



Valentin Suder

CRYPTOGRAPHIE

UNE INTRODUCTION

ORGANISATION GÉNÉRALE DE L'UE

- ▶ 8 séances de 3 heures (mercredi 9:00-12:00)
- ▶ une partie cours (~1h-1h30)
- ▶ une partie exercices/TP (en python) (~1h30-2h)

DÉCOUPAGE EN 5 CHAPITRES

1. La cryptographie «*antique*»
2. La cryptographie «*symétrique*»
3. La cryptographie «*asymétrique*»
4. La cryptographie «*en pratique*»
5. La cryptographie «*quantique et post-quantique*»



LA CRYPTOGRAPHIE

«L'ANTIQUITÉ»

Simon
Singh

Histoire
des
codes secrets

De l'Egypte des Pharaons à l'ordinateur quantique



UNE BONNE
LECTURE

LA STÉGANOGRAPHIE

- ▶ du grec «stégano», couvrir, et «graphie», l'écriture.
C'est l'art de dissimuler un message.
- ▶ Exemples :
 - ▶ tatouer la tête d'un esclave
 - ▶ encre invisible
 - ▶ micropoints
 - ▶ bits de poids faible d'une image
 - ▶ ...

«OUI, MAIS SI QUELQU'UN
TROUVE NOTRE MESSAGE
SECRET, ON FAIT QUOI ?»

Anonyme, ~400bc

LA CRYPTOGRAPHIE, QU'EST-CE QUE C'EST ?

- ▶ Du grec «*crypto*» le secret et «*graphie*» l'écriture.
- ▶ Plus généralement, on parlera de «*cryptologie*» («logique» : la science) qui regroupe
 - ▶ la cryptographie
 - ▶ et la cryptanalyse (du grec «*analyse*»...)

=> L'art du secret

LA SCYTALE

- ▶ Inventé à Sparte en 400 av JC
- ▶ Vous avez besoin :
 - ▶ d'une *lanière de cuir* pour écrire le message
 - ▶ d'un *bâton* pour enrouler la lanière de cuir
- ▶ Pour chiffrer, vous enroulez la lanière autour du bâton et écrivez normalement.
- ▶ Pour déchiffrer, vous avez besoin d'un bâton de même diamètre.



FONCTION DE CHIFFREMENT D'UNE SCYTALE

- ▶ en paramètre de la fonction :
 - ▶ un message (chaîne de caractères)
 - ▶ le diamètre du bâton
1. Calculer sur quelle longueur de bande écrire, et bourrer
 2. Écrire chaque caractère à la bonne position sur la bande
 3. Retourner la chaîne correspondant à la bande

LES CHIFFREMENTS PAR TRANSPOSITION

- ▶ Procédé qui consiste à *changer l'ordre des lettres* d'un message.

- ▶ Message *clair* : MESSAGE
- ▶ Message **chiffré** : ESGMSAE

QUELLE SÉCURITÉ

- ▶ Nombre de façon de permuter **n lettres** : $n!$
- ▶ CLE => CLE, CEL, LCE, LEC, ECL, ELC : $3! = 6$ possibilités
- ▶ pour un mot de 20 lettres : $20 != 2432902008176640000$
soit $\approx 2^{61}$
- ▶ Plutôt bon ! **Problème** : en pratique l'ordonnancement des lettres est *rigoureux* et fixé à l'avance.

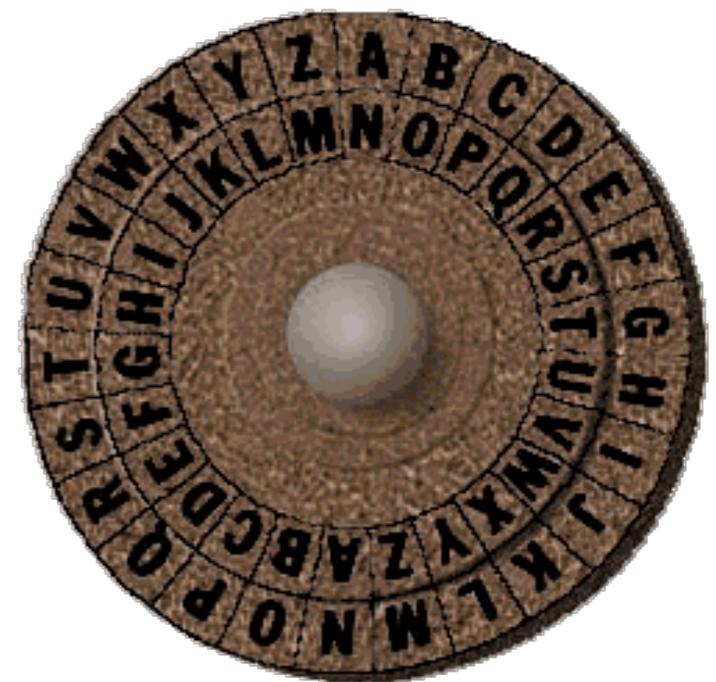
LE CHIFFREMENT DE CÉSAR (100-44 AV JC)

- ▶ A->D, B->E, C->F, D->G, etc...
- ▶ Exemple :
ATTAQUONSCESOIR
DWWDTXRQVFHVRLU
- ▶ DOHDMDGWDHW
ALEA JACTA EST



LES CHIFFREMENTS PAR DÉCALAGE

- ▶ Décalage de l'alphabet par un entier $k \in \{1,2,\dots,25\}$
- ▶ $k = 3$ (chiffrement de césar)
- ▶ Ce chiffrement n'est *pas sûr !!*
Il n'y a que **26 clés** possibles.



FONCTION DE CHIFFREMENT/DÉCHIFFREMENT PAR DÉCALAGE

- ▶ en paramètre de la fonction :
 - ▶ un message (chaîne de caractères)
 - ▶ la clé k de décalage
1. Ecrire l'alphabet dans une liste
 2. Pour chaque lettre du message, trouver l'indice de son décalé par k, et écrire le résultat dans le message chiffré
 3. Retourner le message chiffré.

LES CHIFFREMENTS PAR SUBSTITUTION

- ▶ **Substitution** : chaque lettre du message en clair est remplacée par une *autre lettre, chiffre ou symbole*.
- ▶ **Substitution mono-alphabétique** : le même alphabet est conservé tout au long du chiffrement et déchiffrement.

La **clé secrète** est la permutation entre les alphabets.

- ▶ Exemples :
 - ▶ Chiffre de César
 - ▶ Carré de Polybe
 - ▶ Chiffre des Templiers

LE CARRÉ DE POLYBE

- ▶ Polybe : historien grec, 200-125 av JC
- ▶ Texte clair :
CARRE DE POLYBE
- ▶ Texte chiffré :
13 11 36 36 15 14 15 34 33 26 51 12 15

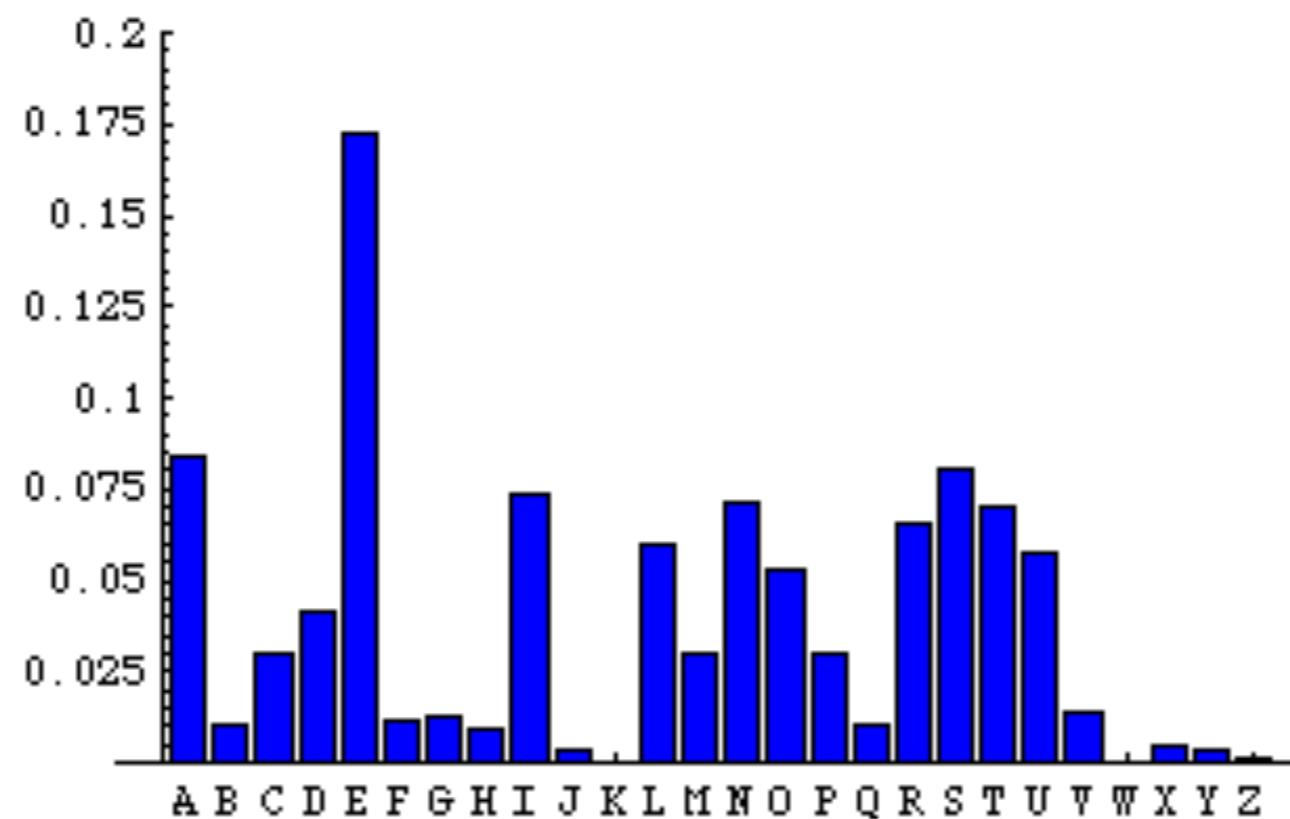
	1	2	3	4	5	6
1	A	B	C	D	E	F
2	G	H	I	J	K	L
3	M	N	O	P	Q	R
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

FONCTION QUI CHIFFRE AVEC UN CARRÉ DE POLYBE

- ▶ en paramètre de la fonction :
 - ▶ un message clair (une chaîne de caractères)
 - ▶ un carré de Polybe (sous forme de chaîne de caractères)
1. Écrire la correspondance entre les indices de la chaîne et du carré de Polybe.
 2. Pour chaque lettre du message clair, écrire les 2 chiffres correspondants dans la chaîne du message chiffré.

L'ANALYSE DE FRÉQUENCE

- ▶ Dans une langue donnée, et dans un texte de longueur suffisante, chaque lettre apparaît avec une certaine fréquence.



ATTAQUER UN CHIFFRE DE CÉSAR

- ▶ Retrouver la clé et le message clair derrière le texte suivant :

SGOZXK IUXHKG_A, YAX AT GXHXK
VKXINK, ZKTGOZ KT YUT HKI AT LXUSGMK.
SGOZXK XKTGXJ, VGX R'UJKAX GRRKINK,
RAO ZOTZ G VKA VXKY IK RGTMGMK : KZ
HUTPUAX, SUTYOKAX JA IUXHKG_A, WAK
BUAY KZKY PURO ! WAK BUAY SK
YKSHRK_F HKGA ! YGTY SKTZOX, YO BUZXK
XGSGMK YK XGVVUXZK G BUZXK
VRASGMK, BUAY KZKY RK VNKTOD JKY
NUZKY JK IKY HUOY.

LE CHIFFREMENT DE VIGENÈRE

- ▶ Inventé par Blaise de Vigenère (1523-1596) ou alors Giovan Batista Belaso (1564) (?)
- ▶ On choisit un mot clé, qui nous permettra de choisir quel décalage choisir, lettre par lettre.
- ▶ Résiste à l'analyse de fréquence...
- ▶ Mais quand même attaquable (Charles Babbage (1792-1871))
 1. Déterminer la longueur de la clé
 2. Faire une analyse de fréquence par lettre de la clé

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		
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		
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	A		
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B		
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C		
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D		
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E		
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F		
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G		
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G		
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H		
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I		
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J		
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K		
N	N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
R	R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
S	S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
T	T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
U	U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
V	V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	
W	W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	
X	X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	
Y	Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
Z	Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	

FONCTION DE CHIFFREMENT DE VIGENÈRE

- ▶ paramètre de la fonction :
 - ▶ un message clair (une chaîne de caractères)
 - ▶ un mot clé (une chaîne de caractère)
1. Utiliser la fonction de chiffrement de César
 2. Pour chaque lettre du mot clé, utiliser le décalage associé
 3. Répéter en boucle jusqu'à ce que le message clair soit entièrement lu et chiffré.

MÉLANGE DE TRANSPOSITION ET DE SUBSTITUTION

LE CHIFFREMENT ENIGMA

- ▶ Utiliser par les allemands pendant la seconde guerre mondiale
- ▶ Comporte :
 - ▶ un clavier
 - ▶ un tableau lumineux
 - ▶ des rotors (3 ou 5)
 - ▶ un tableau de connexion
 - ▶ des réflecteurs
- ▶ Finalement cassé par Alan Turing par la méthode des mots probables

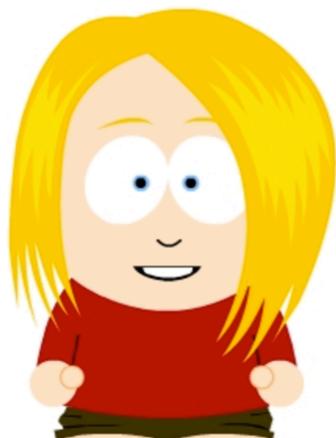


DES USAGES DE SÉCURITÉ DIFFÉRENTS :

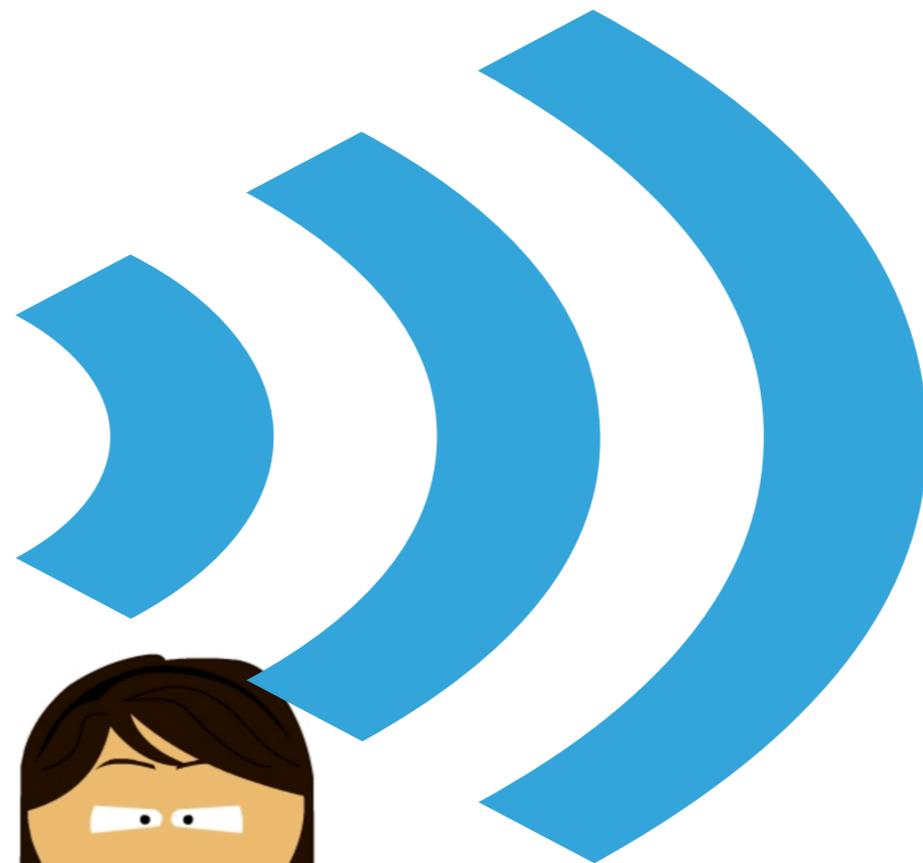
- ▶ **Confidentialité** : un adversaire ne doit pas être capable de *lire* le message.
- ▶ **Authenticité** : un adversaire ne doit pas être capable de *modifier l'origine* d'un message.
- ▶ **Intégrité** : un adversaire ne doit pas être capable de *contrefaire* un message.

