Cryptologie

IF202
2020-2021
Corentin Travers

Administration

Organisation

- Page du cours : moodle Info/2ième année/IF202

- CM en télé-enseignement, TDs a priori en présentiel (machine1/machine2/sans machine/à distance le vendredi)

- Chargés de TDs: Pierre Ramet pierre.ramet@labri.fr Gwénolé Lucas gwenole.lucas@inria.fr Corentin Travers ctravers@enseirb.fr

- évaluation: Contrôle continu DMs + TP noté (selon condition sanitaire)

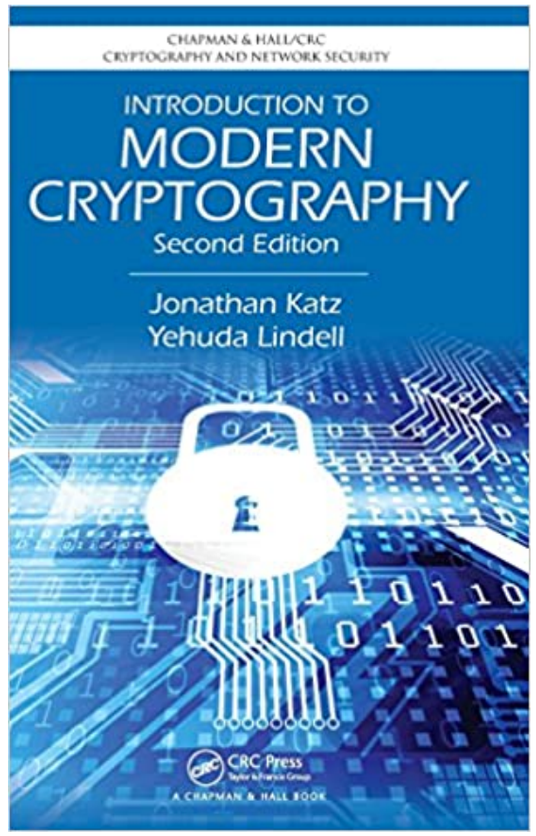
Intro

Références

- J. Katz, Y. Lindell *Introduction to modern cryptography*

- D. Boneh, V. Shoup *A Graduate Course in Applied Cryptography https://toc.cryptobook.us/book.pdf*

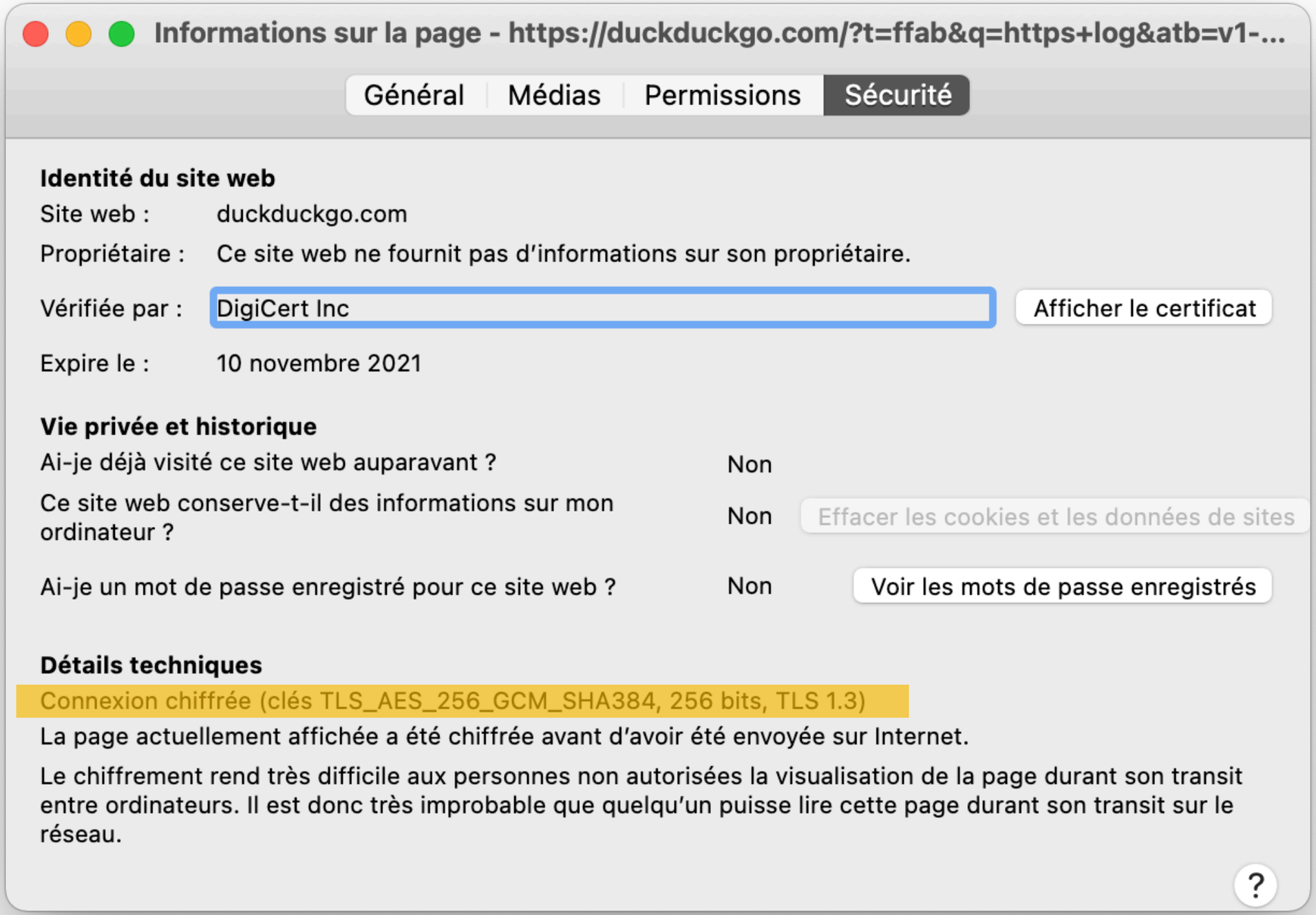
- V. Shoup *A Computational Introduction to Number Theory and Algebra https://www.shoup.net/ntb/ntb-v2.pdf*



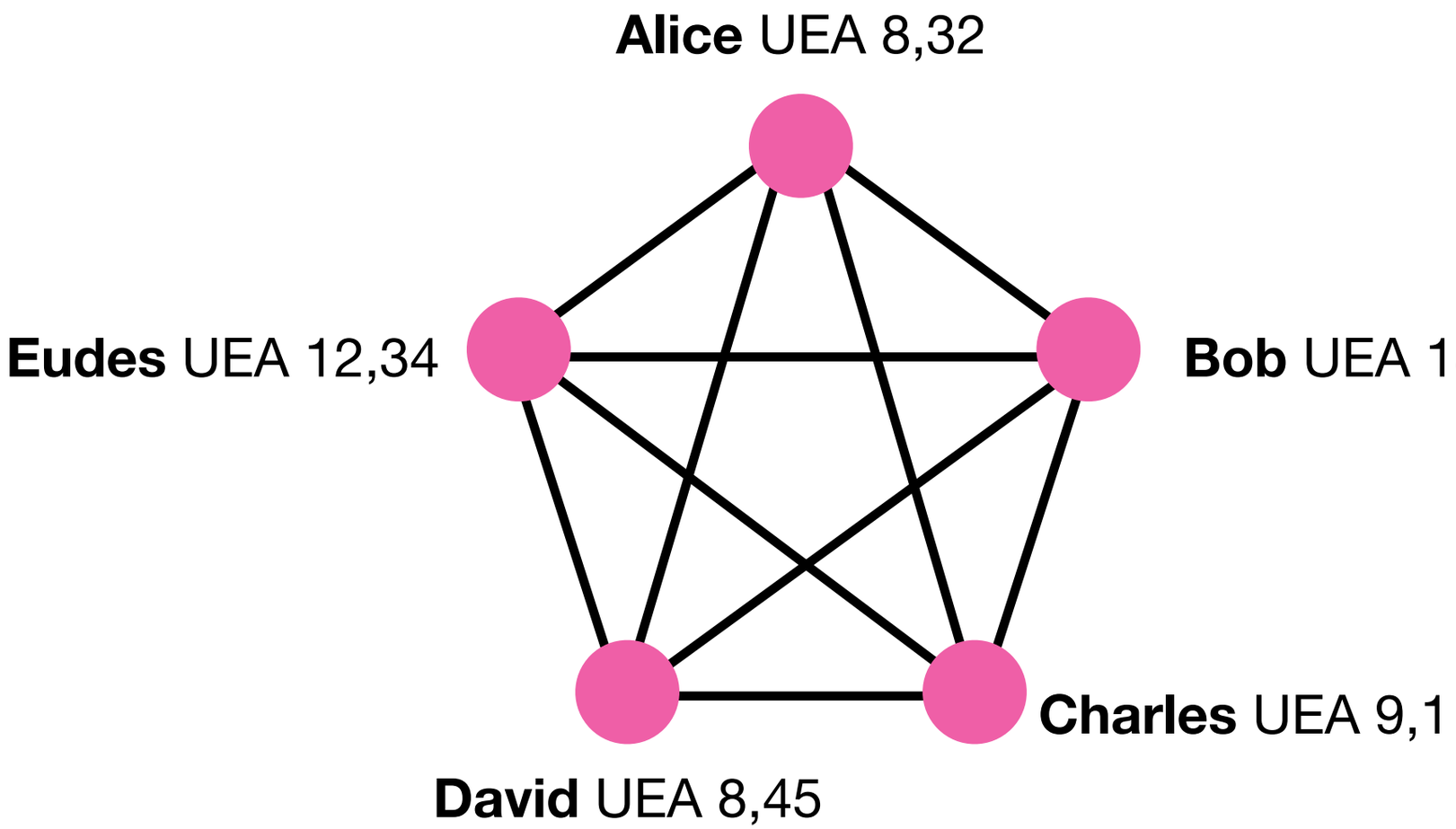
Objectifs

- Fondations pour la cybersécurité
- « Briques de base » pour la construction de systèmes sûrs
- Notion de sécurité
- Quelques applications modernes : blockchain, calcul réparti sûr, calcul réparti préservant la vie privée, etc.

Communication sûre

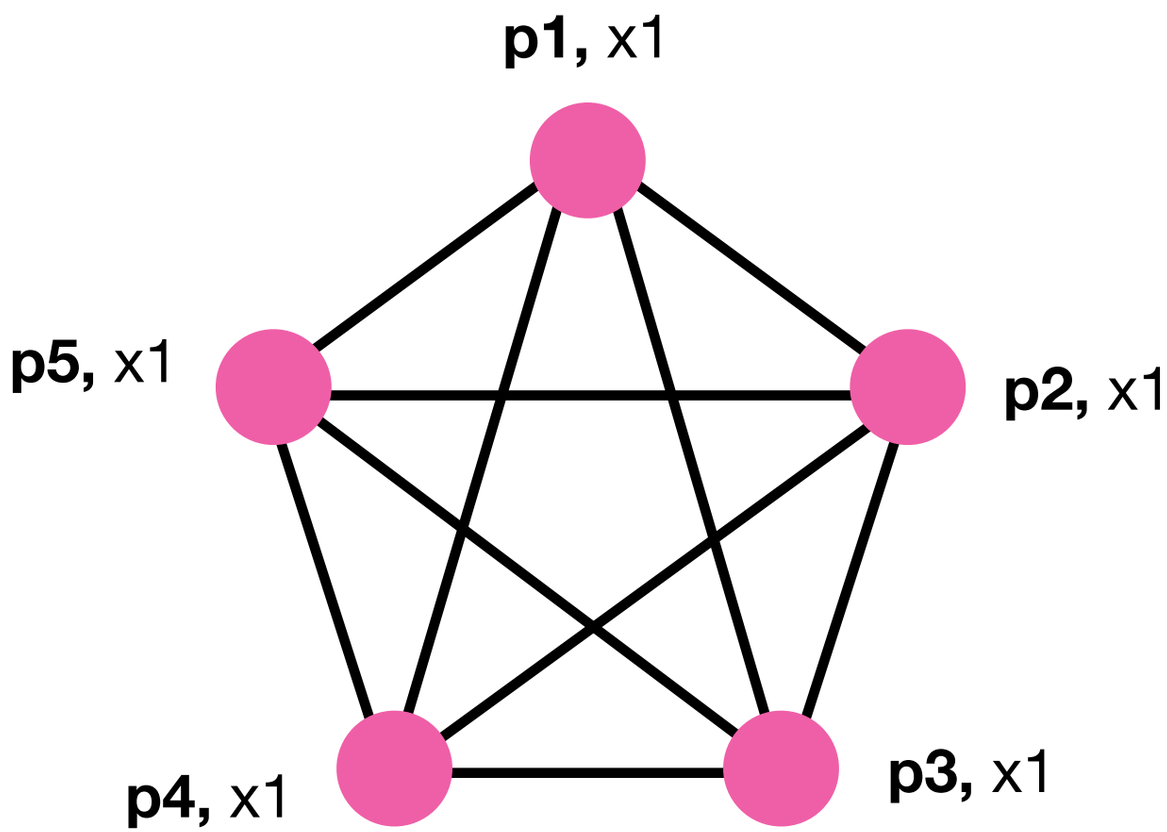


Calcul réparti sûr



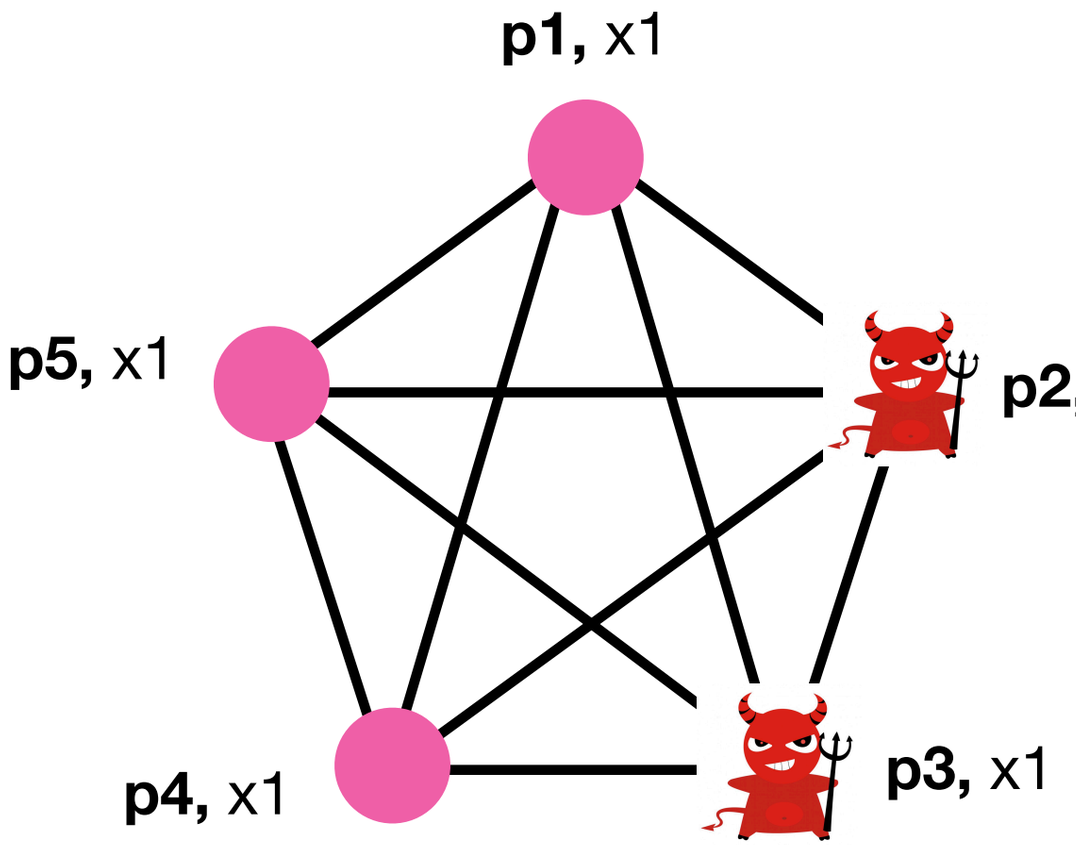
A,B,C,D,E souhaitent connaître combien parmi eux sont au rattrapage
sans révéler leurs notes

Calcul réparti sûr



calculer f(x1,…,xn)
A l’issue du protocole : pi connaît f(x1,…,xn) mais ne connaît pas xj, j != i

Calcul réparti sûr



calculer f(x1,…,xn)
A l'issue du protocole : pi connaît f(x1,…,xn) mais ne connaît pas xj, j != i

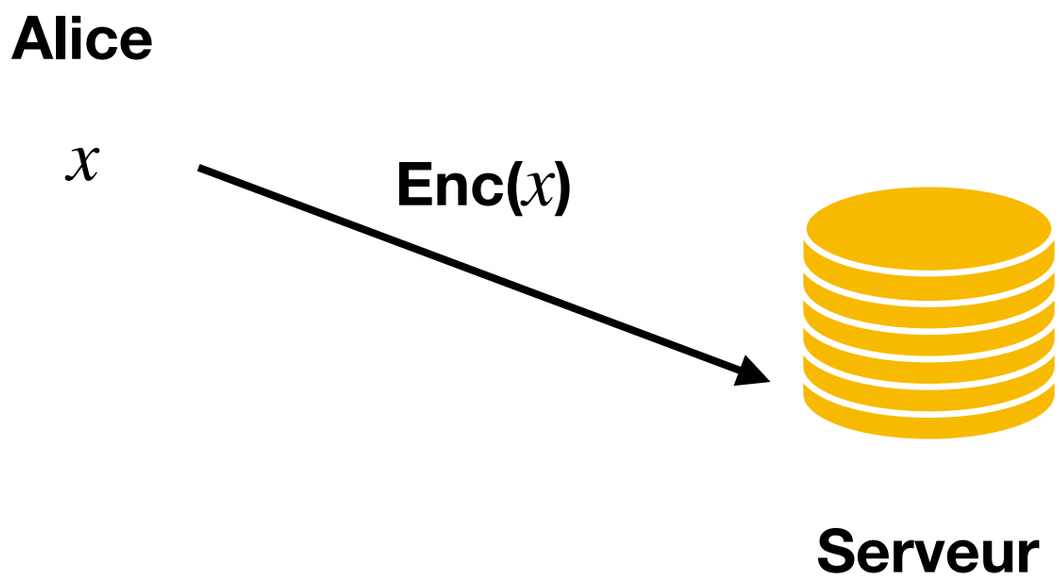
Chiffrement homomorphe

Alice

x

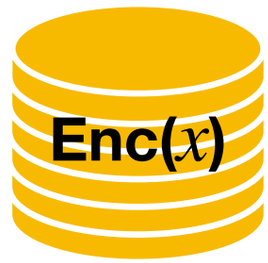
Serveur

Chiffrement homomorphe



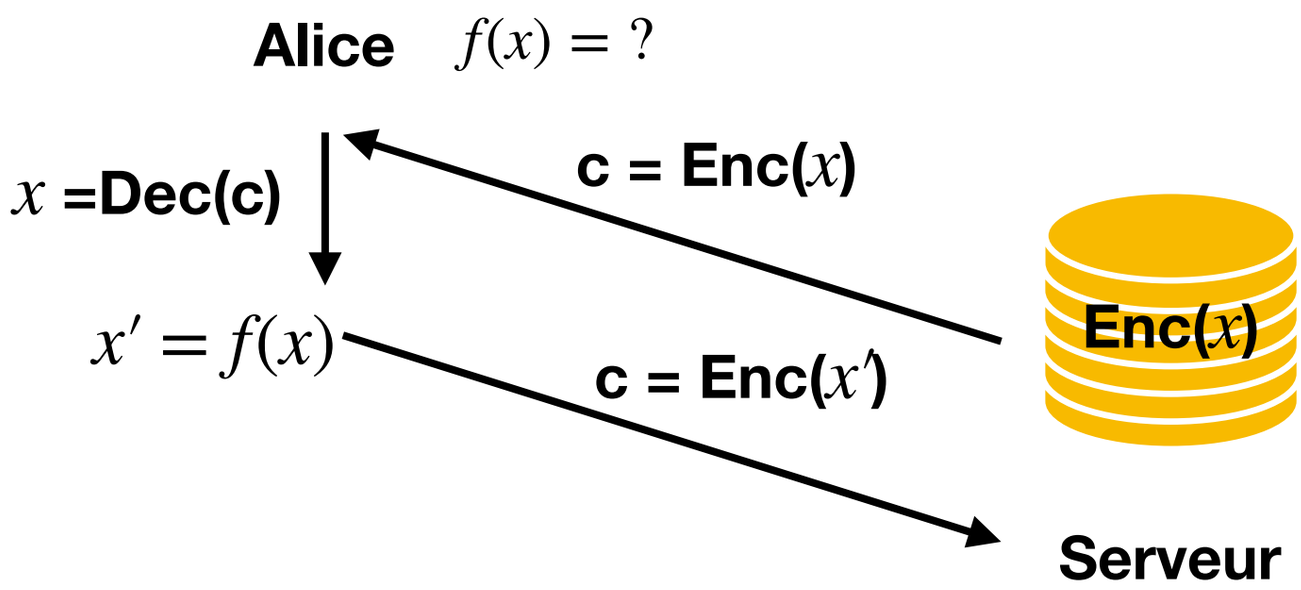
Chiffrement homomorphe

Alice $f(x) = ?$



Serveur

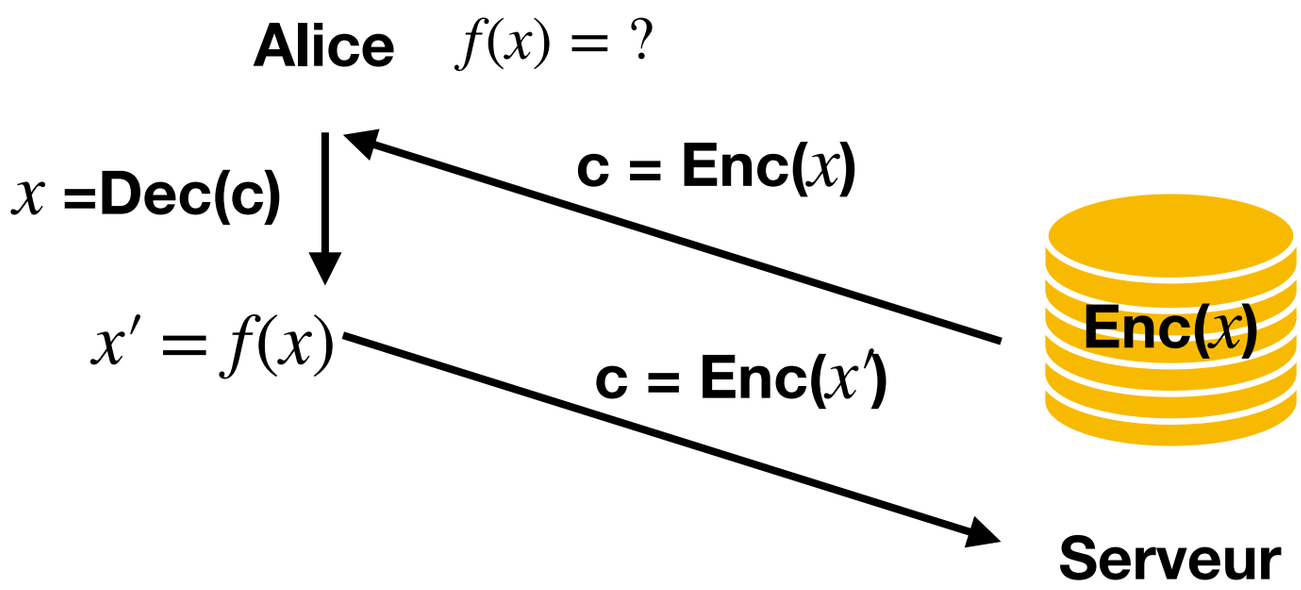
Chiffrement classique : modification



- Serveur stocke les données chiffrées d'Alice
- Modification de x : récupérer la version chiffrée,

déchiffrer, modifier, chiffrer à nouveau et transmettre au serveur

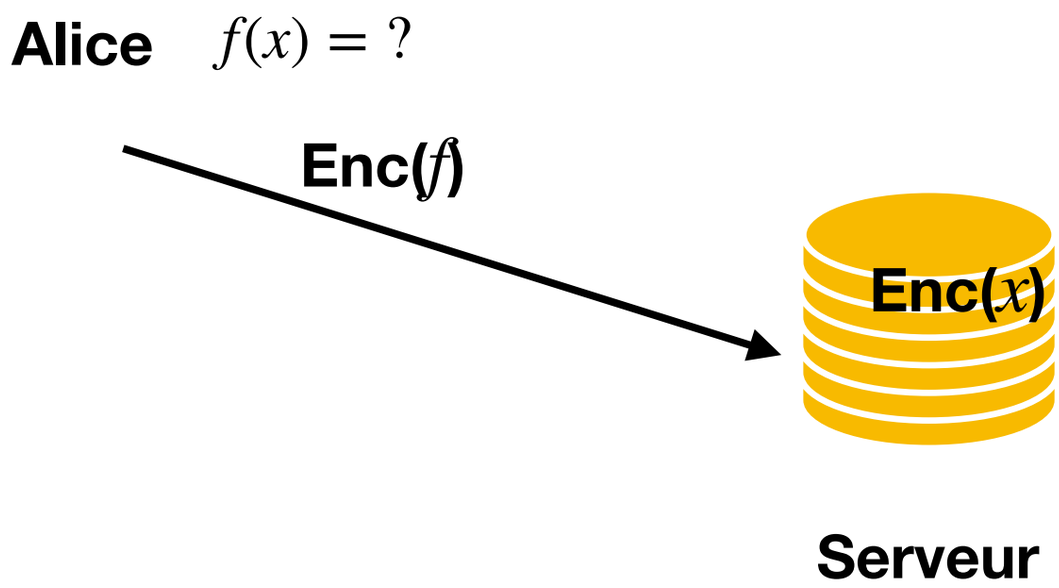
Chiffrement classique : modification



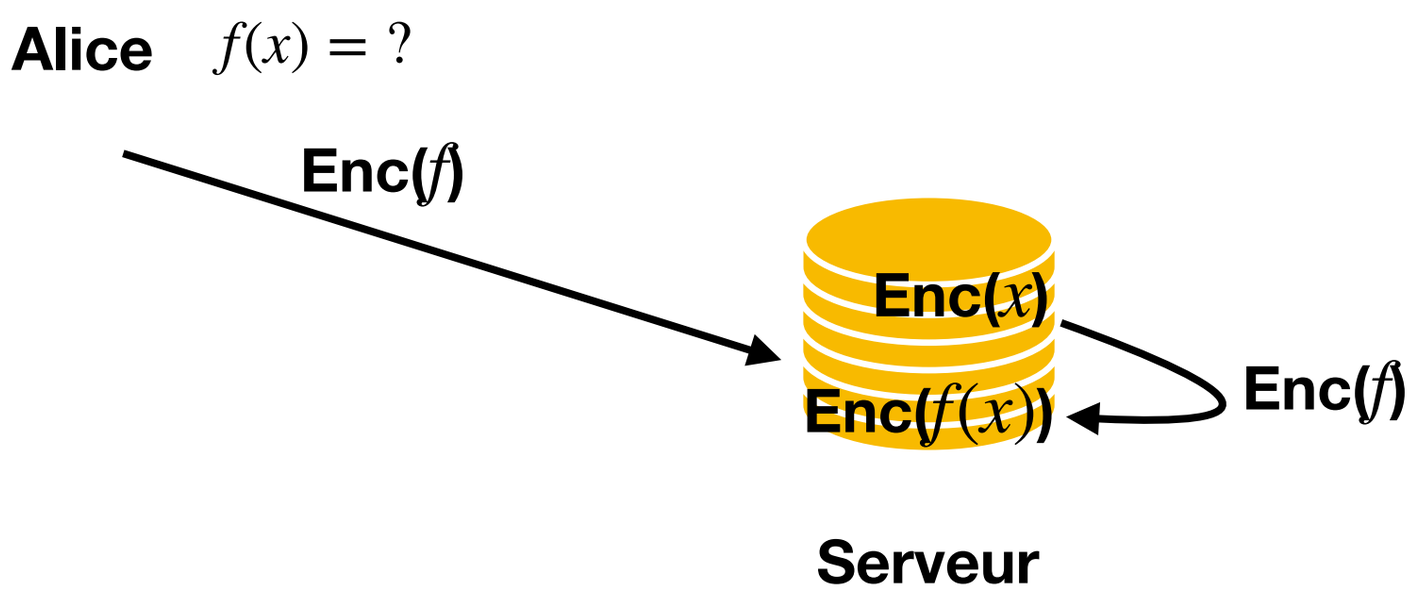
- Serveur stocke les données chiffrées d'Alice
- Modification de x : récupérer la version chiffrée,

déchiffrer, modifier, chiffrer à nouveau et transmettre au serveur

Chiffrement homomorphe : modification



Chiffrement homomorphe : modification



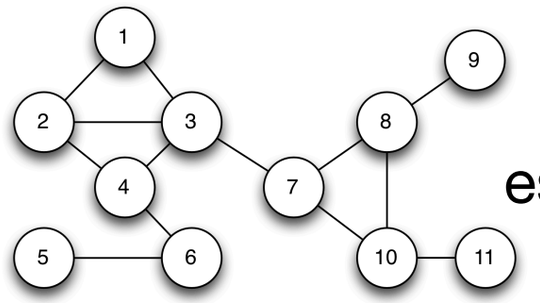
Le serveur ne peut pas connaître

- x
- $f(x)$
- **ni même** f

Prouveur/vérifieur

Valérie **Paul**

G

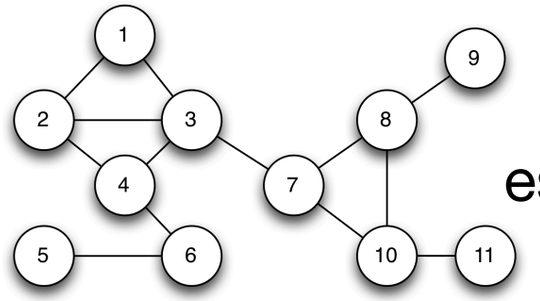


est-ce que G est 3-coloriable ?

