptare).

## Alte primitive criptografice

- ▶ Am studiat sisteme de criptare care asigură confidențialitatea între 2 participanți;
- ▶ Există însă și sisteme de criptare de tip **broadcast** (**broadcast encryption**):
- ▶ Acestea permit comunicarea criptată (unidirecțională) peste un canal de tip *broadcast* (către toți participanții) a.î. numai participanții autorizați să poată decripta;
- ▶ Exemple de utilizare: transmisiuni TV criptate;
- ▶ Noțiuni similare:
  - ▶ **multicast encryption:** comunicarea criptată este bidirecțională;
  - ▶ **threshold encryption:** pentru decriptare este necesar să coopereze un număr de participanți care să depășească un anumit prag.

## Alte primitive criptografice

- ▶ Am studiat construcțiile MAC care asigură integritatea în criptografia simetrică;
- ▶ În criptografia asimetrică există **semnăturile digitale**, care atestă în plus și originea mesajului.

## Alte primitive criptografice

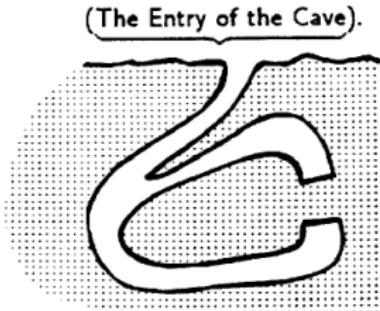
- ▶ Am studiat protocolul de schimb Diffie-Hellman care stabilește o cheie comună între 2 participanți;
- ▶ Există însă și **protocole de stabilire a cheilor de grup**;
- ▶ Acestea permit stabilirea unei chei comune între mai mulți participanți;
- ▶ Exemple de utilizare: comunicație criptată, (video-)conferințe, acces la resurse ...

## Alte primitive criptografice

- ▶ **Schemele de partajare a secretelor** permit partajarea unui secret în mai multe componente distribuite unor participanți astfel încât numai mulțimile *autorizate* de participanți să poată reconstitui secretul;
- ▶ Exemple de utilizare: controlul accesului, stocarea fișierelor în cloud, ...
- ▶ Alte primitive criptografice la nivel de grup: **multiparty computation, protocoale de vot electronic** ...

## Alte primitive criptografice

- ▶ Protocole de tip **zero-knowledge** permit unei entități (*prover*) să demonstreze ceva unei alte entități (*verifier*);

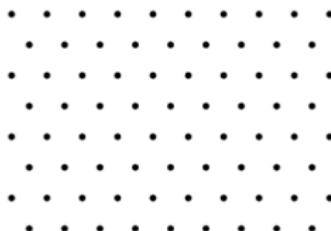


[J.J.Quisquater, L.C.Guillou, T.A.Berson,  
How to Explain Zero-Knowledge Protocols to Your Children]

- ▶ O noțiune similară o reprezintă **commitment scheme**.

## Mai mult despre criptografie

- ▶ Am studiat criptografia bazată pe teoria numerelor și criptografia bazată pe curbe eliptice;
- ▶ Dar există și alte tipuri de criptografie, precum **criptografia bazată pe latici**:

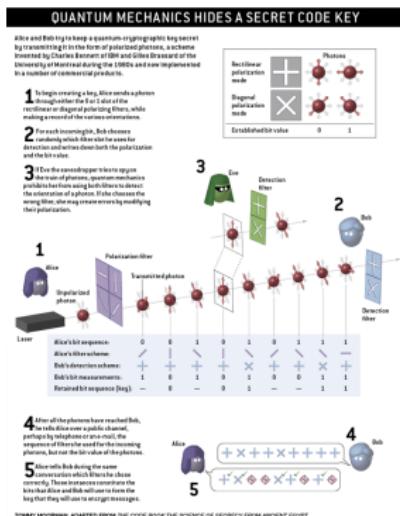


$$\left\{ \sum_{i=1}^n a_i v_i \mid a_i \in \mathbb{Z}, v_i \text{ bază} \right\}$$

- ▶ Probleme dificile: **SVP** (Shortest Vector Problem), **CVP** (Closest Vector Problem), ...

# Mai mult despre criptografie

## ► Criptografia cuantică:



<http://blogs.scientificamerican.com/guest-blog/files/2012/11/quantum.gif>

## Important de reținut!

- ▶ Există încă multe aspecte criptografice interesante de studiat!