CONFIDENTIALITÉ

- ▶ **Échanger** des messages en présence d'adversaire
- ▶ **Stocker** des informations sensibles



Alice



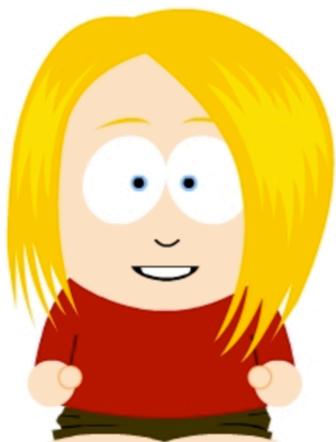
Eve



Bob

AUTHENTICITÉ

- ▶ S'assurer de la *provenance* et de l'*authenticité* d'un message.



Alice



Bob

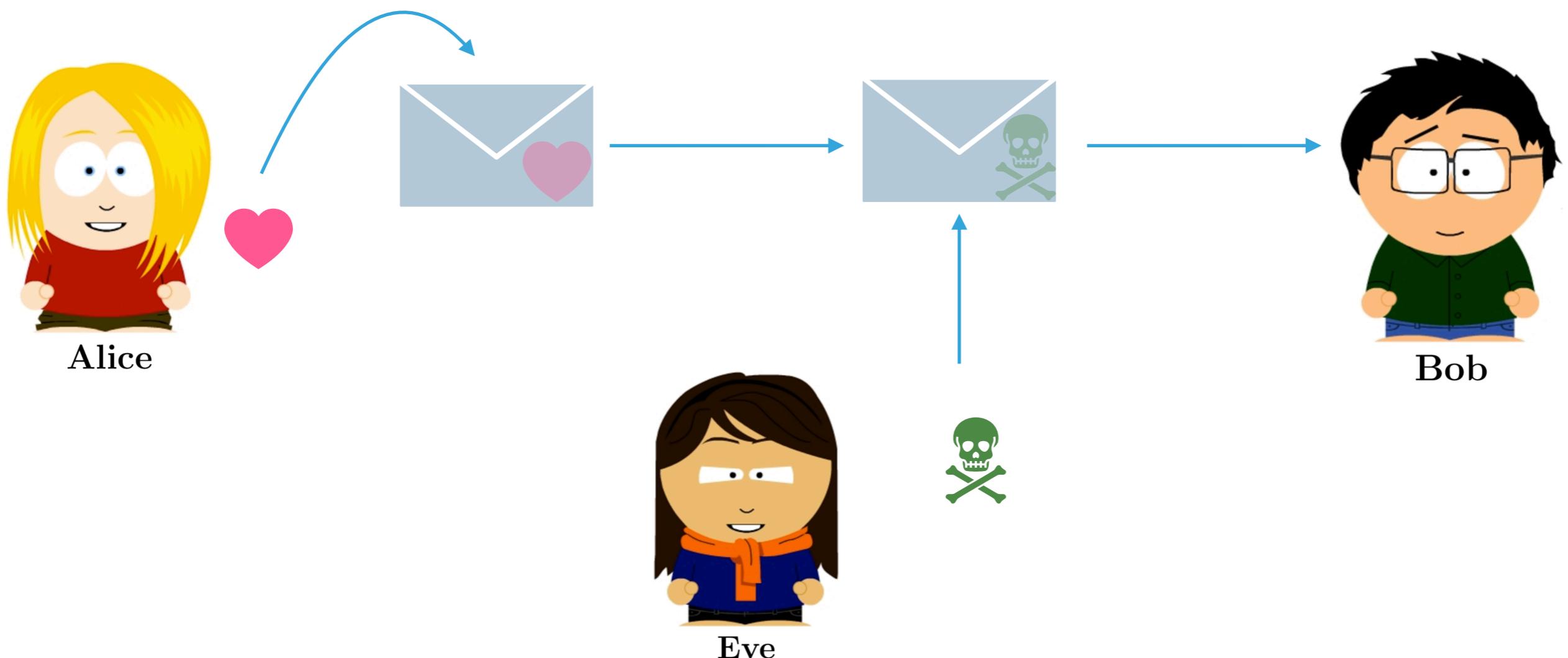


Eve



INTÉGRITÉ

- ▶ S'assurer de la non-modification d'un message, *accidentelle* ou *intentionnelle*.



«LA SÉCURITÉ D'UN
CRYPTOSYSTÈME NE DEVRAIT
REPOSER QUE SUR LA CLÉ. TOUT DU
SYSTÈME DOIT ÊTRE PUBLIQUE.»

«C-À-D : L'ENNEMI
CONNAIT LE SYSTÈME...»

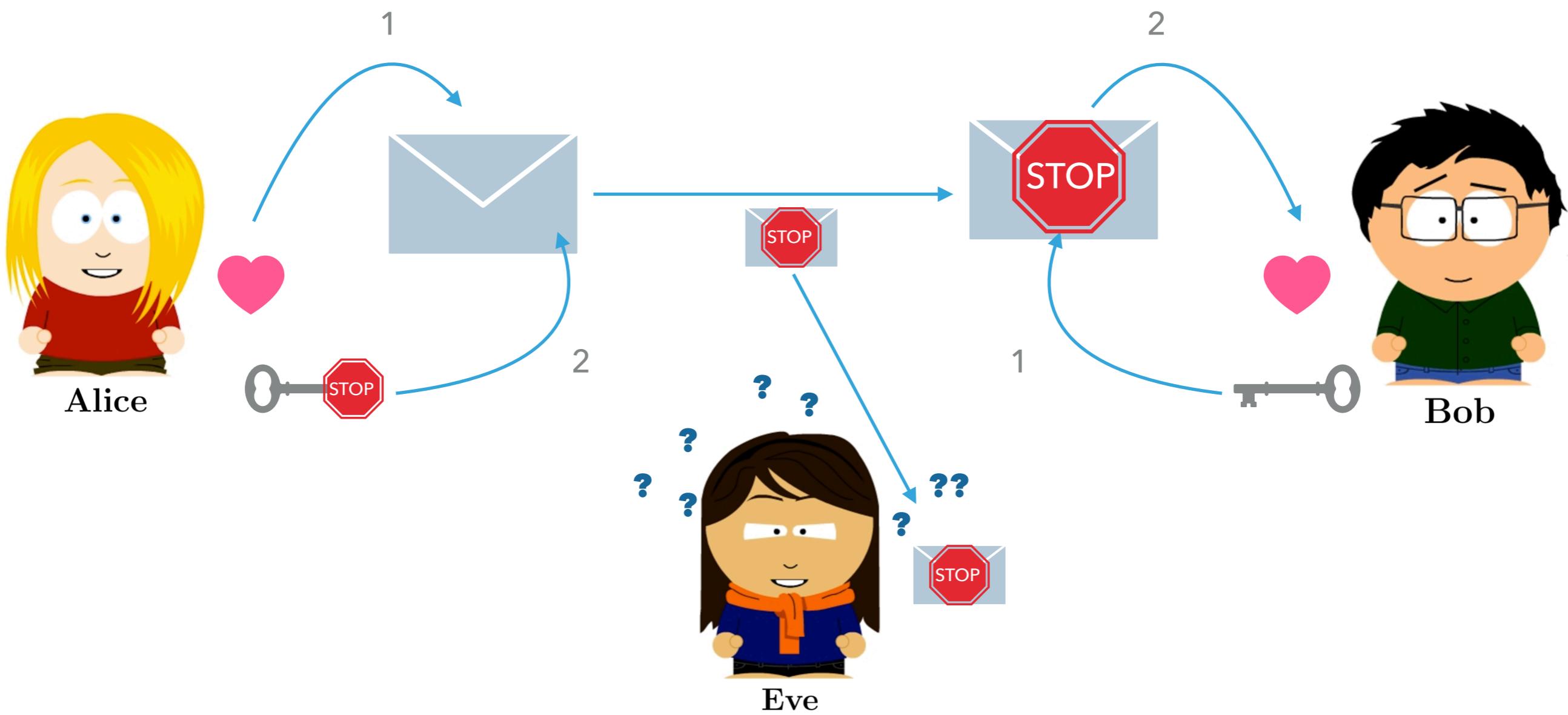
Kerckhoffs (ca. 1883)

LA CRYPTOGRAPHIE SYMÉTRIQUE



DÉFINITION :

- «L'ensemble des algorithmes et procédés dont le *secret est identique* des 2 côtés du canal de communication.»



PRINCIPALES FAMILLES D'ALGORITHMES :

▶ Chiffrement par flot

rapide - à la volée - léger

Chiffrer des données au fur et à mesure qu'elles arrivent ([voix](#))

▶ Chiffrement par bloc

permet de chiffrer une grande quantité de données

Chiffrer des données de taille connue ([logiciel, disque dur, ...](#))

▶ Fonction de hachage

permet de créer des empreintes courtes

Vérifier l'authenticité et/ou l'intégrité des données ([téléchargement, preuves, ...](#))

PRINCIPALES FAMILLES D'ALGORITHMES

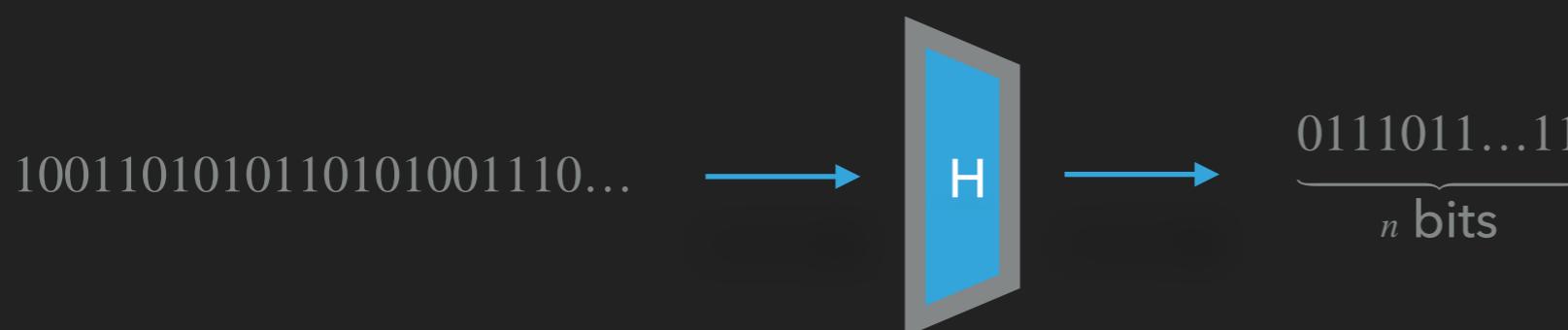
- ▶ chiffrement par flot :



- ▶ chiffrement par bloc :



- ▶ fonction de hachage :



message **clair** :

0100110101001011010010...



message **chiffré** :

1110010101101010010010101...

CRYPTOGRAPHIE SYMÉTRIQUE

LES CHIFFREMENTS PAR FLOT

HISTORIQUEMENT : LE CHIFFREMENT DE VERNAM (1917)

- ▶ Également appelé «masque jetable» ou «One Time Pad» (OTP) en anglais.

Le chiffré d'un message $m = (m_1, \dots, m_n)$ par une clé $K = (K_1, \dots, K_n)$ est

$$c = E_K(m) = (m_1 \oplus K_1, \dots, m_n \oplus K_n)$$

ONE-TIME-PAD

$$c = E_K(m) = (m_1 \oplus K_1, \dots, m_n \oplus K_n)$$

- ▶ Créer une fonction qui prend un message et une clé (sous forme de chaîne de caractères), et qui réalise le xor bit à bit.
- ▶ Pensez à vérifier que les 2 chaînes sont de la même longueur
- ▶ À faire avec des chaînes binaires, puis avec des chaînes de caractères...
- ▶ Utiliser les fonctions : *ord* - *format* - *chr*
pour formater les chaines de caractères

SÉCURITÉ INCONDITIONNELLE

► Théorème de Shannon :

Un système cryptographique tel que : $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ assure une confidentialité parfaite si et seulement si :

- Pour chaque $k \in \mathcal{K}$, et $\forall m_1, m_2 \in \mathcal{P}$ on a $Pr(E_k(m_1) = c) = Pr(E_k(m_2) = c)$
- et $\forall m \in \mathcal{P}, c \in \mathcal{C}, \exists ! k \in \mathcal{K}$ tel que $E_k(m) = c$

► Théorème de Shannon :

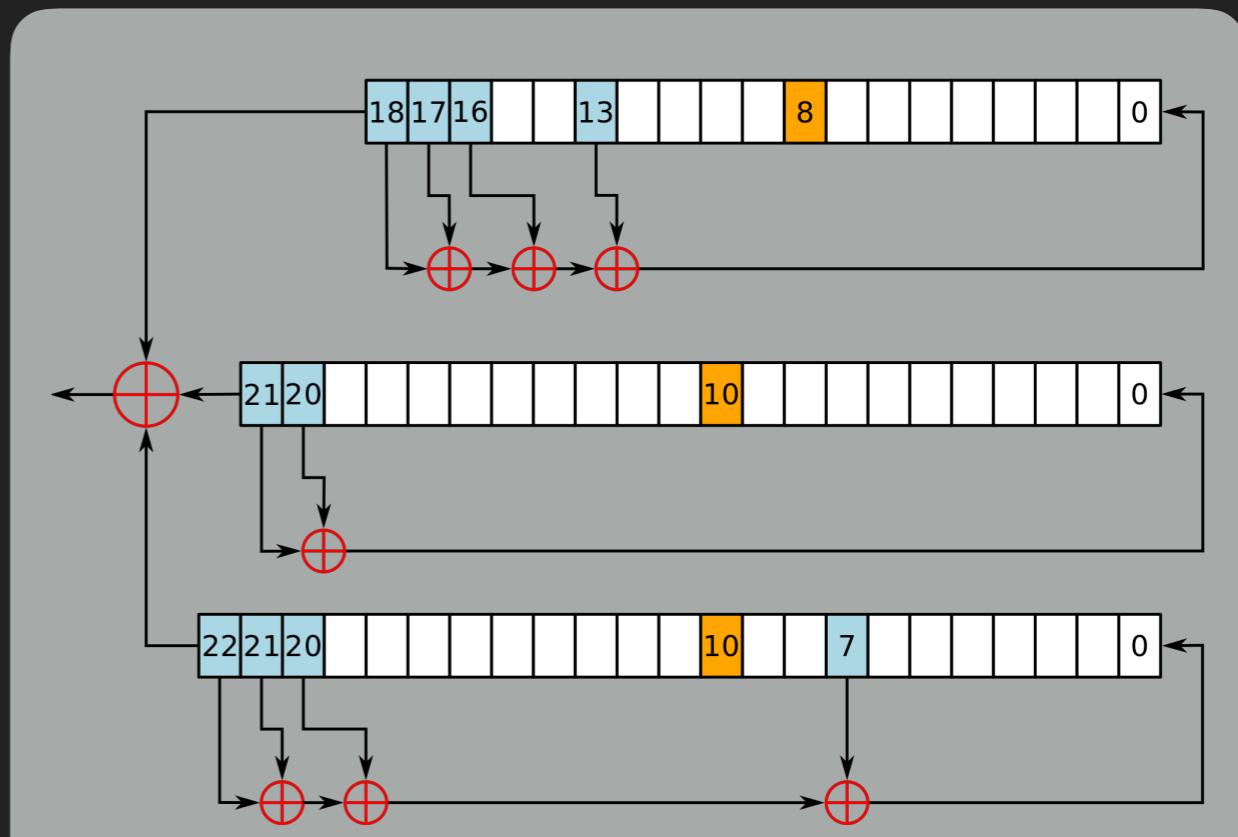
Si toutes les clés sont **équiprobables** et si elles ne sont utilisées qu'**une et une seule fois** alors le chiffre de Vernam assure une **confidentialité parfaite**.