Prouveur/vérifieur

Valérie **Paul**

G



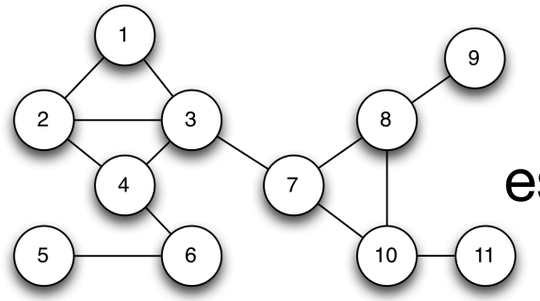
est-ce que G est 3-coloriable ?

oui !

Prouveur/vérifieur

Valérie **Paul**

G



est-ce que G est 3-coloriable ?

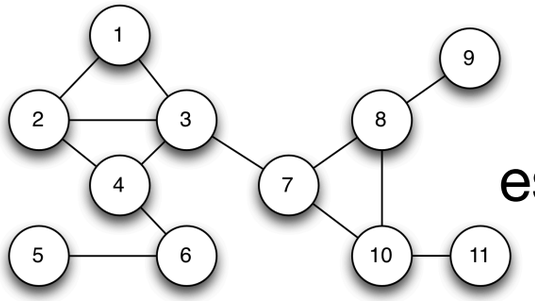
oui !

Prouve-le !

Prouveur/vérifieur

Valérie **Paul**

G

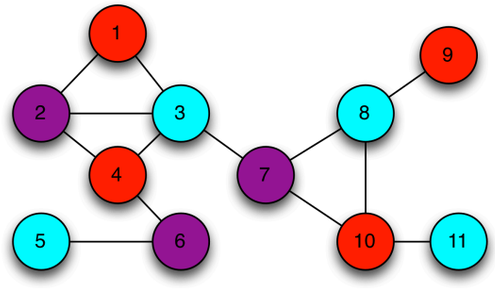


est-ce que G est 3-coloriable ?

oui !

Prouve-le !

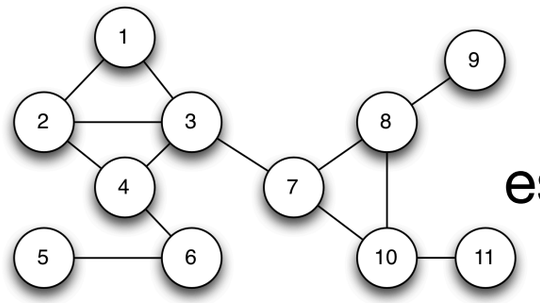
G colorié



Zero Knowledge Proof

Valérie **Paul**

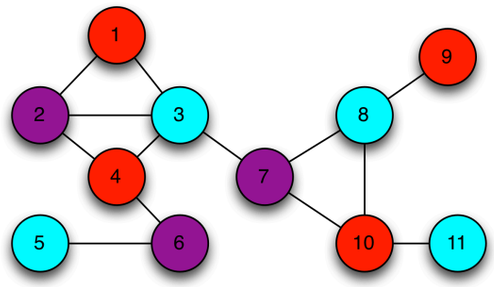
G



est-ce que G est 3-coloriable ?

oui !

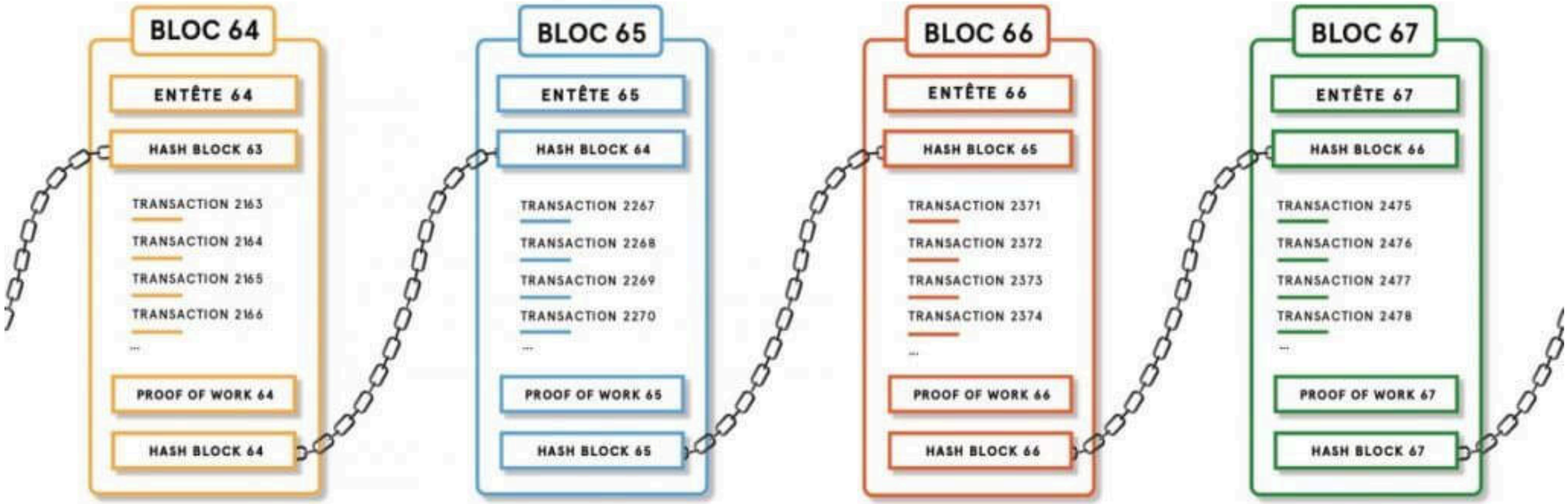
Prouve-le !



Protocole à divulgation nulle

- interactif
- ZK: **V** n’apprend pas comment 3-colorier **G**
- **V** convaincu si **P** sait 3-colorier **G**
- **V** détecte si **P** ment

Blockchain



Registre « append-only » réparti

- **hash cryptographique**
- **clés publiques**
- **signatures**

Chiffrement à clé privée

Communication sûre

Réseau (non sûr)

Alice

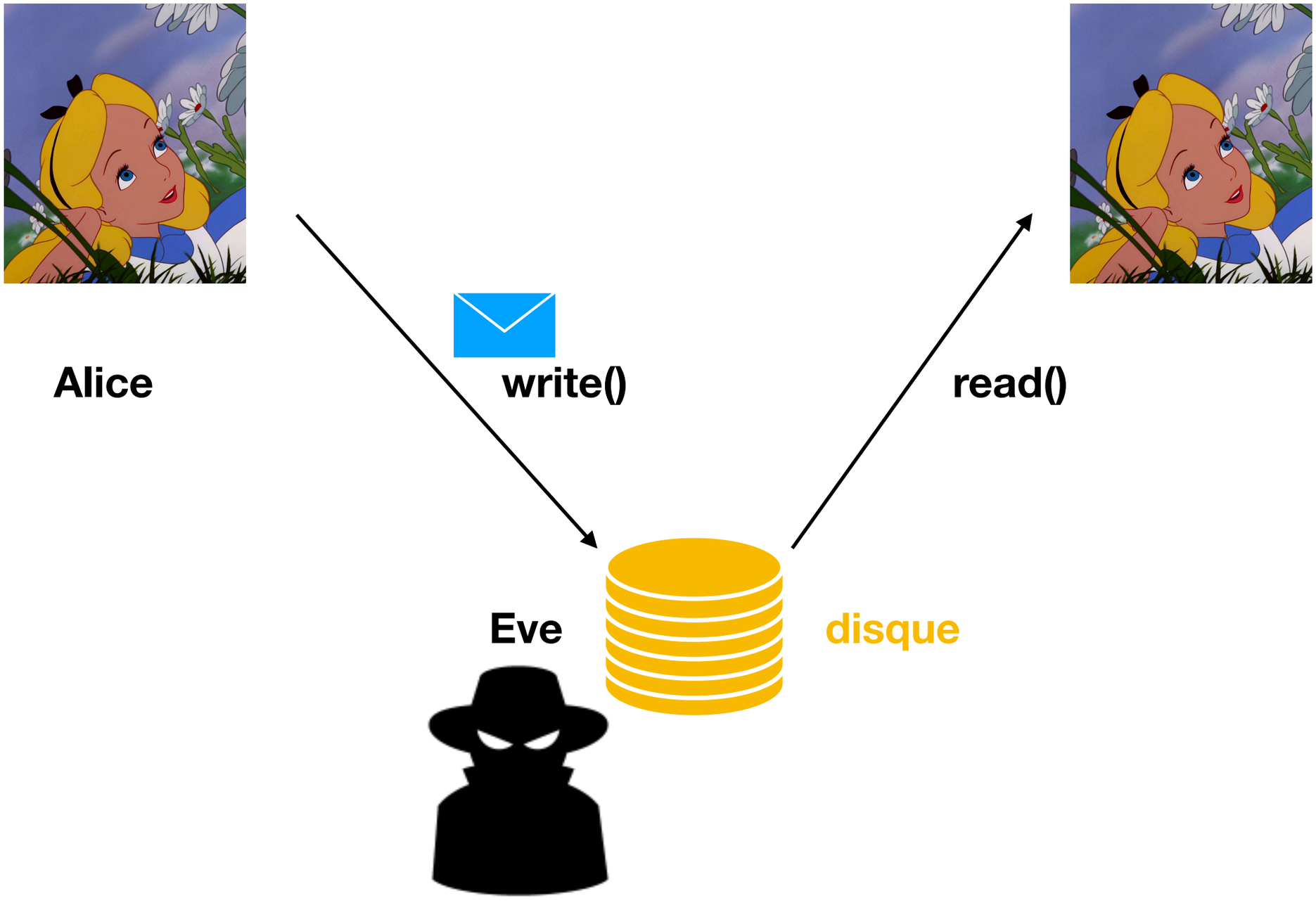
Eve

Bob

Communication sûre ?

- **Confidentialité** : Eve ne peut pas connaître **m**
- **Intégrité** : message reçu par Bob = **m**
- **Authenticité** : message reçu par Bob provient bien d’Alice

Communication sûre



Confidentialité

- Clé privée : Alice et Bob partagent un secret
- Clé publique : Pas de secret partagé

- Un seul message/plusieurs messages

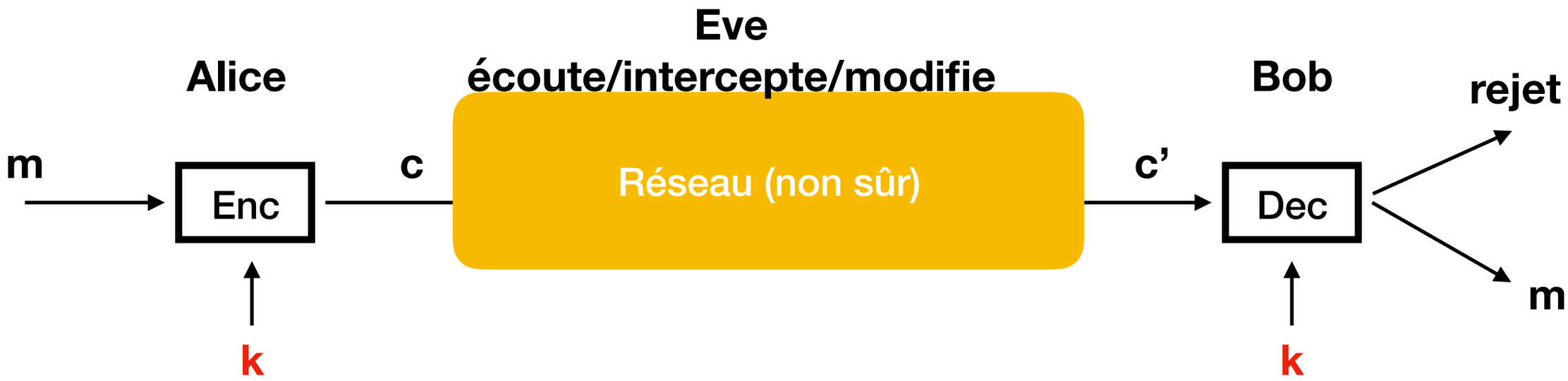
- Définitions : sécurité parfaite, sécurité sémantique/ calculatoire

- « Pouvoir » de l’Espion : écoute, modification des messages

Cryptosystème

- **Enc, Dec :** algorithmes de chiffrement/déchiffrement
- **k :** clé secret partagé par Alice et Bob
- **m :** message **en clair**
- **c :** message **chiffré**

Cryptosystème



- **Enc, Dec :** algorithmes de chiffrement/déchiffrement
- **k :** clé secret partagé par Alice et Bob
- **m :** message **en clair**
- **c :** message **chiffré**
- **c':** message **c** après modification éventuelle par Eve

Cryptosystème

Cryptosystème : (Enc,Dec,K,M,C)

- **Enc : K x M -> C** chiffrement peut-être probabiliste
- **Dec : K x C -> M** déchiffrement
- **K** espace des clés
- **M** espace des messages en clair
- **C** espace des messages chiffrés

Cryptosystème

Cryptosystème : (Enc,Dec,K,M,C)

- **Enc : K x M -> C** chiffrement peut-être probabiliste
- **Dec : K x C -> M** déchiffrement
- **K** espace des clés
- **M** espace des messages en clair
- **C** espace des messages chiffrés

Validité $\forall m \in M, \forall k \in K : \mathrm{Dec}(k, \mathrm{Enc}(k, m)) = m$

Quelques exemples historiques (non sûrs)

Notations

- A = {‘a’,…,’z’} parfois identifié avec {0,1,…,25}

- $M = C = A^{\ell}$ où ℓ est un entier

César

Chiffrement : remplacer chaque lettre du message en clair
par la lettre à distance 3

c e s a r

F H V D U

César

Chiffrement : remplacer chaque lettre du message en clair
par la lettre à distance 3

c e s a r

F H V D U

Enc($m_1 m_2 \ldots m_\ell$**) =**$C_1 C_2 \ldots C_\ell$ avec $C_i = (m_i + 3) \mod 26$

Aisément cassé : K = {3} !

Décalage

clé : entier $d \in \{0,\ldots,25\}$

Chiffrement : remplacer chaque lettre du message en clair par la lettre à distance d

b	o	n	j	o	u	r
6	6	6	6	6	6	6
H	U	T	P	U	A	X

Enc$(d, m_1 m_2 \ldots m_\ell) = C_1 C_2 \ldots C_\ell$ avec $C_i = (m_i + d) \mod 26$

Sûr ? **|K| = 26**

Attaque « brute-force » : Essayer toutes les clés possibles

Substitution

clé : permutation $p : \{0,\ldots,25\} \to \{0,\ldots,25\}$

Chiffrement : remplacer chaque lettre x du message en clair par la lettre $p(x)$

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

m	b	o	n	j	o	u	r
c	N	M	L	B	M	U	R

Enc($p, m_1 m_2 \ldots m_\ell$) = $C_1 C_2 \ldots C_\ell$ avec $C_i = p(m_i)$

Sûr ? $|K| = 26!$

Substitution

clé : permutation $p : \{0,\ldots,25\} \rightarrow \{0,\ldots,25\}$

Chiffrement : remplacer chaque lettre x du message en clair par la lettre $p(x)$

Attaque recherche exhaustive : essayer toutes les clés possibles ?

|K| = **26!** $\simeq 4.10^{26}$

tester 10^6 **clés/sec**

temps pour trouver la clé ? $\simeq 1{,}28.10^{13}$ **ans** **1 machine**

$\simeq 1{,}28.10^{9}$ **ans** **10 000 machines**

Attaque par analyse fréquentielle

répartition (en %) des lettres en français/anglais

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Français	9,42	1,02	2,64	3,39	15,87	0,95	1,04	0,77	8,41	0,89	0,00	5,34	3,24	7,15	5,14	2,86	1,06	6,46	7,90	7,26	6,24
Anglais	8,08	1,67	3,18	3,99	12,56	2,17	1,80	5,27	7,24	0,14	0,63	4,04	2,60	7,38	7,47	1,91	0,09	6,42	6,59	9,15	2,79

	V	W	X	Y	Z
Français	2,15	0,00	0,30	0,24	0,32
Anglais	1,00	1,89	0,21	1,65	0,07

SEGELAZEWAOPQJLNKFAPZAJYUYHKLAZEACNWPQEPAAYNEPAYKKLANWPERAIAJP

Lettres	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	**Total**
Occurrences	12	0	1	0	7	1	1	1	1	3	4	4	0	4	1	7	2	1	1	0	1	0	3	0	4	3	62
Fréquences	19,4	0	1,6	0	11,3	1,6	1,6	1,6	1,6	4,8	6,5	6,5	0	6,5	1,6	11,3	3,2	1,6	1,6	0	1,6	0	4,8	0	6,5	4,8	**100**

p(e) = A, p(i) = E ou P, p(a) = E ou P, etc.

source: Wikipedia

Substitution

- Chiffrement par substitution mono-alphabétique :
  - Espace des clés très grand : attaque brute-force irréalisable
  - une lettre est toujours chiffrée de la même façon
  - Mais : caractéristique du clair (répartition des fréquences) conservée dans le chiffré -> Attaque facile

Substitution poly-alphabétique

- Chiffrement d’une lettre dépend
- de la lettre
- **et** de sa position dans le clair

Vigenère

Clé = enseirb

m =	j	e	n	a	i	m	e	p	a	s	l	i	n	f	o	r	m	a	t	i	q	u	e
clé	**e**	**n**	**s**	**e**	**i**	**r**	**b**	**e**	**n**	**s**	**e**	**i**	**r**	**b**	**e**	**n**	**s**	**e**	**i**	**r**	**b**	**e**	**n**
c =	N	R	F	E	Q	D	F	T	N	K	P	Q	E	G	S	E	E	E	B	Z	R	Y	R

K = $\{k \in A^n\}$

Enc($k, m_0 \ldots m_\ell$**) =** $C_0 \ldots C_\ell$ **avec** $C_i = m_i + k_{i \mod n} \mod 26$

Dec($k, C_0 \ldots C_\ell$**) =** $m_0 \ldots m_\ell$ **avec** $m_i = C_i - k_{i \mod n} \mod 26$

Vigenère (XVIème)

Clé = enseirb

m =	j	e	n	**a**	i	m	e	p	**a**	s	l	i	n	f	o	r	m	**a**	t	i	q	u	e
clé	**e**	**n**	**s**	**e**	**i**	**r**	**b**	**e**	**n**	**s**	**e**	**i**	**r**	**b**	**e**	**n**	**s**	**e**	**i**	**r**	**b**	**e**	**n**
c =	N	R	F	**E**	Q	D	F	T	**N**	K	P	Q	E	G	S	E	E	**E**	B	Z	R	Y	R

K = $\{k \in A^{\ell}\}$

Enc($k, m_0 \ldots m_\ell$**) =** $C_0 \ldots C_\ell$ **avec** $C_i = m_i + k_{i \mod \ell} \mod 26$

Dec($k, C_0 \ldots C_\ell$**) =** $m_0 \ldots m_\ell$ **avec** $m_i = C_i - k_{i \mod \ell} \mod 26$

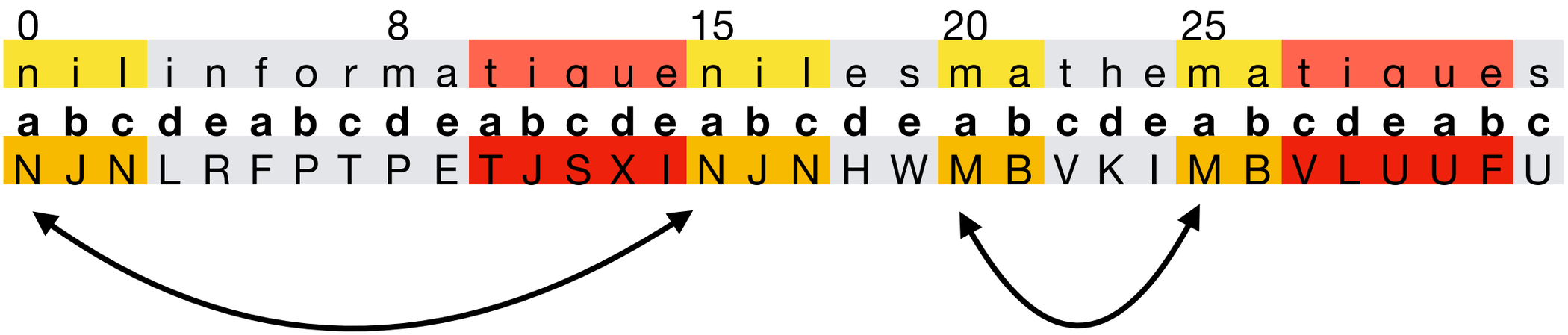
Vigenère : Attaque

Vigenère : Attaque

- **Taille n de la clé connue :** Analyse fréquentielle
- n **inconnu ?**

Vigenère : Attaque

- **Taille n de la clé connue :** Analyse fréquentielle
  lettres $m_i, m_{i+n}, m_{i+2n}, \ldots$ chiffrées par le même décalage
- **n inconnu ?**



distance entre sous-mots identiques =? multiple de n

attaque de Kasiski/Babbage (XIXème)

Cryptosystèmes historiques

- Décalage, substitution mono/poly-alphabétique

- Attaques faciles : brute-force, ou fondées sur le fait que le chiffré conserver des caractéristiques du clair

- Attaque à *texte chiffré seul*

Typologie des attaques

c : message à déchiffrer par l'attaquant

1. **Texte chiffré seul** (*ciphertext-only attack)* Attaquant connaît uniquement des messages chiffrés c1,…,cx

2. **Texte clair connu** (*known-plaintext attack)* Attaquant connaît (m1,c1), …, (mx,cx) mi = Enc(k,ci), ci != **c**

3. **Texte clair choisi** (*chosen-plaintext attack*) Attaquant connaît (m1,c1), …, (mx,cx) mi = Enc(k,ci) **choisi** par l'attaquant

4. **Texte chiffré choisi** (*chosen-ciphertext attack*) Attaquant connaît (m1,c1), …, (mx,cx), ci = Dec(k,mi) **choisi** par l'attaquant ci !=**c**

En pratique, on souhaite résister aux attaques de **type 3. voire 4.**

Qu’est ce qu’un cryptosytème sûr ?

Qu’est ce qu’un cryptosytème sûr ?

- Commet définir le définir ?
- Attaque 1. : texte chiffré seul
- étant donné **c** = Enc(k,m), l’attaquant ne peut pas …

1.trouver la clé m ?

2.trouver m ?

3.trouver les 3premières lettres de m ?

4.???

Qu’est ce qu’un cryptosytème sûr ?

- Commet définir le définir ?
- Attaque 1. : texte chiffré seul
- étant donné **c** = Enc(k,m), l’attaquant ne peut pas …

1.trouver la clé m ?

2.trouver m ?

3.trouver les 3premières lettres de m ?

4.calculer n’importe quel prédicat non-trivial sur m ?

