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

3. Sisteme mecanice de criptare

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

$\longrightarrow$

	M	A	R	T
$\downarrow$	E	J	I	A
	S	C	P	T

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

Cifruri de substitiuție

Definitie

Un cifru de substutuție 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 substitiuție monoalfabetice

- cifrul lui Cezar >>> curs
- substituție simplă >>> curs
- 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

- $\mathcal{K} = \{0, 1, \ldots, 25\}$
- $\mathcal{M} = \{a, b, \ldots, z\}^*$
- $\mathcal{C} = \{A, B, \ldots, Z\}^*$
- $\mathrm{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$

$$Enc_k(m) = m + k \pmod{26}$$

- $\mathrm{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

$$Dec_k(c) = c - k \pmod{26}$$

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

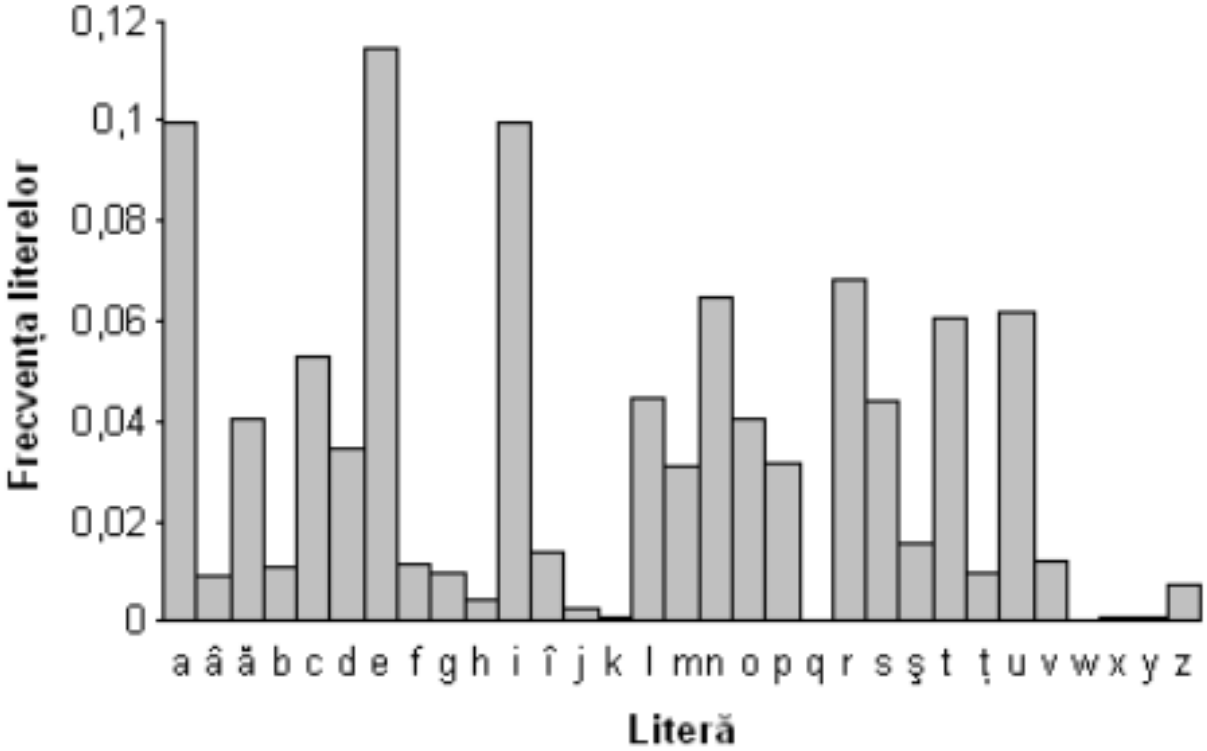
n	o	p	q	r	s	t	u	v	w	x	y	z
S	V	Y	B	E	H	K	N	Q	T	W	Z	C

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 substitiuție polialfabetice / poligrafice

- sistemul Playfair >>> seminar
- sistemul Hill >>> seminar
- sistemul Vigenére >>> laborator

Sisteme mecanice

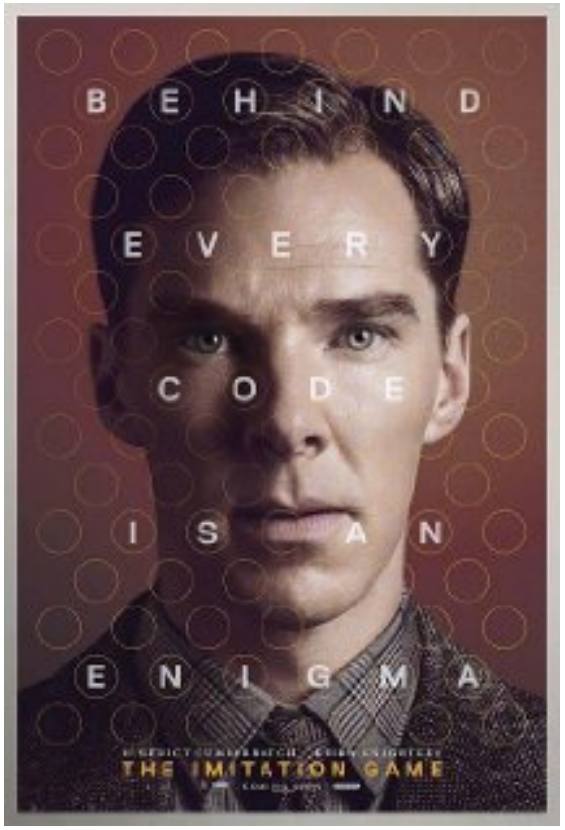
[Bletchley Park 2012]

Sisteme mecanice

[Bletchley Park 2012]

Enigma vs. The Imitation Game

Enigma (2001)

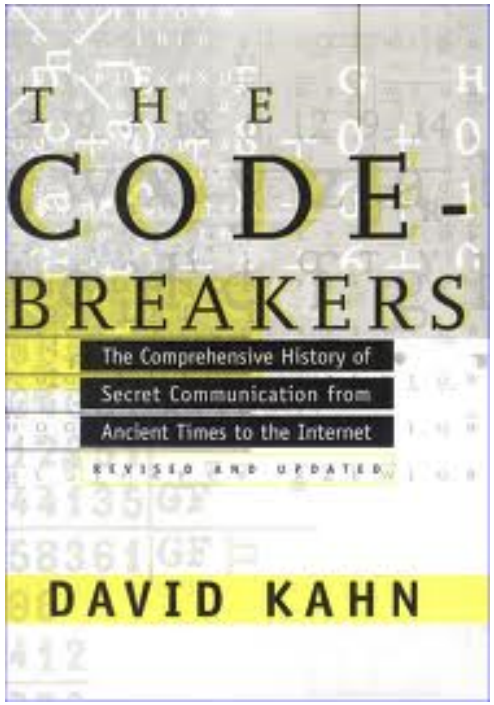


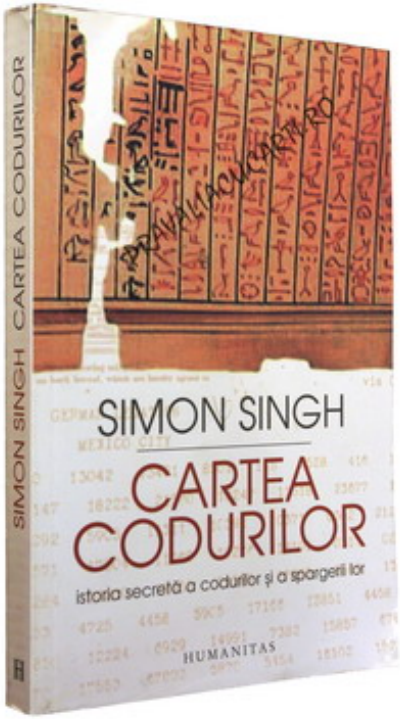
The Imitation Game(2014)

Important de reținut!

- Tipuri de cifruri: transpoziție, substituție
- Sisteme mecanice de criptare
- Astfel de sisteme sunt total nesigure!

Referințe bibliografice





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

Criptografie și Securitate

- Prelegerea 4 -
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 în fața unui adversar cu putere computațională nelimitată
- Insă...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 afla 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 in 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:	0	1	1	0	0	1	1	1	1	$\oplus$
cheie:	1	0	1	1	0	0	1	1	0	
text criptat:	1	1	0	1	0	1	0	0	1	

- avantaj - criptare și decriptare rapide
- dezavantaj - cheia foarte lungă (la fel de lungă precum textul clar)
- Este OTP sigur?

Teoremă

Schema de criptare OTP este perfect sigură.

- securitatea perfectă nu este imposibilă dar...
- cheia trebuie să fie la fel de lungă precum mesajul
- incoveniente 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

- Prelegerea 8 -
Sisteme de criptare bloc

Adela Georgescu, Ruxandra F. Olimid

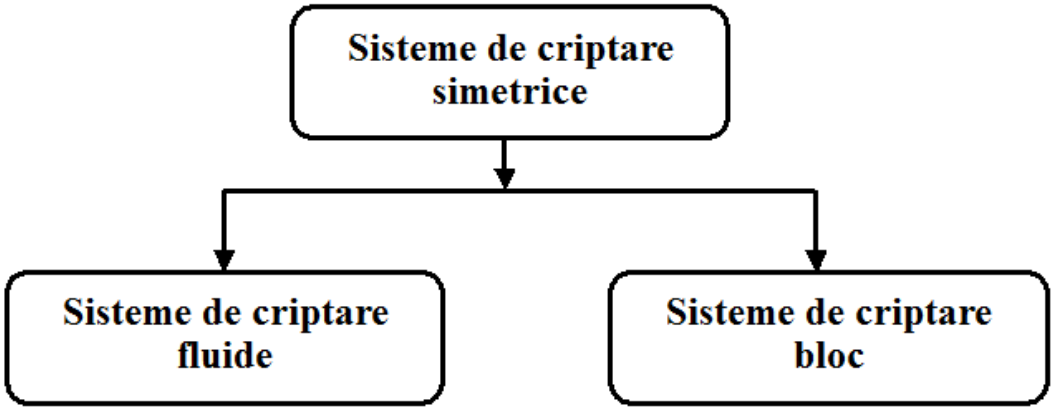
Facultatea de Matematică și Informatică
Universitatea din București

Cuprins

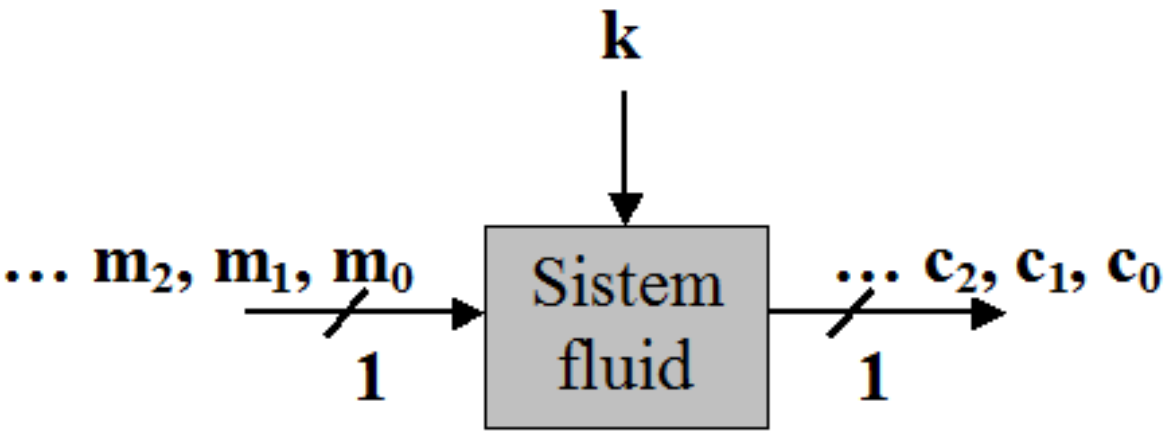
1. Definiție
2. Construcție
3. Moduri de utilizare
4. Exemple