UTILISATION EN PRATIQUE

- ▶ La clé doit faire la même longueur que le message
- ▶ Une nouvelle clé doit être générée pour toute nouvelle correspondance
- ▶ Comment générer aléatoirement de telles clés ?
- ▶ Le chiffrement étant symétrique, il s'agit de s'échanger ces clés, de manière sécurisée

REGISTRES À DÉCALAGES PAR RÉTROACTIONS (LINÉAIRES)

- ▶ Idée :
Utiliser une suite *courte et finie* de bits pour créer une suite chiffrante (un masque) qui possède le plus de propriétés aléatoires possibles
- ▶ Solution :
Les registres à décalages par rétroactions (linéaires ou non)
- ▶ Pourquoi ?
Des opérations très simples et qui s'exécutent très rapidement



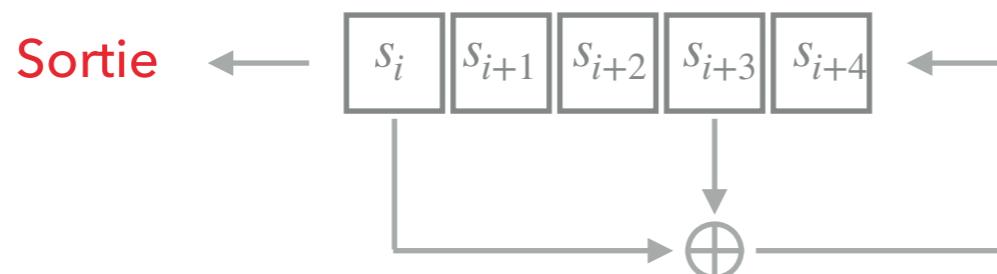
A5/1, Standard de chiffrement de GSM

LINEAR FEEDBACK SHIFT REGISTER (LFSR)

- ▶ Un LFSR de longueur L est un générateur comprenant L registres, qui produit une suite de symboles à partir :
 - ▶ de L symboles initiaux : s_0, s_1, \dots, s_{L-1}
 - ▶ d'une fonction de récurrence (ou rétroaction) :
$$s_i c_L + s_{i+1} c_{L-1} + \dots + s_{i+L-1} c_1 = s_L$$
- ▶ La période de la suite générée est d'au plus $2^L - 1$
- ▶ Elle est **exactement** $2^L - 1$ ssi son polynôme de rétroaction est **primitif**

$$P(X) = c_L X^L + \dots + c_2 X^2 + c_1 X + 1 \in \mathbb{F}_q[X]$$

LFSR (LINEAR FEEDBACK SHIFT REGISTER)



- ▶ À chaque top d'horloge :
 - ▶ le registre s_i est ajouté à la suite chiffrante
 - ▶ le registre s_{i+L} est calculé à partir des autres puis inséré
 - ▶ tous les autres registres sont décalés
- ▶ La *suite générée* est : $s_0 \ s_1 \ s_2 \ s_3 \ \dots$

LFSR BINAIRES

- ▶ Créer une classe LFSR. Chaque instance sera initialisée avec une chaîne binaire représentant le polynôme de rétroaction (sans le +1).
 - ▶ **len** : la longueur du LFSR
 - ▶ **pol** : la fonction de rétroaction
 - ▶ **S** : la suite chiffrante
 - ▶ **state** : l'état interne des registres du LFSR
- ▶ Créer une méthode '**reset**' qui permet de réinitialiser '**S**' à vide et l'état interne '**state**' à la chaîne binaire passée en argument
- ▶ Créer une méthode '**next**' qui permet de calculer le prochain bit de la suite chiffrante, m-à-j '**S**' et '**state**'.
- ▶ c.f. démonstration

```
class LFSR:  
    def __init__(self,P):  
        self.len = len(P)  
        self.pol = P  
        self.S = ''  
        self.state = ''
```

ALGORITHME DE BERLEKAMP-MASSEY (1969)

- ▶ Berlekamp avait un algorithme qui permettait de décoder les codes BCH, à partir d'évaluations polynomiales.
- ▶ En 1969, Massey l'adapte afin de retrouver le polynôme de rétroaction de longueur L à partir d'une suite binaire de longueur $2L$.
- ▶ En fait, il calcul le plus petit LFSR qui permet de générer une suite binaire donnée, en considérant que cette dernière est une période complète (ou une répétition), ce que l'on appelle la *complexité linéaire*.

ALGORITHME DE BERLEKAMP-MASSEY

- ▶ Entrée : $s_0, s_1, s_2, \dots, s_{N-1}$
- ▶ Sortie : $P(X)$ sous la forme $c_L, c_{L-1}, \dots, c_1, 1$
 - ▶ $P(X) = 1; L=0; m=-1; g(X)=1;$
 - ▶ Pour tout n dans $[0..N-1]$ Faire
 - ▶ Si $\bigoplus_{i=1}^L c_i s_{n-i} \oplus s_n = 1$ Faire
 - ▶ $t(X) = P(X); P(X) = P(X) + g(X)X^{n-m};$
 - ▶ Si $2L \leq n$ Faire
 - ▶ $L=n+1-L; m=n; g(X) = t(X);$
 - ▶ Retourner $P(X)$

TRACE D'EXECUTION

N	S_N	d	L	$P(X)$	m	$g(X)$
			0		1 -1	1
0	0	0	0		1 -1	1
1	0	0	0		1 -1	1
2	1	1	3	$X^3 + 1$	2	1
3	1	1	3	$X^3 + X + 1$	2	1
4	0	1	3	$X^3 + X^2 + X + 1$	2	1
5	1	1	3	$X^2 + X + 1$	2	1
6	1	0	3	$X^2 + X + 1$	2	1
7	1	1	5	$X^5 + X^2 + X + 1$	7	$X^2 + X + 1$
8	0	1	5	$X^5 + X^3 + 1$	7	$X^2 + X + 1$

L'ALGORITHME DE BERLEKAMP-MASSEY

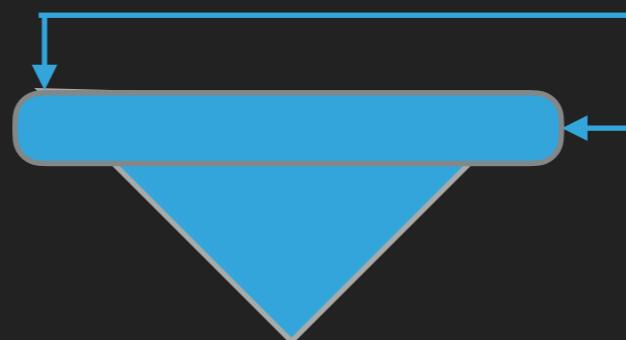
- ▶ Quelques conseils :
 - ▶ Maintenez la longueur de P toujours à N/2
 - ▶ Pour faire la multiplication, transformez la chaîne en entier et faites un shift :

```
P = bin(int(P,2) ^ int(g,2)<<(n-m))[2:]
```

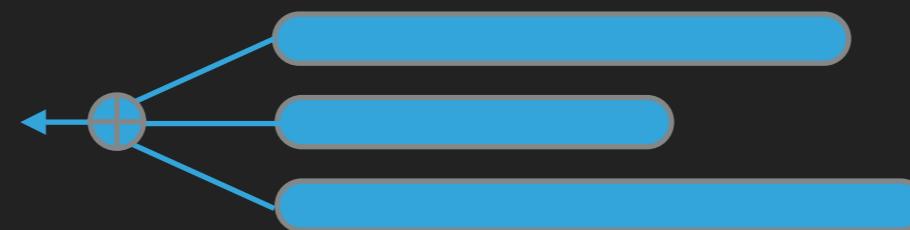
- ▶ Enlevez le +1 lorsque vous retournez le polynôme de rétroaction

MÉTHODES CLASSIQUES

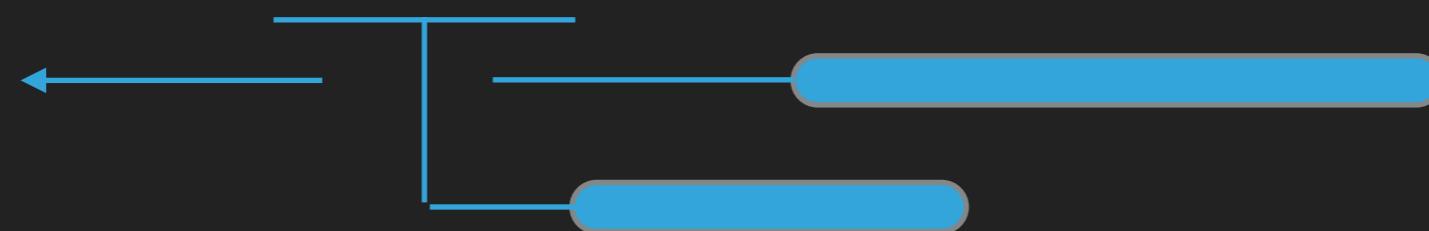
- ▶ Par registre filtré :



- ▶ Par registres combinés

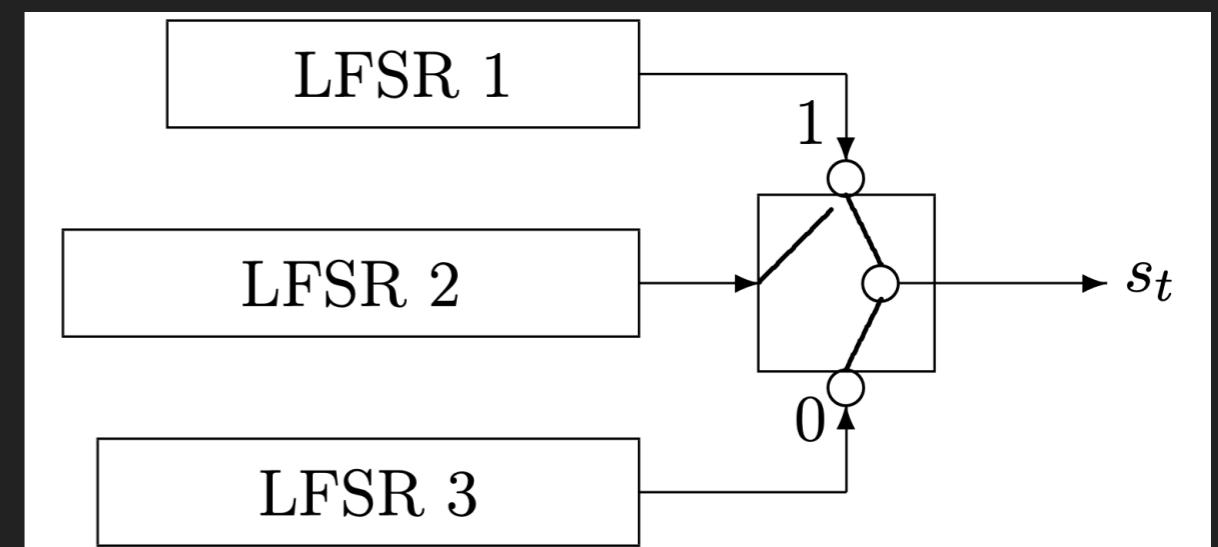


- ▶ Par registre à décalage irrégulier



LE GÉNÉRATEUR DE GEFFE (1973)

- ▶ 3 générateurs de longueurs différentes
Originellement : 7,13 et 12
- ▶ La fonction de combinaison peut s'écrire
$$f(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_3$$
- ▶ Bien, mais vulnérable aux attaques par corrélation.

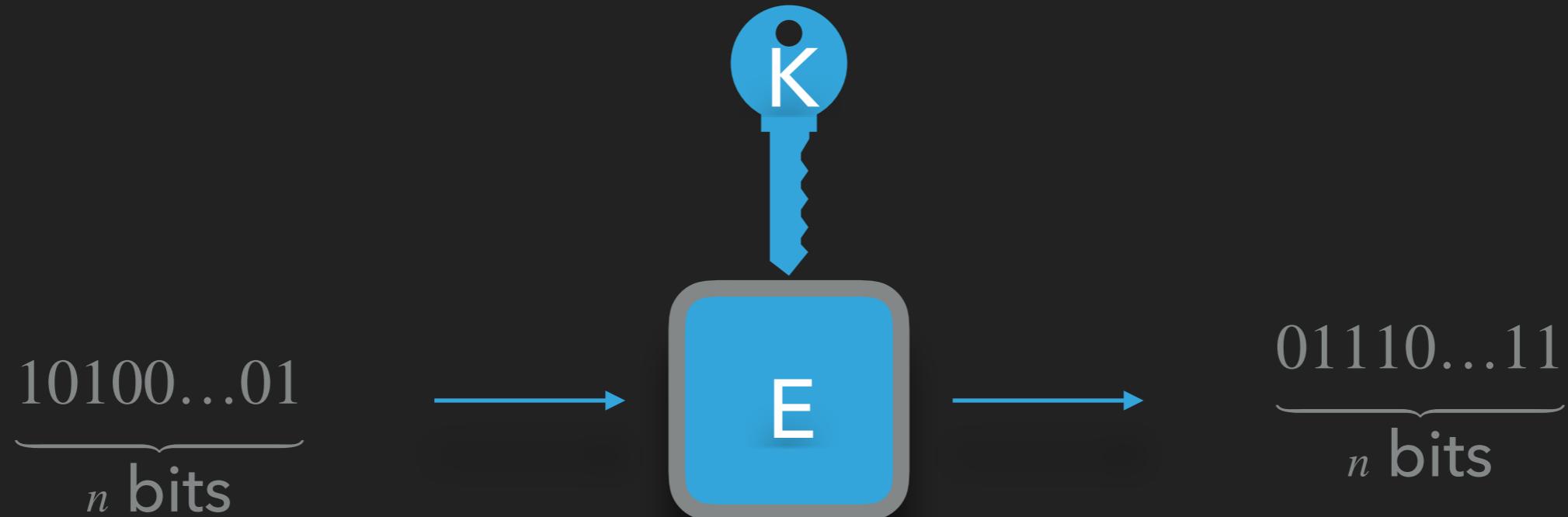


LE GÉNÉRATEUR DE GEFFE

- ▶ Implémentez une classe Geffe qui serait initialisée par 3 polynômes de rétroaction
- ▶ Ré-utilisez la classe LFSR
- ▶ Imaginez une classe où vous pourriez aussi choisir la fonction de combinaison (après que vous ayez vu les attaques par corrélation...)