Chiffrement de Vernam (1917)
masque jetable

Rappel $\oplus$ (XOR)

ou exclusif

$\oplus$	0	1
0	0	1
1	1	0

$$x \oplus y = y \oplus x$$

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

$$x \oplus x = 0\ldots0 \qquad x \oplus 0\ldots0 = x$$

Masque jetable

$$M = C = K = \{0,1\}^n$$

$$\mathrm{Enc}(k, m) = c \text{ avec } c_i = k_i \oplus m_i$$

$$\mathrm{Dec}(k, c) = m \text{ avec } m_i = k_i \oplus c_i$$

Exemple

m	0	1	0	1	1
k	1	0	0	1	0
c	1	1	0	0	1

Validité

$$\mathrm{Dec}(k, \mathrm{Enc}(k, m)) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = 0^n \oplus m = m$$

Sécurité inconditionnelle

Attaquant **Eve**

- connaît **c** chiffré
- sait que le message clair **m** $\in \{m_0, m_1\}$ avec
  $m_0 =$ « attaque at dawn » $m_1 =$ « attaque at dusk »
- sait que Pr[**m** $= m_0$] = Pr[**m**=m_1] = 1/2

Sécurité inconditionnelle : la connaissance de **c** *n'apporte aucune information sur* **m**

Exemple

- S'il existe k_0, k_1 telles que $\mathrm{Enc}(k_0, m_0) = c = \mathrm{Enc}(k_1, m_1)$, la connaissance de **c** ne permet d'être certain que **m** = m_0 ou **m** = m_1

- Supposons qu'il existe
  - 800 clés k_0 telles que $\mathrm{Enc}(k_0, m_0) = c$
  - 600 clés k_1 telles que $\mathrm{Enc}(k_1, m_1) = c$

- **Eve** devine que **m** = m_0
  - Proba. qu'elle devine correctement = $\dfrac{800}{800 + 600} \simeq 0{,}57 > 1/2$
  - Proba. de deviner correctement sans connaître **c** = 1/2
  - Proba. de deviner correctement en connaissant **c** = 0,57
  - **Non inconditionnellement sûr**

Sécurité inconditionnelle
Définition

Soit **A** $= (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

k variable aléatoire correspondant au tirage aléatoire uniforme d'une clé dans K

pour $m \in M$, Enc(**k**,m) variable aléatoire correspondant à l'application de la fonction de chiffrement à la v.a. **k** pour le message m

Sécurité inconditionnelle
Définition

Soit **A** $= (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

k variable aléatoire correspondant au tirage aléatoire uniforme d'une clé dans K

pour $m \in M$, Enc(**k**,m) variable aléatoire correspondant à l'application de la fonction de chiffrement à la v.a. **k** pour le message m

Définition

A est inconditionnellement sûr ssi
$\forall m_0 \neq m_1 \in M$ et $\forall c \in C$, on a
$\Pr[\mathrm{Enc}(\mathbf{k}, m_0) = c] = \Pr[\mathrm{Enc}(\mathbf{k}, m_1) = c]$

Vernam est inconditionnellement sûr

Vernam

$$M = C = K = \{0,1\}^n$$
$$\mathrm{Enc}(k, m) = c \text{ avec } c_i = k_i \oplus m_i$$
$$\mathrm{Dec}(k, c) = m \text{ avec } m_i = k_i \oplus c_i$$

Vernam est inconditionnellement sûr

Vernam/Masque jetable

$M = C = K = \{0,1\}^n$

$\mathrm{Enc}(k, m) = c$ avec $c_i = k_i \oplus m_i$

$\mathrm{Dec}(k, c) = m$ avec $m_i = k_i \oplus c_i$

Soit $m_0 \neq m_1 \in M$ et $c \in C$

$$\Pr[\mathrm{Enc}(\mathbf{k}, m_i) = c] = \Pr[\mathbf{k} \oplus m_i = c] \qquad i \in \{0,1\}$$
$$= \Pr[\mathbf{k} = c \oplus m_i]$$
$$= \frac{1}{2^n}$$

Sécurité inconditionnelle définition alternative

Soit **A** $= (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

k variable aléatoire distribution uniforme sur K

m variable aléatoire distribuée sur M

k et **m** sont indépendantes

c variable aléatoire $\mathrm{Enc}(\mathbf{k},\mathbf{m})$

Théorème

- Si **A** est inconditionnellement sûr alors **c** et **m** sont indépendantes (i.e., $\Pr[\mathbf{c} = c \,|\, \mathbf{m} = m] = \Pr[\mathbf{c} = c]$)
- Si **c** et **m** sont indépendantes et $\forall m \in M, \Pr[\mathbf{m} = m] > 0$ alors **A** est inconditionnellement sûr

Théorème

- Si **A** est inconditionnellement sûr alors **c** et **m** sont indépendantes (i.e., $\Pr[\mathbf{c} = c \,|\, \mathbf{m} = m] = \Pr[\mathbf{c} = c]$)
- Si **c** et **m** sont indépendantes et $\forall m \in M, \Pr[\mathbf{m} = m] > 0$ alors **A** est inconditionnellement sûr

Démo

Sécurité inconditionnelle
Bad news

Théorème (Shannon)

Soit **A** $= (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème.
Si **A** est inconditionnellement sûr alors $|K| > |M|$

Sécurité inconditionnelle
Bad news

Théorème (Shannon)

Soit **A** $= (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème.
Si **A** est inconditionnellement sûr alors $|K| > |M|$

Conséquence

- Alice veut envoyer 2Go de vidéo à Bob

-> nécessite qu'ils s'accordent au préalable sur une clé de 2 Go

- Sécurité inconditionnelle quasi-impossible à mettre en oeuvre

en pratique

Sécurité inconditionnelle
Bad news

Théorème (Shannon)

Soit **A** $= (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème.
Si **A** est inconditionnellement sûr alors $|K| > |M|$

Démonstration

Supp. $|K| < |M|$

Soit $k_0 \in K, m_0 \in M$ et soit $c = \mathrm{Enc}(k_0, m_0)$

définissons $M_c = \{m \in M : \exists k_1 \in K, \mathrm{Dec}(k_1, c) = m\}$
autrement dit $M_c = \{\mathrm{Dec}(k_1, c) : k_1 \in K\}$

Puisque $|K| < |M|$, $\exists m_1 \in M \backslash M_c$

On a $\Pr[\mathrm{Enc}(\mathbf{k}, m_0) = c] > 0$ et $\Pr[\mathrm{Enc}(\mathbf{k}, m_1) = c] = 0$

A n'est donc pas inconditionnellement sûr

Sécurité inconditionnelle limites

- Adversaire : attaque à *texte chiffré seul. Ne connaît qu'un seul message chiffré*

- Nécessite des clés au moins aussi longues que les messages en clair: difficile à mettre en oeuvre en pratique

- Masque recyclable : Soit $c_1 = m_1 \oplus k, c_2 = m_2 \oplus k$
  $c_1 \oplus c_2 = ?$

Sécurité inconditionnelle limites

- Adversaire : attaque à *texte chiffré seul. Ne connaît qu'un seul message chiffré*

- Nécessite des clés au moins aussi longues que les messages en clair: difficile à mettre en oeuvre en pratique

- Masque recyclable : Soit $c_1 = m_1 \oplus k, c_2 = m_2 \oplus k$

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$$
$$= m_1 \oplus m_2 \oplus k \oplus k = m_1 \oplus m_2$$

Ne dépend pas de la clé k

Sécurité : version jeu

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

Jeu à 2 joueurs **: Adversaire** et **Challenger**

Partie (ou Expérience) $b \in 0,1$

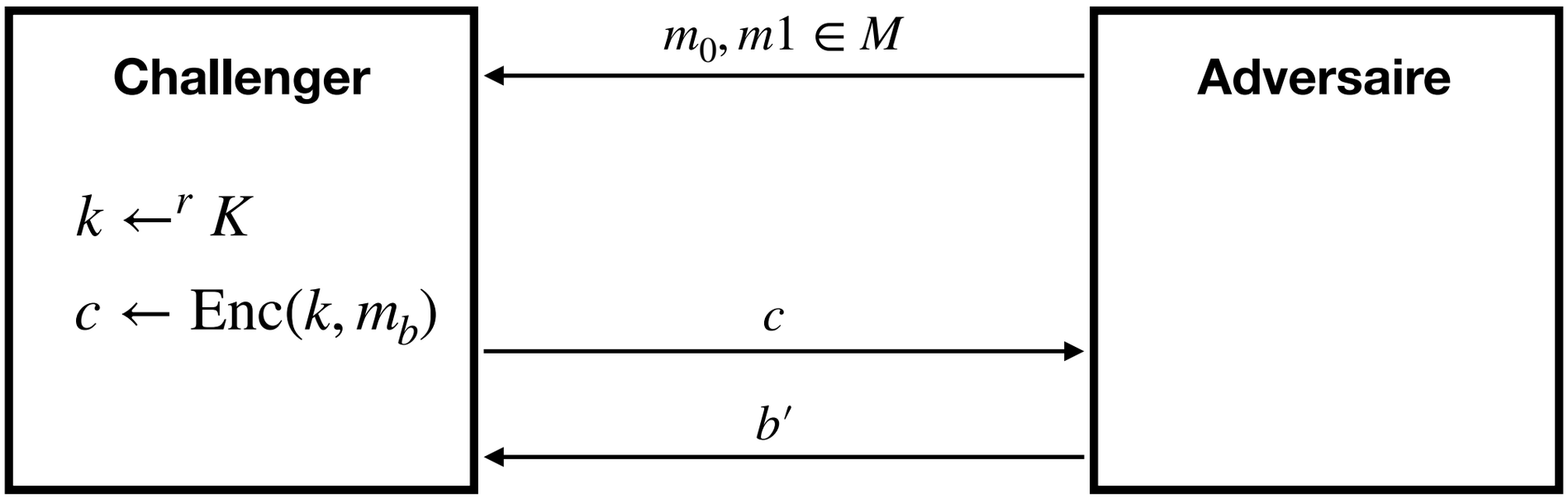
- **A** choisit $m_0, m_1 \in M$ avec $|m_0| = |m_1|$ et les envoie à **C**
- **C** tire une clé k aléatoire de façon uniforme et envoie $c = \mathrm{Enc}(k, m_b)$ à **A**
- **A** produit un bit b'

Adversaire gagne si $b' = b$

Sécurité : version jeu

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

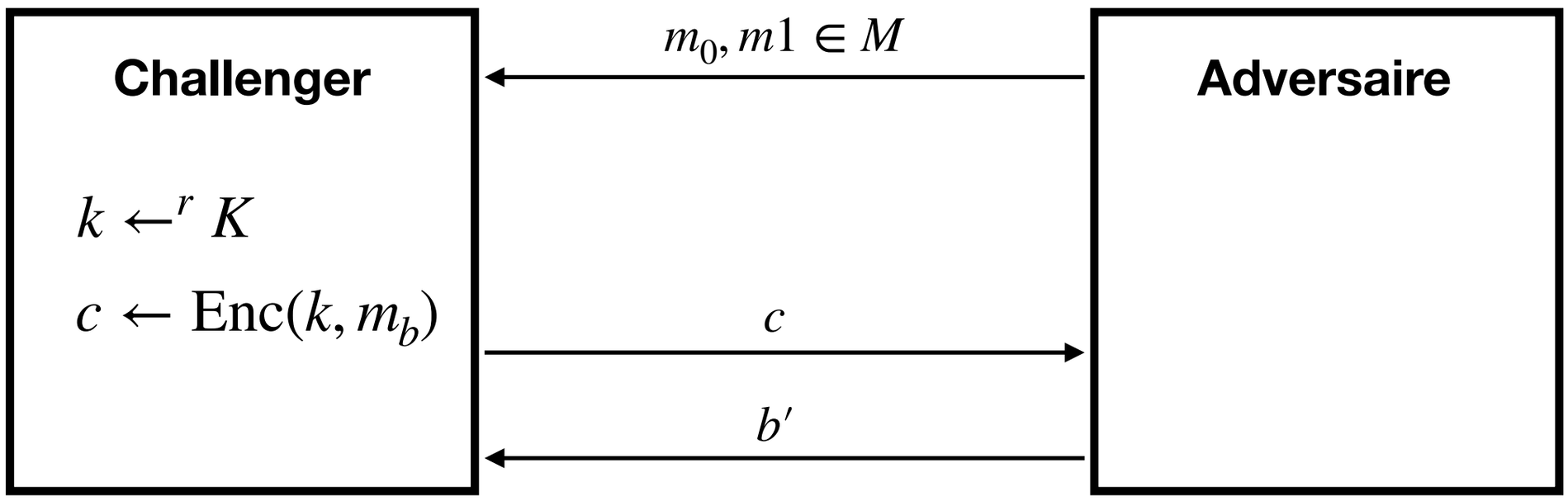
Jeu à 2 joueurs **: Adversaire** et **Challenger**



Sécurité : version jeu

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

Jeu à 2 joueurs **: Adversaire** et **Challenger**



W_b **A** produit $b' = 1$ dans l'expérience b

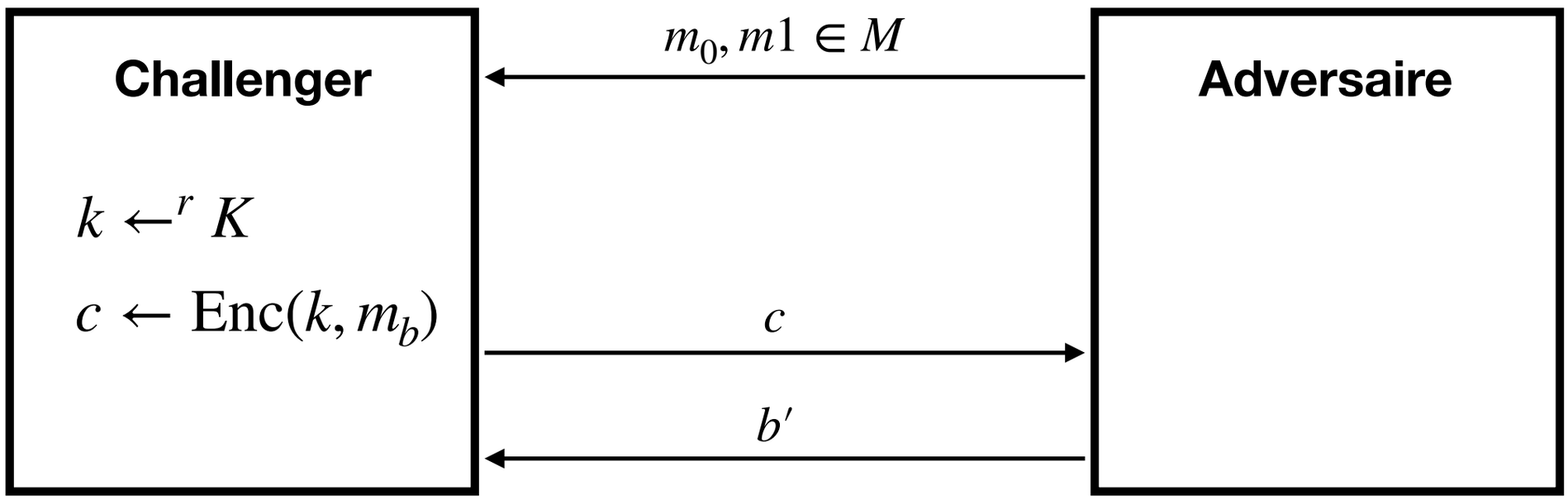
Avantage $Avtg(\mathbf{A}, \mathscr{E}) = |\Pr[W_1] - \Pr[W_0]|$

A gagne dans l'expérience 1

A perd dans l'expérience 0

Sécurité inconditionnelle : version jeu

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème



W_b **A** produit $b' = 1$ dans l'expérience b

Avantage $Avtg(\mathbf{A}, \mathscr{E}) = |\Pr[W_1] - \Pr[W_0]|$

théorème

Si pour tout **A** $Avtg(\mathbf{A}, \mathscr{E}) = 0$ alors $\mathscr{E}$ inconditionnellement sûr

Sécurité inconditionnelle : version jeu

théorème

Si pour tout **A** $Avtg(\mathbf{A}, \mathscr{E}) = 0$ alors $\mathscr{E}$ inconditionnellement sûr

- Pas de limite sur les capacités et temps de calcul de **A**
- Avantage nul : pas de stratégie meilleure que produire b' au hasard

Sécurité inconditionnelle : version jeu

théorème

Si pour tout **A** $Avtg(\mathbf{A}, \mathscr{E}) = 0$ alors $\mathscr{E}$ inconditionnellement sûr

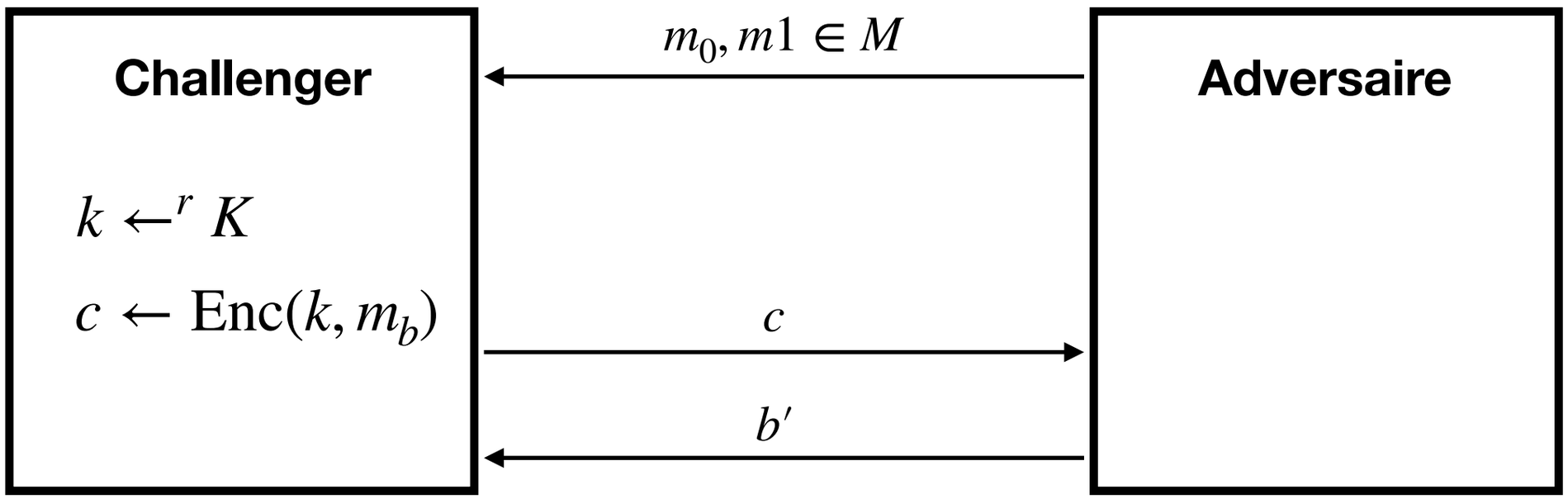
- Pas de limite sur les capacités et temps de calcul de **A**
- Avantage nul : pas de stratégie meilleure que produire b' au hasard

En pratique

- Ressources calculatrices bornées : **A** *efficace*
- Avantage *négligeable* : probabilité extrêmement faible que **A** gagne

Sécurité sémantique

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème



W_b **A** produit $b' = 1$ dans l'expérience b

Avantage $Avtg(\mathbf{A}, \mathscr{E}) = |\Pr[W_1] - \Pr[W_0]|$

Définition

$\mathscr{E}$ est **sémantiquement sûr** ssi pour tout **A** efficace,

$Avtg(\mathbf{A}, \mathscr{E}) = |\Pr[W_1] - \Pr[W_0]| \leq \mathsf{negl}$.

Efficace/négligeable

- **A** efficace : calcul réaliste
  - 100 x 365 x 24 x 3600 x 10^15 opérations ***ok*** (100 ans TOP10 supercomputer)
  - 10^80 x 13.10^9 x 365 x 24 x 3600 x 10^15 ***not ok*** (TOP10 supercomputer par atome dans l'univers pendant 13.10^9 an)
- p **negligeable** : probabilité si faible qu'elle peut être considérée nulle en pratique
  - 1/6 non-négligeable
  - 1/2118760 non-négligeable (prob. gain euro million)
  - 1/2^100 négligeable

Efficace/négligeable

- Un peu plus formellement :
- n : paramètre de sécurité (e.g., taille des clés)
- Efficace : complexité *polynomiale* temps/espace en $O(n^d)$
- $f : \mathbb{N} \to \mathbb{N}$ négligeable si $\forall c > 0, \lim_{n \to +\infty} f(n)n^c = 0$

Négligeable ?

- $f : n \to \dfrac{1}{14590n^4 + n^6 \log n}$
- $f : n \to \dfrac{1}{14590n^{100000}}$
- $f : n \to \dfrac{1}{2^n}$
- $f : n \to \dfrac{1}{2\sqrt{n}}$
- $f : n \to \dfrac{1}{n^{\log n}}$

Sécurité sémantique

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

- Adversaire *choisit* m0, m1

- et est capable de distinguer avec proba. faible, mais non-négligeable leur chiffrement en temps/espace raisonnable

- Alors $\mathscr{E}$ *n'est pas sémantiquement sûr*

Sécurité sémantique

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, M, C)$ un cryptosystème

- Adversaire *choisit* m0, m1

- et est capable de distinguer avec proba. faible, mais non-négligeable leur chiffrement en temps/espace raisonnable

- Alors $\mathscr{E}$ *n'est pas sémantiquement sûr*

- attaque à textes clairs *choisis*

- attaque probabiliste

- ce qu'exactement l'attaquant est capable de déterminer du clair n'est pas important

- attaquant dispose de ressources de calcul considérable (gouvernement/Google/etc. vs individu)

Chiffrement de flot

stream cipher

Masque jetable (Vernam)
Rappel

Masque jetable

$$M = C = K = \{0,1\}^n$$
$$\mathrm{Enc}(k, m) = c \text{ avec } c_i = k_i \oplus m_i$$
$$\mathrm{Dec}(k, c) = m \text{ avec } m_i = k_i \oplus c_i$$

inconditionnellement sûr mais :

- **clé k aussi longue le message**
- **clé choisie aléatoirement, de façon uniforme**
- **clé à usage unique**

Chiffrement de flot

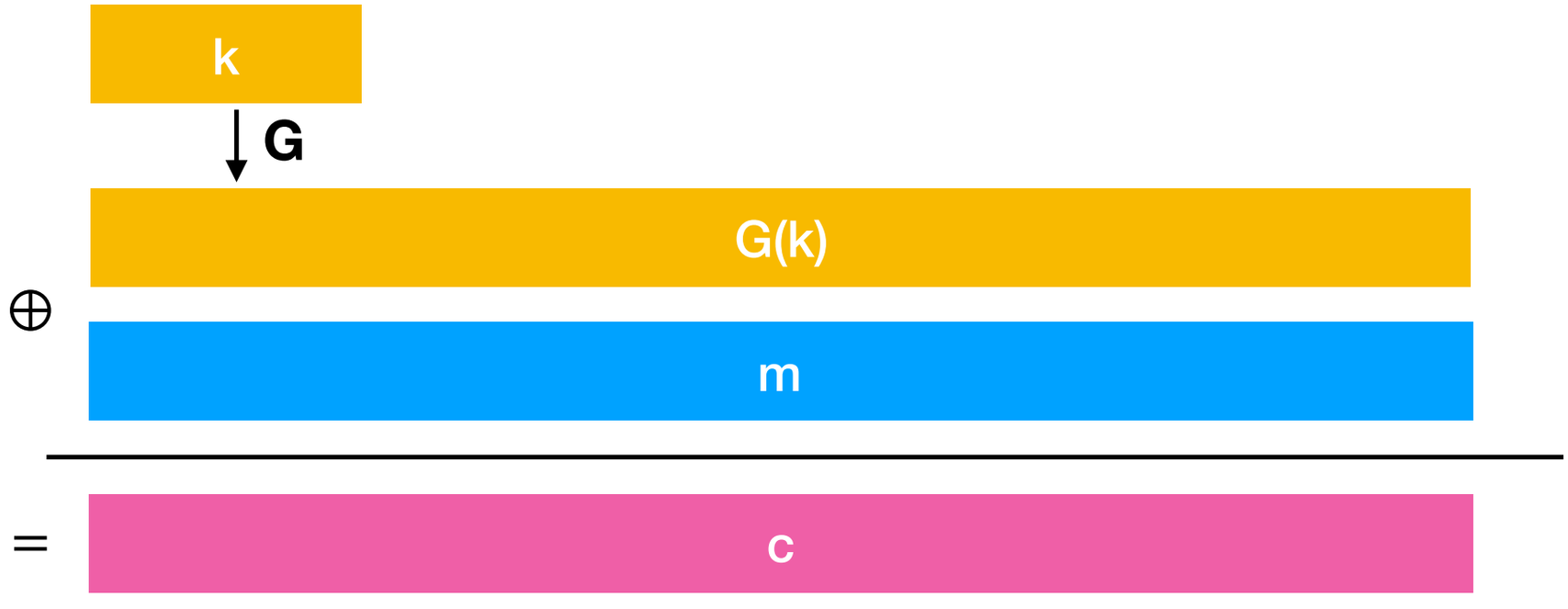
$G : \{0,1\}^n \rightarrow \{0,1\}^N, n \ll N$ **générateur pseudo aléatoire**

$M = C = \{0,1\}^N$

$K = \{0,n\}^n$

$\mathrm{Enc}(k, m) = G(k) \oplus m$

$\mathrm{Dec}(k, c) = G(k) \oplus c$

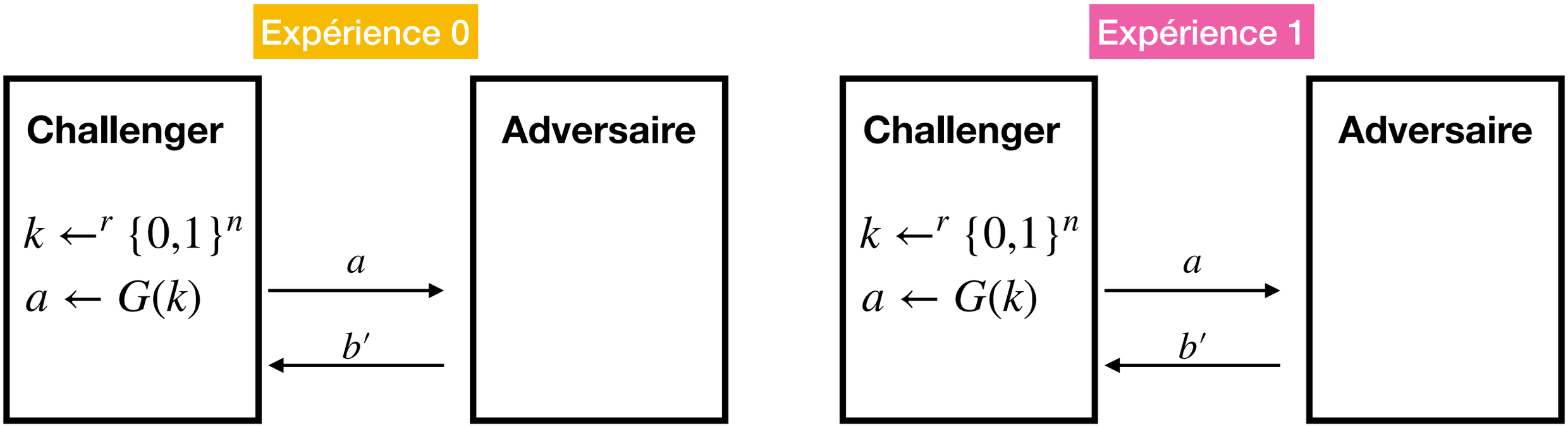


Générateur pseudo-aléatoire

$$G : \{0,1\}^n \rightarrow \{0,1\}^N, n < N$$

- **Il existe un algorithme efficace pour calculer G**
- **Calculatoirement difficile de distinguer la sortie de G d'une source aléatoire**