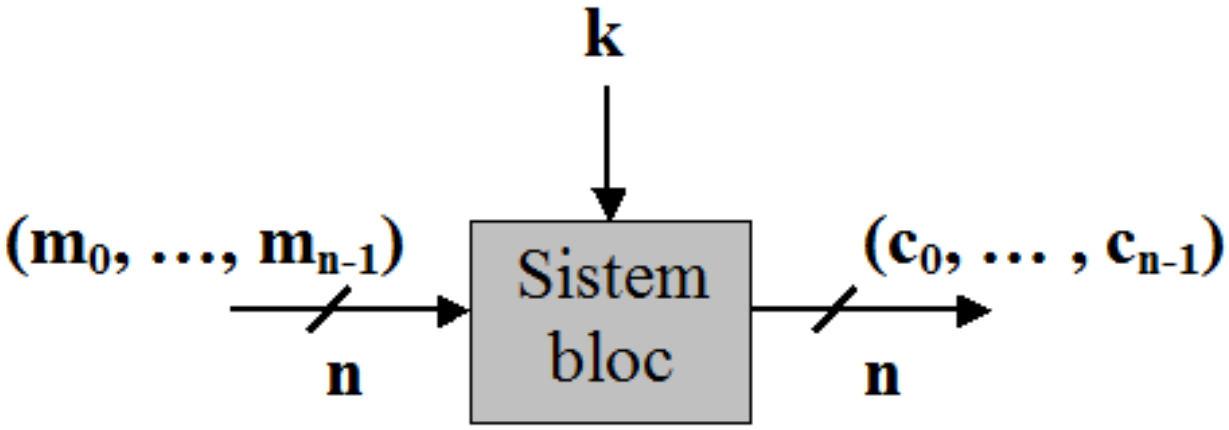
Criptografia simetrică

- 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

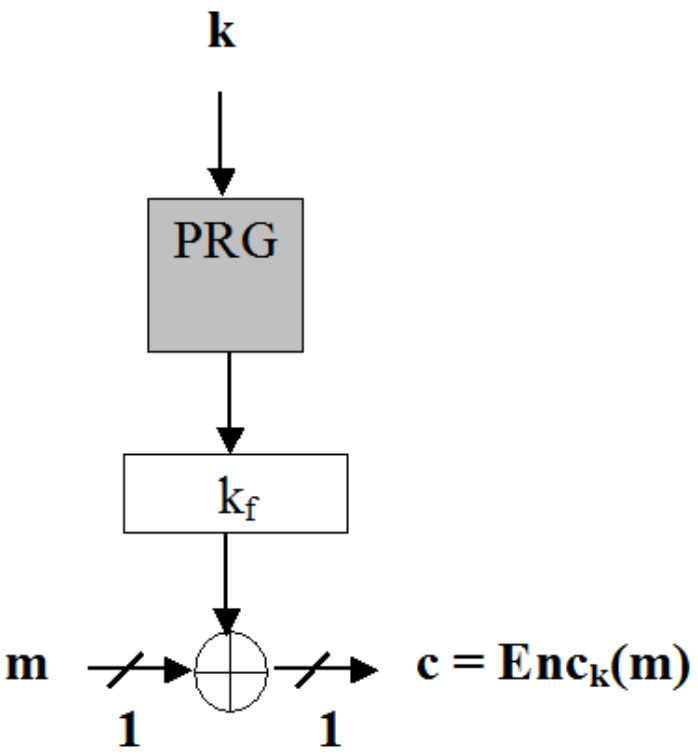
- Introducem noțiunea de permutare pseudoaleatoare sau PRP*(PseudoRandom Permutation)*
- În analogie cu ce știm deja:
  - **PRP** sunt necesare pentru construcția **sistemelor bloc**

    asa cum

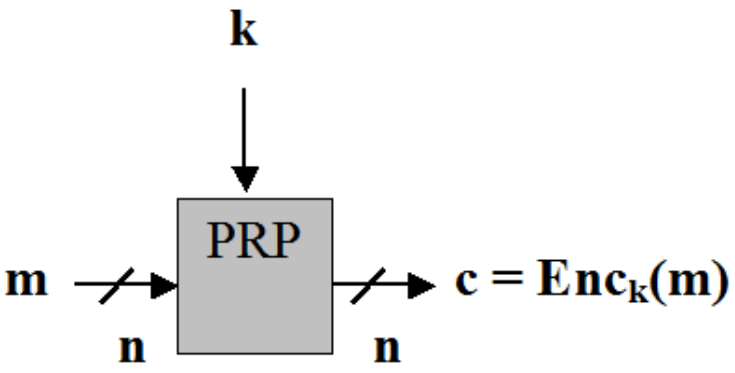
  - **PRG** sunt necesare pentru construcția **sistemelor fluide**

Sisteme bloc

Sisteme fluide



Sisteme bloc



PRP

- Ramâne să definim noțiunea de permutare pseudoaleatoare sau PRP *(PseudoRandom Permutation)*;
- Acesta 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**.

PRP

Definitie

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

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

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.:*

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

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

Notații

- $F_k(x) = F(k, x)$
  o cheie este în general (aleator) aleasă și apoi fixată
- $Perm(X)$ = mulțimea tuturor funcțiilor bijective de la $\mathcal{X}$ la $\mathcal{X}$
- $\mathcal{X} = \{0, 1\}^n$
- $\mathcal{D}$ = *Distringuisher* 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**;
- Acesta este o funcție **cu cheie** care este **indistinctibilă** față de o funcție aleatoare (cu același domeniu și mulțime de valori).

PRF

Definitie

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

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

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.:*

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

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

Notații

- $F_k(x) = F(k, x)$
  o cheie este în general (aleator) aleasă și apoi fixată
- $Func(X, Y)$ = mulțimea funcțiilor de la $\mathcal{X}$ la $\mathcal{Y}$
- $\mathcal{X} = \{0,1\}^n$, $\mathcal{Y} = \{0,1\}^n$
  considerăm în general că *PRF păstrează lungimea*
- $\mathcal{D}$ = *Distringuisher* 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 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

Întrebare: Care dintre aceste construcții este trivială?

Construcții

Răspuns: **PRP** $\Rightarrow$ **PRF**

PRP este o particularizare a $PRF : \mathcal{X} \times \mathcal{K} \rightarrow Y$ 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** $\surd$
  Pornind de la PRP se poate construi PRF
- **PRF $\Rightarrow$ PRP**
  Pornind de la PRF se poate construi PRP

$PRF \Rightarrow PRG$

- Considerăm $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ PRF;
- Construim $G : \mathcal{K} \to \{0,1\}^{nl}$ *PRG* sigur:

$$G(k) = F_k(0)||F_k(1)||\ldots||F_k(l-1)$$

- 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 ln.
- Avantaj: Construcția este *paralelizabilă*

Construcții

- **PRF $\Rightarrow$ PRG** $\surd$
  Pornind de la PRF se poate construi PRG
- **PRG $\Rightarrow$ PRF**
  Pornind de la PRG se poate construi PRF
- **PRP $\Rightarrow$ PRF** $\surd$
  Pornind de la PRP se poate construi PRF
- **PRF $\Rightarrow$ PRP**
  Pornind de la PRF se poate construi PRP

$PRG \Rightarrow PRF$

- Considerăm $G : \mathcal{K} \to \mathcal{K}^2$ 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 $F : \mathcal{K} \times \{0,1\} \to \mathcal{K}$ *PRF*:

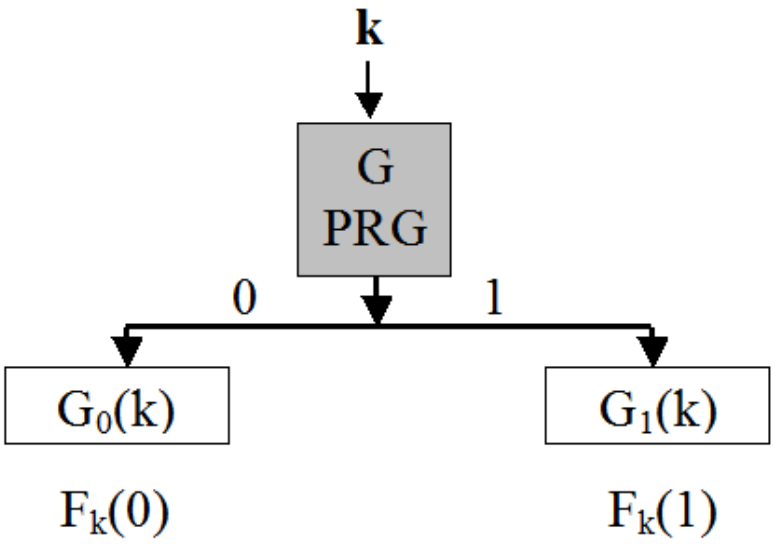
$$F_k(0) = G_0(k), F_k(1) = G_1(k)$$

  $F_k(0)$ 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.