ATTAQUE PAR CORRÉLATION (SUR LE GÉNÉRATEUR DE GEFFE)

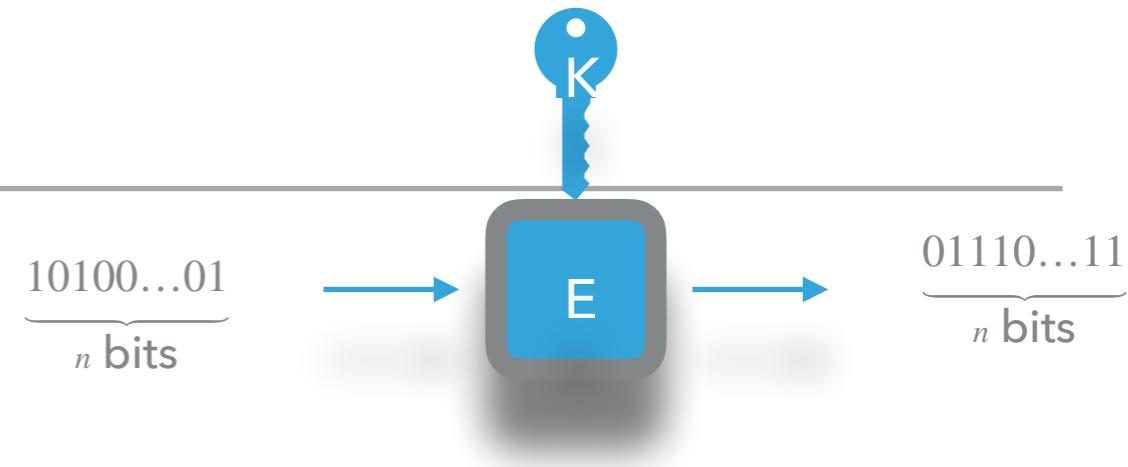
- ▶ La sortie du LFSR # i à l'instant t est notée : u_t^i
- ▶ le bit de suite chiffrante à l'instant t est noté :
$$f(u_t^1, u_t^2, u_t^3) = u_t^1 u_t^2 \oplus u_t^3 u_t^2 \oplus u_t^3 = s_t$$
- ▶ Choisissez des polynômes primitifs pour les 3 LFSR (de longueurs resp. 7, 13 et 12) (n'en changez plus !)
- ▶ Calculez u_t^1 et u_t^3 pour $t \in [0..99]$ pour chaque valeurs de registres possible
- ▶ Comparez pour quelles valeurs 75% correspondent à la suite chiffrante ($2^7 - 1$ pour le premier et $2^{12} - 1$ pour le troisième)
- ▶ Retrouvez l'initialisation du LFSR 2 par recherche exhaustive.



CRYPTOGRAPHIE SYMÉTRIQUE

LES CHIFFREMENTS PAR BLOC

DÉFINITION : CHIFFREMENT PAR BLOC



Une famille de fonctions :

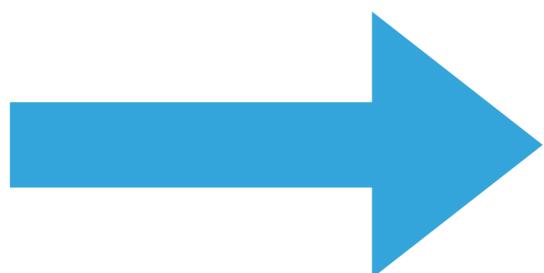
$$E : \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$$

$$M, \quad K \mapsto E_K(M) = C$$

telle que E_K soit **bijective** quelque soit K.

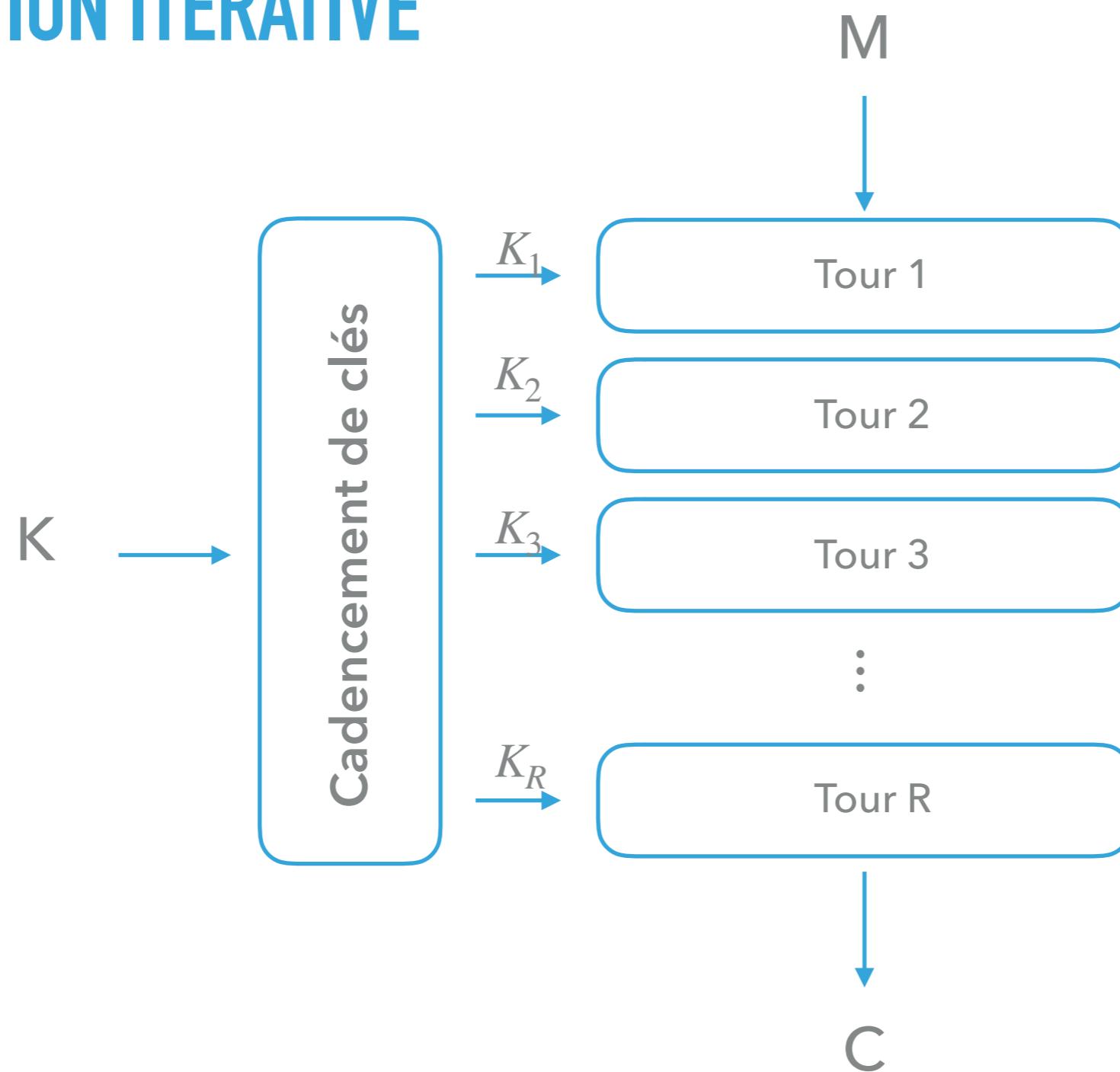
▶ Typiquement, pour des applications pratiques :

$n > 64$ et $k > 80$



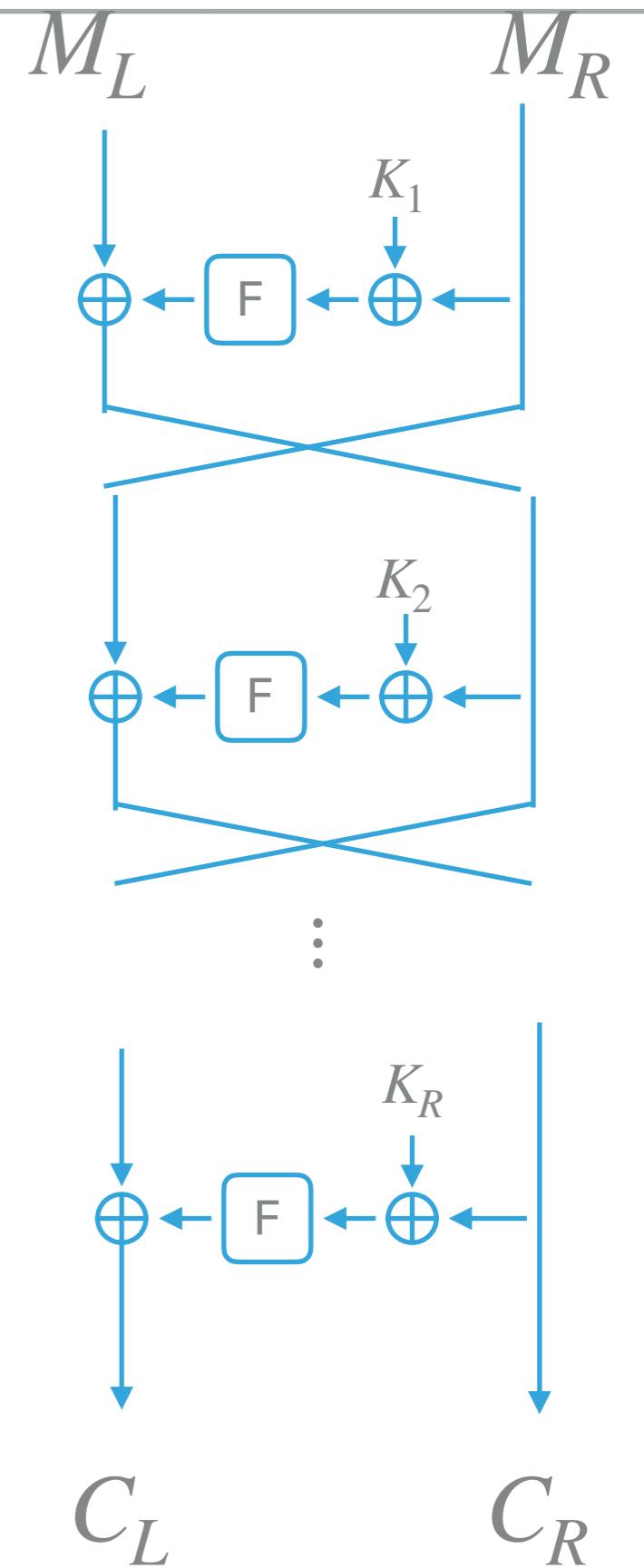
Un chiffrement par bloc, c'est 2^{80} permutations,
permutant 2^{64} mots !! (au moins)

CONSTRUCTION ITÉRATIVE



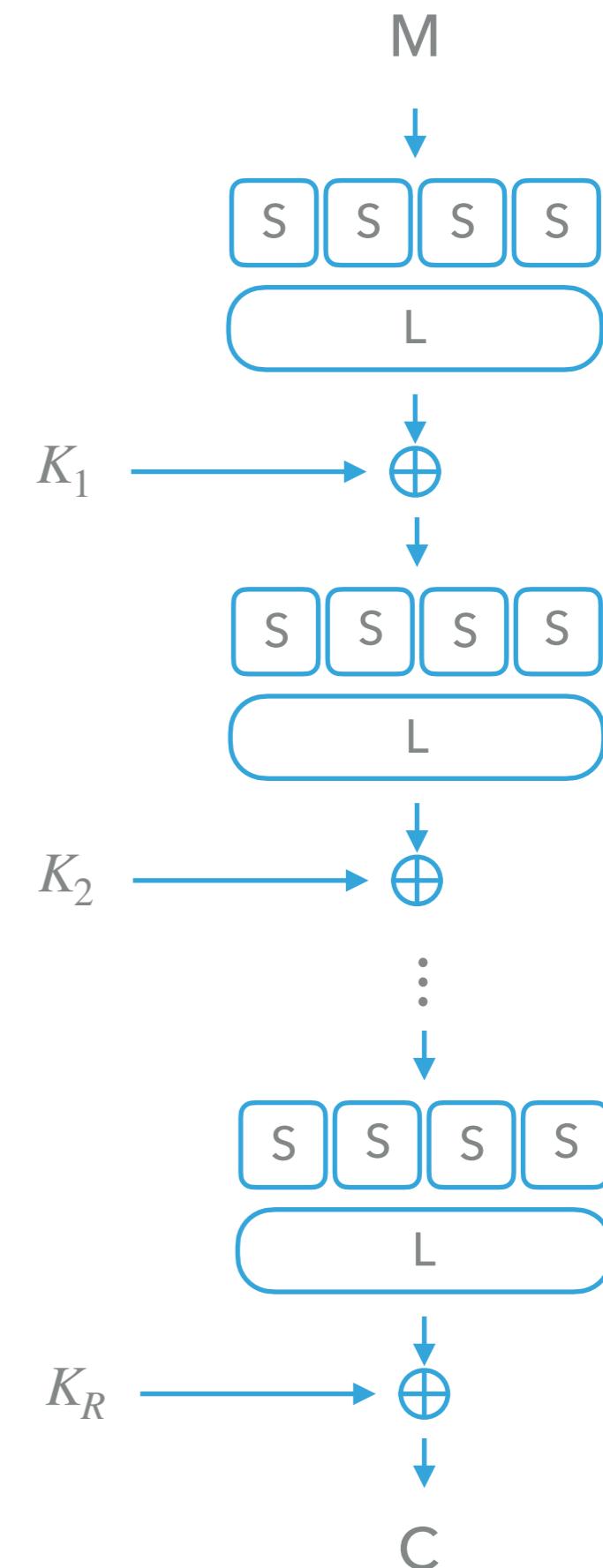
1. FEISTEL NETWORK

- ▶ F : fonction appelée fonction de tour fonctions (hautement) non-linéaires qui assurent la *confusion*
- ▶ Pour déchiffrer, il suffit de changer l'ordre des clés.
- ▶ Entre autre, F n'a pas besoin d'être inversible
- ▶ Exemple : DES (Data Encryption Standard) ~'70