Distinguer pseudo-aléatoire/aléatoire

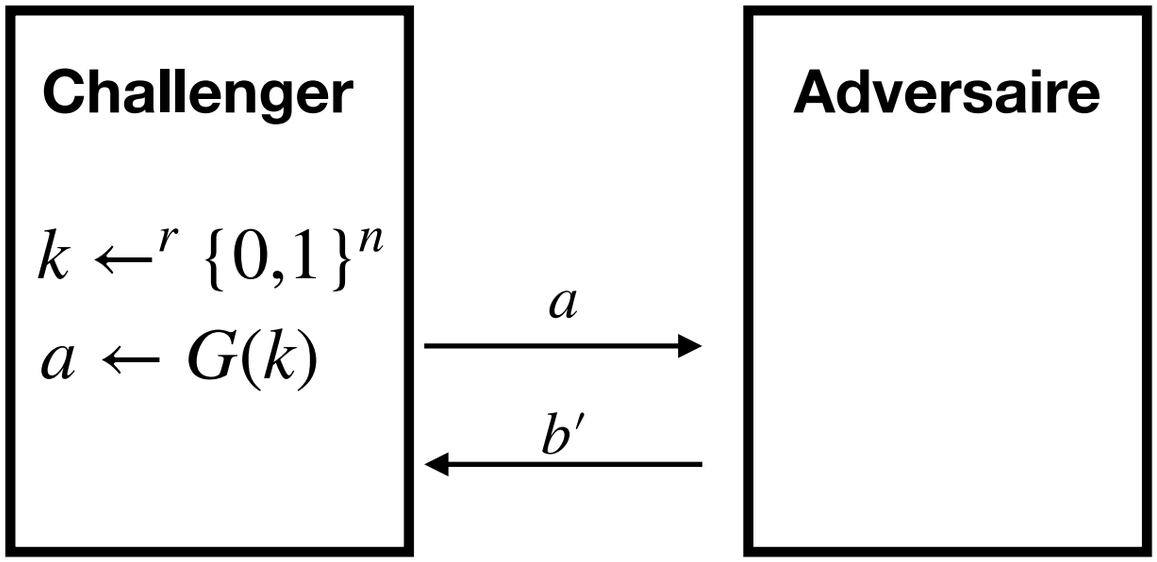


W_b **A** produit $b' = 1$ dans l'expérience b

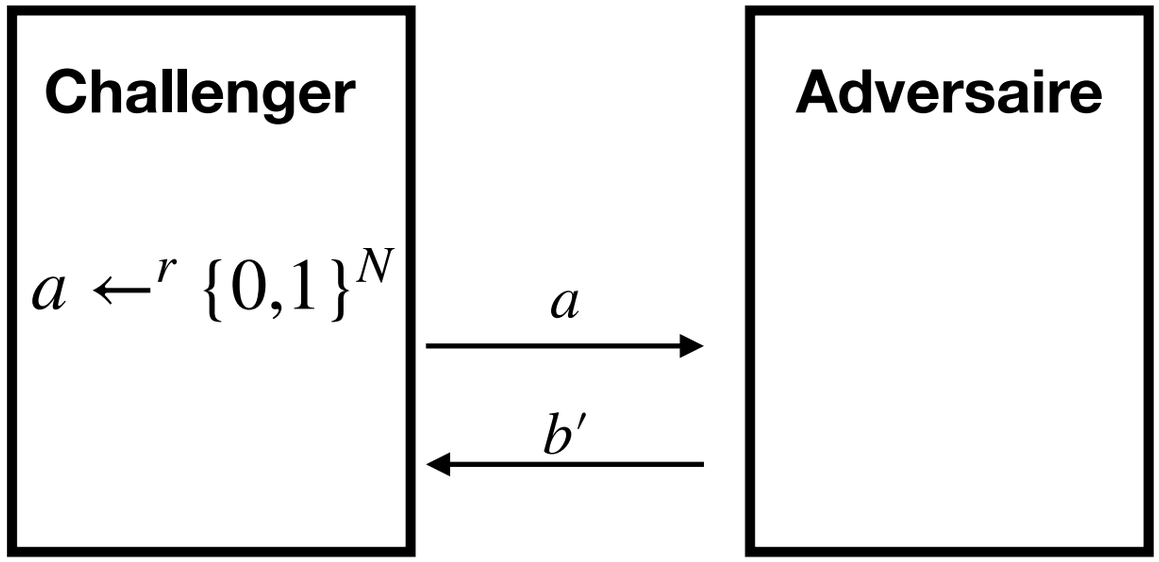
Avantage $Avtg(\mathbf{A}, G) = |\Pr[W_1] - \Pr[W_0]|$

Distinguer pseudo-aléatoire/aléatoire

Expérience 0



Expérience 1



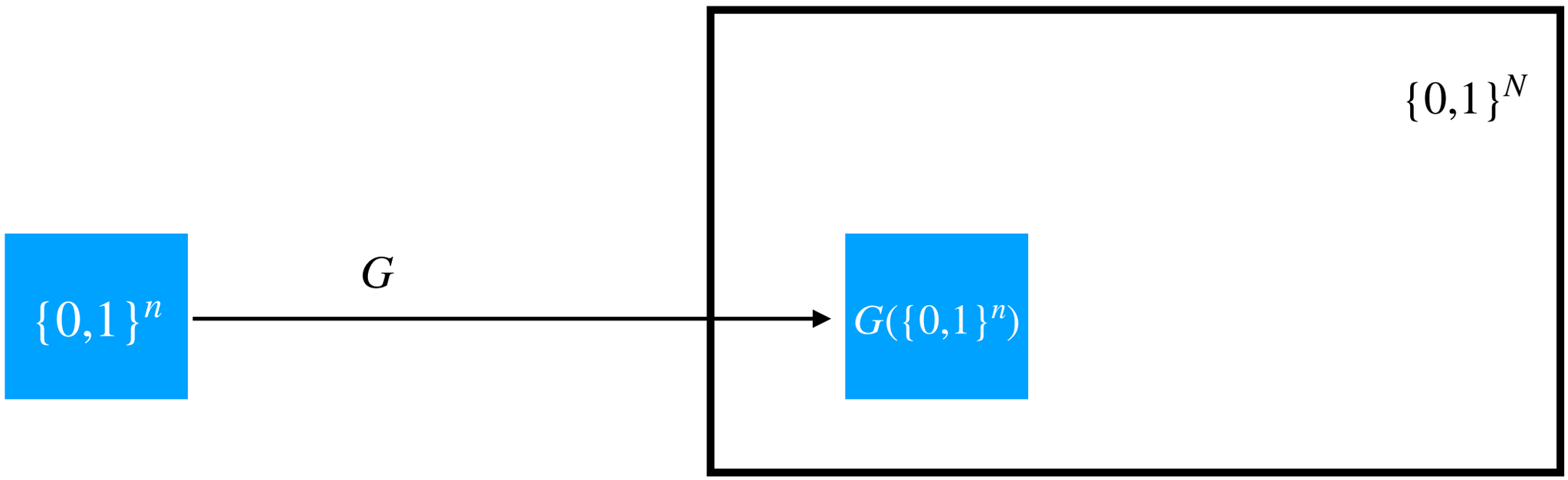
W_b **A** produit $b' = 1$ dans l'expérience b

Avantage $Avtg(\mathbf{A}, G) = |\Pr[W_1] - \Pr[W_0]|$

A gagne dans l'expérience 1

A perd dans l'expérience 0

Distinguer pseudo-aléatoire/aléatoire

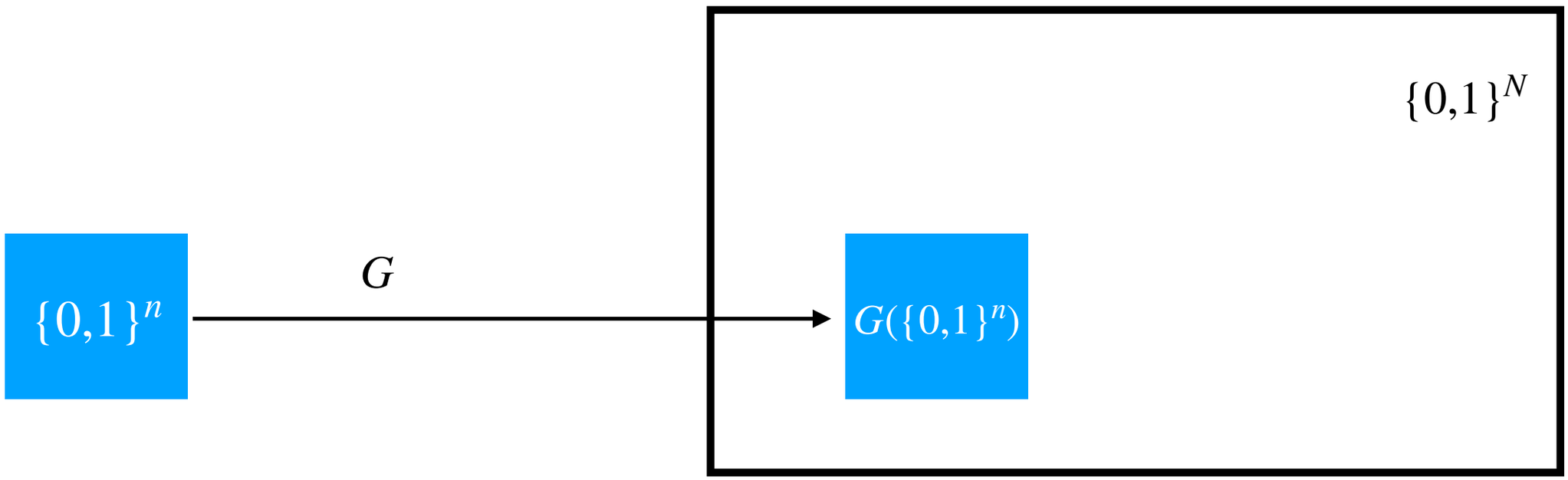


adv A$(a \in \{0,1\}^N) : \{0,1\}$

if $\exists x : G(x) = a$ **alors** retourner 0
sinon retourner 1

$$Avtg(\mathbf{A}, G) = |\Pr[W_1] - \Pr[W_0]|$$ **??? négligeable ? non-négligeable ?**

Distinguer pseudo-aléatoire/aléatoire

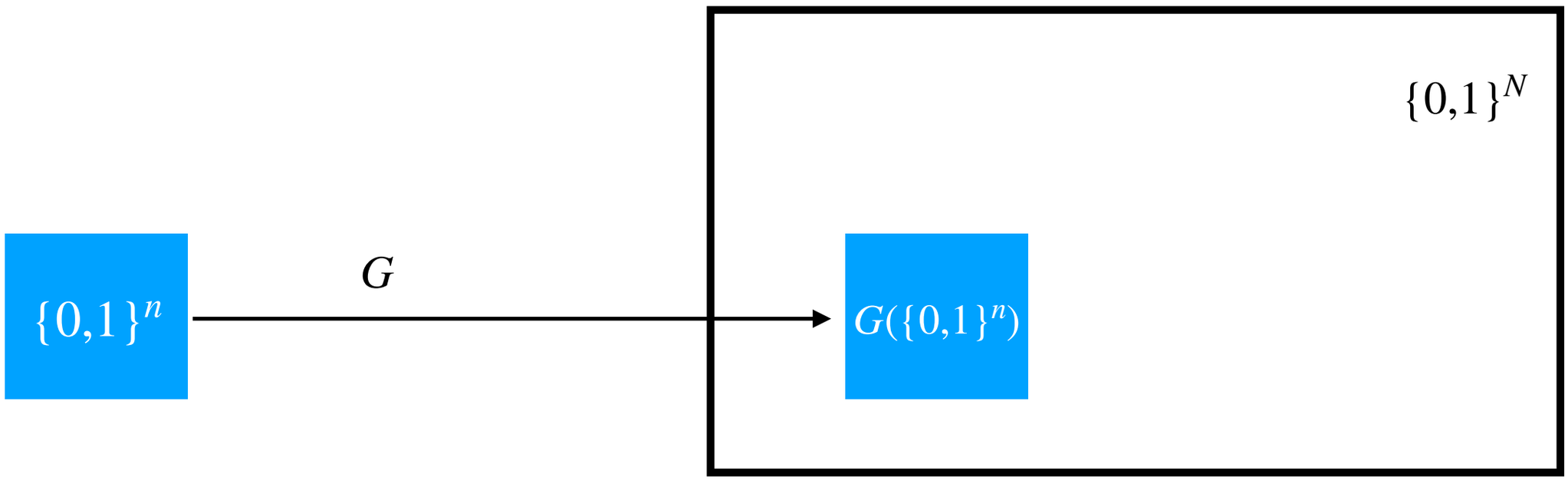


adv A$(a \in \{0,1\}^N) : \{0,1\}$

if $\exists x : G(x) = a$ **alors** retourner 0
sinon retourner 1

Exp. 0 : a pseudo-aléatoire $\Pr[W_0] = \Pr[\mathbf{b}' = 1] = 0$

Distinguer pseudo-aléatoire/aléatoire



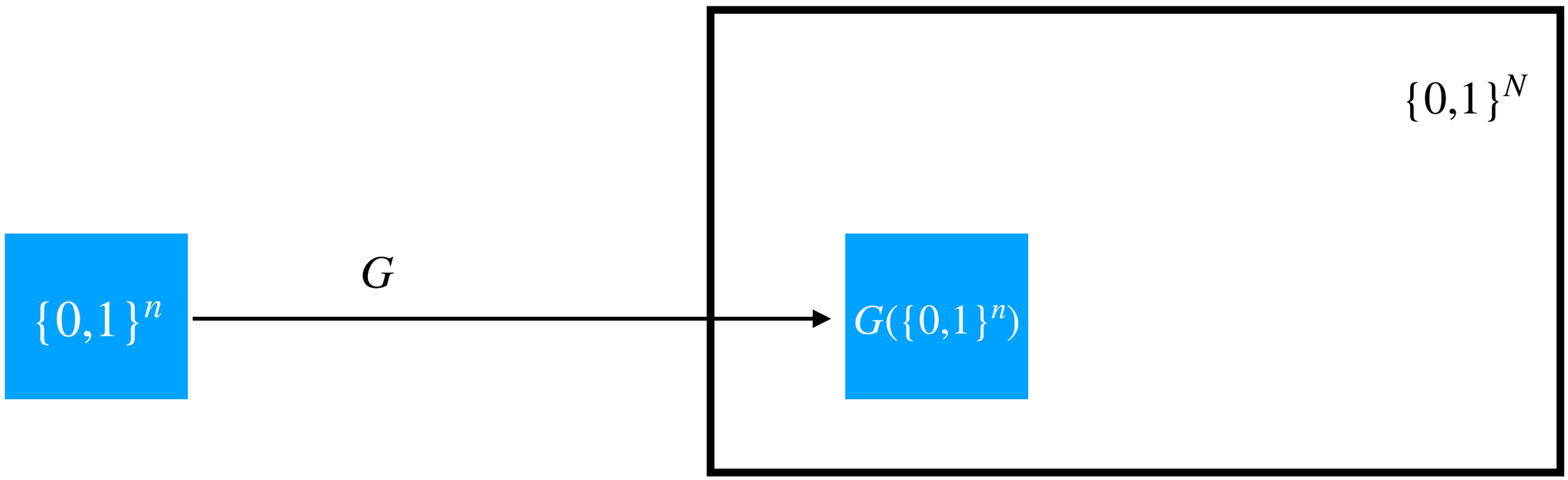
adv A$(a \in \{0,1\}^N) : \{0,1\}$

if $\exists x : G(x) = a$ **alors** retourner 0
sinon retourner 1

Exp. 0 : a pseudo-aléatoire $\quad \Pr[W_0] = \Pr[\mathbf{b}' = 1] = 0$

Exp. 1: a aléatoire $\quad \Pr[W_1] = \Pr[\mathbf{b}' = 1] = \Pr[a \notin G(\{0,1\}^n] = 1 - 2^n/2^N$

Distinguer pseudo-aléatoire/aléatoire



adv A$(a \in \{0,1\}^N) : \{0,1\}$

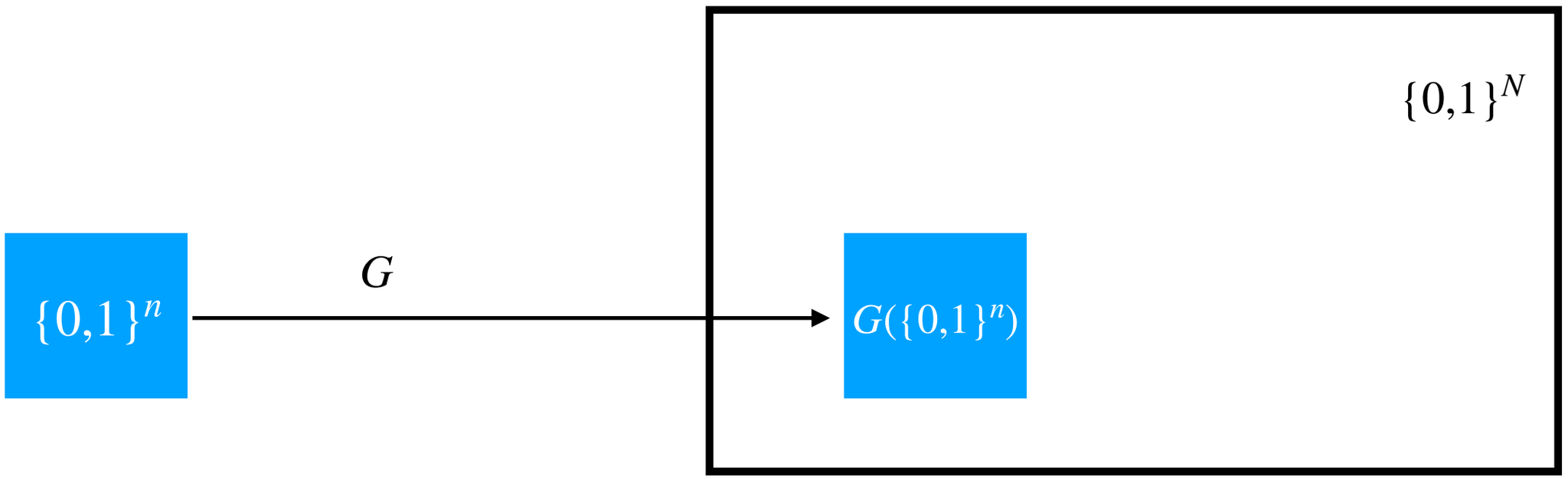
if $\exists x : G(x) = a$ **alors** retourner 0
sinon retourner 1

Exp. 0 : a pseudo-aléatoire $\quad \Pr[W_0] = \Pr[\mathbf{b}' = 1] = 0$

Exp. 1: a aléatoire $\quad \Pr[W_1] = \Pr[\mathbf{b}' = 1] = \Pr[a \notin G(\{0,1\}^n] = 1 - 2^n/2^N$

$$Avtg(\mathbf{A}, G) = |\Pr[W_1] - \Pr[W_0]| = 1 - 2^n/2^N$$

Distinguer pseudo-aléatoire/aléatoire



adv A$(a \in \{0,1\}^N) : \{0,1\}$

if $\exists x : G(x) = a$ **alors** retourner 0
sinon retourner 1

$O(2^n)$ non efficace

Exp. 0 : a pseudo-aléatoire $\Pr[W_0] = \Pr[\mathbf{b}' = 1] = 0$

Exp. 1: a aléatoire $\Pr[W_1] = \Pr[\mathbf{b}' = 1] = \Pr[a \notin G(\{0,1\}^n] = 1 - 2^n/2^N$

$Avtg(\mathbf{A}, G) = |\Pr[W_1] - \Pr[W_0]| = 1 - 2^n/2^N$ non négligeable

Générateur sûr

$G : \{0,1\}^n \to \{0,1\}^N, n < N$

définition

G est sûr si pour tout adversaire A efficace
$Avtge(\mathbf{A}, G) = \text{negl}$.

$\mathscr{E} = (\text{Enc}, \text{Dec}, K, M, C)$ avec $\text{Enc}(k, m) = G(k) \oplus m$ **et** $\text{Dec}(k, c) = G(k) \oplus c$

théorème

Si G est sûr alors $\mathscr{E}$ est sémantiquement sûr

Chiffrements de flot: limites

Soit $c_1 = G(k) \oplus m_1, c_2 = G(k) \oplus m_2$

$c_1 \oplus c_2 = (G(k) \oplus m_1) \oplus (G(k) \oplus m_2) = m_1 \oplus m_2$

ne dépend pas de la clé !

ne pas réutiliser le flot de bits pseudo-aléatoire !

Chiffrements de flot: limites

Soit $c_1 = G(k) \oplus m_1, c_2 = G(k) \oplus m_2$

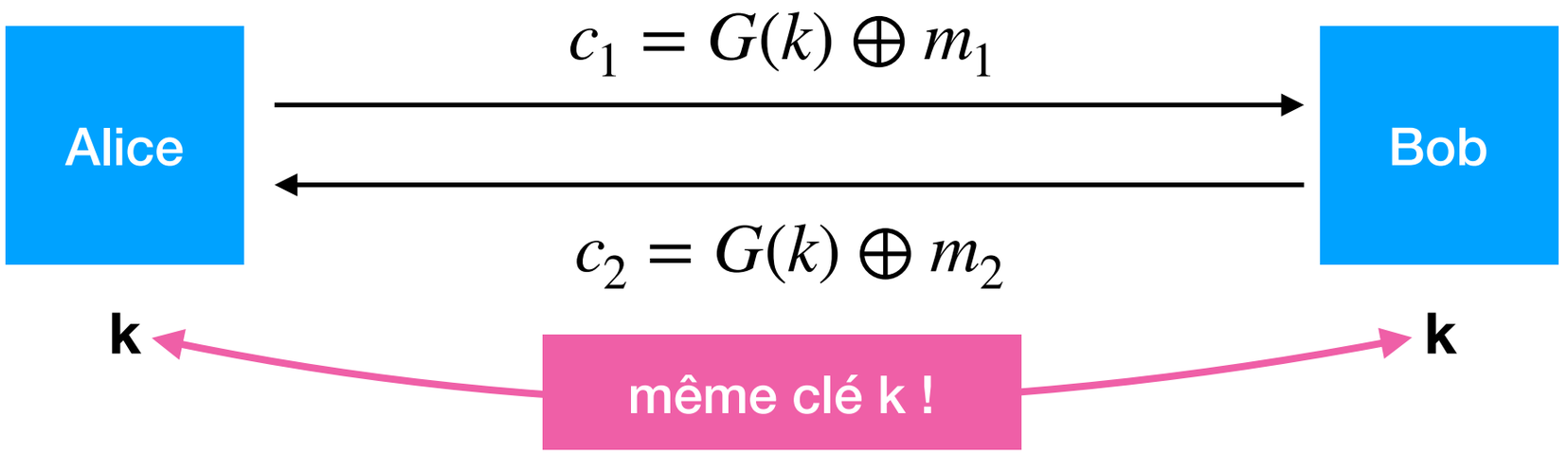
$c_1 \oplus c_2 = (G(k) \oplus m_1) \oplus (G(k) \oplus m_2) = m_1 \oplus m_2$

ne dépend pas de la clé !

ne pas réutiliser le flot de bits pseudo-aléatoire !

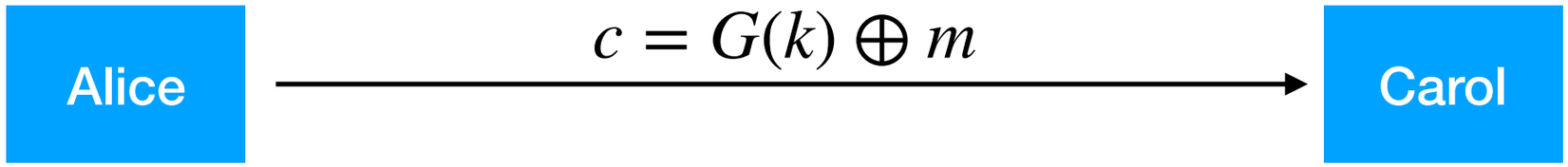
PPTP (point to point tunneling protocol)

implémentation dans windows NT



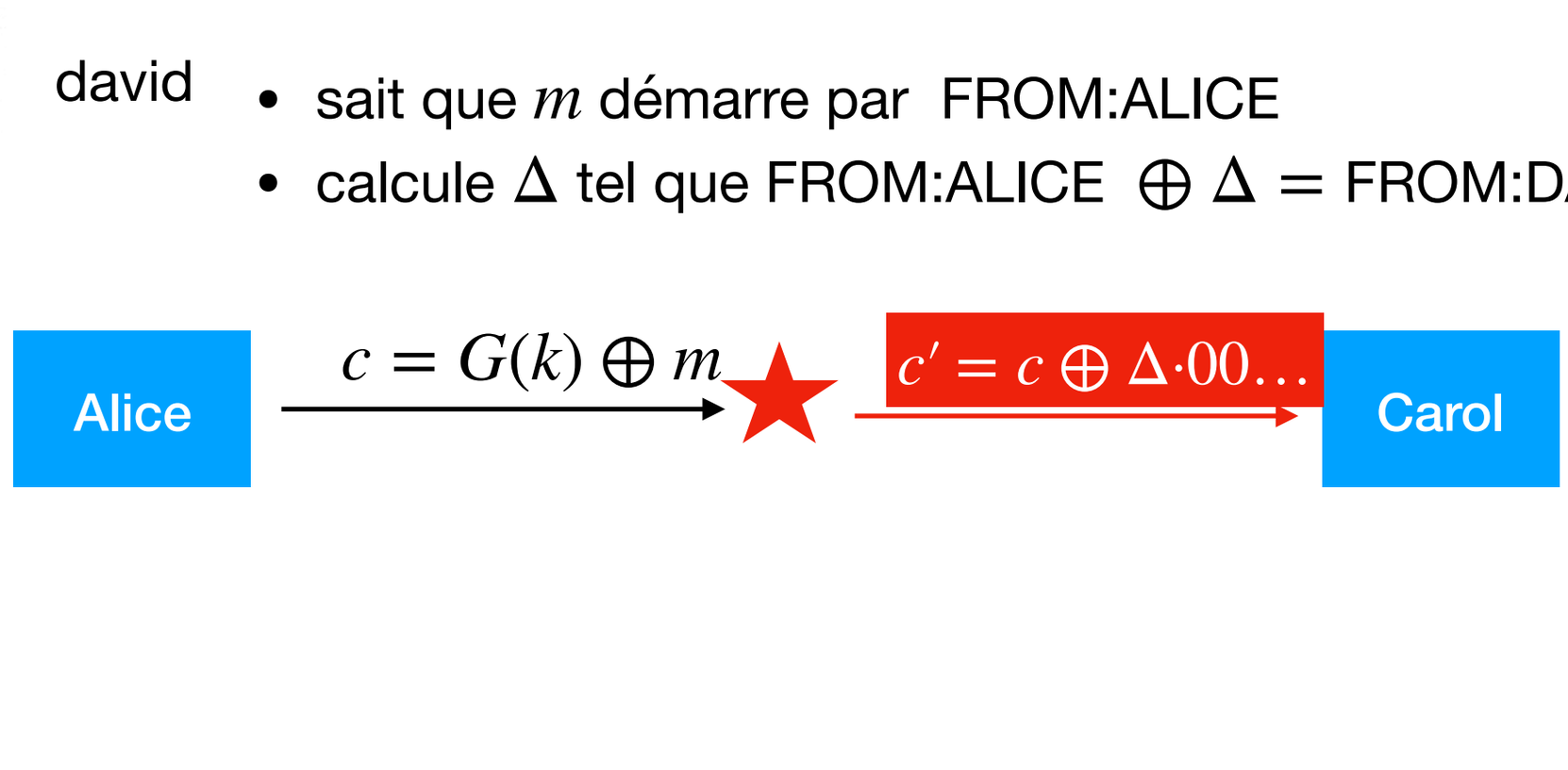
Chiffrements de flot: malléabilité

$m =$ F R O M : A L i C E _ D M 1 _ I F 2 0 2 ...