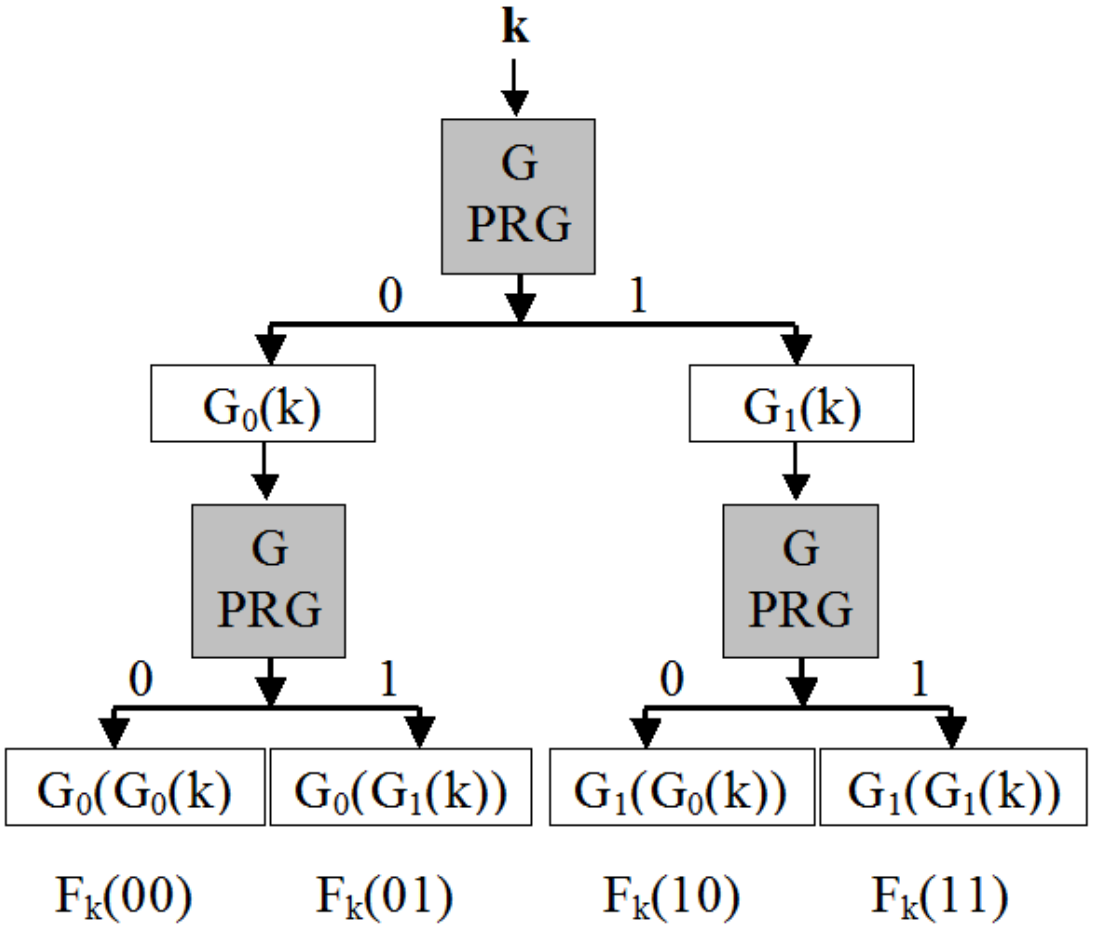
$PRG \Rightarrow PRF$

- 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 $F_k(x) = F_k(x_0, \ldots, x_{n-1})$ 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** $\surd$
  Pornind de la PRF se poate construi PRG
- **PRG $\Rightarrow$ PRF** $\surd$
  Pornind de la PRG se poate construi PRF
- **PRP $\Rightarrow$ PRF** $\surd$
  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)

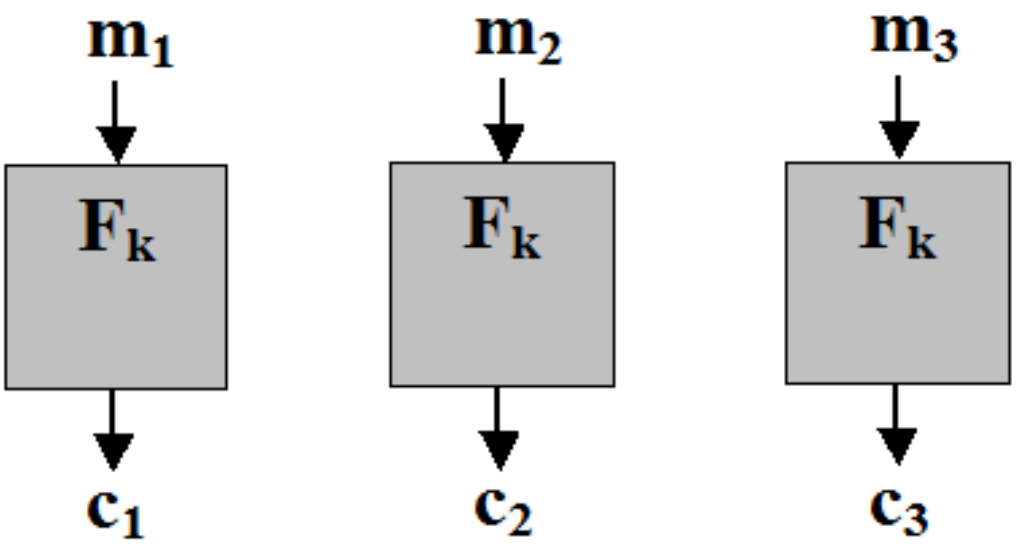
Dacă $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *este PRF, se poate construi* $F' : \mathcal{K} \times \{0,1\}^2 \to \{0,1\}^2$ *PRP.*

- 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: $\mathbf{1\ 0 \ldots 0}$;
- 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.

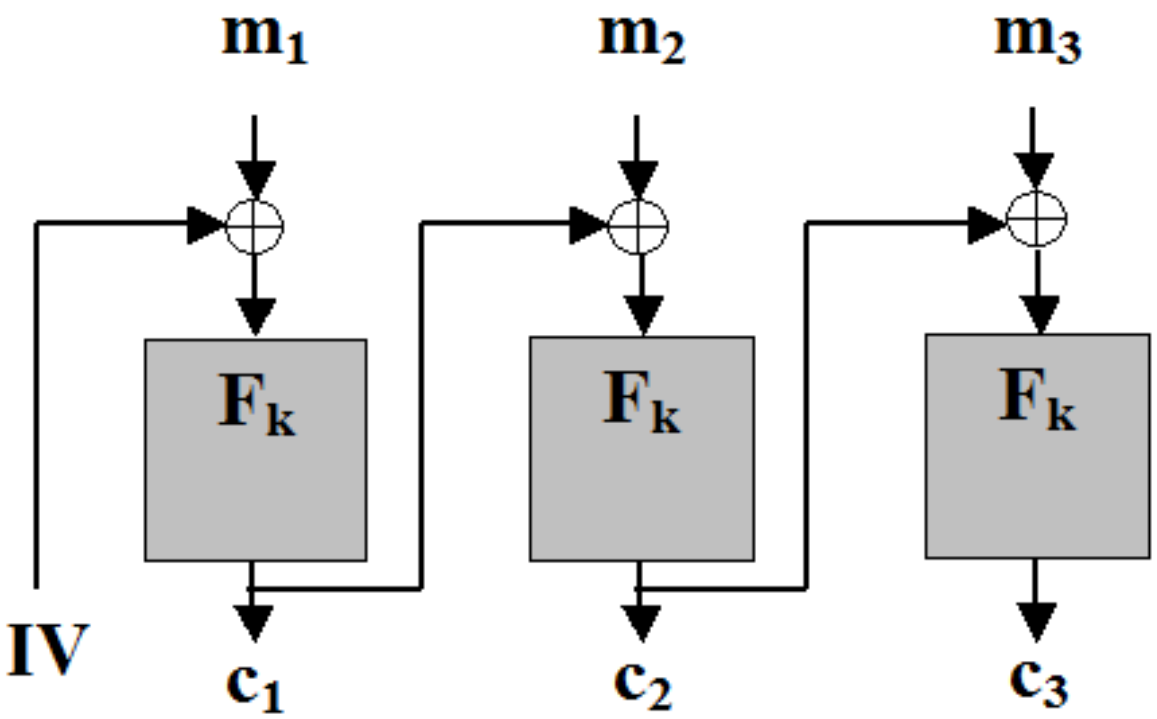
Modul ECB (Electronic Code Book)



Modul ECB (Electronic Code Book)

- Pare modul cel mai **natural** de a cripta mai multe blocuri;
- Pentru decriptare, F_k trebuie să fie **inversabliă**;
- Este **paralelizabil**;
- 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!

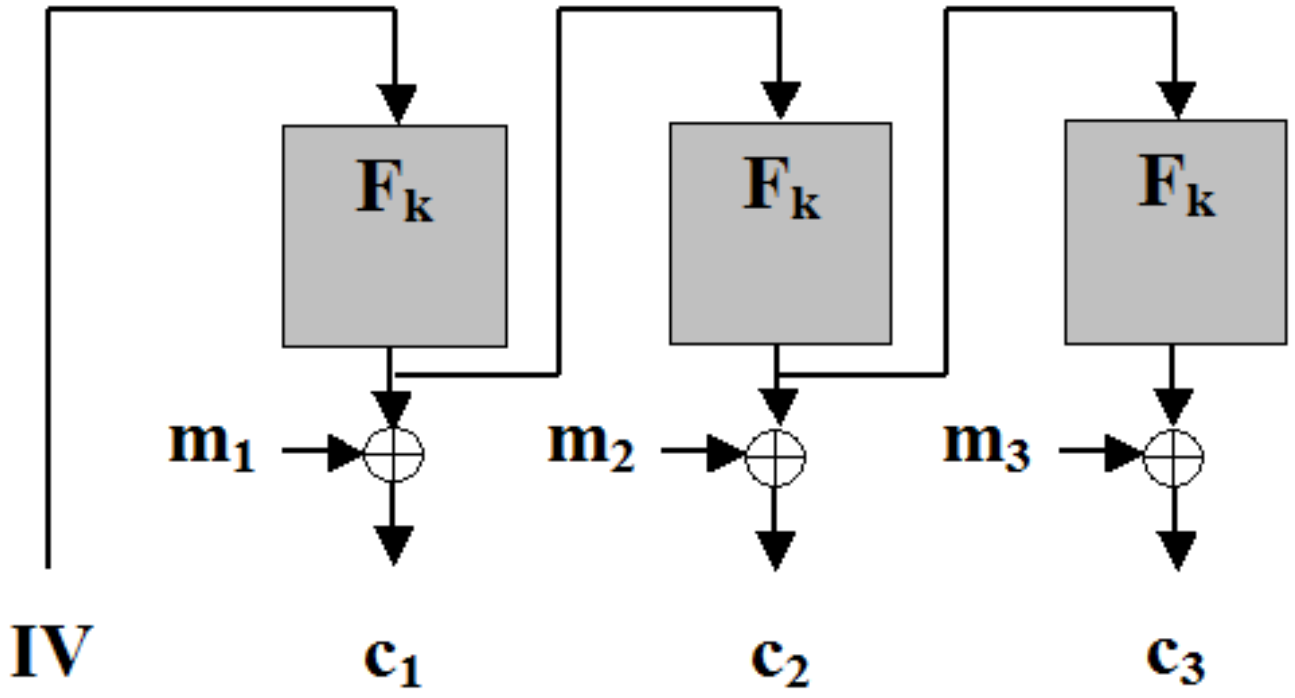
Modul CBC (Cipher Block Chaining)



Modul CBC (Cipher Block Chaining)

- IV este o ales în mod aleator la criptare;
- IV se transmite în clar pentru ca este necesar la decriptare;
- Pentru decriptare, F_k trebuie să fie **inversabliă**;
- Este **secvențial**, un dezavantaj major dacă se poate utiliza procesarea paralelă.