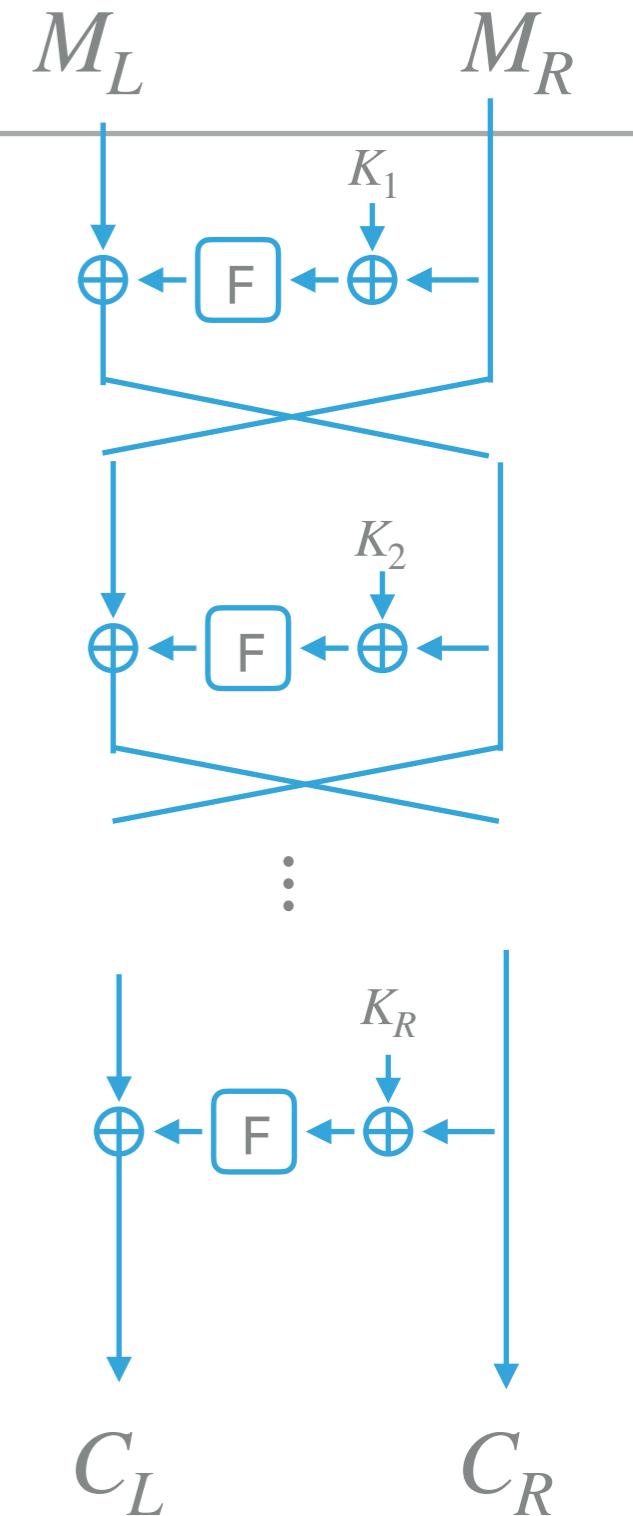
2. SUBSTITUTION PERMUTATION NETWORK

- ▶ S : fonctions appelées des **boîtes S** (ou SBoxes)
fonctions (hautement) non-linéaires
qui assurent la **confusion**
- ▶ L : fonction linéaire qui assurent la **diffusion**
- ▶ Pour déchiffrer, il faut évidemment
que toutes les opérations soient
inversibles
- ▶ Exemple : AES (Advanced
Encryption Standard) ~'00



LE SCHÉMA DE FEISTEL

- ▶ Créez une classe FEISTEL, à partir :
 - ▶ d'une fonction F (sous forme de fonction)
 - ▶ d'un nombre de tour
 - ▶ d'une fonction de cadencement de clé
- ▶ Et qui possède :
 - ▶ une méthode pour chiffrer (message, clé)
 - ▶ une méthode pour déchiffrer



PRINCIPALES ATTAQUES

- ▶ Attaques linéaires
- ▶ Attaques différentielles
- ▶ Attaques algébriques

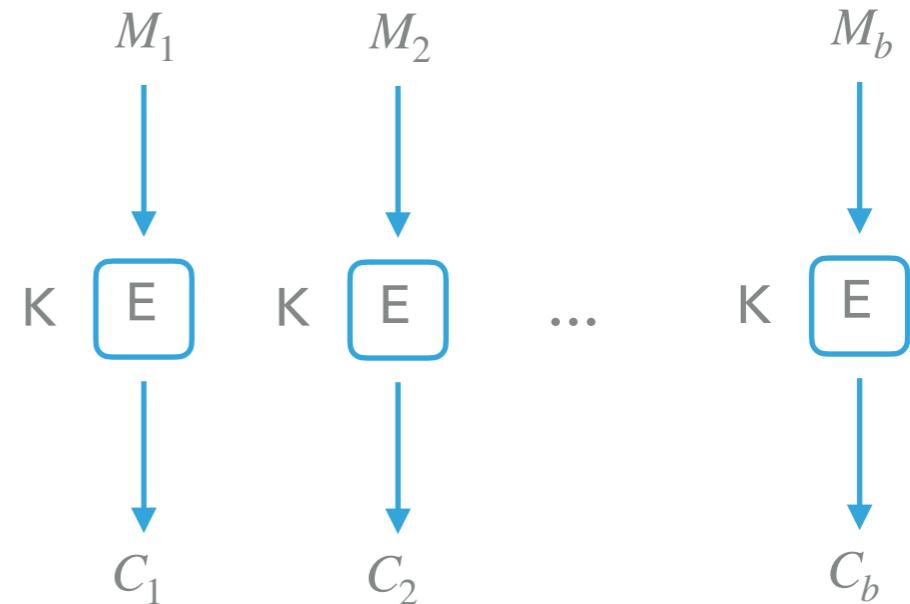
CHIFFREMENT PAR BLOC

MODES OPÉATOIRES

LES DIFFÉRENTS MODES OPÉRATOIRES

- ▶ Les messages sont de tailles plus grandes qu'un seul bloc de 80-128 bits en général
- ▶ D'où la nécessité de savoir comment les combinés entre eux
- ▶ Il existe 5 modes **principaux** (selon le NIST):
 - ▶ *Electronic Code Book Mode (ECB)*
 - ▶ *Cipher Block Chaining Mode (CBC)*
 - ▶ *Cipher Feedback Mode (CFB)*
 - ▶ *Output Feedback Mode (OFB)*
 - ▶ *Counter Mode (CTR)*

ELECTRONIC CODE BOOK MODE

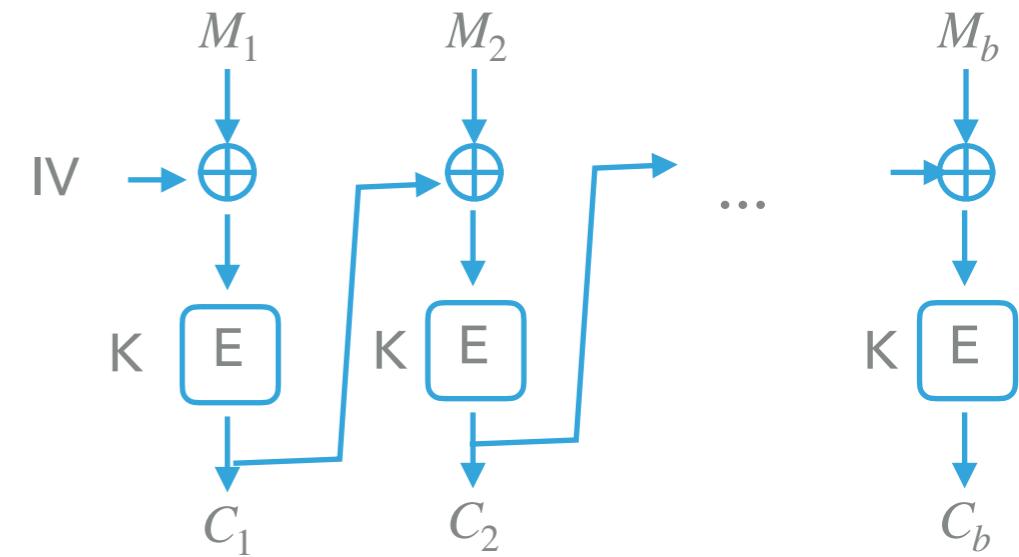


- ▶ Les erreurs ne se propagent pas
- ▶ Susceptible aux attaques statistiques
- ▶ Parallélisable
- ▶ Problème de synchronisation

CIPHER BLOCK CHAINING MODE (CBC)

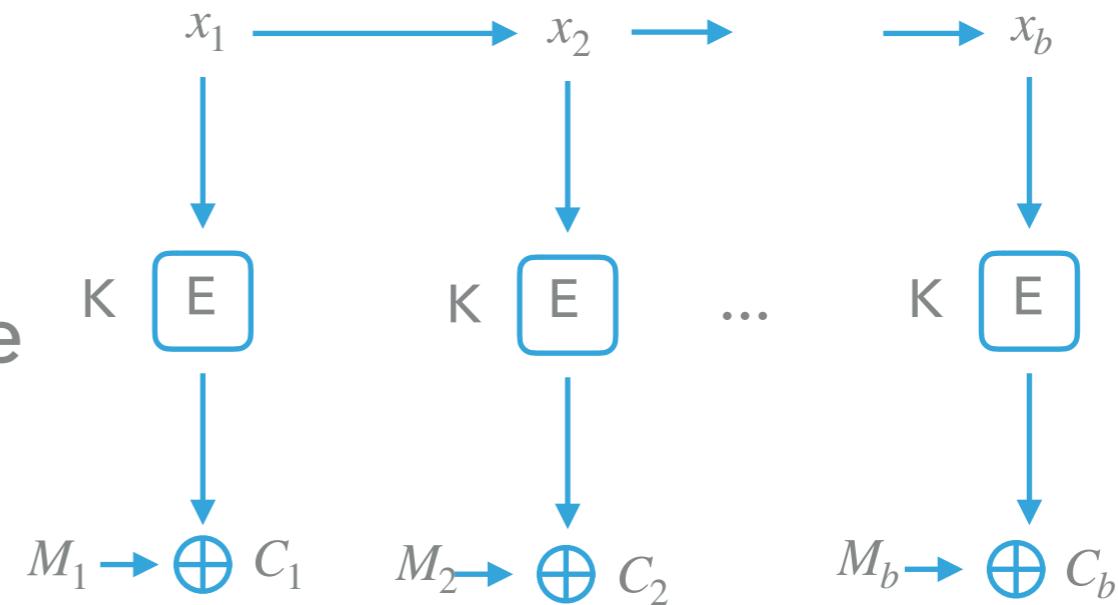
- ▶ Ne peut être parallélisé
- ▶ Un bloc manquant ou avec erreur suspend toute la procédure
- ▶ Petite fuite d'information :

$$C_i = C_j \Rightarrow m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$$



COUNTER MODE (CTR)

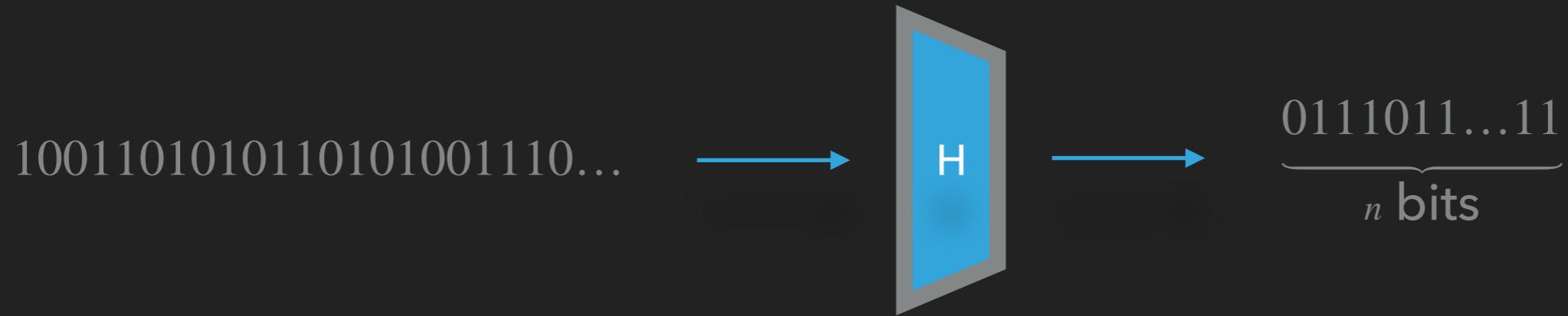
- ▶ L'idée est d'utiliser un chiffrement par bloc 'comme un' chiffrement par flot



- ▶ Les erreurs sur les messages ne se propagent pas
- ▶ Possibilité de choisir le bloc à déchiffrer
- ▶ Possibilité de penser à paralléliser

MODES OPÉRATOIRES

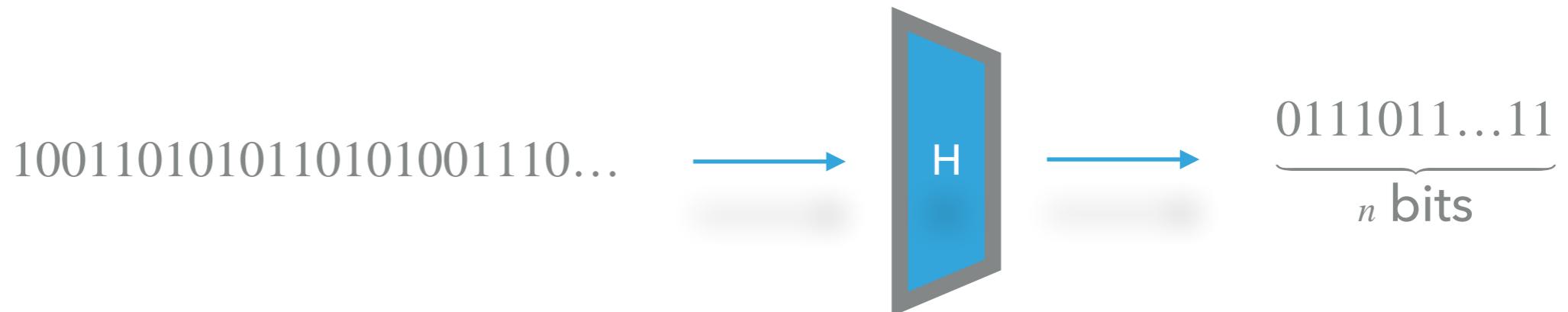
- ▶ Créer une classe CBC qui s'initialise avec :
 - ▶ Un chiffrement par bloc (typiquement une instance de FEISTEL)
- ▶ Créer une classe CTR qui s'initialise (aussi) avec :
 - ▶ Un chiffrement par bloc (typiquement une instance de FEISTEL)



CRYPTOGRAPHIE SYMÉTRIQUE

FONCTIONS DE HACHAGE

DÉFINITION



Une fonction de hachage est une fonction :

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

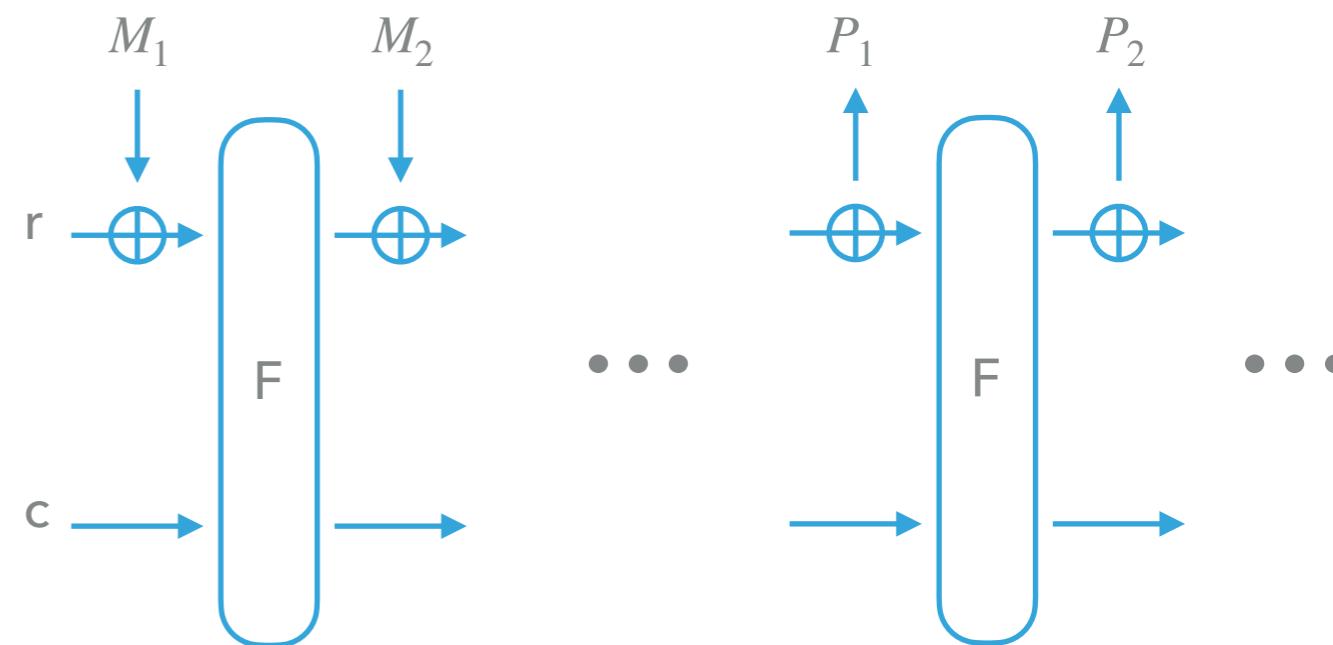
Pour des utilisations cryptographiques, on voudra que deux messages se ressemblant aient des hachés très différents.