Chiffrements de flot: malléabilité

$m =$ F|R|O|M|:|A|L|i|C|E|_|D|M|1|_|I|F|2|0|2|...

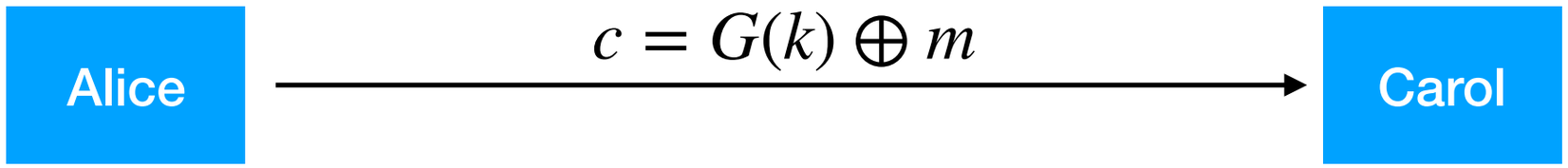


david

- sait que m démarre par FROM:ALICE
- calcule Δ tel que FROM:ALICE $\oplus\ \Delta =$ FROM:DAVID

Chiffrements de flot: malléabilité

$m =$ F|R|O|M|:|A|L|i|C|E|_|D|M|1|_|I|F|2|0|2|...



david

- sait que m commence par FROM:ALICE
- calcule Δ tel que FROM:ALICE $\oplus\ \Delta =$ FROM:DAVID

$$\begin{aligned}\mathrm{Dec}(k, c') &= G(k) \oplus c' \\ &= G(k) \oplus \Delta \cdot 00\ldots \oplus c \\ &= m \oplus \Delta \cdot 00\ldots\end{aligned}$$

$=$ F|R|O|M|:|D|A|V|I|D|_|D|M|1|_|I|F|2|0|2|...

GLIBC random()

init(s): r[1..34]

$r[0] \leftarrow s$
pour $i : 1 \leq i \leq 30$
$\quad r[i] \leftarrow a \times r[i-1] \mod 2^{32} - 1$
pour $i : 31 \leq i \leq 33$
$\quad r[i] \leftarrow r[i-31]$
retourner r

rand() i-ème appel

retourner $o[i] = r[i+344] \gg 1$ **où**
$r[i] = r[i-3] + r[i-31] \mod 2^{32}$

GLIBC random()

init(s): r[1..34]

$r[0] \leftarrow s$
pour $i : 1 \leq i \leq 30$
$\quad r[i] \leftarrow a \times r[i-1] \mod 2^{32} - 1$
pour $i : 31 \leq i \leq 33$
$\quad r[i] \leftarrow r[i-31]$
retourner r

sûr ?

rand() i-ème appel

retourner $o[i] = r[i+344] \gg 1$ **où**
$r[i] = r[i-3] + r[i-31] \mod 2^{32}$

GLIBC random()

init(s): r[1..34]

$r[0] \leftarrow s$
pour $i : 1 \leq i \leq 30$
$r[i] \leftarrow a \times r[i-1] \mod 2^{32} - 1$
pour $i : 31 \leq i \leq 33$
$r[i] \leftarrow r[i-31]$
retourner r

Non sûr

rand() i-ème appel

retourner $o[i] = r[i+344] \gg 1$ **où**
$r[i] = r[i-3] + r[i-31] \mod 2^{32}$

sûr ?

prévisible :

$$o[i] = o[i-31] + o[i-3] \mod 2^{31}$$

ou

$$o[i] = o[i-31] + o[i-3] + 1 \mod 2^{31}$$

RC4 (1987)

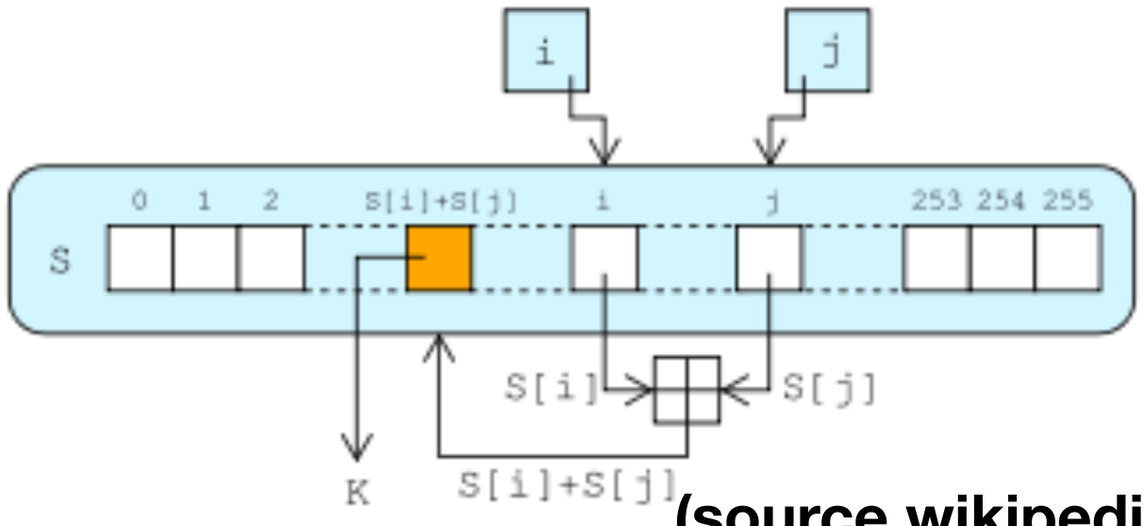
Etat S : 256 octets
« pointeurs » i,j : 0 <= i,j <= 255

init(s) |s| 40-128 bits

```
for i from 0 to 255 : S[i] = i
j := 0
for i from 0 to 255
     j := (j + S[i] + key[i mod |s|]) mod 256
     swap values of S[i] and S[j]
```

Génération

```
i := 0; j:= 0
repeat
 i := (i + 1) mod 256
 j := (j + S[i]) mod 256
 swap values of S[i] and S[j]
 output S[(S[i] + S[j]) mod 256]
```



(source wikipedia)

RC4

Utilisation : TLS, WEP

Faiblesses

Biais

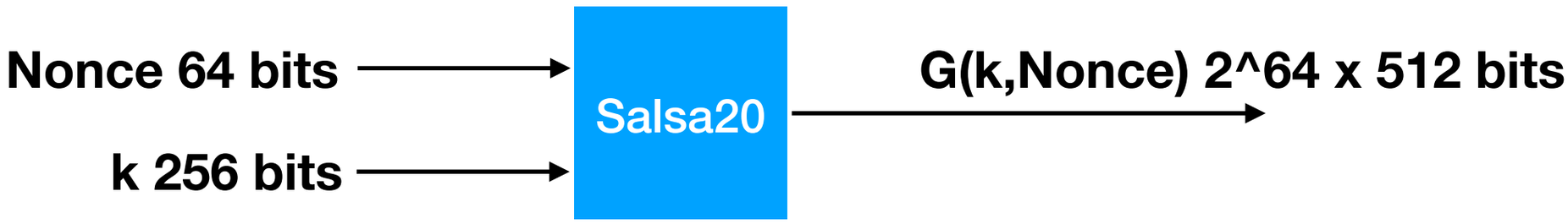
- $\Pr[(o_i, o_{i+1})] = (0,0) \geq 1/256^2 + 1/256^3$
- $\Pr[(o_i, o_{i+1})] = (0,1) \geq 1/256^2 + 1/256^3$

Attaque clés proches

- **Similarités entre RC4(s) et RC4(s') quand s et s' diffèrent peu**
- **Attaque du protocole WEP**

Salsa20 (2005)

- **issu d'un processus ouvert de standardisation (eSTream)**



G(k) /* Nonce = 0 ici */

```
for j from 0 to L-1 : S[i] = i
    h[j] := pad(k,j,0) /* 512 bits */
    r[j] := pi(h[j]) XOR h[j]
    output (r[0],…,r[L-1])
```

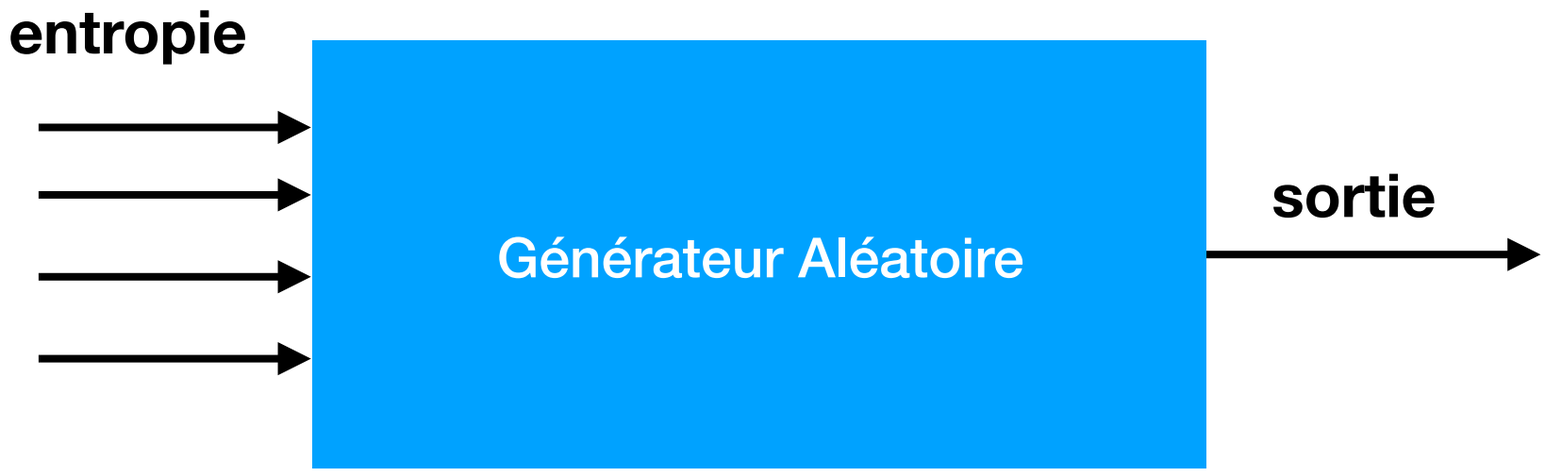
Considéré sûr en pratique

Générer des bits (vraiment) aléatoires

- Il est parfois nécessaire de générer des bits (vraiment) aléatoires : clé, tirage aléatoire dans certains alg. de chiffrements, etc.

- Comment en pratique cela se fait-il ?
  - `/dev/random`
  - hardware random generator `RDRAND` (Intel)

Générer des bits (vraiment) aléatoires



Sources d'entropie

- **clavier**
- **souris**
- **interruptions matérielles**

Chiffrement par bloc

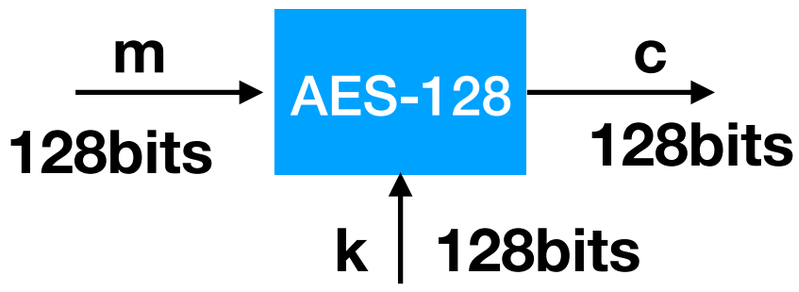
block cipher

Chiffrement par bloc

- (Enc,Dec,K,M,C) chiffrement par bloc
  - M= C = {0,1}^b, K={0,1}^k
  - b : taille de *bloc*

- Exemples

	k (bits)	b (bits)
DES	56	64
AES-128	128	128
AES-256	256	128



Permutation pseudo-aléatoire

- Soit $k \in K$. $E_k : \{0,1\}^b \to \{0,1\}^b$

$$E_k : x \to \mathrm{Enc}(k, x)$$

E_k **est une permutation de** $\{0,1\}^b$

$$\mathrm{Dec}(k, c) = E_k^{-1}(c)$$

<u>Objectif</u>: concevoir Enc **(et donc)** Dec **tel que** $\mathrm{E_k}$ **soit une permutation pseudo -aléatoire**

- $f : \{0,1\}^b \to \{0,1\}^b$ permutation pseudo-aléatoire si calculatoirement difficile de distinguer f d'une permutation vraiment aléatoire

Distinguer permutation pseudo-aléatoire/vraiment aléatoire

permutation

$E_k : \{0,1\}^b \rightarrow \{0,1\}^b, k \in K$

permutation

$f : \{0,1\}^b \rightarrow \{0,1\}^b$

tirée aléatoirement

Distinguer permutation pseudo-aléatoire/vraiment aléatoire

permutation

$E_k : \{0,1\}^b \rightarrow \{0,1\}^b, k \in K$

permutation

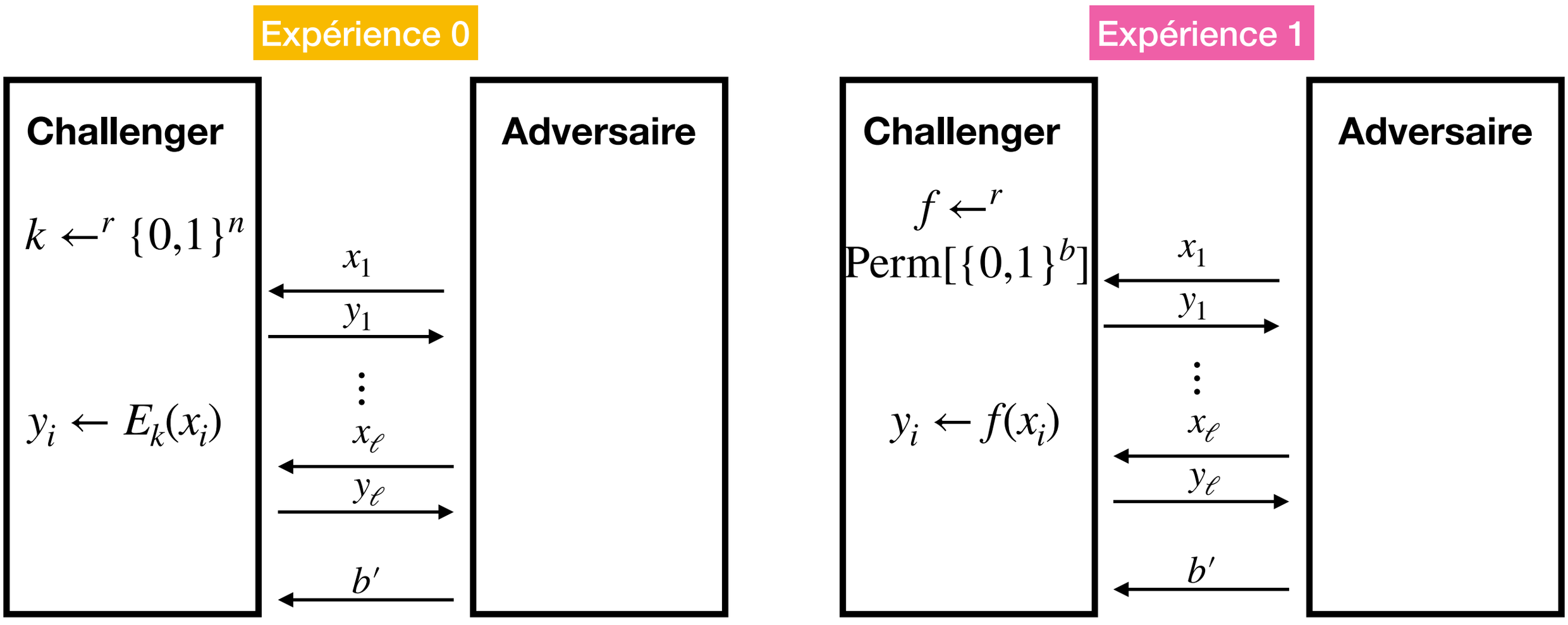
$f : \{0,1\}^b \rightarrow \{0,1\}^b$

tirée aléatoirement

Exemple AES-128

- $K = \{0,1\}^{128}$ $2^{128} \simeq 3{,}4.10^{38}$ **permutation** E_k **possibles**
- $b = 128$ $(2^{128})! \simeq 10^{10^{40{,}11}}$ **permutations**

Distinguer permutation pseudo-aléatoire/aléatoire



W_b **A** produit $b' = 1$ dans l'expérience b

Avantage $Avtg(\mathbf{A}, \mathscr{E}) = |\Pr[W_1] - \Pr[W_0]|$

avec $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$

Chiffrement par bloc sûr

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

définition

$\mathscr{E}$ est <u>sûr</u> ssi pour tout adversaire A efficace, $Avtg(\mathbf{A}, \mathscr{E}) = \mathrm{negl}\,.$

Conséquence : recouvrement de clé difficile

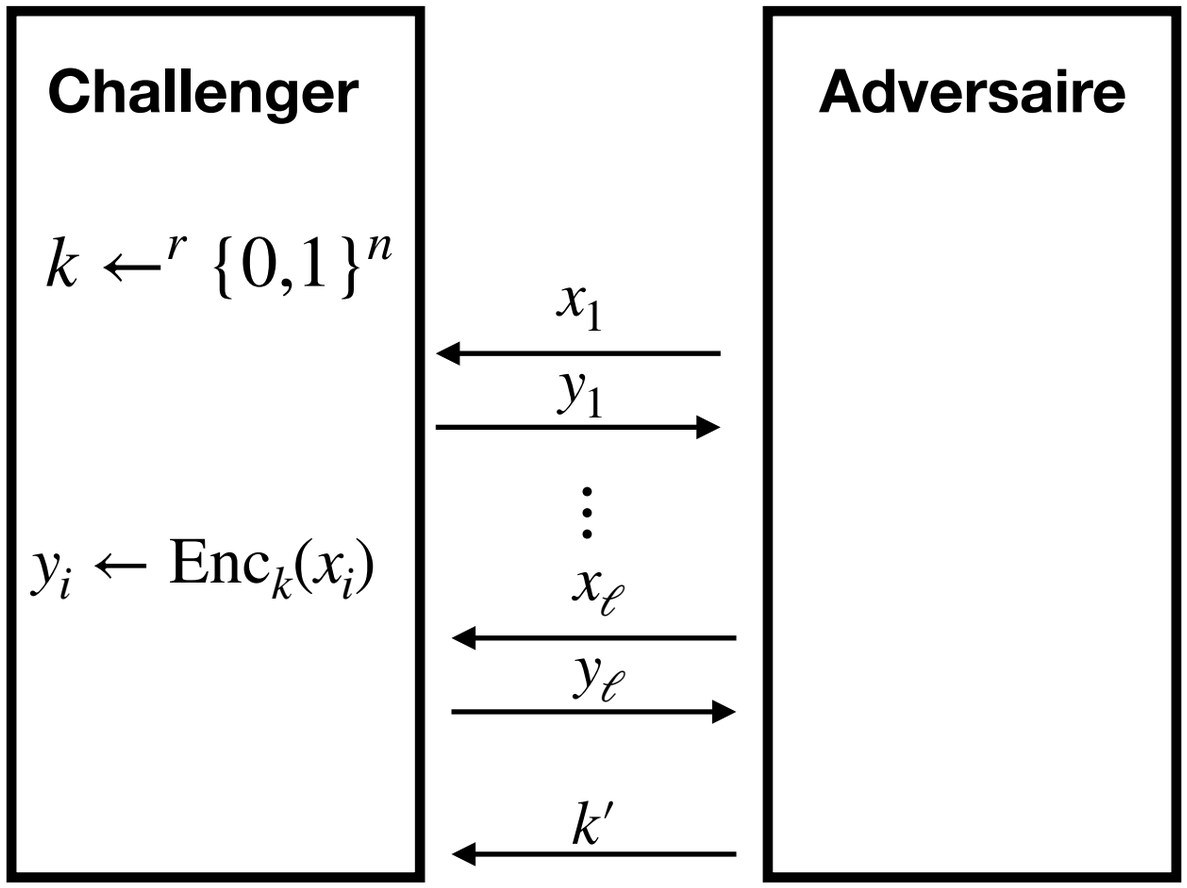
Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

Théorème

Si $\mathscr{E}$ est <u>sûr</u> alors il n'existe pas d'attaque efficace pour recouvrer la clé

La taille n des clés doit donc être suffisamment grande pour qu'une attaque par recherche exhaustive soit infaisable en pratique

Recouvrement de clé : modélisation



W **A** produit $k' = k$

Probabilité succès $\Pr[W]$

Recouvrement de clé => non sûr

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

- Soit **A** un adversaire qui recouvre la clé avec proba. *p*
- Jeu distinction perm. pseudo aléatoire/aléatoire : Adversaire **B**
  - simule **A** pour obtenir une clé k'
  - soit $\{x_1, \ldots, x_q\} = X$ les requêtes effectuées par **A**. Soumet $x_{q+1} \notin X$ au challenger et obtient y_{k+1}
  - si $y_{q+1} = \mathrm{Enc}(k', x_{q+1})$ produit $b' = 0$ (non aléatoire) et $b' = 1$ (aléatoire) sinon

Recouvrement de clé => non sûr

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

- Soit **A** un adversaire qui recouvre la clé avec proba. *p*
- Jeu distinction perm. pseudo aléatoire/aléatoire : Adversaire **B**
  - simule **A** pour obtenir une clé k'
  - soit $\{x_1, \ldots, x_q\} = X$ les requêtes effectuées par **A**. Soumet $x_{q+1} \notin X$ au challenger et obtient y_{k+1}
  - si $y_{q+1} = \mathrm{Enc}(k', x_{q+1})$ produit $b' = 0$ (non aléatoire) et $b' = 1$ (aléatoire) sinon

Avantage $Avtg(\mathbf{B}, \mathscr{E}) \geq p - \mathrm{negl}\,.$

Sécurité sémantique

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

- **Sécurité sémantique** : Adv. capable de distinguer Enc(k,m0) et Enc(k,m1) avec proba. non negl.
- **Sûr** : Adv. capable de distinguer interaction avec perm. pseudo aléatoire Enc(k,.) et perm. aléatoire f avec proba. non negl.

théorème

si $\mathscr{E}$ est sûr alors $\mathscr{E}$ est sémantiquement sûr

démo. (idée)

- contraposée : non sémantiquement sûr => non sûr
- soit A adv. jeu sécurité sémantique
- adv. B (interagit avec Enc(k,.) ou f) : simule A.
- Si A gagne, B produit « non aléatoire », « aléatoire » sinon

Sécurité sémantique

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

- **Sécurité sémantique** : Adv. capable de distinguer Enc(k,m0) et Enc(k,m1) avec proba. non negl.
- **Sûr** : Adv. capable de distinguer interaction avec perm. pseudo aléatoire Enc(k,.) et perm. aléatoire f avec proba. non negl.

théorème

si $\mathscr{E}$ est sûr alors $\mathscr{E}$ est sémantiquement sûr

limite

taille des messages = taille de bloc b

démo. (idée)

- contraposée : non sémantiquement sûr => non sûr
- soit A adv. jeu sécurité sémantique
- adv. B (interagit avec Enc(k,.) ou f) : simule A.
- Si A gagne, B produit « non aléatoire », « aléatoire » sinon

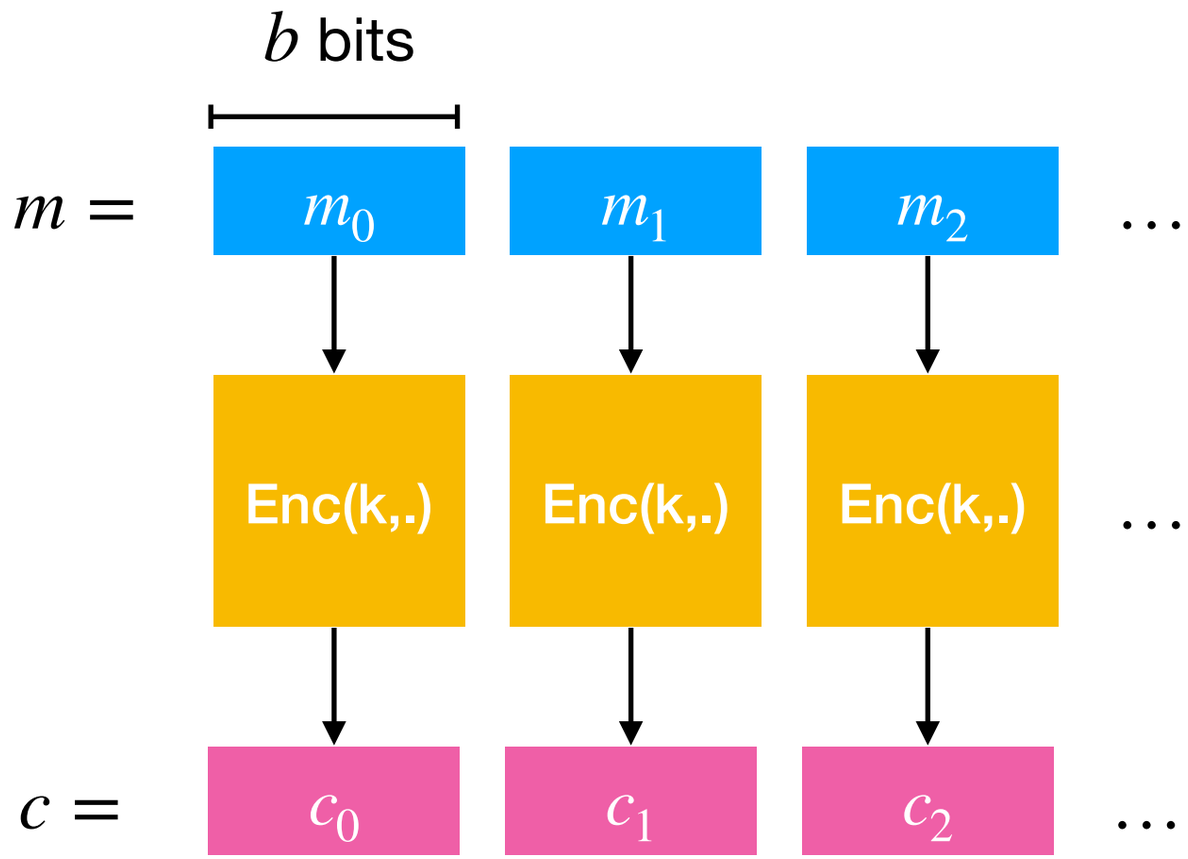
Chiffrement de messages de taille arbitraire