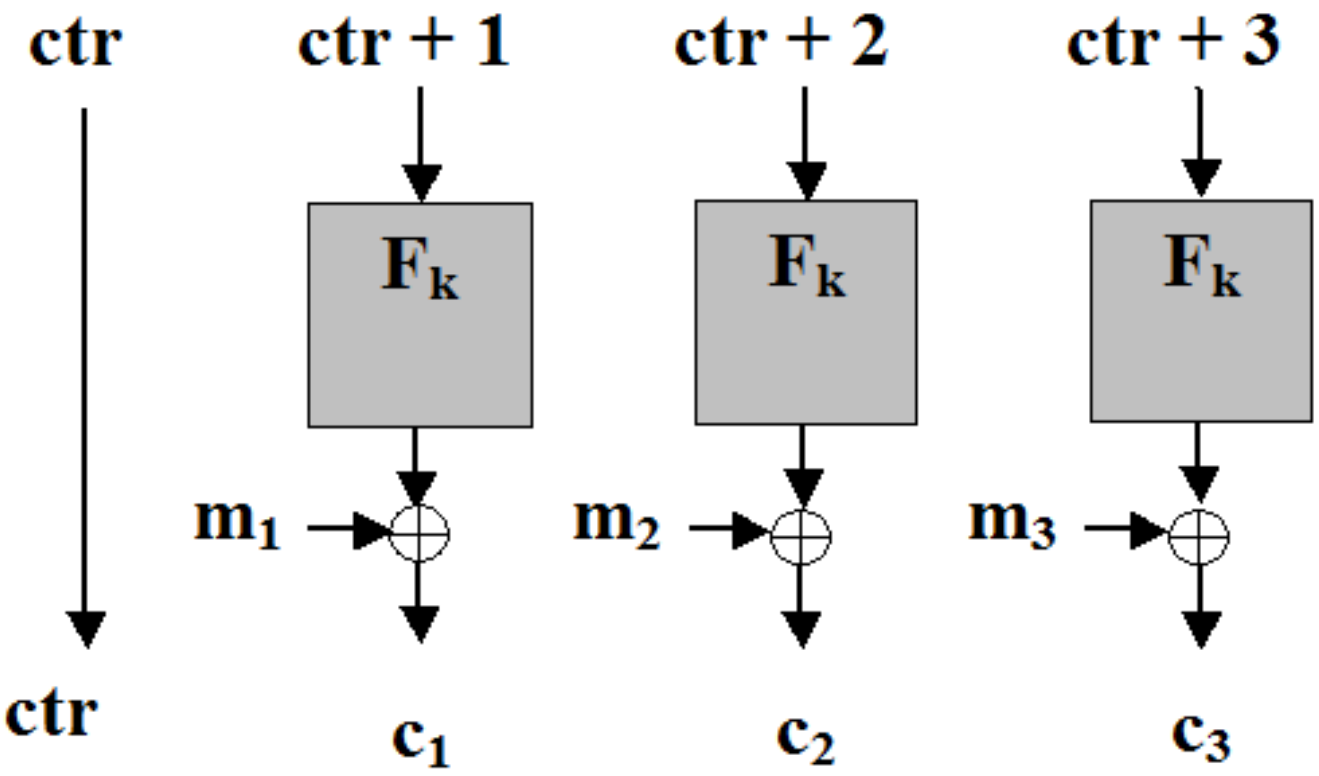
Modul OFB (Output FeedBack)



Modul OFB (Output FeedBack)

- Generează o secvență pseudoaleatoare care se XOR-ează mesajului clar;
- IV este o ales în mod aleator la criptare;
- IV se transmite în clar pentru ca este necesar la decriptare;
- F_k nu trebuie neapărat să fie inversabliă;
- 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 o ales în mod aleator la criptare;
- *ctr* se transmite în clar pentru ca este necesar la decriptare;
- F_k nu trebuie neapărat să fie inversabliă;
- 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

- Prelegerea 9 -
Construcții practice pentru PRP

Adela Georgescu, Ruxandra F. Olimid

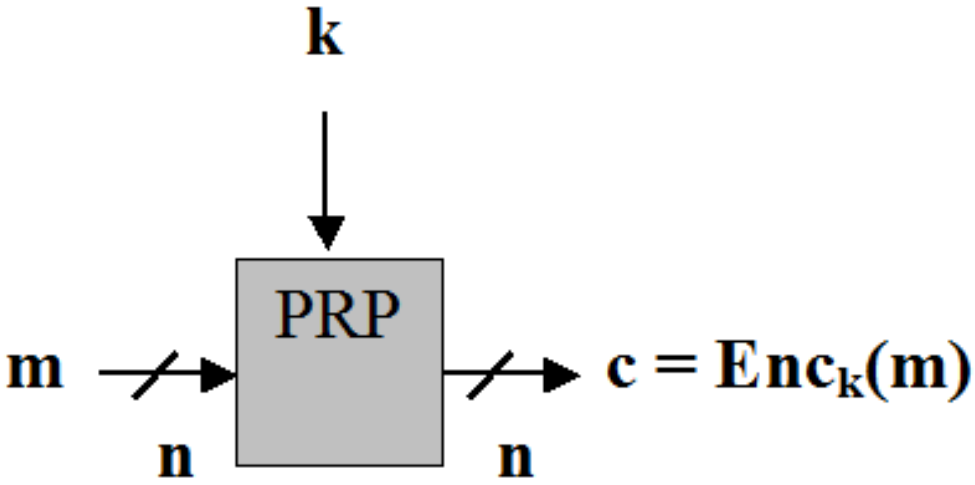
Facultatea de Matematică și Informatică
Universitatea din București

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 $f_1, \ldots, f_{16}$ pe câte 8 biți;
- Pentru $x = x_1 || \ldots || x_{16}$, $x \in \{0,1\}^{128}$ $x_i \in \{0,1\}^8$:

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

- 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.

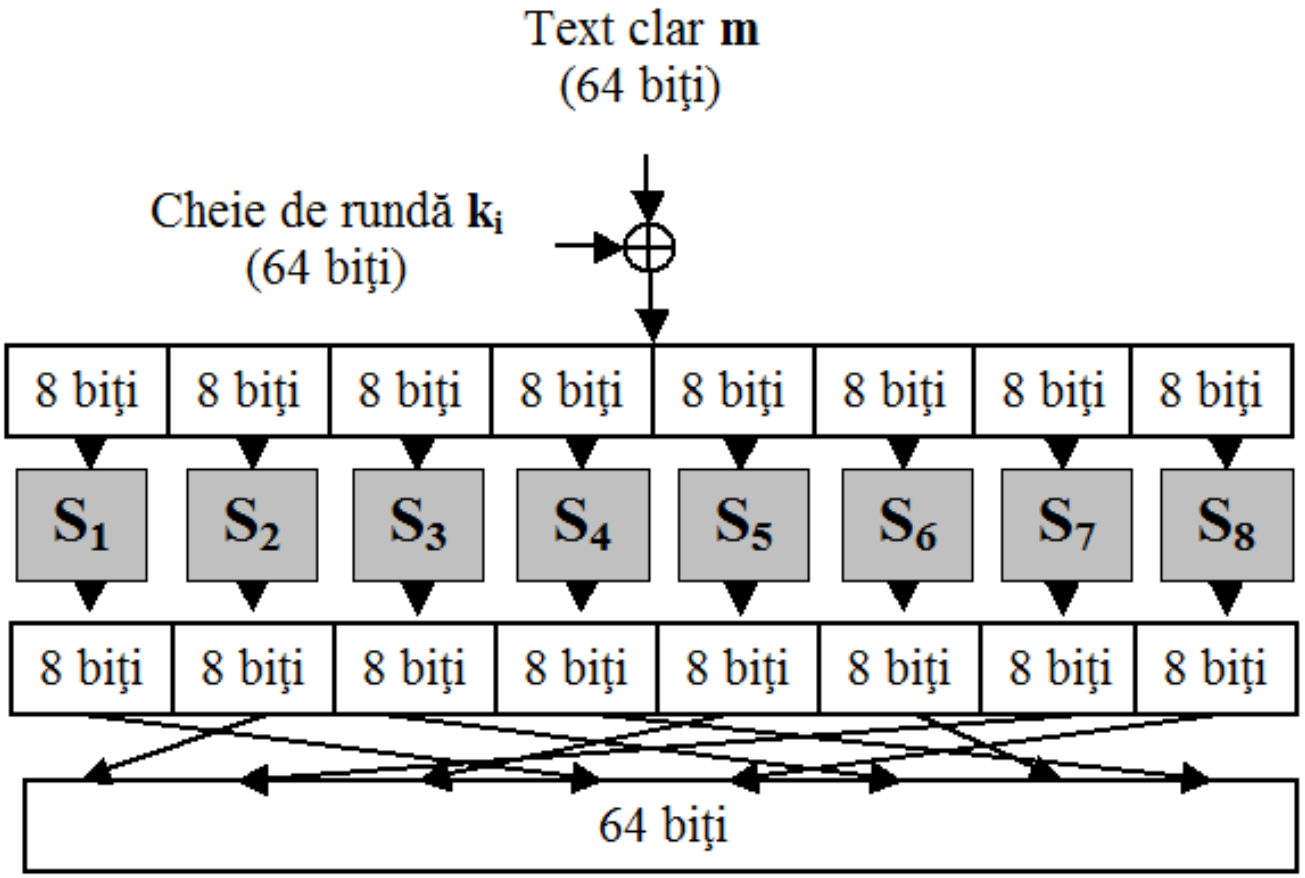
Rețele de substituție - permutare

- 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 tutoror 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

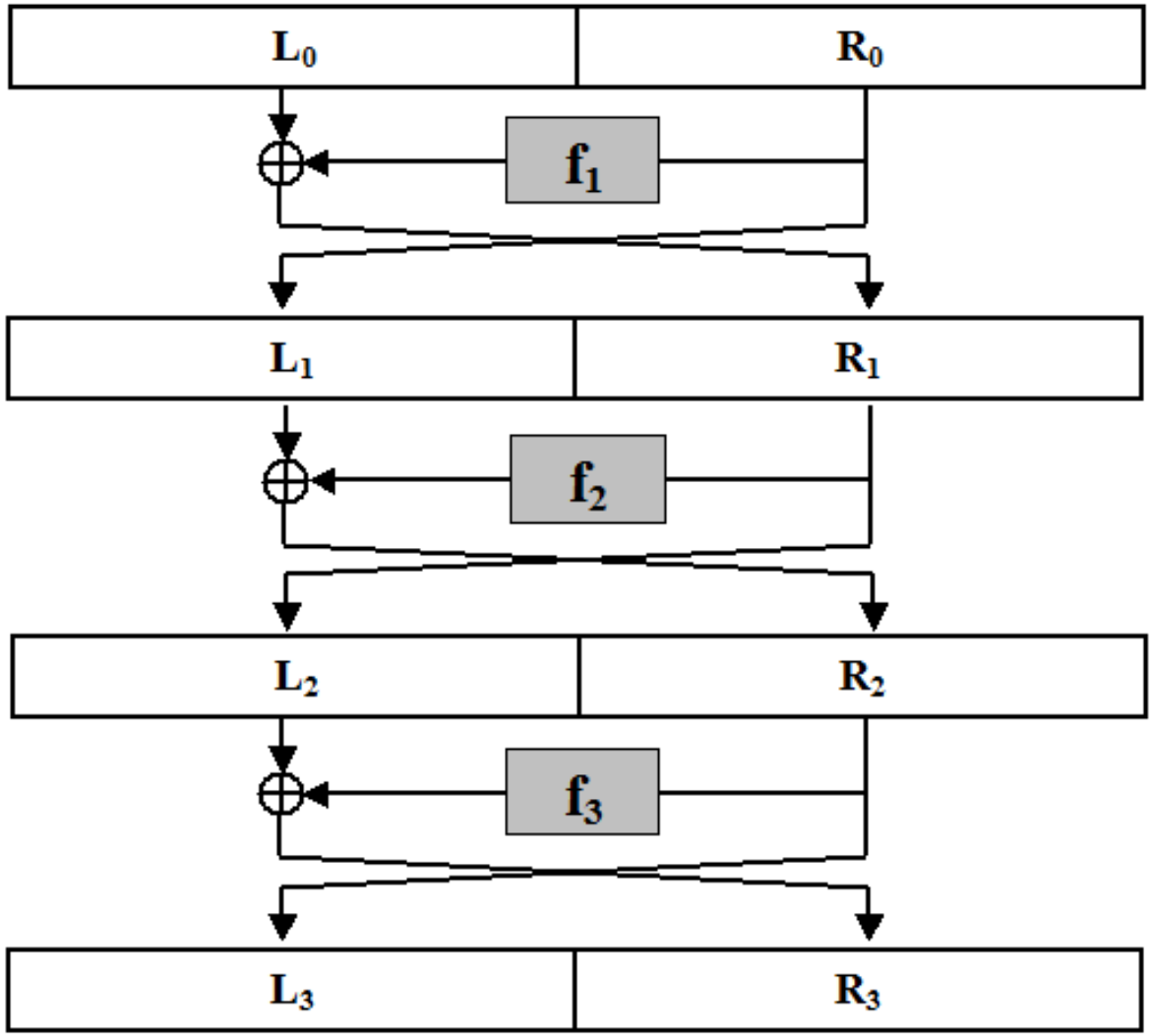
- Se aseamănă rețelelor de substituție-permutare în sensul că păstrează aceleași elementele componente: S-box, permutare, procesul de derivare a cheii, runde;
- Se diferențiază de rețelele de substituție-permutare prin proiectarea de nivel înalt;
- Introduc avantajul major că S-box-urile NU trebuie să fie inversabile;
- Permit așadar 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:

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

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

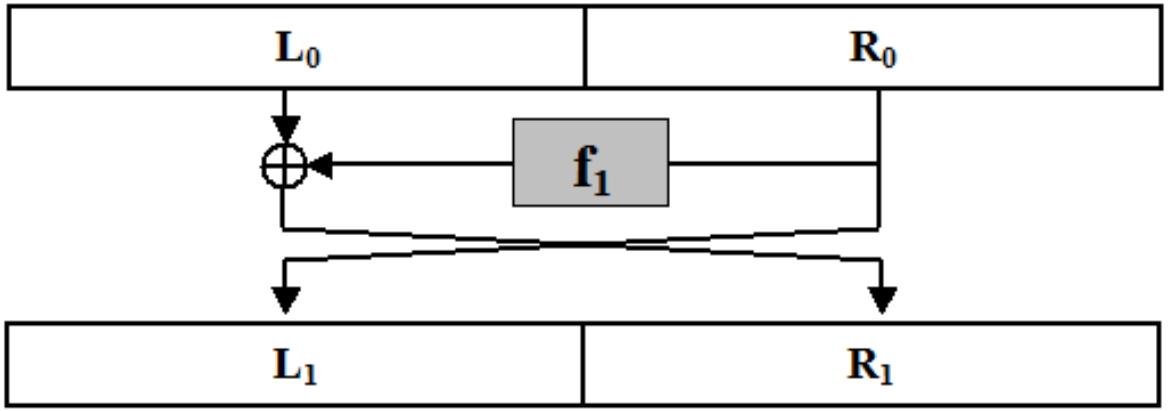
Construcții

Am omis în cursurile anterioare ultima construcție:

- **PRF $\Rightarrow$ PRG** $\surd$
  Pornind de la PRF se poate construi PRG
- **PRG $\Rightarrow$ PRF** $\surd$
  Pornind de la PRG se poate construi PRF
- **PRP $\Rightarrow$ PRF** $\surd$
  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 $\{0,1\}^n$,...
- ... 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 11 -
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;
- Insă, nu ne interesează doar ca adversarul să nu aibă acces la mesajele trimise, ci...
- Vrem să garantăm integritatea mesajelor (sau autentificarea mesajelor)
- Aceasta înseamnă ca mesajul primit de Bob este exact mesajul trimis de Alice.

Comunicare sigură și integritatea mesajelor

- Iată un exemplu:
- Să consideram 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

- In exemplul precedent, problema este doar de integritate a mesajelor, și nu de confidențialitate (comanda nu e secretă);
- In 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

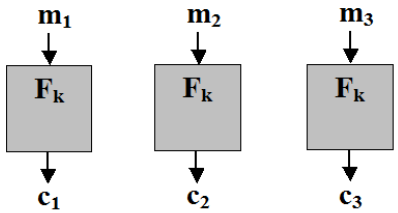
- Criptarea, în general, **NU** oferă integritatea mesajelor!
- Dacă un mesaj este transmis criptat de-a lungul unui canal de comunicare, nu înseamnă că un adversar nu poate modifica/altera mesajul așa încât modificarea să aiba sens în textul clar;
- Verificăm, în continuare, că nici o schemă de criptare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

Criptare vs. autentificarea mesajelor

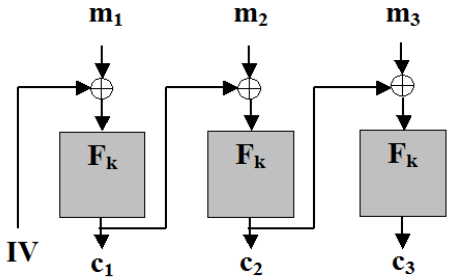
- Criptarea folosind sisteme fluide
  - $Enc_k(m) = G(k) \oplus m$, unde G este un PRG;
  - Dacă modificăm un singur bit din textul criptat c, modificarea se va reflecta imediat în același bit din textul clar;
  - Consecințele pot fi grave: de pildă, să considerăm transferul unei sume de bani în dolari 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

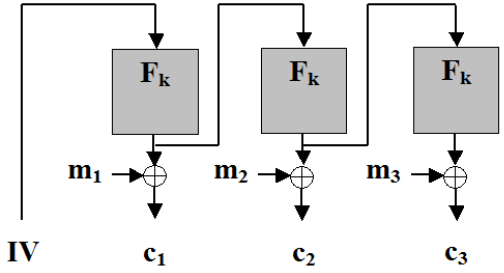
- Criptarea folosind sisteme bloc
  - Intrebare: Atacul de mai sus se poate aplica și pentru sistemele bloc cu modurile de operare studiate?



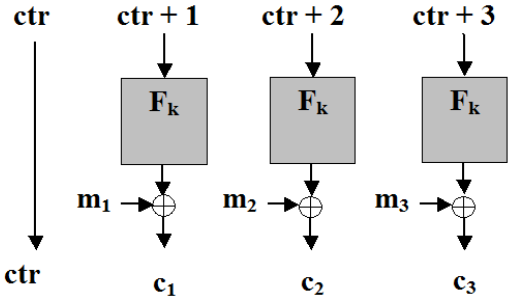
(a) ECB



(b) CBC



(c) OFB



(d) CTR

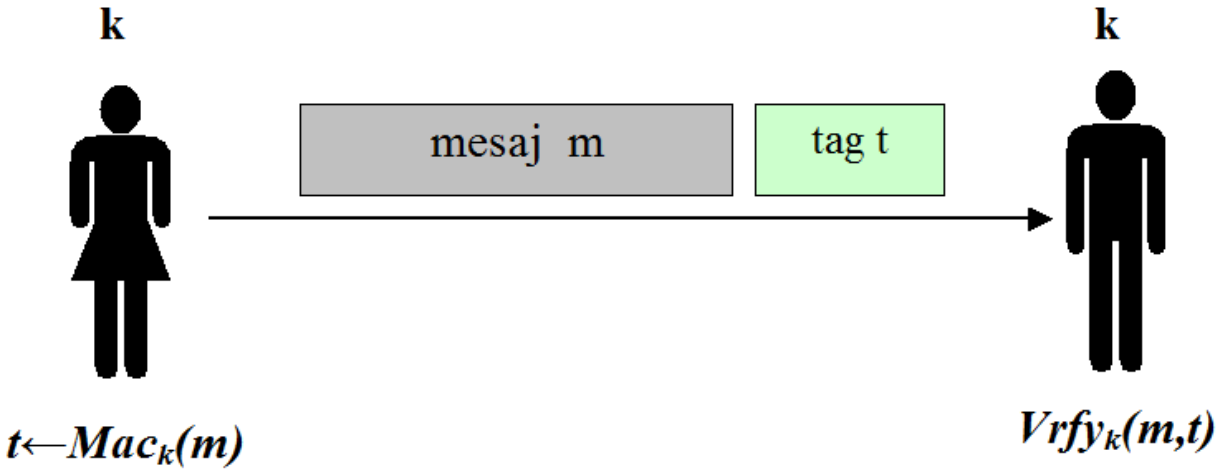
Criptare vs. autentificarea mesajelor

- 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 si c_{i-1} nu au fost modificate).

Coduri de autentificare a mesajelor - MAC

- Așa cum am vazut, criptarea nu rezolvă problema autentificarii 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 prestabilească de comun acord o cheie secretă.

MAC - Definiție



- Alice și Bob stabliesc o cheie secretă k pe care o partajează;
- Când Alice vrea să îi trimită un mesaj m lui Bob, calculează mai întâi un tag t 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`;
- In 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* $(\mathrm{Mac}, \mathrm{Vrfy})$ *unde:*

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

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

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 împreuna cu tag-urile aferente.
- Adversarul poate influența conținutul mesajelor (direct sau indirect), fiind deci un adversar *activ*.

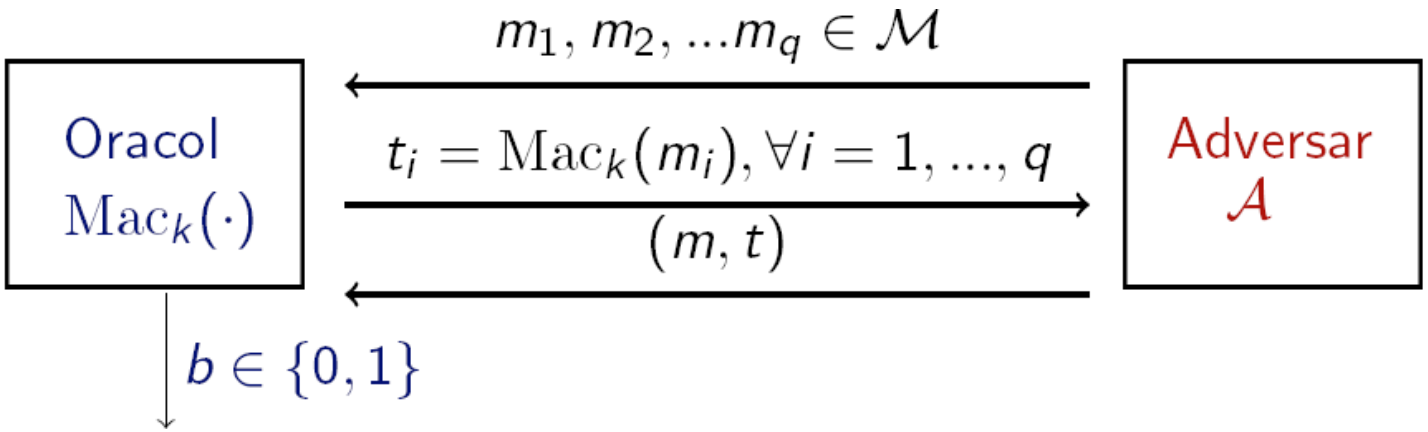
Securitate MAC - formalizare

- Formal, îi dăm adversarului acces la un *oracol* $\mathrm{Mac}_k(\cdot)$;
- Adversarul poate trimite orice mesaj m dorit către oracol și primește înapoi un tag corespunzător $t \leftarrow \mathrm{Mac}_k(m)$;
- 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: $\mathrm{Vrfy}_k(m, t) = 1$;
  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 $\pi = (\mathrm{Mac}, \mathrm{Vrfy})$, în care considerăm un adversar $\mathcal{A}$ și parametrul de securitate n;