► Exemples :

- SHA-1 (cassé! c.f. shattered.io)
- SHA-256 (ok)
- SHA-3
- MD5 (cassé)

CONSTRUCTION EN ÉPONGE

- ▶ Choisir des valeurs initiales r, c
- ▶ 2 phases :
 - ▶ Absorption
 - ▶ Extraction
- ▶ Tronquer la sortie à la longueur voulue
- ▶ Permet techniquement de construire n'importe quelle primitive symétrique



TYPES D'ATTAQUES, NIVEAU DE SÉCURITÉ

- ▶ Il existe trois types d'attaques conceptuelles :
 - ▶ Attaque par collision : Trouver deux messages tels que $M_1 \neq M_2, H(M_1) = H(M_2)$
 - ▶ Attaque en pré-image : Étant donné y , trouver x tel que $H(x) = y$
 - ▶ Attaque en seconde pré-image : Étant donné x , trouver un différent x' tel que $H(x) = H(x')$

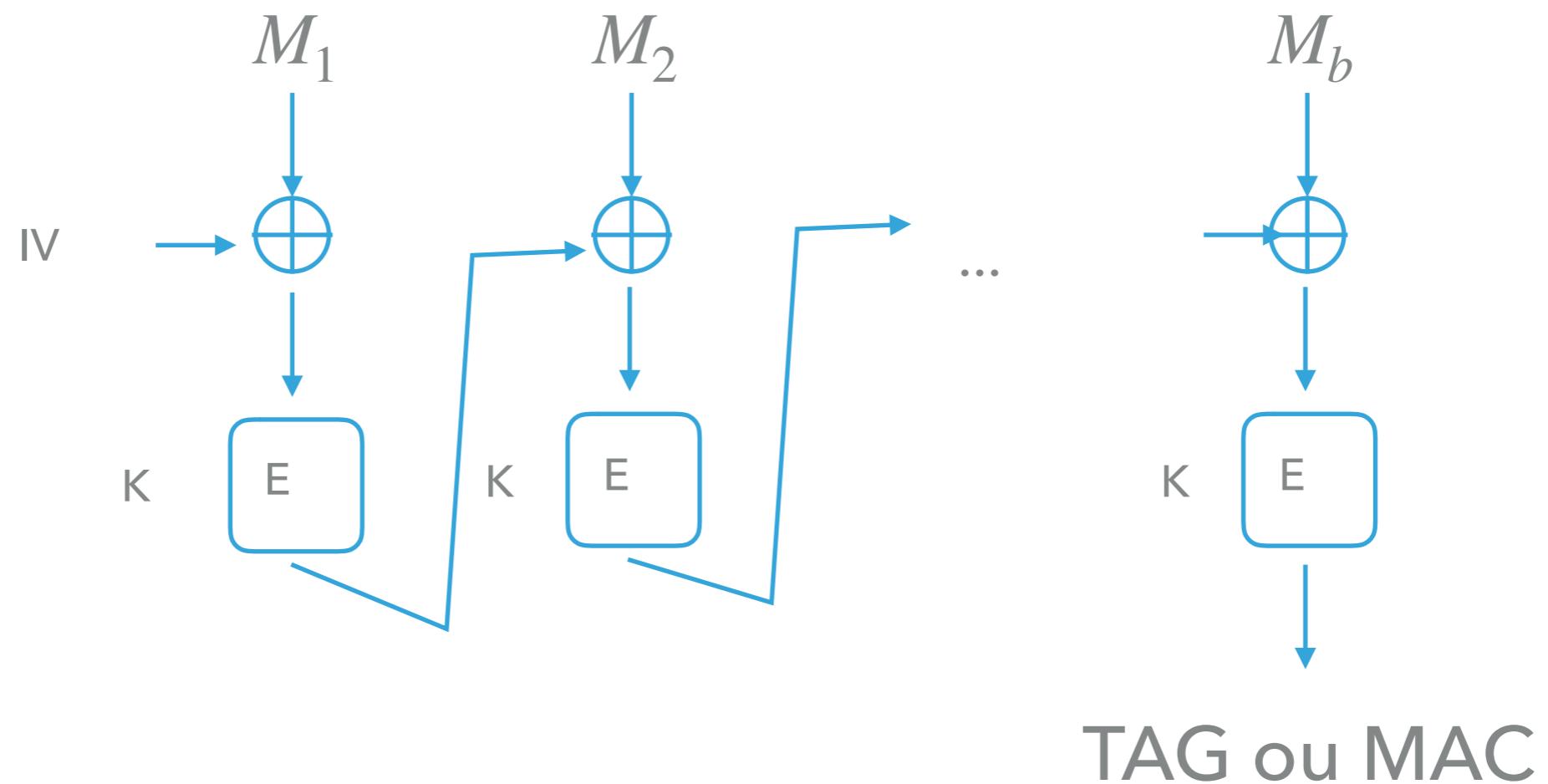
CHIFFREMENT & HACHAGE

CHIFFREMENT AUTHENTIFIÉ

CHIFFREMENT AUTHENTIFIÉ

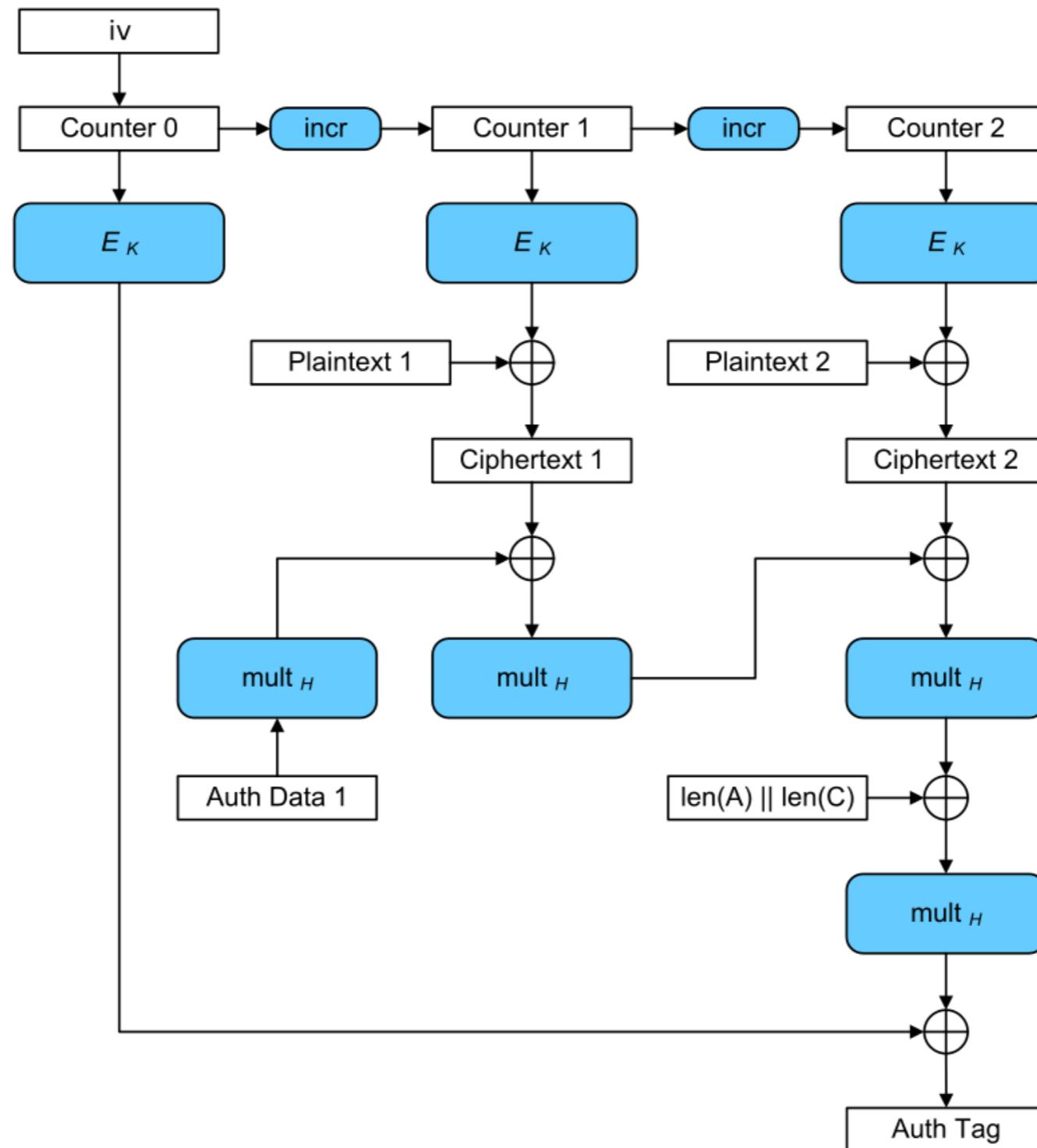
- ▶ Permet d'assurer à la fois :
 - ▶ la confidentialité
 - ▶ l'authenticité
 - ▶ l'intégrité
- ▶ Une compétition a lieu en ce moment (CAESAR) pour définir un portfolio

CBC-MAC



MAC : Message Authentication Code

GALOIS COUNTER MODE (GCM)