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

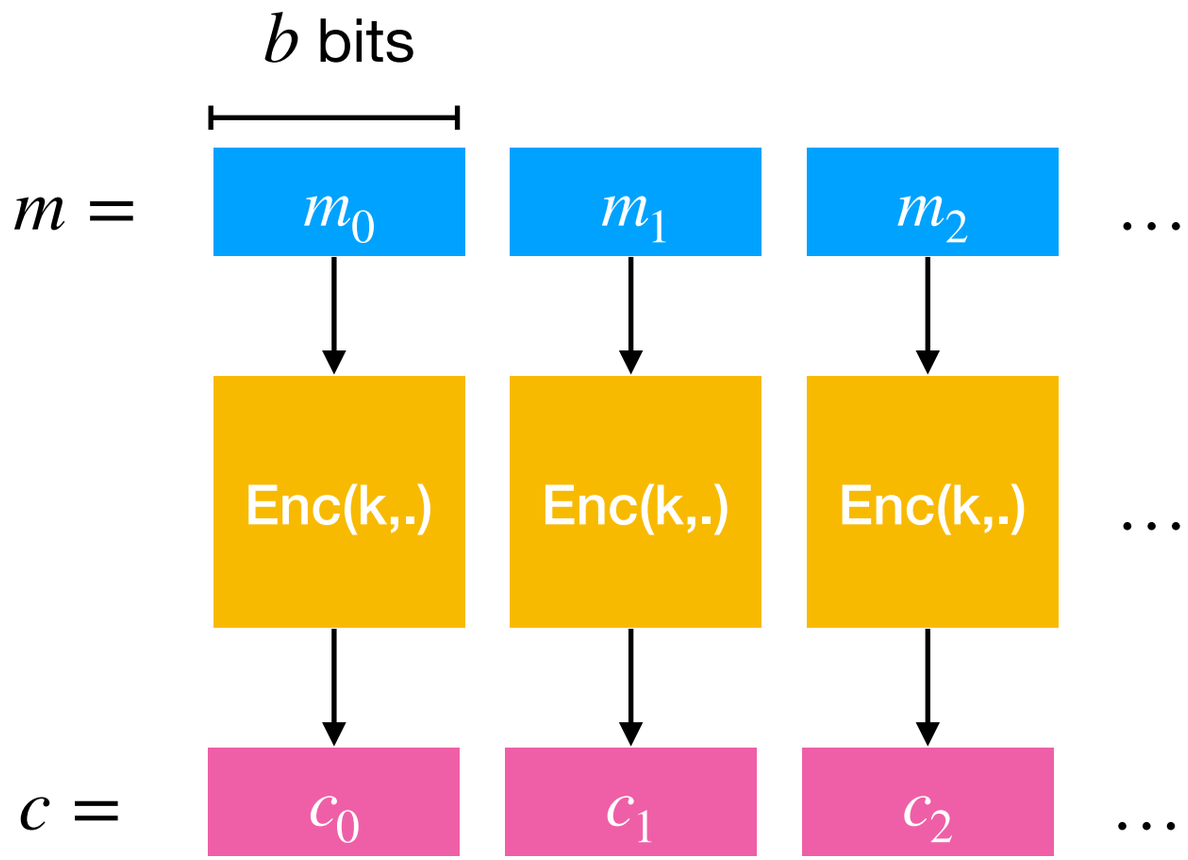
typiquement , b = qqs centaines de bits

Pb: chiffrement de messages de taille $\ell > b$

Electronic Code Book



Electronic Code Book



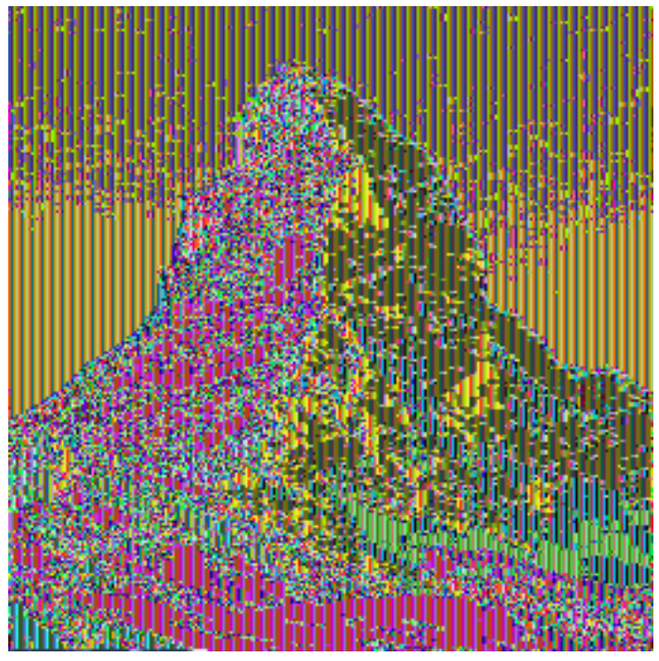
Sécurité ?

non sémantiquement sûr

Electronic Code Book

$m =$

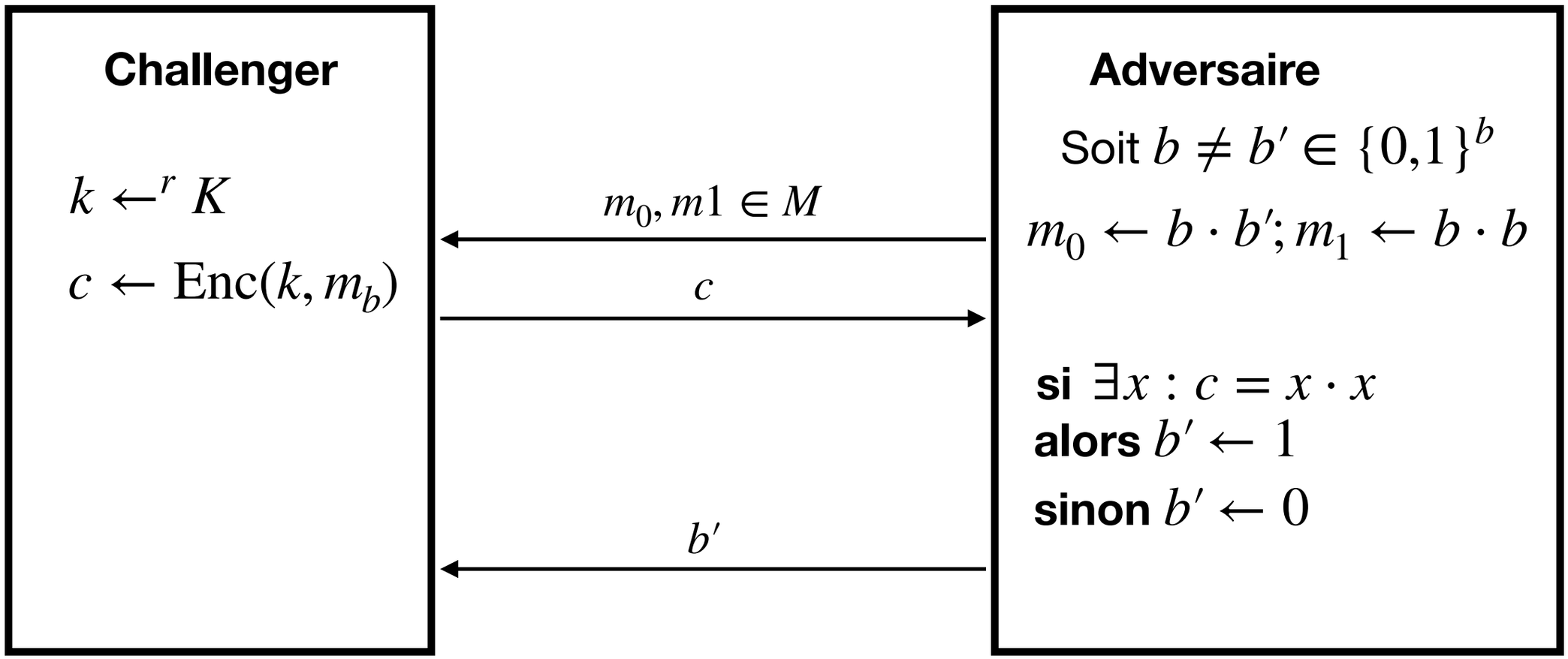
$c =$



non sémantiquement sûr

source : wikipédia

ECB : (In)sécurité sémantique



W_b **A** produit $b' = 1$ dans l'expérience b

Avantage $Avtg(\mathbf{A}, \mathscr{E}) = |\Pr[W_1] - \Pr[W_0]| = 1$

Chiffrement de messages de taille arbitraire

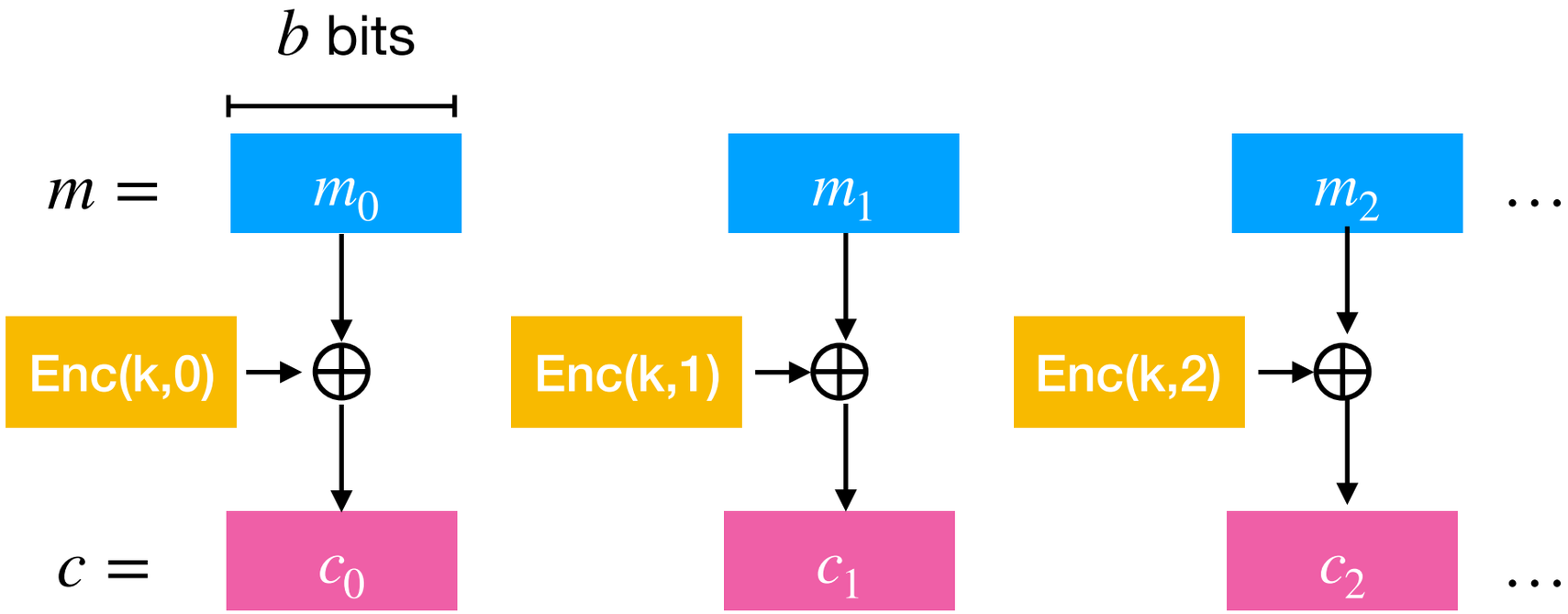
Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$ un chiffrement par bloc

typiquement , b = qqs centaines de bits

Pb: chiffrement de messages de taille $\ell > b$

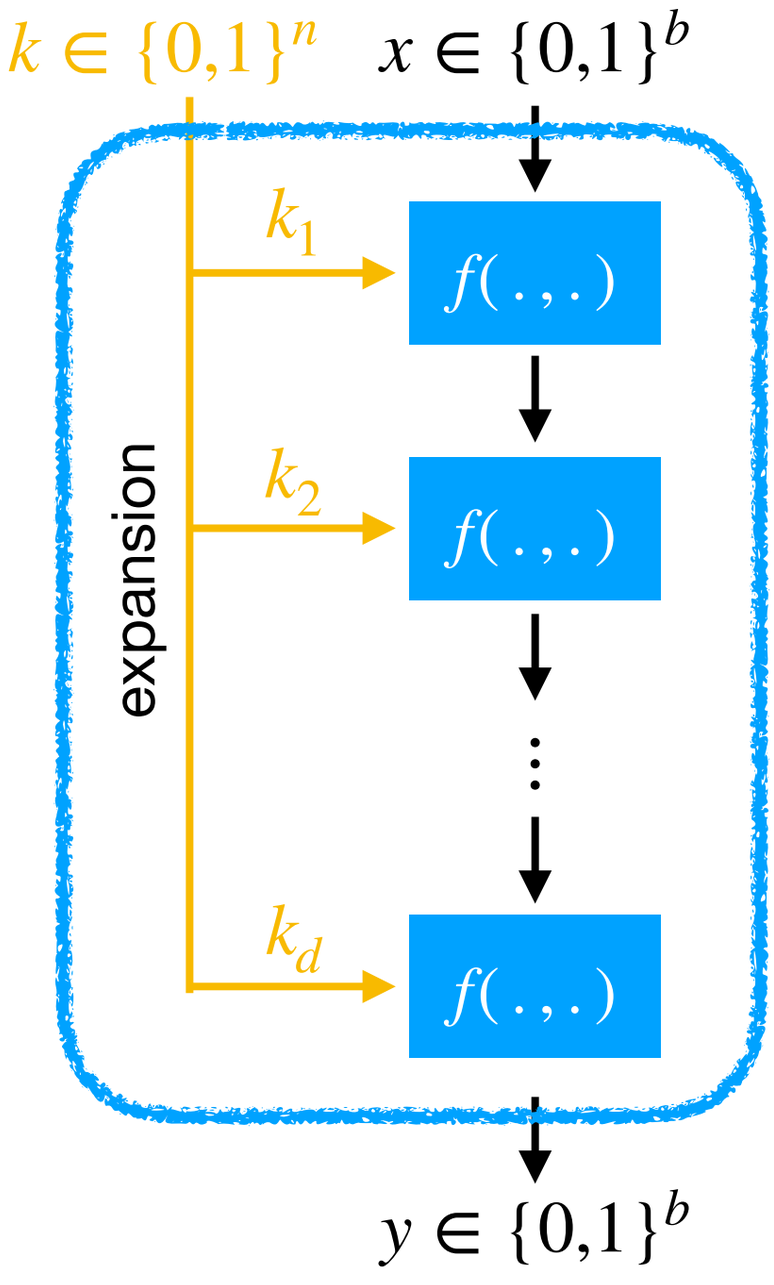
- mode **ECB** non sémantiquement sûr
- d'autres modes existent : **CBC** Cipher Block Chaining**, CFB** cipher feedback**, OFB** output feedback**, CTR** CouTeR mode**,** etc.
- **cf. TD**

Counter mode



Chiffrements par bloc en pratique

Patron de conception



$\mathrm{Enc}(k, x)$

- **d itérations (aka rondes)**
- k_i **clé de la ronde** i
- $y = f(k_d, f(k_{d-1}, \ldots, f(k_2, f(k_1, x)), \ldots))$

DES/AES

	clé (en bits)	bloc (en bits)	rondes
DES	56	64	16
3DES	168	64	48
AES-128	128	128	10
AES-256	256	128	14

- DES (Data Encryption Standard) 1977
- 3DES (triple DES) : 3 applications de DES avec 3 clés
- AES (Advanced Encryption Standard) 2001

DES

	clé (en bits)	bloc (en bits)	rondes
DES	56	64	16

- Proposé par IBM en 1975. 1ere Version : b = n = 128bits
- Adopté comme standard US en 1977
- **Non sûr aujourd'hui** : taille des clés (56bits) trop petite. Attaque par recherche exhaustive faisable
- 3DES (3 applications de DES, clés 3x56 = 168bits) standard US approuvé jusqu'en 2030

Réseau de Feistel

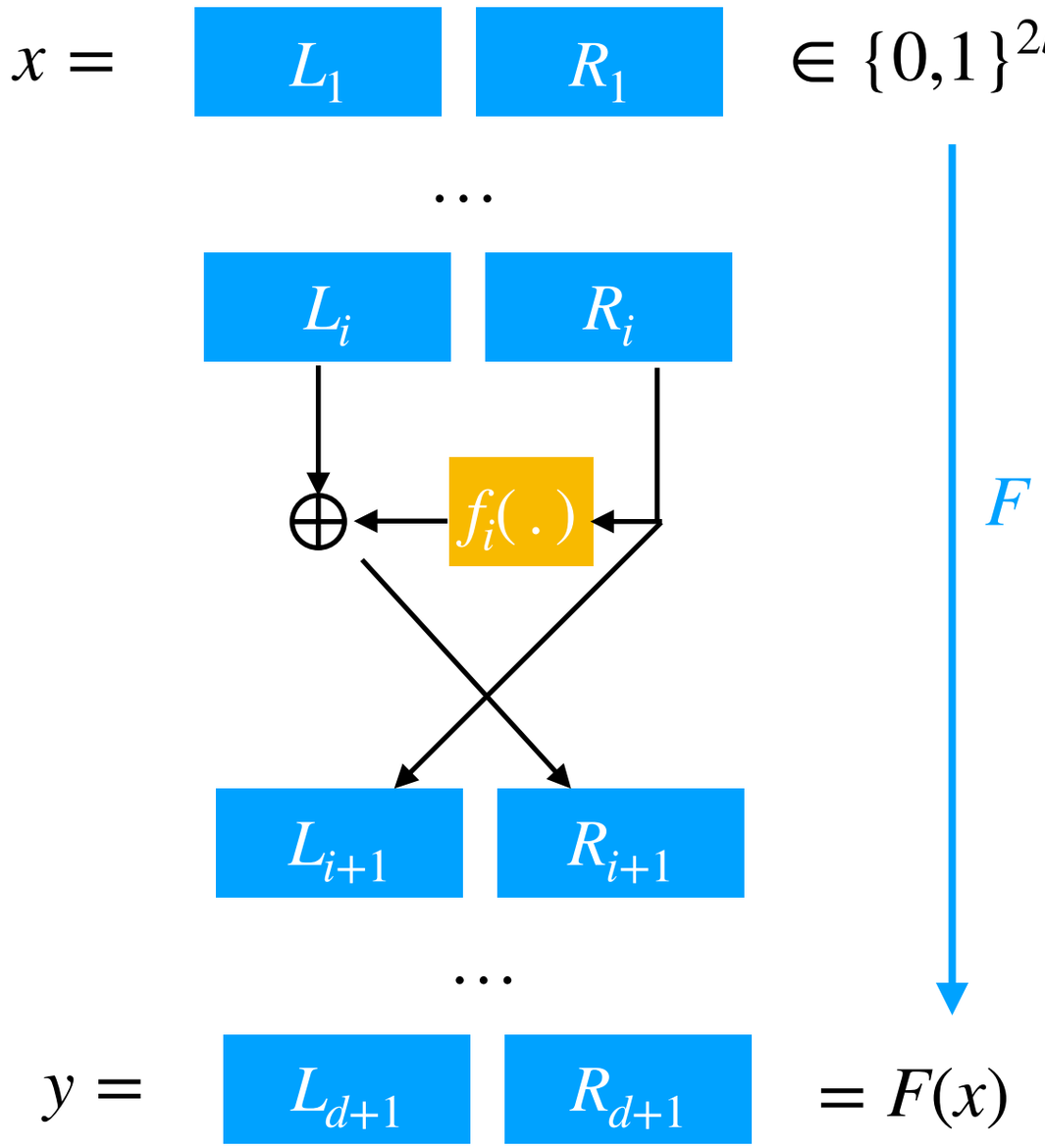
Soit $f_1, \ldots, f_d : \{0,1\}^b \to \{0,1\}^b$ des fonctions arbitraires

but: construire $F : \{0,1\}^{2b} \to \{0,1\}^{2b}$ **permutation**

Réseau de Feistel

Soit $f_1, \ldots, f_d : \{0,1\}^b \to \{0,1\}^b$ des fonctions arbitraires

but: construire $F : \{0,1\}^{2b} \to \{0,1\}^{2b}$ **<u>permutation</u>**



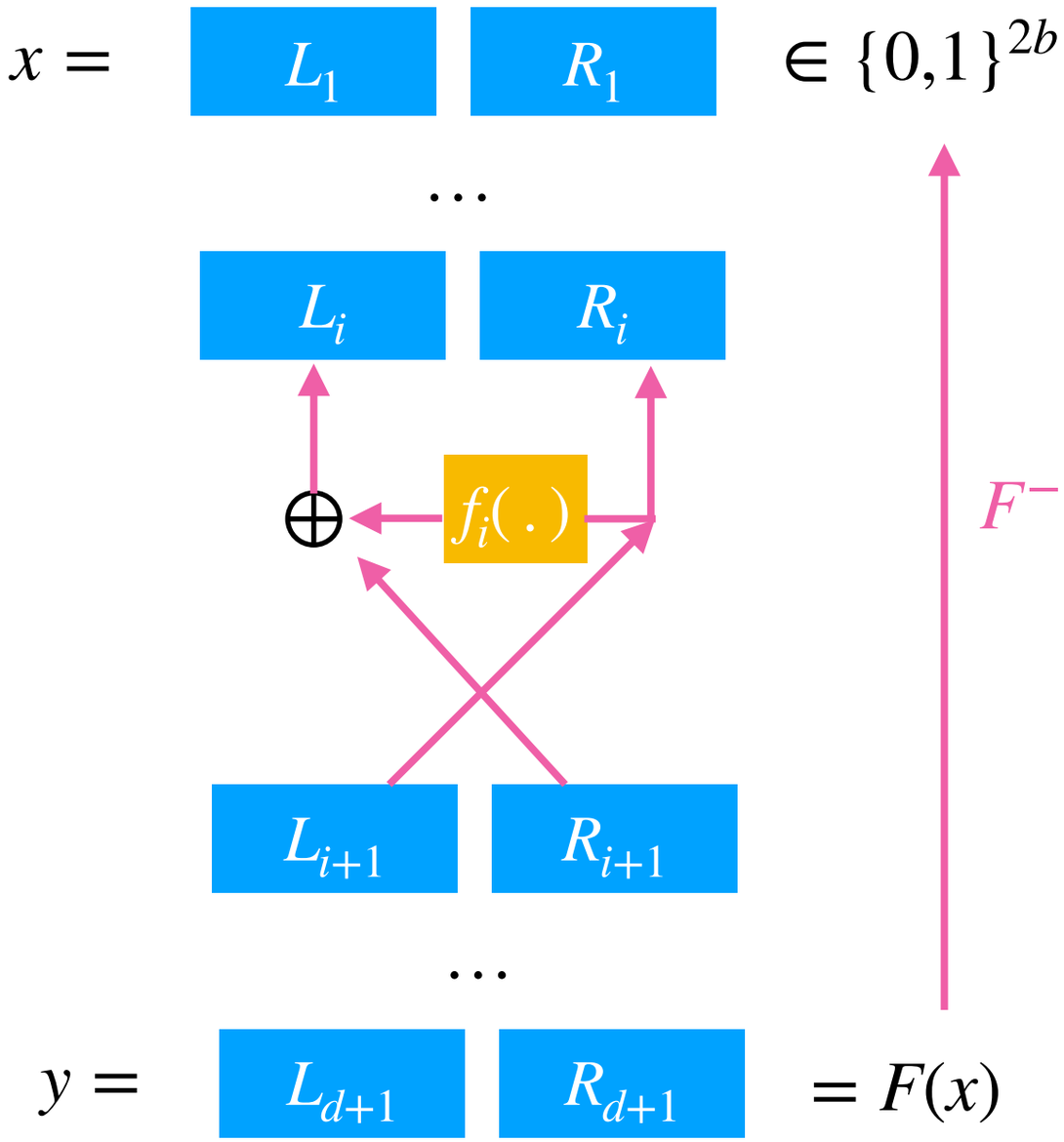
$$R_{i+1} = L_i \oplus f(R_i)$$

$$L_{i+1} = R_i$$

Réseau de Feistel

Soit $f_1, \ldots, f_d : \{0,1\}^b \to \{0,1\}^b$ des fonctions arbitraires

but: construire $F : \{0,1\}^{2b} \to \{0,1\}^{2b}$ **<u>permutation</u>**



$$R_{i+1} = L_i \oplus f(R_i)$$

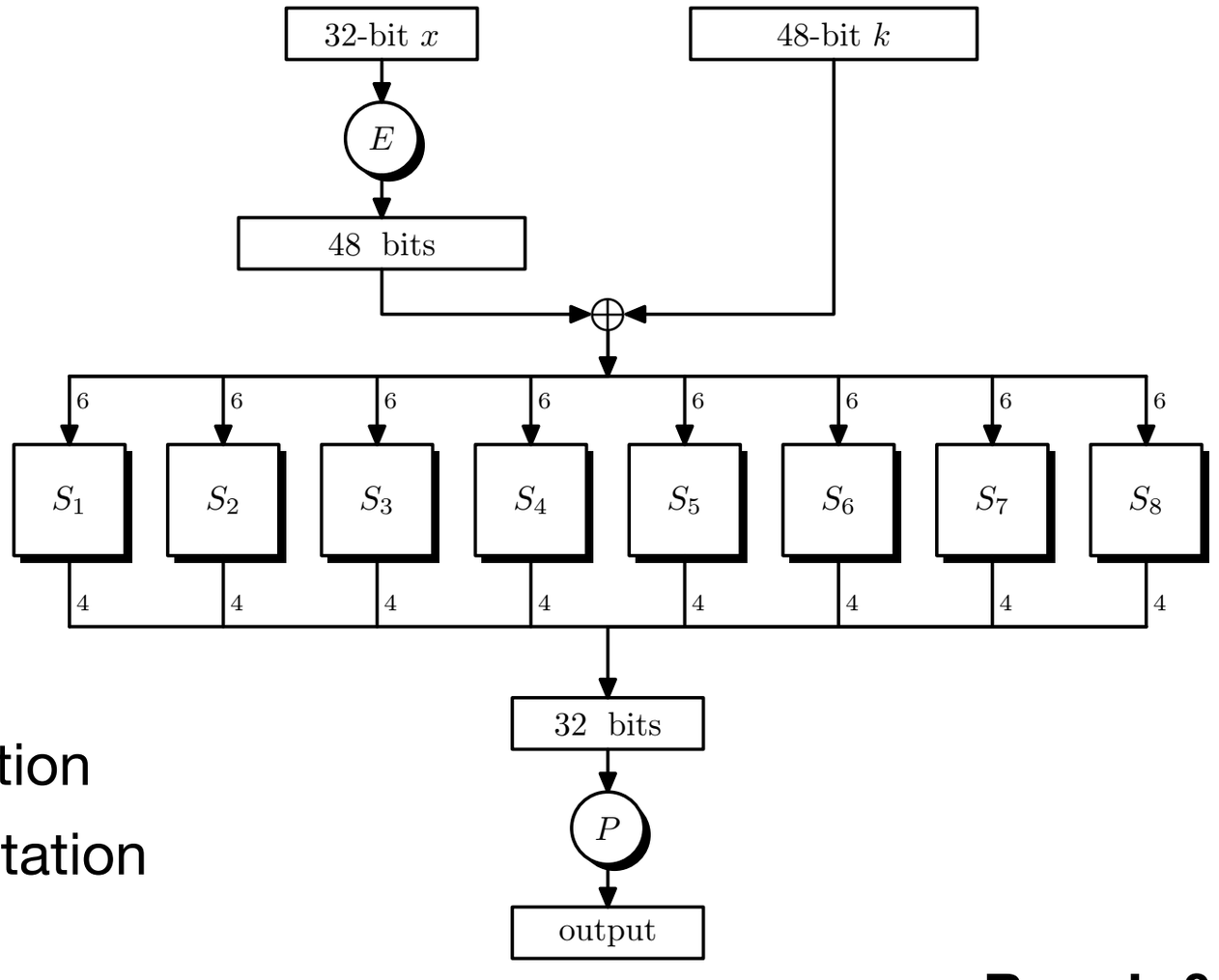
$$L_{i+1} = R_i$$

$$R_i = L_{i+1}$$

$$L_i = R_{i+1} \oplus f(L_{i+1})$$

DES

- Réseau de Feistel à 16 rondes
- Fonction de ronde $f_i : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$