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



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

Securitate MAC

Definiție

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

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

Atacuri prin replicare

- Întrebare: De ce este necesară a doua 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;
- Intrebare: 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 trimite 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 așa 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 algortimul de verificare;
- Mai degrabă, protecția împotriva replicării trebuie făcută la nivel înalt de către aplicatiile care folosesc MAC-uri;
- Două tehnici comune de protejare împotriva atacurilor prin replicare folosesc *secvențe de numere* sau *ștampilă 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 ștampilelor de timp: la recepția mesajului, destinatarul verifică dacă ștampila de timp inclusă se află într-un interval de timp acceptabil;
- Aceasta presupune ca părțile 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 $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *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;
- Insă 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 $(\pi' = (\mathrm{Mac}', \mathrm{Vrfy}'))$ 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 $m_1, ..., m_d$ și autentificam blocurile folosind π';
- Iată câteva modalități de a face aceasta:

MAC-uri pentru mesaje de lungime variabilă

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

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

- 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*:

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

- 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*:

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

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

MAC-uri pentru mesaje de lungime variabilă

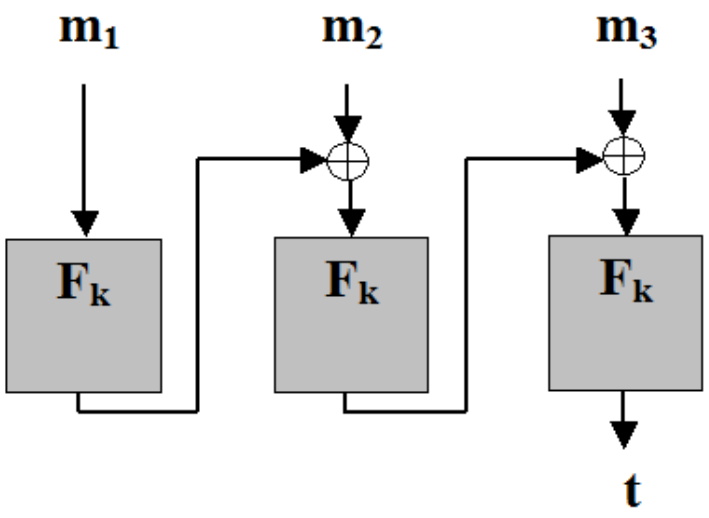
- O soluție pentru atacurile anterioare o reprezintă adăugarea de informație suplimentară în fiecare bloc, în afara numărului de secvență:
  - un identificator aleator de mesaj - previne combinarea blocurilor din mesaje diferite
  - lungimea mesajului - previne modificarea lungimii mesajelor

CBC-MAC

- Soluția este ineficientă și greu de folosit în practică;
- Insă, 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.



CBC-MAC

Definiție

Fie F o funcție pseudoaleatoare. Un CBC-MAC este format dintr-o pereche de algoritmi polinomiali probabiliști $(\mathrm{Mac}, \mathrm{Vrfy})$*:*

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

   Intoarce 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 $t = \mathrm{Mac}_k(m)$*.*

Rămâne valabilă condiția de corectitudine:
$\forall m \in \mathcal{M},\ k \in \mathcal{K},\ \mathrm{Vrfy}_k(m, \mathrm{Mac}_k(m)) = 1.$

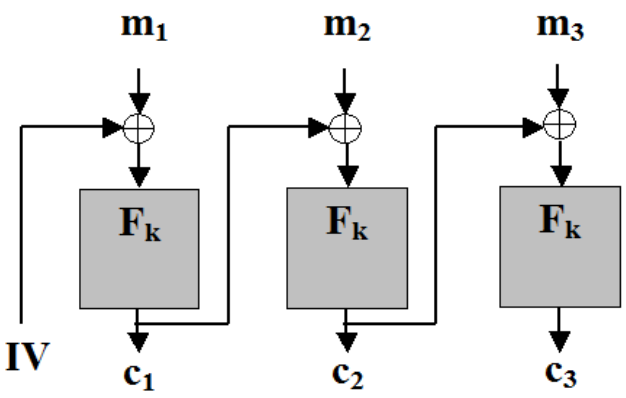
Securitatea CBC-MAC

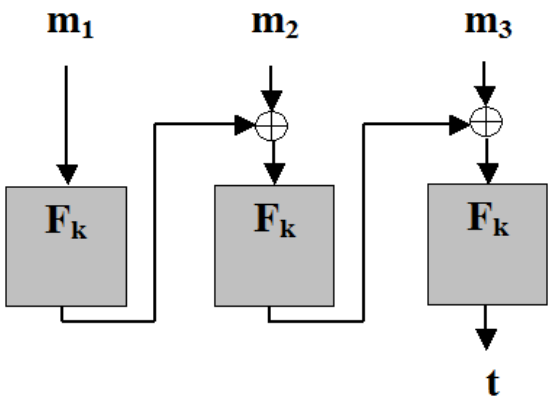
Teoremă

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 vs. Criptare in modul 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 demonstrate ca fiind sigure:

1. Calculează $k_l = F_k(l)$; Apoi folosește CBC-MAC cu cheia k_l; 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 așa încât să se aleagă două chei $k_1, k_2 \in \{0,1\}^n$; se autentifica 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

- Prelegerea 14 -
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**, **ștergere**;
- 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

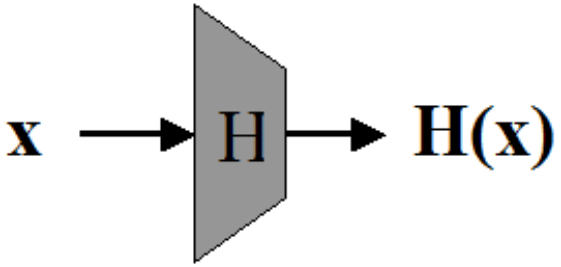
- Răspuns: Probleme sunt multe: adresele nu sunt mereu identic introduse, numele nu sunt unice, ...
- ... dar pe noi ne interesează următorul aspect:
- Întrebare: Ce se întâmplă dacă pentru 2 valori $x \neq x'$, $H(x) = H(x')$?
- Răspuns: În acest caz, apare o coliziune - ambele valori se vor stoca în aceeași celulă.

Funcții Hash

- În criptografie, o funcție hash rămâne o funcție care comprimă secvențe de lungime variabilă în secvențe de lungime fixă, insă:
- 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 $Hash^{coll}_{\mathcal{A},H}(n)$

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

Rezistența la coliziuni

Definiție

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

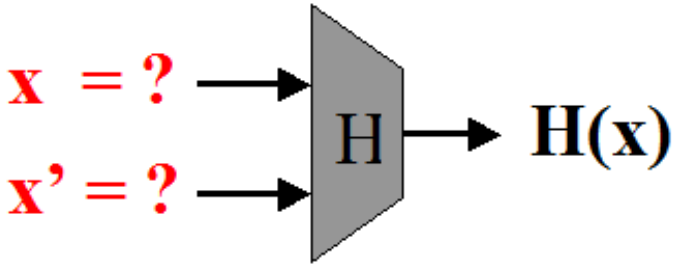
$$Pr[Hash^{coll}_{\mathcal{A},H}(n) = 1] \leq negl(n).$$

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:

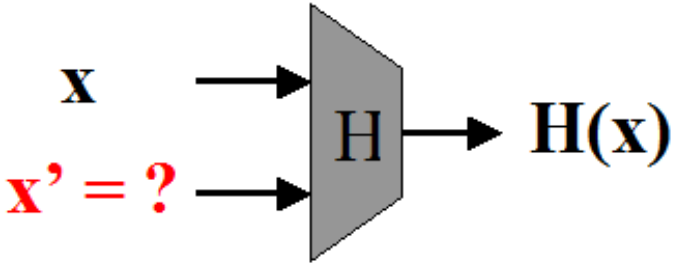
  1. Rezistența la coliziuni: este cea mai puternică noțiune de securitate și deja am definit-o formal;
  2. Rezistența la a doua preimagine: presupune că fiind dat x este dificil de determinat $x' \neq x$ a.î. $H(x) = H(x')$
  3. Rezistența la prima preimagine: presupune că fiind dat $H(x)$ este imposibil de determinat x.

Rezistența la coliziuni



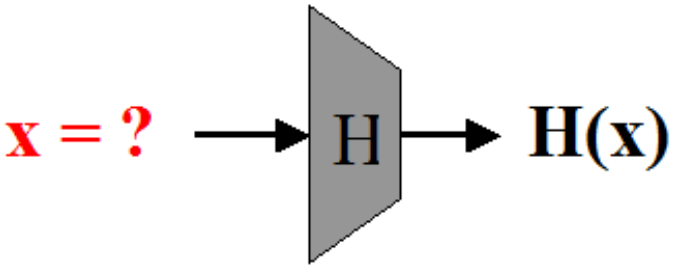
- **Provocare:** Se cer 2 valori $x \neq x'$ a.î. $H(x) = H(x')$;
- **Atac**: "Atacul zilei de naștere" necesită $\approx 2^{l(n)/2}$ evaluări pentru H.

Rezistența la a doua preimagine



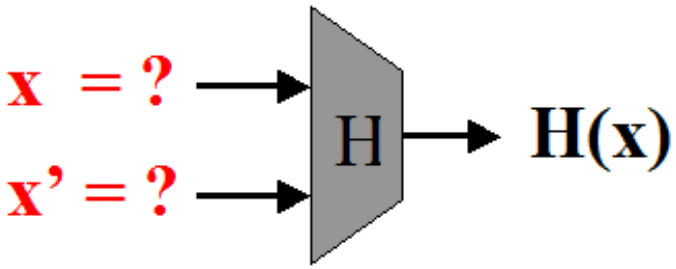
- **Provocare:** Fiind dat x, se cere x' a.î. $H(x) = H(x')$;
- **Atac**: Un atac generic necesită $\approx 2^n$ evaluări pentru H.

Rezistența la prima preimagine



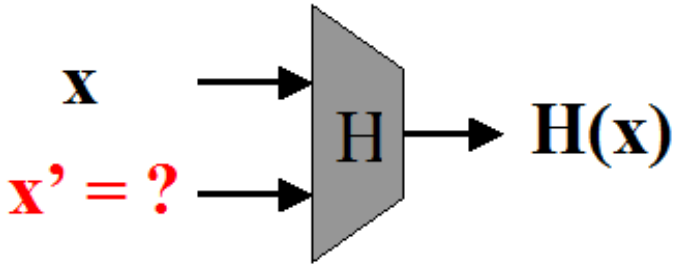
- **Provocare:** Fiind dat $y = H(x)$, se cere x a.î. $H(x) = y$;
- O astfel de funcție se numește și *calculabilă într-un singur sens (one-way function)*;
- **Atac**: Un atac generic necesită $\approx 2^n$ evaluări pentru H.