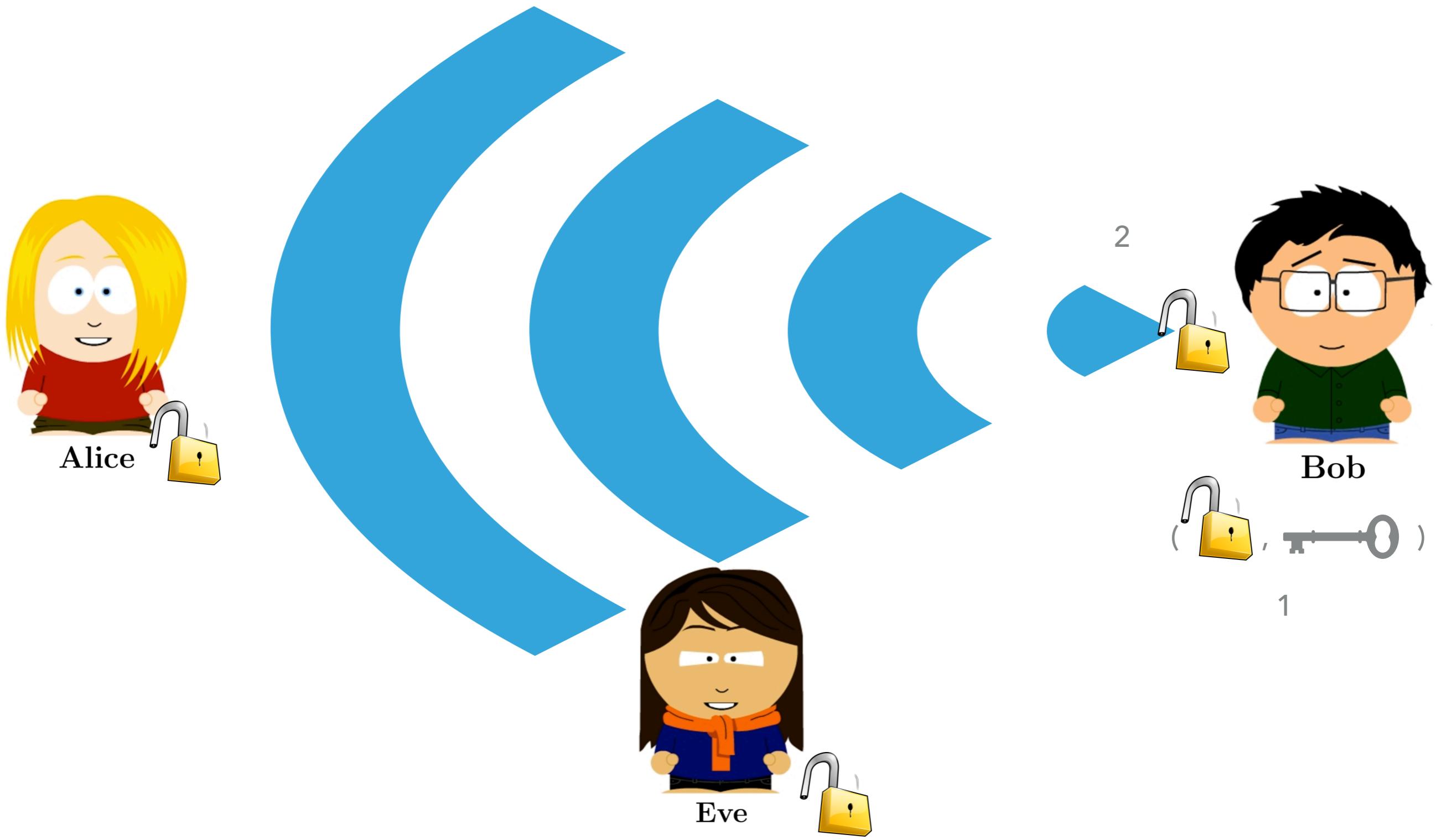
LA CRYPTOGRAPHIE ASYMÉTRIQUE



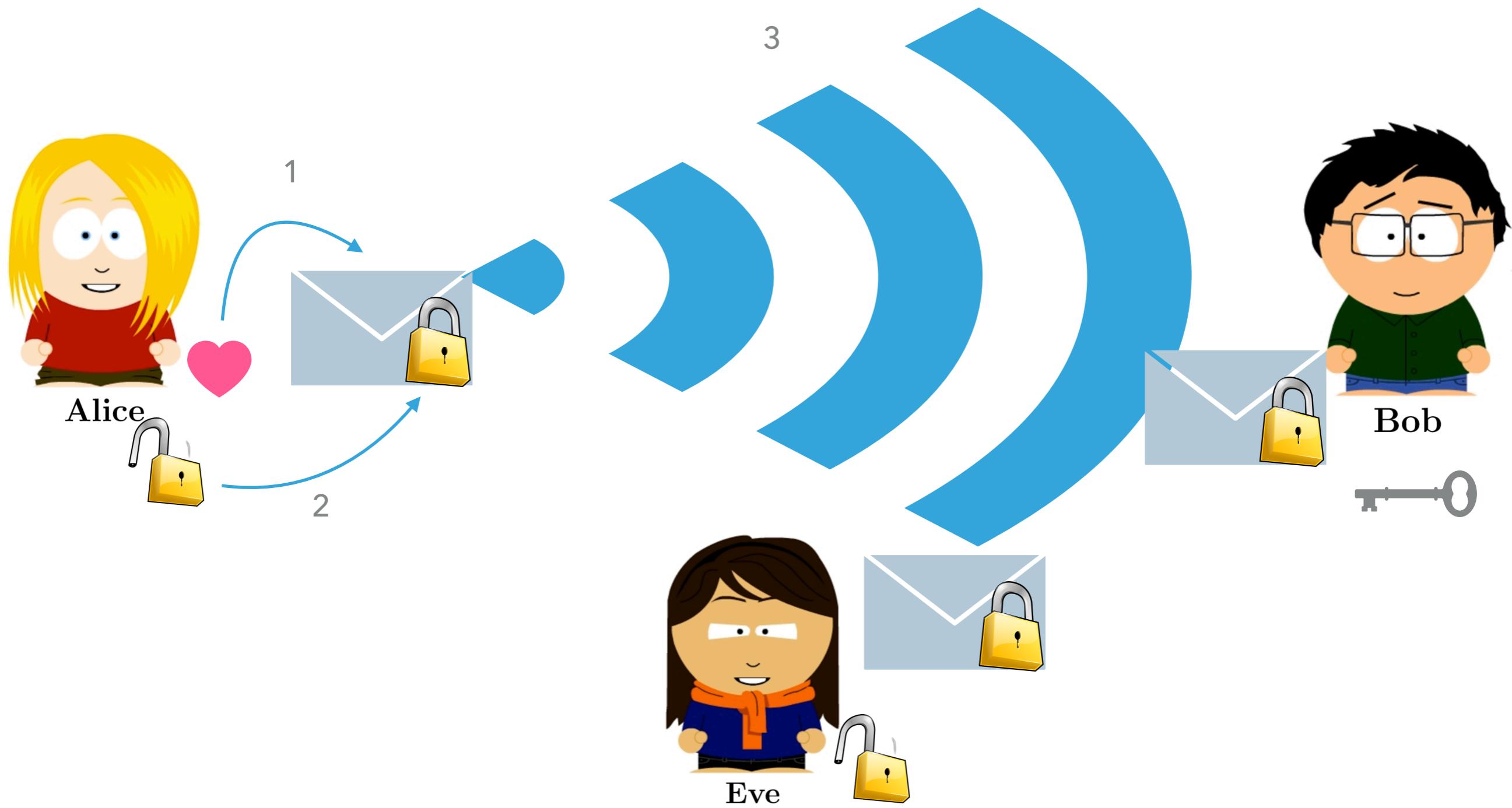
DÉFINITION :

- ▶ En opposition à la cryptographie *symétrique* où le même secret est détenu entre les différents participants, en cryptographie **asymétrique**, il existe des données **publiques** et des données **privées**.
- ▶ Née dans les années '70 suite à un besoin d'échanger des clés de chiffrement (pour OTP par exemple)

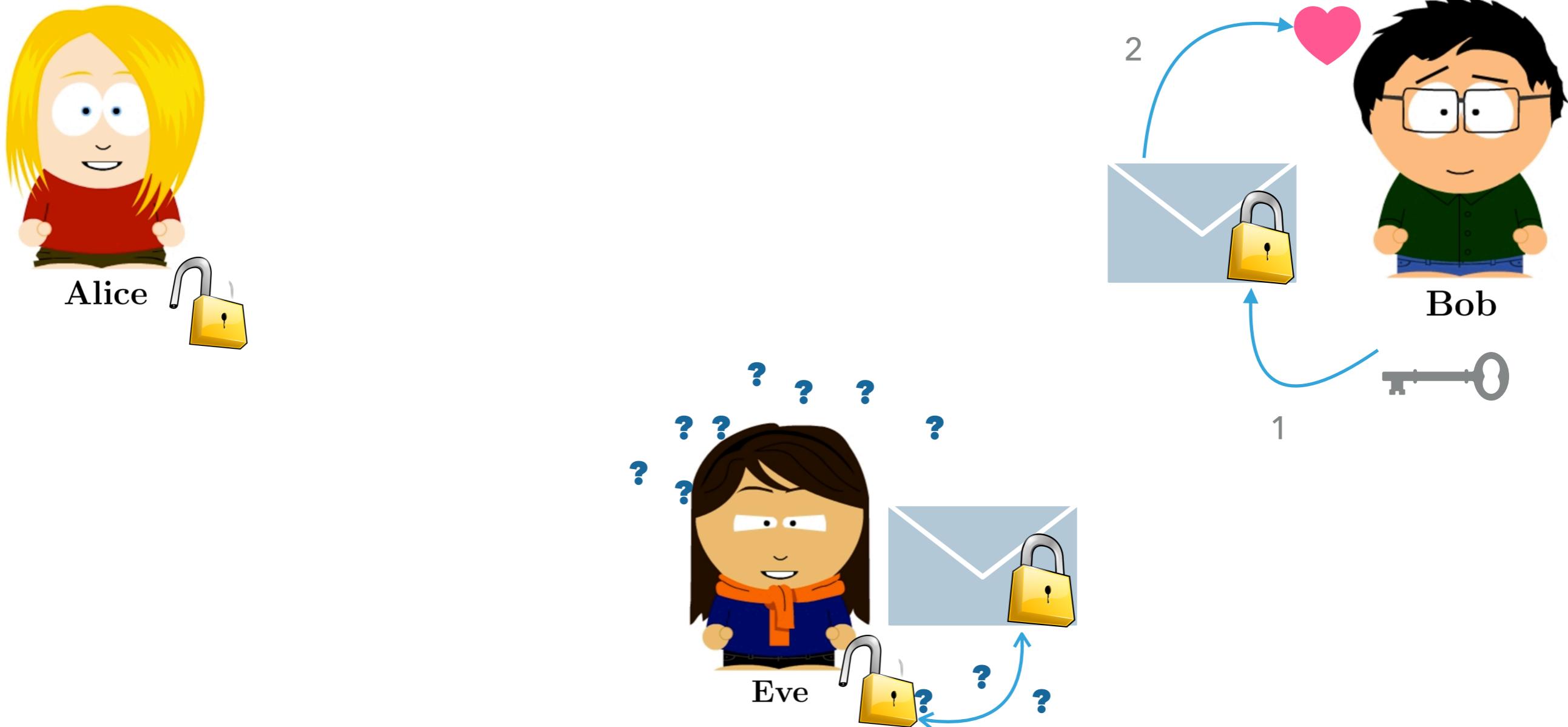
PRINCIPE : ENVOIE DES CLÉS



PRINCIPE : CHIFFREMENT



PRINCIPE : DÉCHIFFREMENT

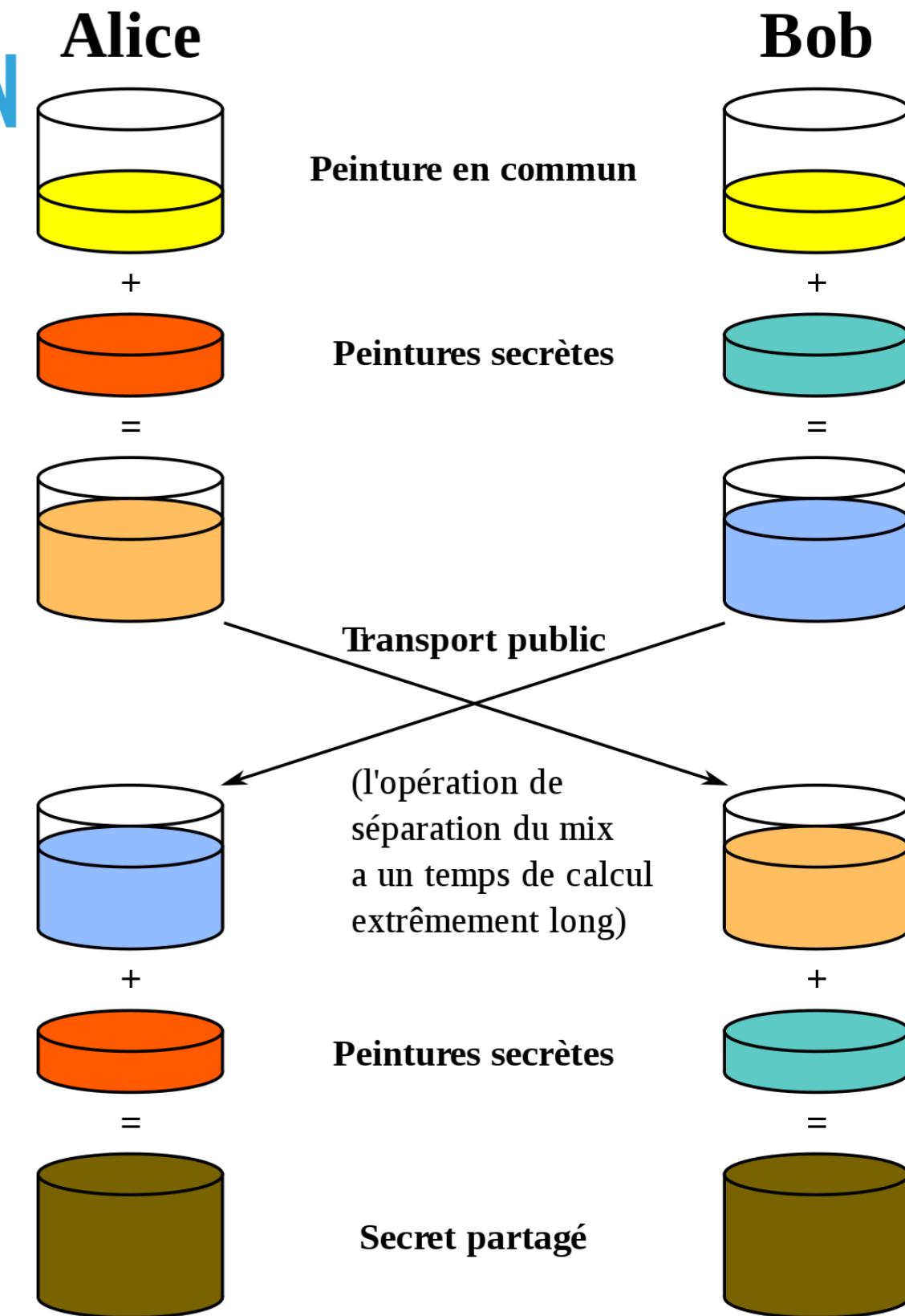


LES PROBLÉMATIQUES AUXQUEL CELA RÉPOND

- ▶ Permet d'échanger des clés de chiffrement de manière sécurisée.
- ▶ Facilite la gestion des clés de chiffrement.
- ▶ Permet de chiffrer des messages à envoyer.
- ▶ Permet de s'assurer de l'authenticité de l'expéditeur, indépendamment du message envoyé.

ÉCHANGE DE CLÉ DE DIFFIE-HELLMAN

- ▶ Alice et Bob se mettent d'accord sur un paramètre à utiliser en commun
- ▶ Alice choisit un paramètre secret pour elle.
Bob choisit un paramètre secret pour lui.
- ▶ Alice et Bob mélange le paramètre commun avec leur paramètre secret et se l'envoie mutuellement.
- ▶ Par l'ajout de leur propre secret, Alice et Bob se retrouvent avec le même secret.



RAPPEL :

ARITHMÉTIQUE MODULAIRE

NOMBRES PREMIERS

Un nombre premier est un entier positif divisible seulement par 1 et lui même.

- ▶ 1 n'est pas premier
- ▶ Le nombre de premiers $\leq n$ est à peu près $\frac{n}{\ln(n)}$
- ▶ Il existe de très bon tests pour vérifier si un nombre est premier (e.g. Test de Miller-Rabin, Fermat, etc ...)

GÉNÉRATEUR DE NOMBRES PREMIERS MILLER-RABIN

- ▶ Test de primalité de Miller-Rabin (`is_prime`)
 - ▶ Cet algorithme va faire k tests probabilistes pour savoir si un entier n est premier ou non
- ▶ Utilisez cette algorithme pour concevoir un générateur de nombre premier de taille choisie :

```
## generate_prime_number : permet de générer des entiers premiers, probabilistes
##   de longueur length bits.
def generate_prime_number(length=1024):
```

- ▶ Pensez à vérifier que vos entiers font bien length bits (!!)

MODULO

- ▶ L'opération modulo renvoie le reste de la division entière entre 2 entiers
- ▶ Exemple : $5 \equiv 2 \pmod{3}$
 - ▶ $5 = q \times 3 + 2$
 - ▶ se lit 5 est congrue à 2 modulo 3
- ▶ $a \equiv b \pmod{m} \iff a - b \equiv 0 \pmod{m}$
- ▶ Tous les entiers ne sont pas inversibles modulairement
- ▶ % en python

INVERSION MODULAIRE

- ▶ 2 nombres sont inverses l'un de l'autre si $a \times b = 1$
- ▶ 2 nombres sont inverses modulo m l'un de l'autre si
$$a \times b \equiv 1 \pmod{m}$$
- ▶ on note l'inverse de a par a^{-1}
- ▶ un nombre a est inversible modulo m ssi $\text{pgcd}(a,m)=1$
(i.e. ils sont premiers entre eux)
- ▶ Utilisez dorénavant la fonction `inv_modulo` pour réaliser vos inversions modulaires

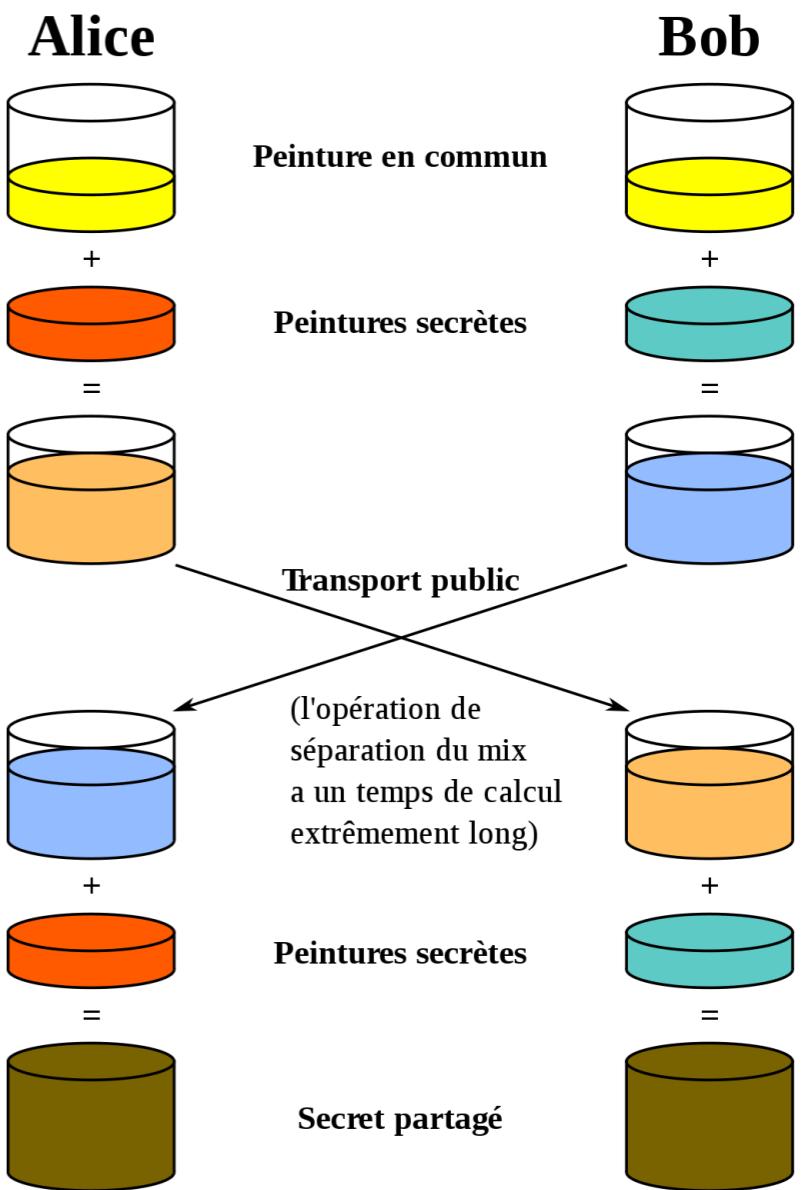
EXPONENTIATION BINAIRE

- ▶ Pour calculer x^n , vous avez a priori besoin de n multiplications
- ▶ Or $x^{a+b} = x^a \cdot x^b$ et $x^{2b} = (x^b)^2$
- ▶ Donc $3^{13} = 3^{(1101)_2} = 3^{8+4+1} = 3^8 \cdot 3^4 \cdot 3^1$
- ▶ De plus, si l'on veut $3^{13} \pmod{17}$
On calcule
$$(3^{13} \pmod{17}) \equiv (3^8 \pmod{17}) \cdot (3^4 \pmod{17}) \cdot 3^1 \pmod{17}$$
- ▶ Fonction `pow(x,n,m)` en python

EXPONENTIATION BINAIRE MODULAIRE

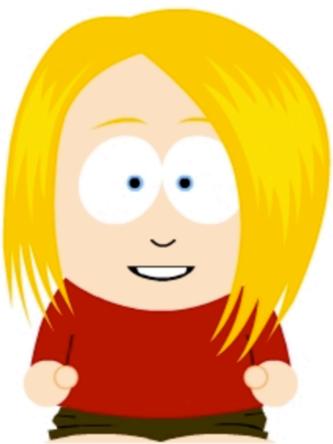
- ▶ Codez `exp_bin(x,n,m)` qui retourne $x^n \text{ mod } m$
- ▶ Si $n==0$, retournez 1
- ▶ Si $n==1$, retournez x
- ▶ Si n est pair, retournez $\text{exp_bin}(x \cdot x, n/2, m)$
- ▶ Si n est impair, retournez $\text{exp_bin}(x \cdot x, n/2, m) \times x$
- ▶ Pensez à faire modulo m (!!)