- $S_i : \{0,1\}^6 \rightarrow \{0,1\}^4$ substitution
- $P : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ permutation

source: Boneh & Shoup

Challenge DES

Challenge

- étant donné$(x_1, y_1 = \mathrm{Enc}(k, x_1)), (x_2, y_2 = \mathrm{Enc}(k, x_2)), (x_3, y_3 = \mathrm{Enc}(k, x_3)), y_4, \ldots, y_n$

- trouver $x_4, \ldots, x_n$ tels que $\mathrm{Enc}(k, x_i) = y_i, 4 \leq i \leq n$

Résultats

- 1997 : projet DESCHALL 96jours, recherche de la clé répartie sur Internet (~78000 participants)

- 1998 : projet distributed.net 41j recherche répartie à plus grande échelle

- 1998 : deepcrack machine spécialisée 56h, 250000$

- 1999 : deepcrack + distributed.net 22h

- 2007 : COPACOBANA 120FGPA 12,6j, ~10000$

3DES

$K = \{0,1\}^{56} \times \{0,1\}^{56} \times \{0,1\}^{56}$

$b = 64$bits (même taille de bloc que DES)

$$3\mathrm{DES}((k_1, k_2, k_3), x) = \mathrm{DES}(k_1, \mathrm{DES}^{-1}(k_2, \mathrm{DES}(k_3, x)))$$
$$3\mathrm{DES}^{-1}((k_1, k_2, k_3), y) = \mathrm{DES}^{-1}(k_3, \mathrm{DES}(k_2, \mathrm{DES}^{-1}(k_1, y)))$$

rétro-compatibilité : $3\mathrm{DES}((k, k, k), x) = \mathrm{DES}(k, x)$

3DES

$K = \{0,1\}^{56} \times \{0,1\}^{56} \times \{0,1\}^{56}$

$b = 64$bits (même taille de bloc que DES)

$$3\mathrm{DES}((k_1, k_2, k_3), x) = \mathrm{DES}(k_1, \mathrm{DES}^{-1}(k_2, \mathrm{DES}(k_3, x)))$$
$$3\mathrm{DES}^{-1}((k_1, k_2, k_3), y) = \mathrm{DES}^{-1}(k_3, \mathrm{DES}(k_2, \mathrm{DES}^{-1}(k_1, y)))$$

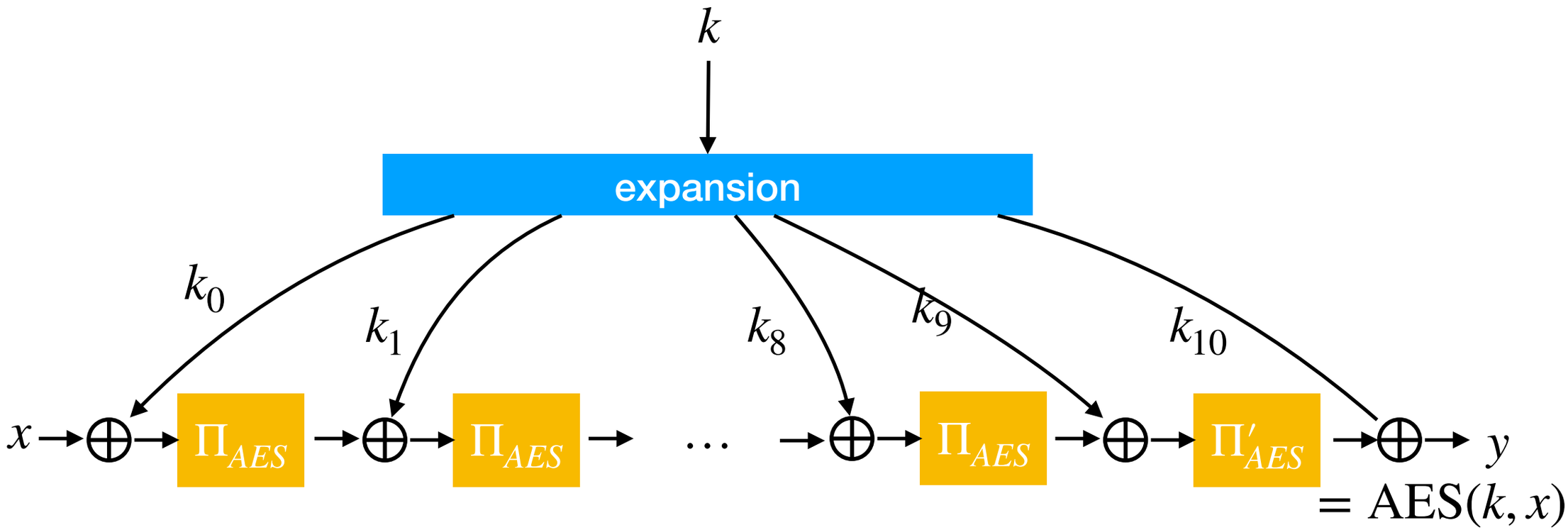
rétro-compatibilité : $3\mathrm{DES}((k, k, k), x) = \mathrm{DES}(k, x)$

pourquoi pas 2DES ? cf. TD

AES

- Advanced Encryption Standard, 2000
- Issu d’un processus de sélection ouvert 1997-2000
- Auteurs Joan Daemen et Vincent Rijmen
- taille de la clé : 128,192 ou 256 bits
- taille de bloc : 128

AES-128



- $\Pi_{AES}, \Pi'_{AES} : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ **permutations**
- 10 rondes

Attaques

- Recherche exhaustive, « brute-force »
- Cryptanalyse linéaire
- Attaque par canaux cachés (« *side-channel attacks* »)
- Recherche exhaustive par machine quantique

Recherche exhaustive

- Etant donné
  $(x_1, y_1 = \mathrm{Enc}(k, x_1)), \ldots, (x_\ell, y_\ell = \mathrm{Enc}(k, x_\ell))$, trouver k

- Typiquement pour $\ell = 3$, unique clé solution

- Attaque à texte clair connu

- AES-128 : 2^{56} clés -> 22heures. 2^{128} clés -> ???

Recherche exhaustive

- Etant donné
  $(x_1, y_1 = \mathrm{Enc}(k, x_1)), \ldots, (x_\ell, y_\ell = \mathrm{Enc}(k, x_\ell))$, trouver k

- Typiquement pour $\ell = 3$, unique clé solution

- Attaque à texte clair connu

- AES-128 : 2^{56} clés -> 22heures. 2^{128} clés ->
  $2^{128-56} \times 22\mathrm{h} \simeq 1{,}18.10^{20}$ ans

Cryptanalyse linéaire

- Identifier des **relations linéaires** entre bits de la clé et bits des messages clairs/chiffrés

- Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K = \{0,1\}^n, M = C = \{0,1\}^b)$

- Relation linéaire s'il existe
  $S_1, S_2 \subseteq \{1,\ldots,b\}, S_3 \subseteq \{1,\ldots,n\}, \epsilon$ t.q.

$$\Pr\left[\bigoplus_{i\in S_1} m[i] \oplus \bigoplus_{i\in S_2} \mathrm{Enc}(k,m)[i] = \bigoplus_{j\in S_3} k[j]\right] \geq 1/2 + \epsilon$$

Cryptanalyse linéaire

- DES : relation linéaire avec $\epsilon \simeq 2^{-21}$

- Retrouver 26 bits de la clés avec $\simeq 2^{43}$ messages clairs/ chiffrés avec probabilité $\simeq 0{,}85$

- 56 - 26 = 30 bits restant recouvré par recherche exhaustive

Attaque par canaux cachés

- Mesure (fine) du temps de chiffrement/déchiffrement peut révéler des informations sur la clé

- Energie consommée peut révéler des informations sur la clé

- Attaque sur AES fondée sur effet de cache. même instruction a des temps d'exécutions significativement différents selon que la donnée est présente ou non dans le cache

- AES-128 : 2^{20} mesures, qqs minutes pour retrouver la clé

Recherche exhaustive quantique

- Algorithme de Grover
- Entrée : accès « boite noire » à $f : \mathcal{K} \to \{0,1\}$ avec $f(k) = 1 \iff k = k_0$
- Sortie : k_0
- Complexité ?

Recherche exhaustive quantique

- Algorithme de Grover
- Entrée : accès « boite noire » à $f : K \rightarrow \{0,1\}$ avec $f(k) = 1 \iff k = k_0$
- Sortie : k_0
- Complexité $O(\sqrt{K} \times C_f)$

Recherche exhaustive quantique

- Algorithme de Grover
- Entrée : accès « boite noire » à $f : \mathcal{K} \to \{0,1\}$ avec $f(k) = 1 \iff k = k_0$
- Sortie : k_0
- Complexité $O(\sqrt{K} \times C_f)$
- Application : AES-128
- Soit $m_1, \ldots, m_d, c_1, \ldots, c_d : \forall i, c_i = \mathrm{Enc}(k_0, m_i)$
- Soit $f : K \to \{0,1\}$ définie par
  $f(k) = 1 \iff \forall 1 \leq i \leq d, \mathrm{Enc}(k, m_i) = c_i$
- Grover : clé recouvrée après $\simeq d \times 2^{64}$ évaluations d'AES

Intégrité

Intégrité

m

Réseau (non sûr)

m’

Alice

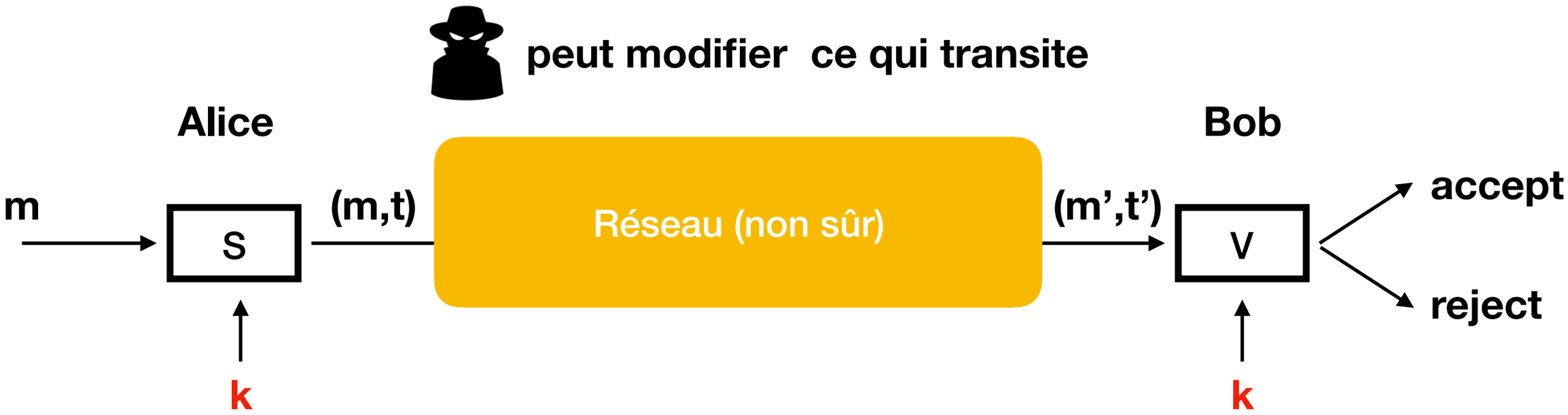
Bob

peut modifier les messages

m= m’ ??

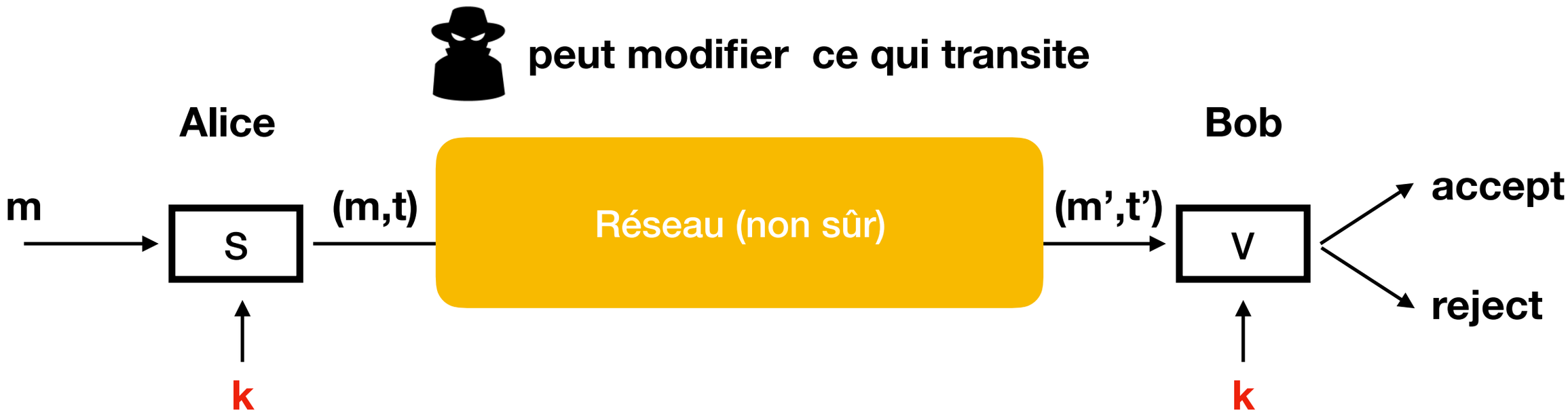
m’ est-il authentique ?
(On ne s’intéresse pas à la confidentialité ici)

Intégrité



- **S, V :** algorithmes de signature/vérification
- **k :** clé secret partagé par Alice et Bob
- **m :** message
- **t :** tag (ou signature ou code d’authentification de message MAC)

Intégrité



- **S, V :** algorithmes de signature/vérification
- **k :** clé secret partagé par Alice et Bob
- **m :** message
- **t :** tag (ou signature ou code d’authentification de message MAC)
- **S** (m:message, k:clé) -> tag
- **V** (m:message, k:clé, t:tag) -> booléen

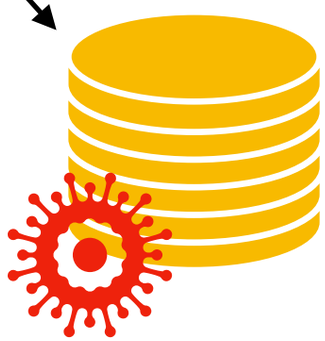
Exemple

Alice

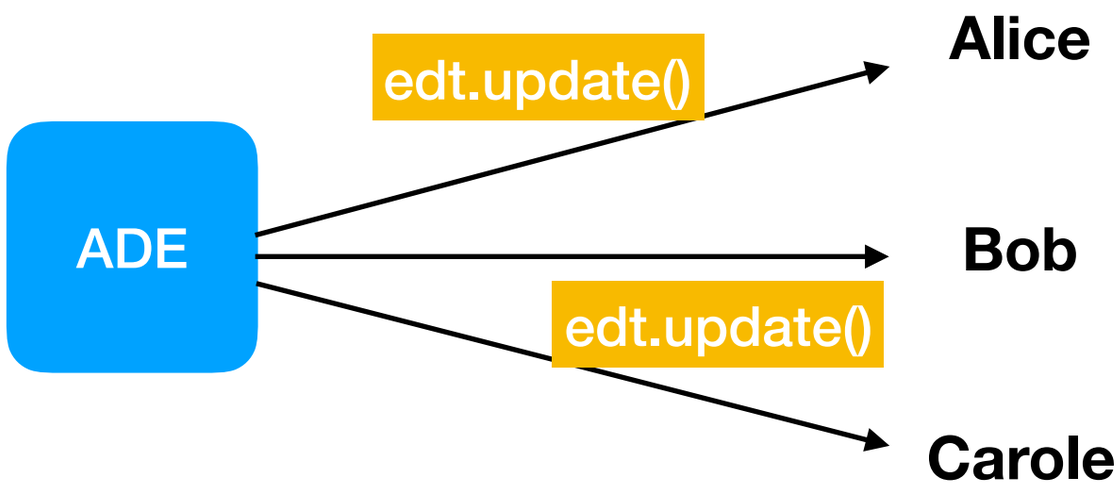
installe programme P

exécute P’

P = P’ ?



Exemple



- **Authenticité des mises à jour ?**
- **Confidentialité non-requise : emploi du temps public**

Système d’authentification de messages

Peut-on concevoir un système d’authentification de messages sans clé secrète ?

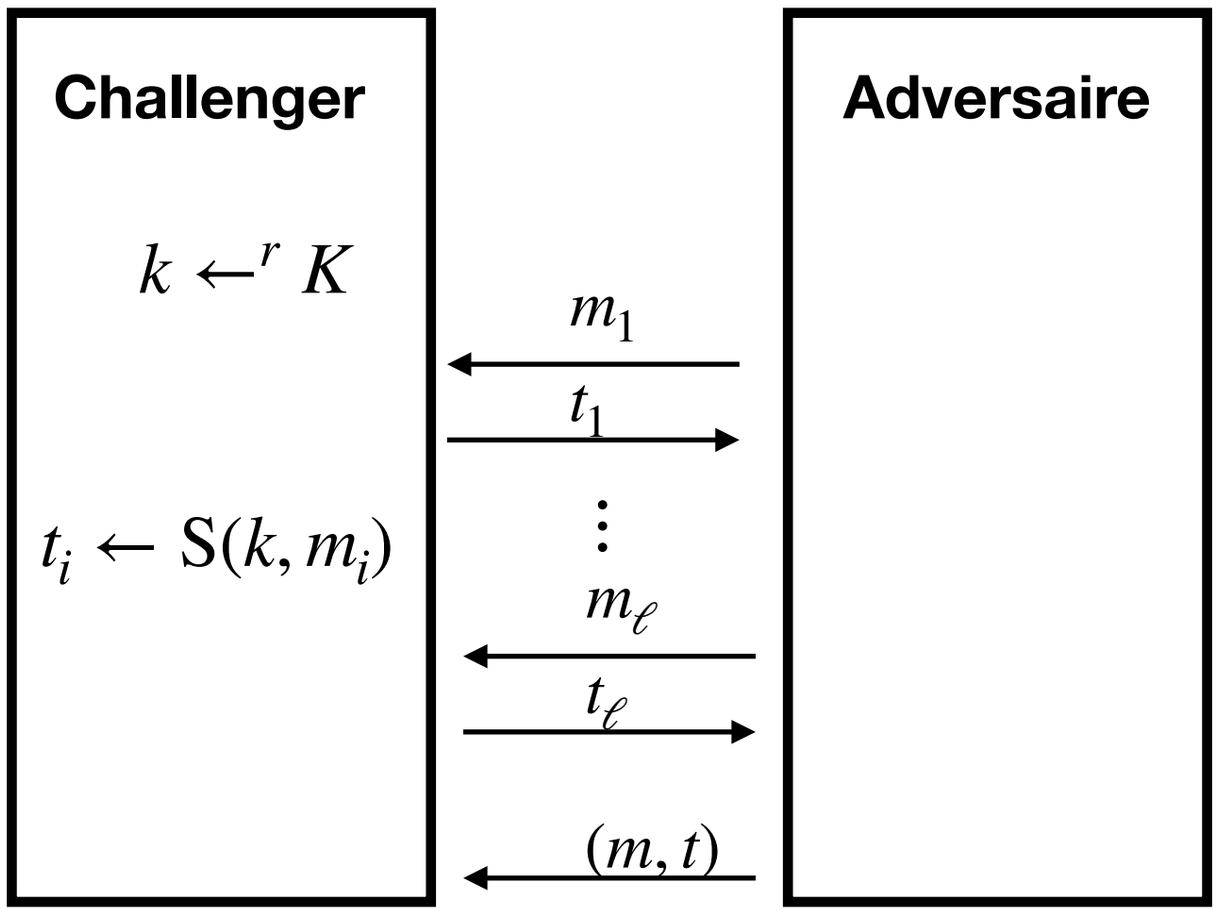
Système d'authentification de messages (MAC)

$(\mathrm{S}, \mathrm{V}, K, M, T)$

- **M: ens. des messages**
- **K : ens. des clés**
- **T : ens. des *tags***
  - **S : K x M → T alg. de signature, probabiliste généralement**
  - **V : K x M x T** $\rightarrow \{0,1\}$ **alg. de verification**

Validité : $\forall m \in M, \forall k \in K, \mathrm{V}(m, k, \mathrm{S}(m, k)) = 1$

Sécurité : contrefaçon existentielle



W **A** produit $(m, t) \in M \times T$ avec $\mathrm{V}(m, t) = 1$et $(m, t) \neq (m_i, t_i), 1 \leq i \leq \ell$

Avantage $\mathrm{Avtg}(A) = \Pr[W]$

Sécurité : contrefaçon existentielle

Soit $I = (\mathrm{S}, \mathrm{V}, K, M, T)$ un système d'authentification de message

I est sûr si et seulement si
pour tout adversaire A *efficace*, $\mathrm{Avtge}(A, I) = \mathsf{negl}$.

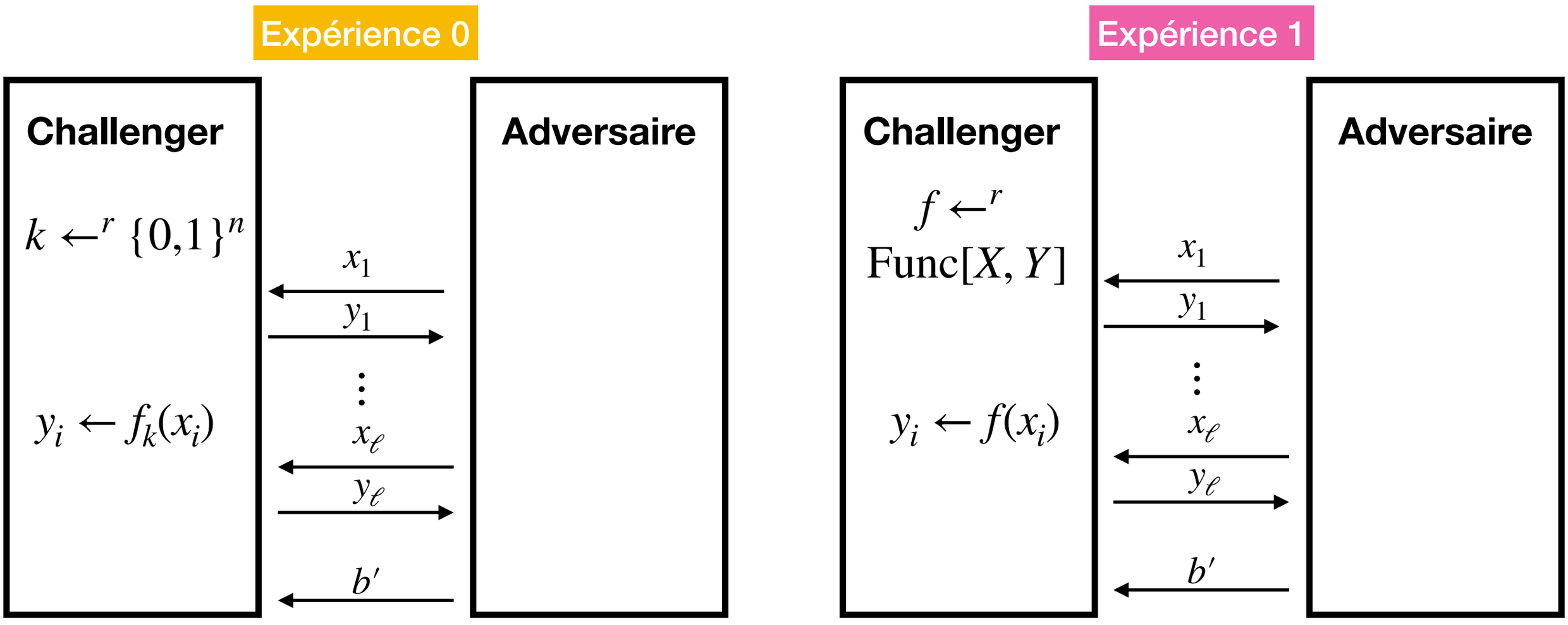
Rappel : fonction-pseudo aléatoire

Soit $F : K \times X \to Y$

Pour $k \in K$**, soit** $f_k : x \to F(k, x)$

f_k **est *pseudo-aléatoire* s'il est calculatoirement difficile de distinguer** f_k **d'une fonction** $X \to Y$ **tiré aléatoirement**

Rappel : fonction pseudo-aléatoire



W_b **A** produit $b' = 1$ dans l'expérience b

Avantage $Avtg(\mathbf{A}, F) = |\Pr[W_1] - \Pr[W_0]|$

Rappel : fonction pseudo-aléatoire

Soit $F : K \times X \to Y$ est pseudo-aléatoire ssi

- **Il existe un alg. *efficace* pour calculer F**
- **Pour tout adversaire A. *efficace*,** $\mathrm{Avtge}(A, F) = \mathrm{negl}$.

Fonction pseudo-aléatoire en pratique : AES, 3DES, etc.