Atacuri în practică

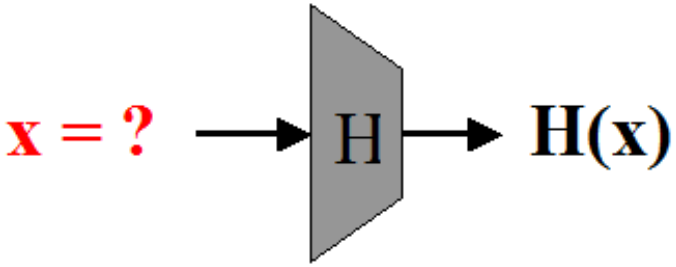


$\{x\}$ = documente originale
$\{x'\}$ = documente false
cineva este de acord să semneze electronic documentul original, însă semnătura devine valabilă și pentru un document fals

---



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



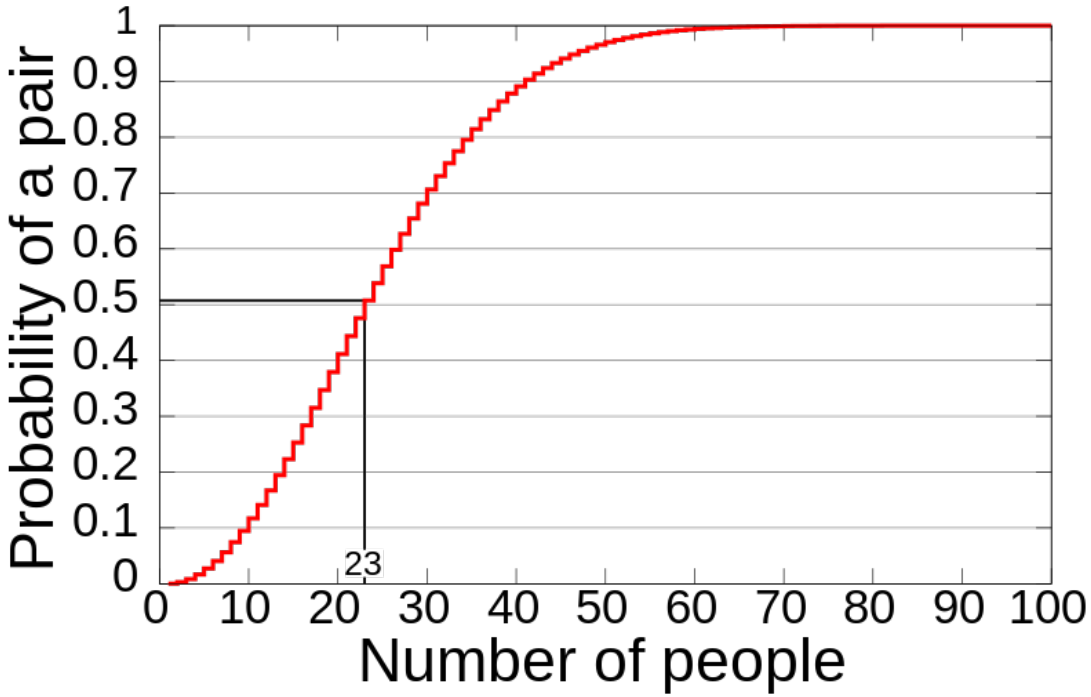
dacă se cunoaște cheia de sesiune k_i calculată din cheia master k
$x = H(k||i)$, atunci se determină cheia k

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 $y_1, \ldots, y_q$;
- 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 ≈ 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 ?

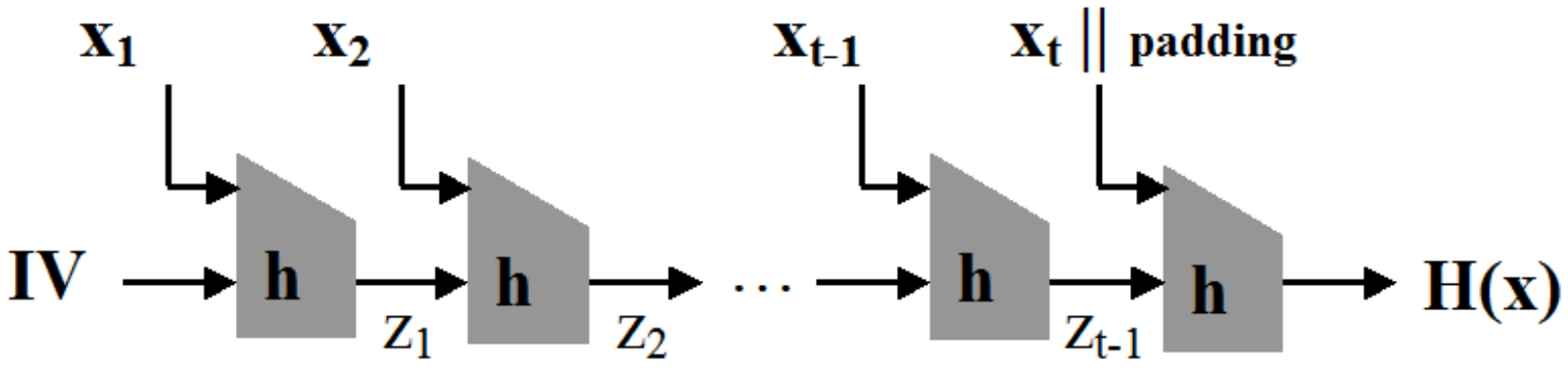
$$H_1 : \{0,1\}^{m_1} \to \{0,1\}^{n_1}, m_1 > n_1, m_1 \approx n_1$$

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

- 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;



Notații

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

Transformarea Merkle-Damgård

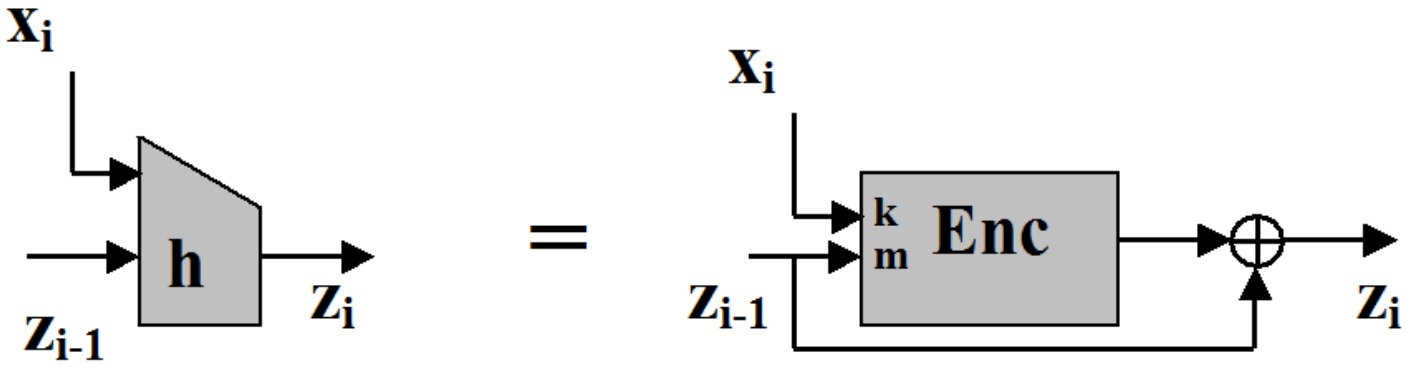
Teoremă

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 $< z_{i-1}, x_i > \neq < z'_{i-1}, x'_i >$ 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 criptează 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 20.1 -
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 = **R**ivest + **S**hamir + **A**dleman

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 satisface $(e, \phi(N)) = 1$, și un element $y \in \mathbb{Z}_N^*$, se cere să se găsească x a.î. $x^e = y \bmod N$.

- 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 $RSA - inv_{\mathcal{A},GenRSA(n)}$

- 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 $x^e = y \bmod N$ 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*

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

GenRSA

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

Algorithm 3 GenRSA

Input: n

Output: N, e, d

1: **print** N cu factorii p și q
2: $\phi(N) = (p-1)(q-1)$
3: **găsește** e a.î. $gcd(e, \phi(N)) = 1$
4: **calculează** $d := e^{-1} \bmod \phi(N)$
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 $ed = 1 \bmod \phi(N)$, se poate calcula factorizarea lui N în timp probabilist polinomial;
- Nu se cunoaște nici o dovadă 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 $c = m^e \mod N$;
  3. **Dec**: dată o cheie secretă (N, d) și un mesaj criptat $c \in \mathbb{Z}_N$, întoarce $m = c^d \mod N$.