ECHANGE DE CLÉS



DIFFIE-HELLMAN

PROTOCOLE



Alice

Un premier p
et un g tel que $\text{pgcd}(g,p-1)=1$



Bob

Choisi aléatoirement : $a < p-2$

Calcule

$$A \equiv g^a \pmod{p}$$

Choisi aléatoirement : $b < p-2$

Calcule

$$B \equiv g^b \pmod{p}$$

Calcule

$$K \equiv B^a \pmod{p}$$

Calcule

$$K \equiv A^b \pmod{p}$$

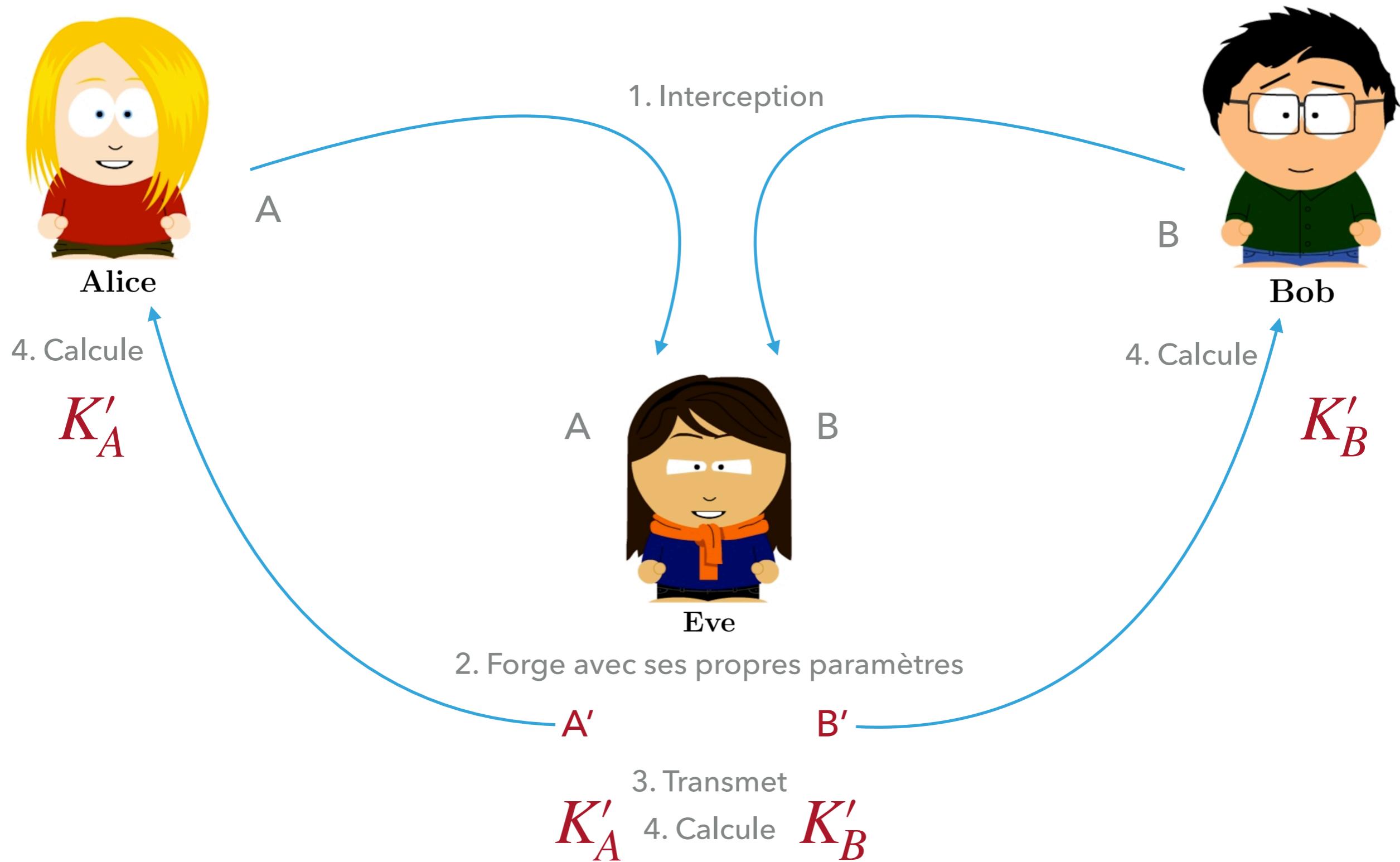
$$\left(B^a \equiv (g^b)^a \equiv g^{a \cdot b} \equiv (g^a)^b \equiv A^b \right)$$

CODONS UN PEU

PROTOCOLE DIFFIE-HELLMAN

Échangez vous des secrets en utilisant le protocole de Diffie-Hellman

ATTAQUE PAR L'HOMME DU MILIEU (MAN IN THE MIDDLE — MITM)



Eve partage une clé avec Alice et une clé avec Bob. Alice et Bob ne partagent pas de clé.

SOUVENEZ VOUS

- ▶ *Ne permet pas* de chiffrer
- ▶ Se base sur la complexité d'un problème mathématique difficile dit du ***logarithme discret*** :
à partir de $A = g^a$ et g , retrouvez a ? (très difficile en général)
- ▶ Pour se prémunir de MITM, il faut arriver à authentifier la provenance des entiers
(échange en personne (?), ou signature (cf plus loin))

CRYPTOGRAPHIE ASYMÉTRIQUE

CHIFFREMENT À CLÉ
PUBLIQUE

PRINCIPE

Être capable de créer soi-même une paire : ( , )

On peut ensuite broadcaster sans soucis



Et garder la clé  bien secrète

LE CHIFFREMENT RSA (RIVEST-SHAMIR-ADLEMAN '77)

- ▶ Générer aléatoirement deux premiers **P** et **Q** de 512 bits chacun (au moins)
- ▶ Calculer leur produit **N = PQ** (qui sera utilisé comme modulo)
- ▶ Calculer $\phi(N) = (p - 1)(q - 1)$
- ▶ Choisir **e** tel que $\text{pgcd}(e, N) = 1$ (ils sont premiers entre eux)
- ▶ Calculer **d** tel que $e \times d \equiv 1 \pmod{\phi(N)}$
- ▶ Clé publique : **(N,e)**  Clé privée : **(d,P,Q)** 

CHIFFREMENT



(N,e)

(d,P,Q) A grey key icon.

M : le message à chiffrer

C : le message chiffré

$$C \equiv M^e \pmod{N}$$

DÉCHIFFREMENT

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

PREUVE : PETIT THÉORÈME DE FERMAT

$$\forall a \text{ entier et } \forall p \text{ premier, } a^{p-1} \equiv 1 \pmod{p}$$

SOUVENEZ VOUS



Se base sur la complexité de la **factorisation**.

Sans la connaissance de **P** et **Q**, impossible de calculer **d**.

Il est possible d'utiliser d'autres problèmes mathématiques et d'autres outils (log discret, courbes elliptiques, isogénie, codes correcteurs, etc ...)

En utilisant votre propre **clé privée** pour chiffrer, vous laisser tout le monde savoir que c'est vous qui l'avait chiffré puisque tout le monde peut vérifier avec la clé publique => **Signature**

CHIFFREMENT RSA

- ▶ Générez des paires de clés RSA
- ▶ Chiffrez et déchiffrez des messages en utilisant ces clés

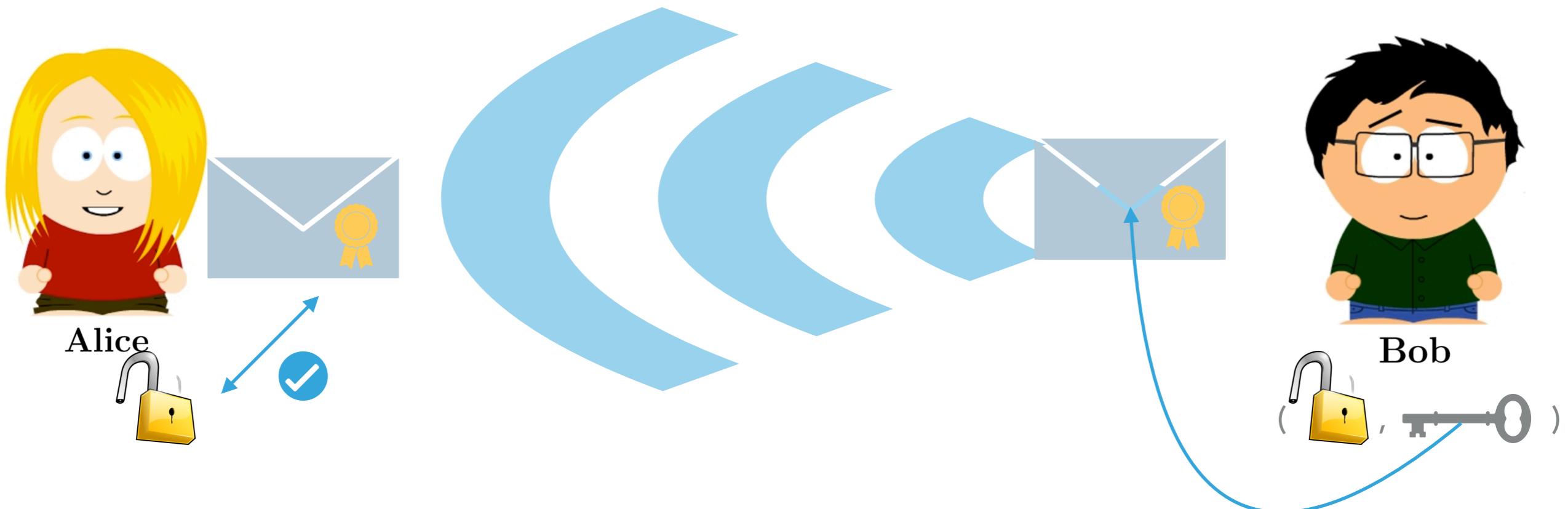
CRYPTOGRAPHIE ASYMÉTRIQUE

SIGNATURE

SIGNATURE

On utilise la **clé secrète**, privée, personnelle, en mode **chiffrement** pour «signer» le document.

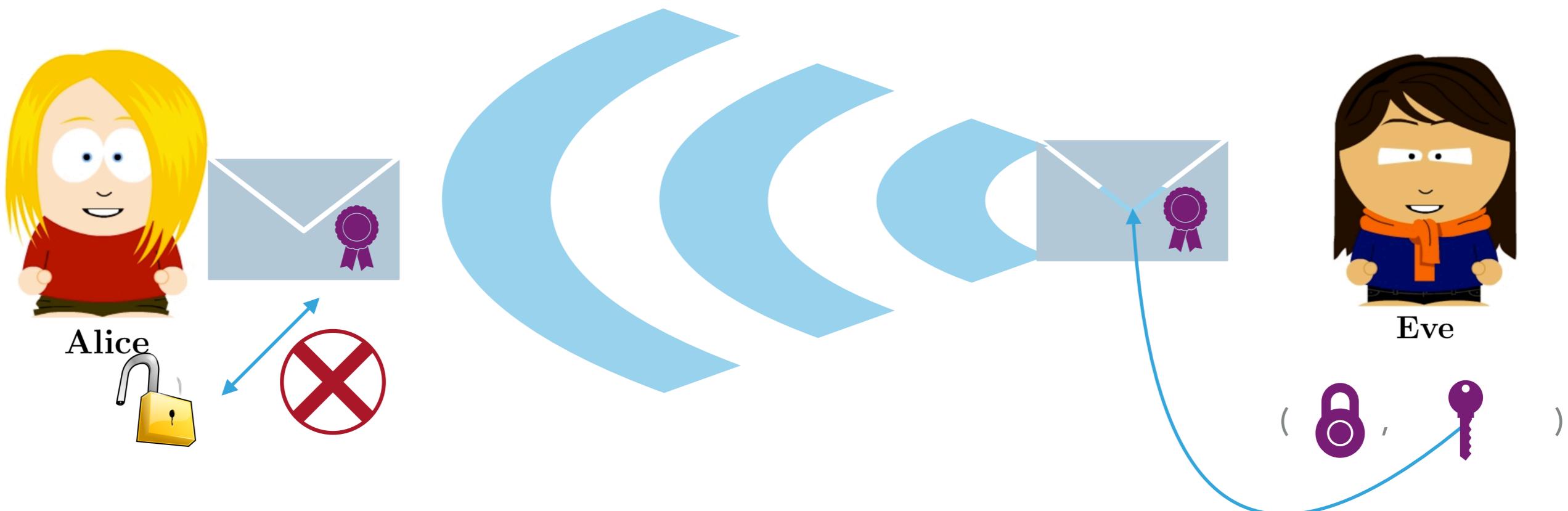
Toutes les personnes possédant la **clé publique** associée, peuvent alors l'utiliser en mode **déchiffrement** pour vérifier la **signature** et donc être sûres de la provenance du document.



SIGNATURE

On utilise la **clé secrète**, privée, personnelle, en mode **chiffrement** pour «signer» le document.

Toutes les personnes possédant la **clé publique** associée, peuvent alors l'utiliser en mode **déchiffrement** pour vérifier la **signature** et donc être sûres de la provenance du document.



SIGNATURE RSA

- ▶ Utilisez vos paires de clés RSA
- ▶ Signez et Vérifiez des messages en utilisant vos clés et celles de vos camarades