Construction : MAC à partir de fonctions pseudo-aléatoires

Soit $F : K \times X \to Y$ pseudo-aléatoire
alors $I = (S, V, K, M, T)$ avec

- $M = X, T = Y$
- $S(k, m) = F(k, m)$
- $V(k, m, t) = 1$ **si** $F(k, m) = t$,0 **sinon**

est un système d'authentification sûr

Construction : MAC à partir de fonctions pseudo-aléatoires

Soit $F : K \times X \rightarrow Y$ pseudo-aléatoire
alors $I = (S, V, K, M, T)$ avec

- $M = X, T = Y$
- $S(k, m) = F(k, m)$
- $V(k, m, t) = 1$ **si** $F(k, m) = t$,0 **sinon**

est un système d'authentification sûr

limite : taille des messages fixée et petite (ie, 128bits pour AES)
solution : étendre le domaine des fonctions pseudo-aléatoires

Construction : MAC à partir de fonctions pseudo-aléatoires

limite : taille des messages fixée et petite (ie, 128bits pour AES)
solution : étendre le domaine des fonctions pseudo-aléatoires

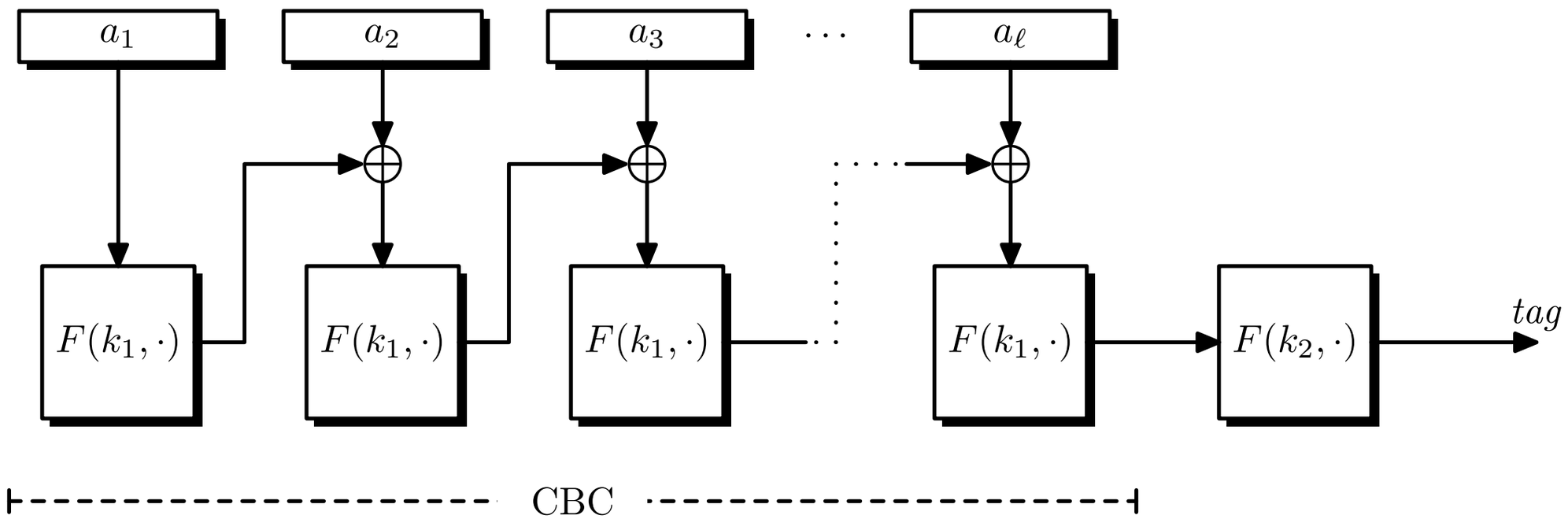
Soit $F : K \times \{0,1\}^b \rightarrow \{0,1\}^b$ **pseudo- aléatoire**
but : construire $F^* : K \times \{0,1\}^{\ell} \rightarrow \{0,1\}^b$**, avec** $\ell \gg b$ **pseudo-aléatoire**

Idée : chainage

ECBC

Encrypted Cipher Block Chaining (ANSI/ISO standard)

$k = (k_1, k_2)$



source : Boneh Shoup

Fonctions de Hachage (cryptographiques)

$H : M \to T$ **fonction de hachage**

- $|T| \ll |M|$
- **Il existe un Alg. efficace pour calculer** H

Fonctions de Hachage (cryptographiques)

$H : M \to T$ **fonction de hachage**

- $|T| \ll |M|$
- **Il existe un Alg. efficace pour calculer** H

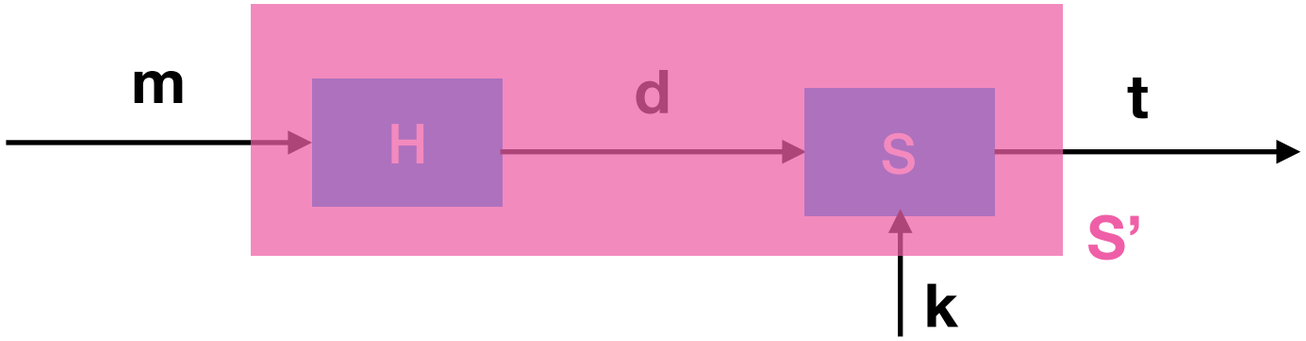
Résistance aux collisions

Pour tout adv. A *efficace,*
proba. produire $m_0 \neq m_1 : H(m_0) = H(m_1)$ **est** negl

Exemple SHA256. $T = \{0,1\}^{256}$, $M = \{0,1\}^{2^{64}}$

Authentification pour de longs messages « hacher puis signer »

- $H : \{0,1\}^{\ell} \rightarrow \{0,1\}^{b}$ ***hachage résistant aux collisions***
- $I = (S, V, K, M = \{0,1\}^{b}, T = \{0,1\}^{b})$ **système d'authentification *sûr***



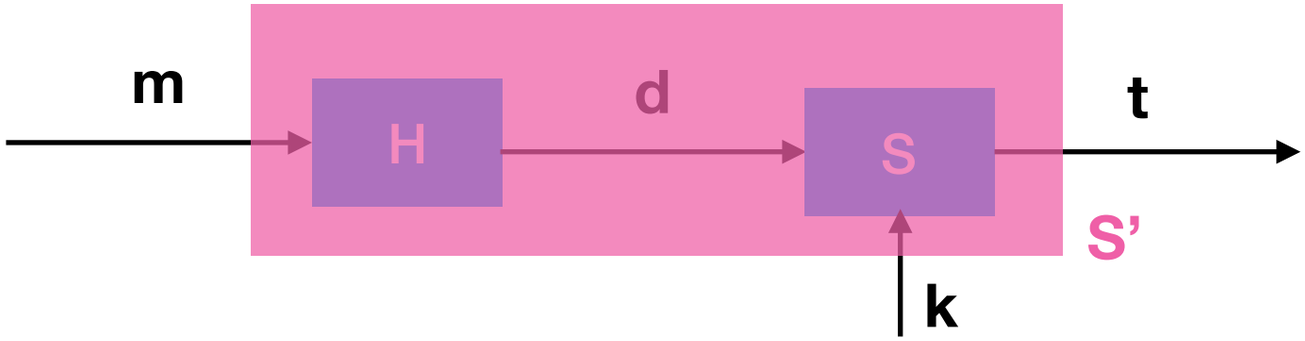
$I' = (S', V', K, M' = \{0,1\}^{\ell}, C)$ **avec**

- $S'(k, m) = S(k, H(m))$
- $V'(k, m, t) = V(k, H(m), t)$

est un système d'authentification sûr

Authentification pour de longs messages « hacher puis signer »

- $H : \{0,1\}^{\ell} \to \{0,1\}^{b}$ ***hachage résistant aux collisions***
- $I = (S, V, K, M = \{0,1\}^{b}, T = \{0,1\}^{b})$ **système d'authentification *sûr***



$I' = (S', V', K, M' = \{0,1\}^{\ell}, C)$ **avec**

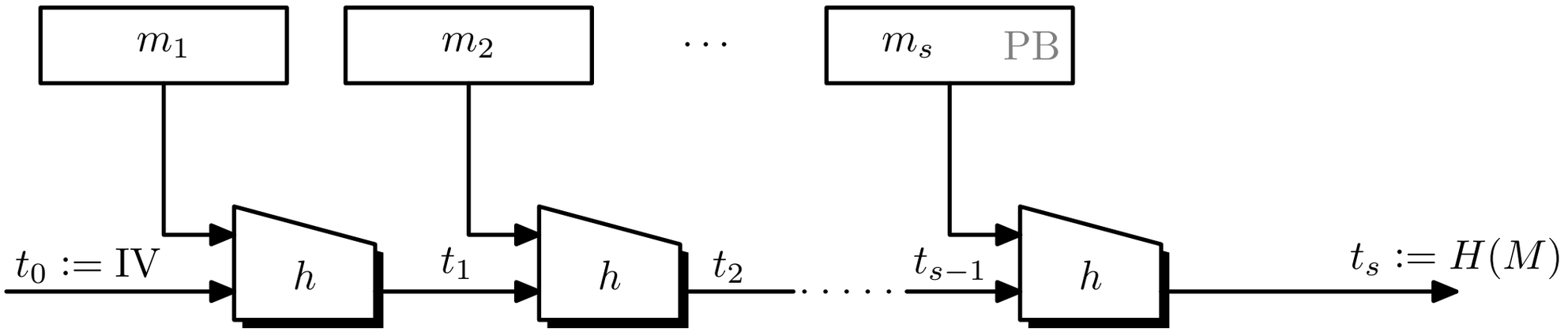
- $S'(k, m) = S(k, H(m))$
- $V'(k, m, t) = V(k, H(m), t)$

est un système d'authentification sûr

limites :

- **Besoin : hachage H et MAC I**
- **Collision pour H casse le système**

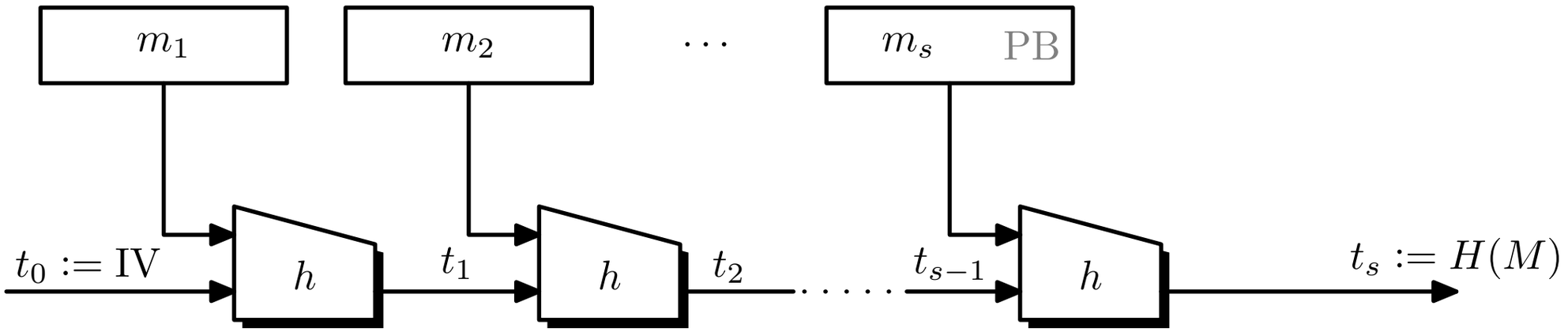
Fonction de hachage: construction de Merkle-Damgard



- $h : K \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{n}$ **fonction de compression**
  $h_k : x \rightarrow h(k, x)$ **est une fonction de hachage**
- **IV : valeur initiale fixée, connue e.g. 0….0**
- **PB : padding block de la forme 10…0 ‖ s**

source : Boneh & Shoup

Fonction de hachage: construction de Merkle-Damgard

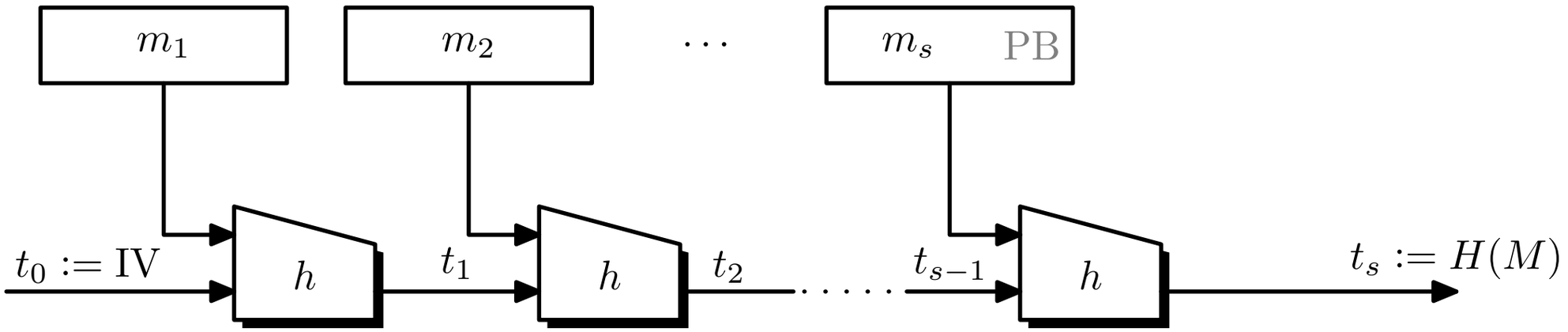


- $h : K \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ **fonction de compression**
  $h_k : x \to h(k,x)$ **est une fonction de hachage**
- **IV : valeur initiale fixée, connue e.G. 0….0**
- **PB : padding block de la forme 10…0 ‖ s**

SHA256

- $\ell = 512, n = 256, K = \{0,1\}^{256}$
- **IV** valeur compliquée sur 256 bits
- s : 64 bits

Fonction de hachage: construction de Merkle-Damgard



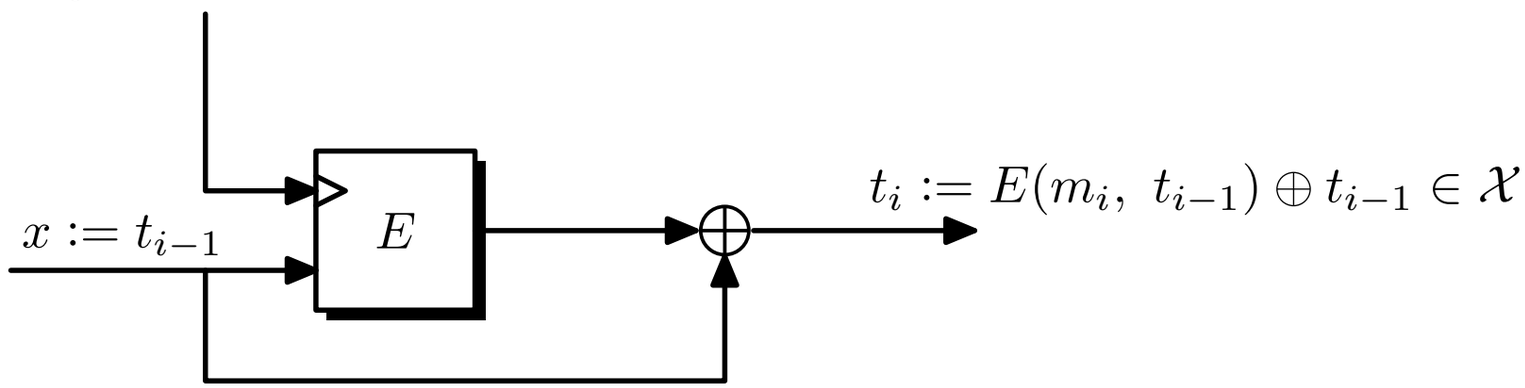
- $h : K \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ **fonction de compression**
  $h_k : x \to h(k,x)$ **est une fonction de hachage**
- **IV : valeur initiale fixée, connue e.G. 0….0**
- **PB : padding block de la forme 10…0 ‖ s**

théorème

Si h est résistant aux collisions alors H l'est aussi

Fonction de compression h construction de Davies-Meyer

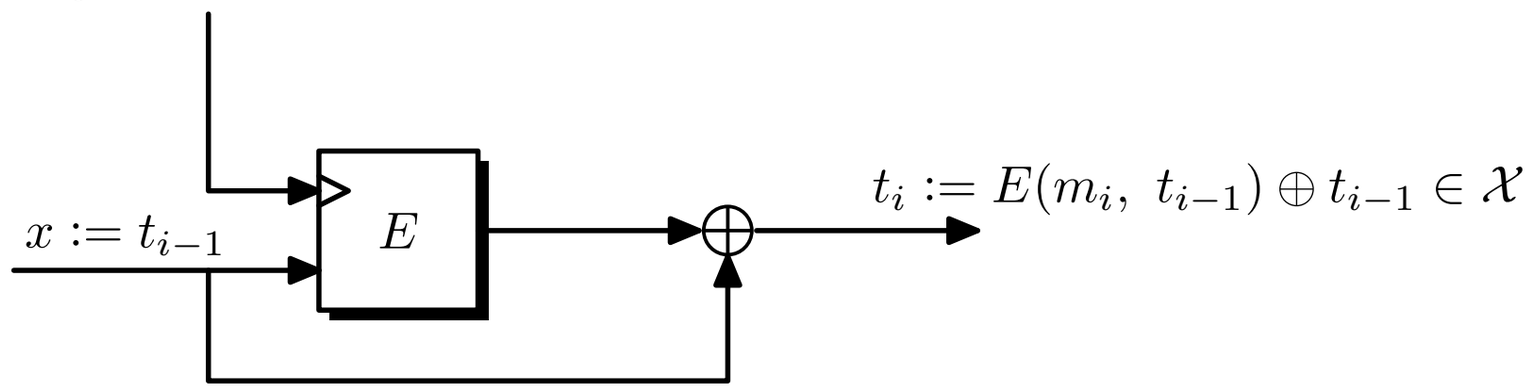
Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, X, Y)$ un chiffrement par bloc avec $X = Y$



$h : X \times K \to X$ définie par $h(x, y) = \mathrm{Enc}(y, x) \bigoplus x$ est une fonction de compression

Fonction de compression h construction de Davies-Meyer

Soit $\mathscr{E} = (\mathrm{Enc}, \mathrm{Dec}, K, X, Y)$ un chiffrement par bloc avec $X = Y$



théorème :

Si $\mathscr{E}$ ~~est sûr~~ alors h est résistant aux collisions

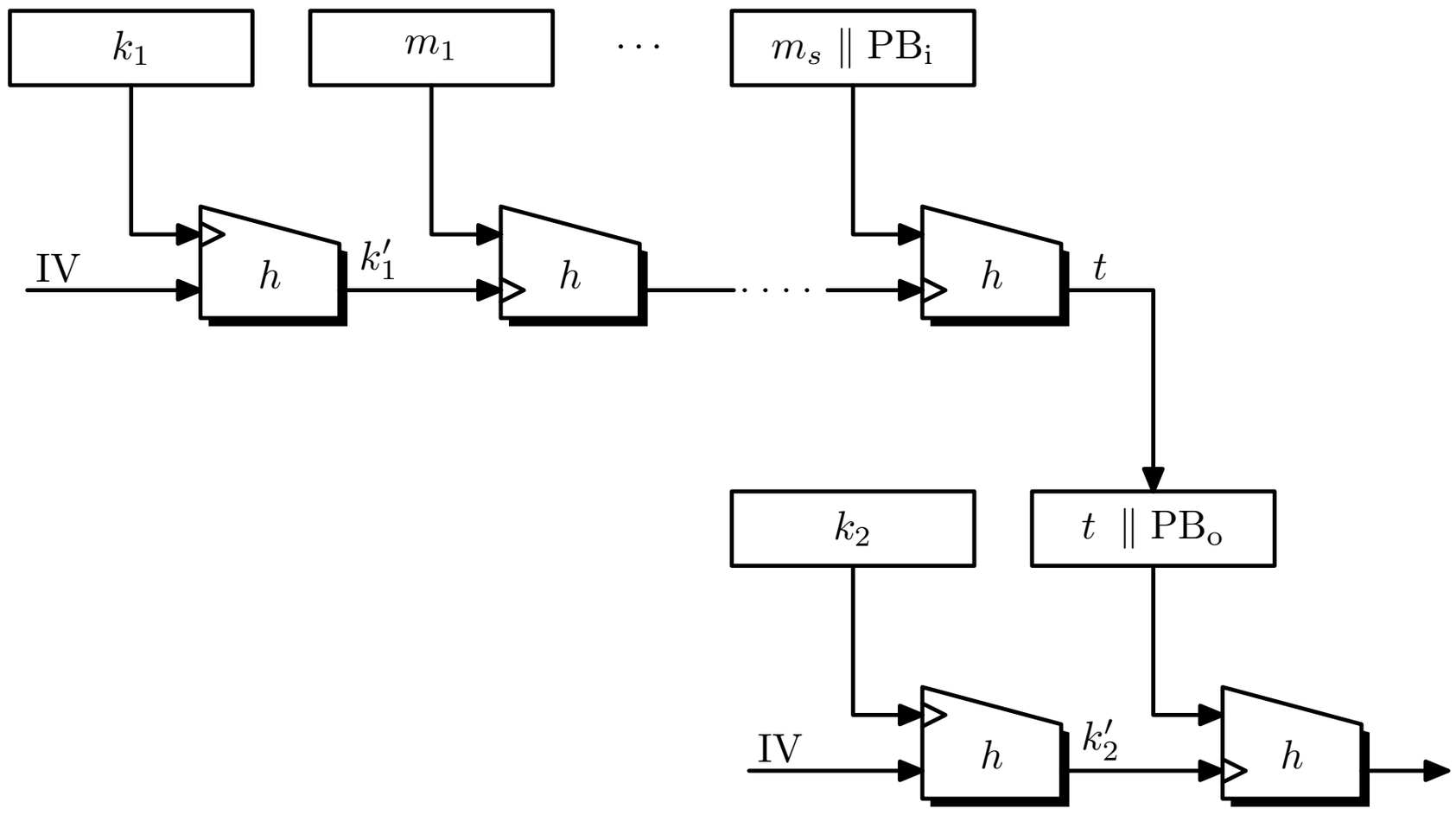
est idéalement sûr

limites : taille de bloc doit être relativement grande (128 bits insuffisant)

HMAC

- Signer des messages longs
- Système d’authentification n’utilisant qu’une fonction de hachage
- Que faire de la clé ??

HMAC



$$S((k_1, k_2), m) = H(k_2 \cdot H(k \cdot m_1 \cdot PB_i) \cdot PB_o)$$

Fonction de Hachage autres notions de sécurité

- Soit $H : M \rightarrow T$ une fonction de hachage

- H est **résistant à la seconde pré-image** si étant donné m, il est calculatoirement difficile de trouver $m' \neq m : H(m) = H(m')$

- H est **à sens unique** si étant donné $t = H(m)$, il est calculatoirement difficile de trouver $m' : H(m') = t$

Fonction de Hachage autres notions de sécurité

- Soit $H : M \rightarrow T$ une fonction de hachage

- H est **résistant à la seconde pré-image** si étant donné m, il est calculatoirement difficile de trouver $m' \neq m : H(m) = H(m')$

- H est **à sens unique** si étant donné $t = H(m)$, il est calculatoirement difficile de trouver $m' : H(m') = t$

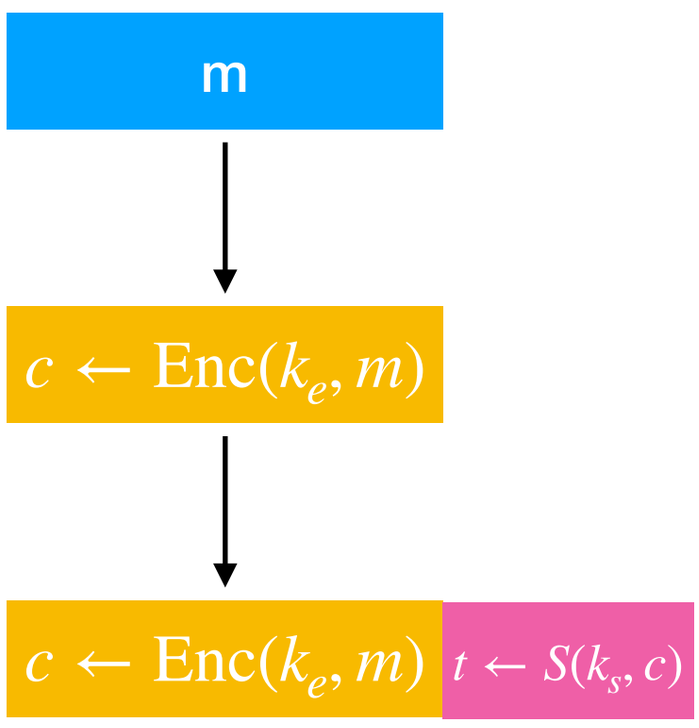
H est à collision difficile $\Longrightarrow$ H est résistant à la seconde préimage $\Longrightarrow$ H est à sens unique

Intégrité et Confidentialité

(authenticated encryption)

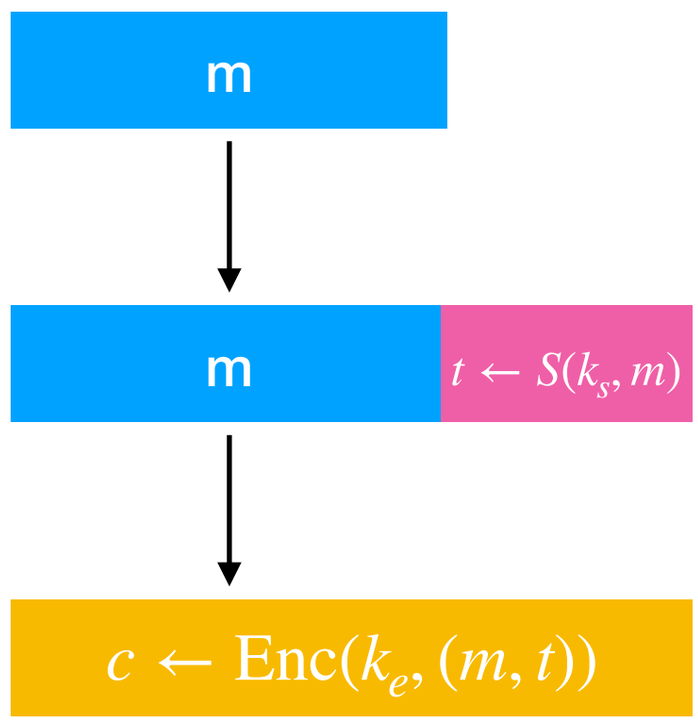
Chiffrer et signer

chiffrer puis signer



TLS 1.2, IPsec, Galois Counter Mode

signer puis chiffrer



TLS 1.0, 802.11i WiFi

Correct : chiffrer puis signer. chiffrement sûr + signature sûre => chiffrement authentifié