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 $x^c \mod N$;
- 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: $x^c \mod N$

```
descompune c în binar: c = Σ_{i=0}^{n-1} c_i 2^i
z ← 1
for i = n − 1 to 0 do
    z ← z^2 mod N
    if c_i = 1 then
        z ← z · x mod N
    end if
end for
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 messaj 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:

$$pk_1 = (N, e_1);\ sk_1 = (N, d_1)$$
$$pk_2 = (N, e_2);\ sk_2 = (N, d_2)$$

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

Sistemul de criptare RSA

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

Construcție

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

- $\mathrm{Enc_{pk}(m)} = (y, c) = (RSA_{pk}(x), Enc_k(m))$, *unde* $k = H(x)$ *și* $x \leftarrow^R \mathbb{Z}_N$;
- $\mathrm{Dec_{sk}(y, c)} = Dec_k(c)$, *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 20.2 -
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 $c = (r||m)^e \mod N$;

3. **Dec**: dată o cheie secretă (N, d) și un mesaj criptat $c \in \mathbb{Z}_N$, calculează $c^d \mod N$ 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 $c' = r^e \cdot c \mod N$;
- Răspunsul serverului indică adversarului dacă c' este valid (i.e. începe cu **02**);
- Adversarul folosește răspunsul primit pentru a determina informații despre m;
- Repetă atacul până determină mesajul m.

OAEP

- octombrie 1998 - Laboratoarele RSA introduc un nou standard PKCS demonstrat CCA-sigur...
- ...în modelul $\mathcal{ROM}$ (Random Oracle Model);
- Este vorba despre PKCS #1 v2.0 sau OAEP = Optimal Asymmetric Encryption Standard;

OAEP

- 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:

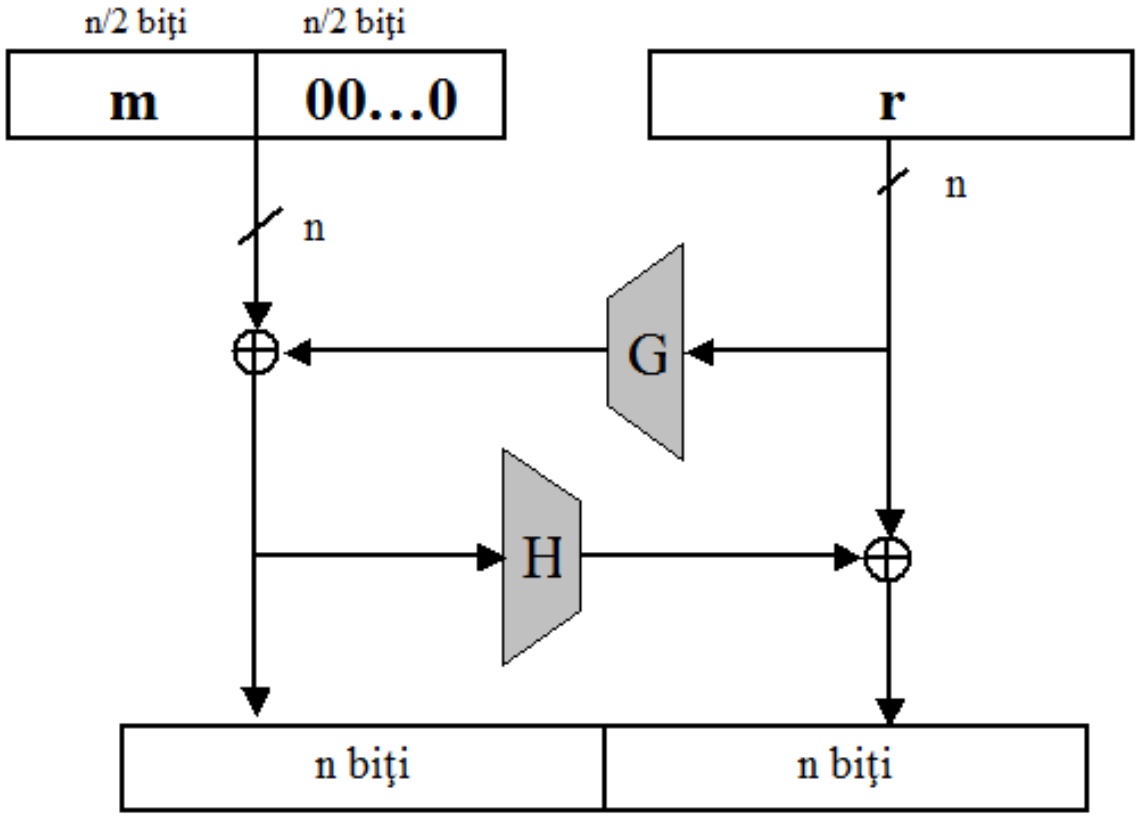
$$(OAEP(m, r))^e \mod N$$

- RSA-OEAP decriptează c folosind cheia secretă (N, d) ca:

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

OAEP

- OAEP este definit astfel:



Notații

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

Important de reținut!

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

Criptografie și Securitate

- Prelegerea 21.1 -
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

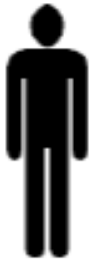
- **Sistemele de criptare cu cheie publică** le-am studiat și le vom mai studia în detaliu;
- **Semnăturile digitale** sunt analogul MAC-urilor din criptografia simetrică (sau corespondentul digital unei semnături reale);
- **Schimbul de chei** îl introducem pentru a facilita introducerea sistemelor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

Schimb de chei

- Un protocol de schimb de chei este un protocol prin care 2 persoane care nu partajează în 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;
- Protocoalele 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

Alice



Bob

$$\xrightarrow{\mathbb{G}, q, g, h_1}$$

$x \leftarrow \mathbb{Z}_q$ $\qquad$ $y \leftarrow \mathbb{Z}_q$

$h_1 := g^x$ $\qquad$ $h_2 := g^y$

$$\xleftarrow{h_2}$$

$k_A := h_2^x$ $\qquad$ $k_B := h_1^y$

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 trimite 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 trimite 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):

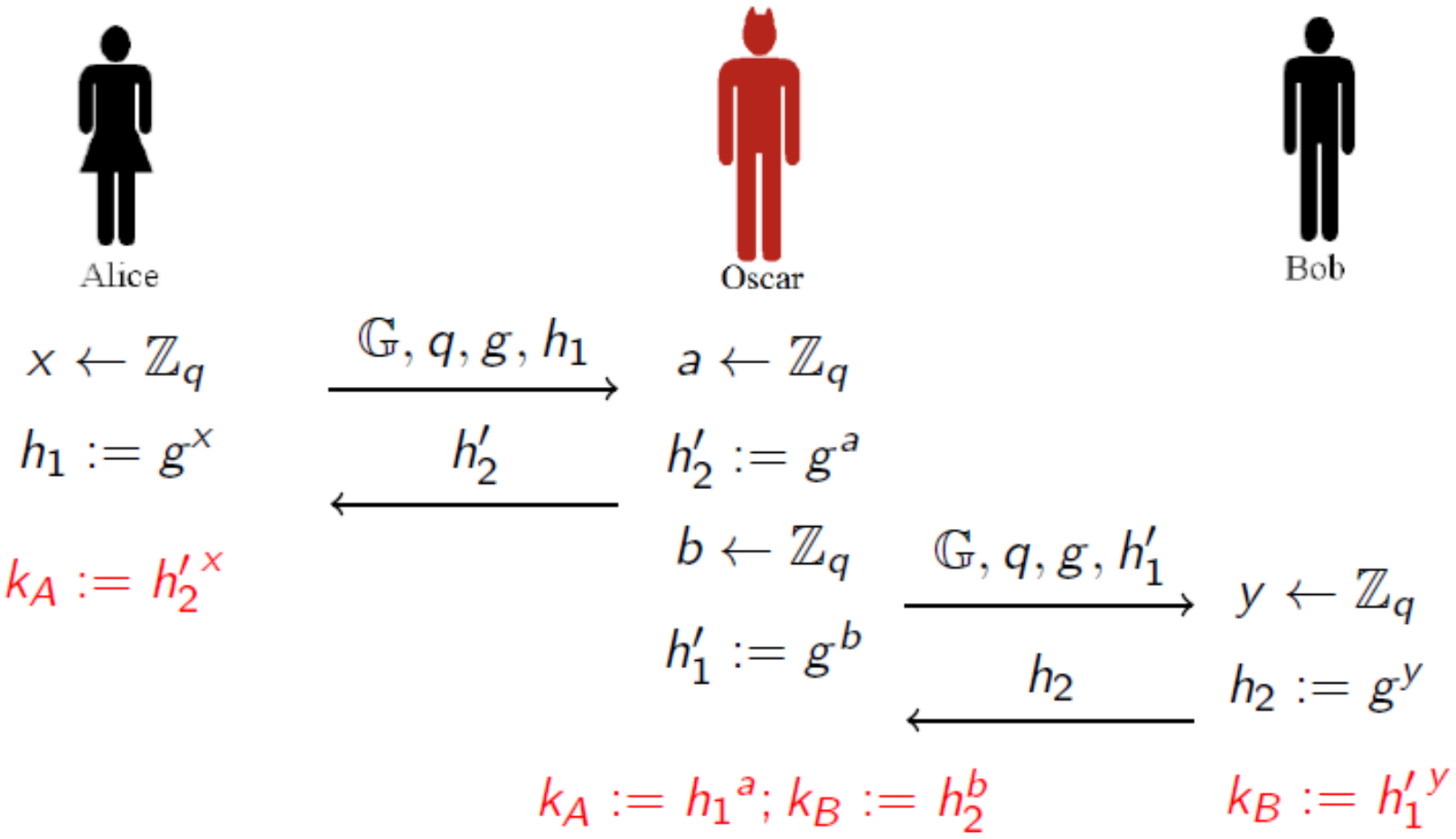
Definiție

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



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 trimite 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 cripografică importantă
- Prezumții criptografice: CDH, DDH
- Schimbul de chei Diffie-Hellman nu rezistă la atacuri active

Criptografie și Securitate

- Prelegerea 21.2 -
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 emițătorul 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 $Dec_s k(c_1 \cdot c_2) = Dec_{sk}(c_1) \cdot Dec_{sk}(c_2)$ 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 întâmpla și la RSA, unde modulul NU trebuie utilizat de mai multe ori.

Securitate - teoremă

Teoremă

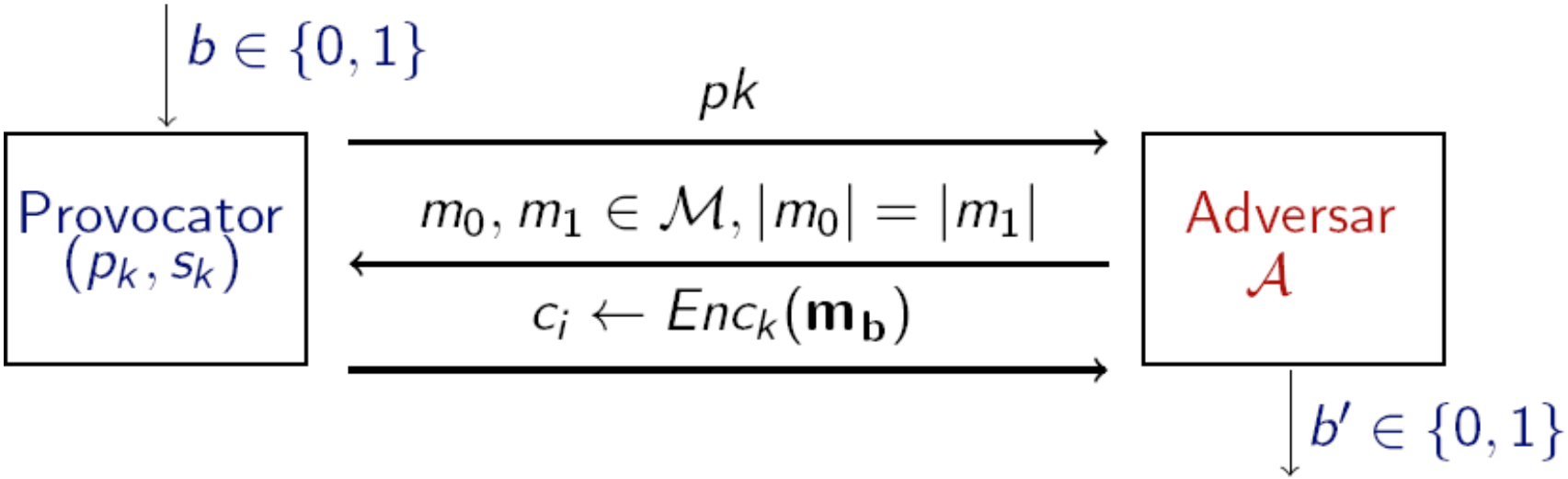
Dacă problema decizională Diffie-Hellman (DDH) este dificilă în grupul $\mathbb{G}$, atunci schema de criptare ElGamal este CPA-sigură.

- Notăm cu Π schema de criptare ElGamal. E suficient să arătăm că schema este sigură la interceptare simplă;
- Fie $\mathcal{A}$ un adversar PPT; notăm cu

$$\epsilon(n) = Pr[PubK^{eav}_{\mathcal{A},\Pi}(n) = 1]$$

probabilitatea ca $\mathcal{A}$ să câstige experimentul de mai jos folosit pentru a defini securitatea la interceptare simplă.

Demonstrație



- Considerăm schema modificată $\tilde{\Pi}$ care diferă de schema Π 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

$$Pr[PubK^{eav}_{\mathcal{A},\tilde{\Pi}}(n) = 1] = \frac{1}{2}$$

- Deși $\tilde{\Pi}$ nu e o schemă de criptare (nu se poate decripta), experimentul $PubK^{eav}_{\mathcal{A},\tilde{\Pi}}(n)$ este bine-definit pentru că folosește doar algoritmul de criptare;
- Arătăm 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. Îi 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 $PubK^{eav}_{\mathcal{A},\tilde{\Pi}}(n)$
- Cum $\mathcal{D}$ întoarce 1 exact atunci când output-ul b' al lui $\mathcal{A}$ este egal cu b, rezultă că:

Demonstrație

$$Pr[D(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] = Pr[PubK^{eav}_{\mathcal{A},\tilde{\Pi}}(n) = 1] = \frac{1}{2}$$

- **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 $PubK^{eav}_{\mathcal{A},\Pi}(n)$

Demonstrație

- Cum $\mathcal{D}$ întoarce 1 exact atunci când output-ul b' al lui $\mathcal{A}$ este egal cu b, rezultă că:

$$Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] = Pr[PubK^{eav}_{\mathcal{A},\Pi}(n) = 1] = \epsilon(n)$$

- Dar cum problema DDH este dificilă, rezultă că există o funcție neglijabilă negl a.î.

$$\mathrm{negl}(n) \geq$$
$$|Pr[D(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - Pr[D(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]|$$

$$= \left|\frac{1}{2} - \epsilon(n)\right|$$

- Adică $\epsilon(n) \leq \frac{1}{2} + \mathrm{negl}(n)$ $\square$.

Important de reținut!

- Sistemul de criptare ElGamal
- Proprietatea de homomorfism