Information, Calcul et Communication

Cryptographie à clé secrète

Olivier Lévêque

École polytechnique fédérale de Lausanne

Scénario

secret K partagé (clé)

Alice

Bob

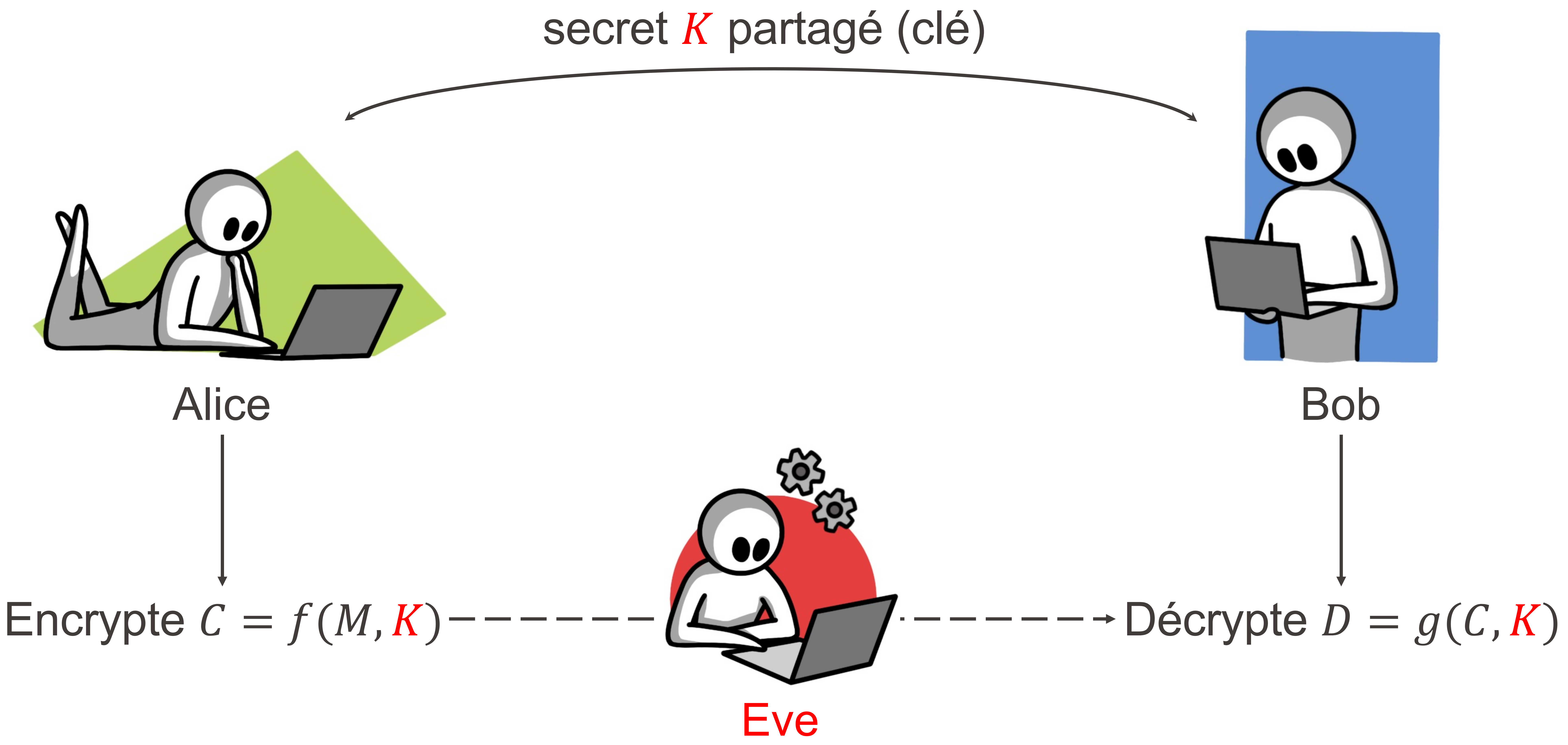
Encrypte $C = f(M, K)$ $\longrightarrow$ Décrypte $D = g(C, K)$

message chiffré

message d'origine

message déchiffré $\to M$

Scénario



Eve intercepte C mais ne sait pas trop quoi en faire sans la clé K…

Clé à usage unique *(«one-time pad»)*

$\begin{cases} 0 \oplus 0 = 1 \oplus 1 = 0 \\ 0 \oplus 1 = 1 \oplus 0 = 1 \end{cases}$

- Voici une recette 100% sûre pour encrypter un message M de n bits :

Pour cela, il faut supposer que la clé secrète K est composée également de n bits **et** générée de la façon suivante :

Chacun des bits K_i est tiré uniformément au hasard
(i.e. $P(K_i = 1) = P(K_i = 0) = {}^1/_2$)
et tous les tirages sont effectués indépendamment,
et aussi indépendamment du message M.

- Alice envoie alors $C = M \oplus K$ (XOR bit par bit : pas de retenue ici)
  - Exemple : Si $M = 01101101$ et $K = 11101100$, alors $C = 10000001$

$+ \; K = 11101100$

$C = 10000001$

pas de retenue

Bob effectue : $C \oplus K = M$

Clé à usage unique *(«one-time pad»)*

- Pour décrypter le message, Bob effectue l'opération :

$$D = C \oplus K = (M \oplus K) \oplus K = M \oplus \underbrace{(K \oplus K)}_{=\text{séquence de } 0\,!} = M$$

- Si Eve intercepte C, elle ne peut rien faire car :

$$P(C_i = 0) = P(M_i \oplus K_i = 0) = P(K_i = M_i) = \frac{1}{2} \quad \forall M_i$$

$$P(C_i = 1) = P(M_i \oplus K_i = 1) = P(K_i \neq M_i) = \frac{1}{2} \quad \forall M_i$$

- Pour Eve, le message C est une séquence de bits tirés uniformément au hasard ! → Le système est sûr à 100% !

Défauts de ce système

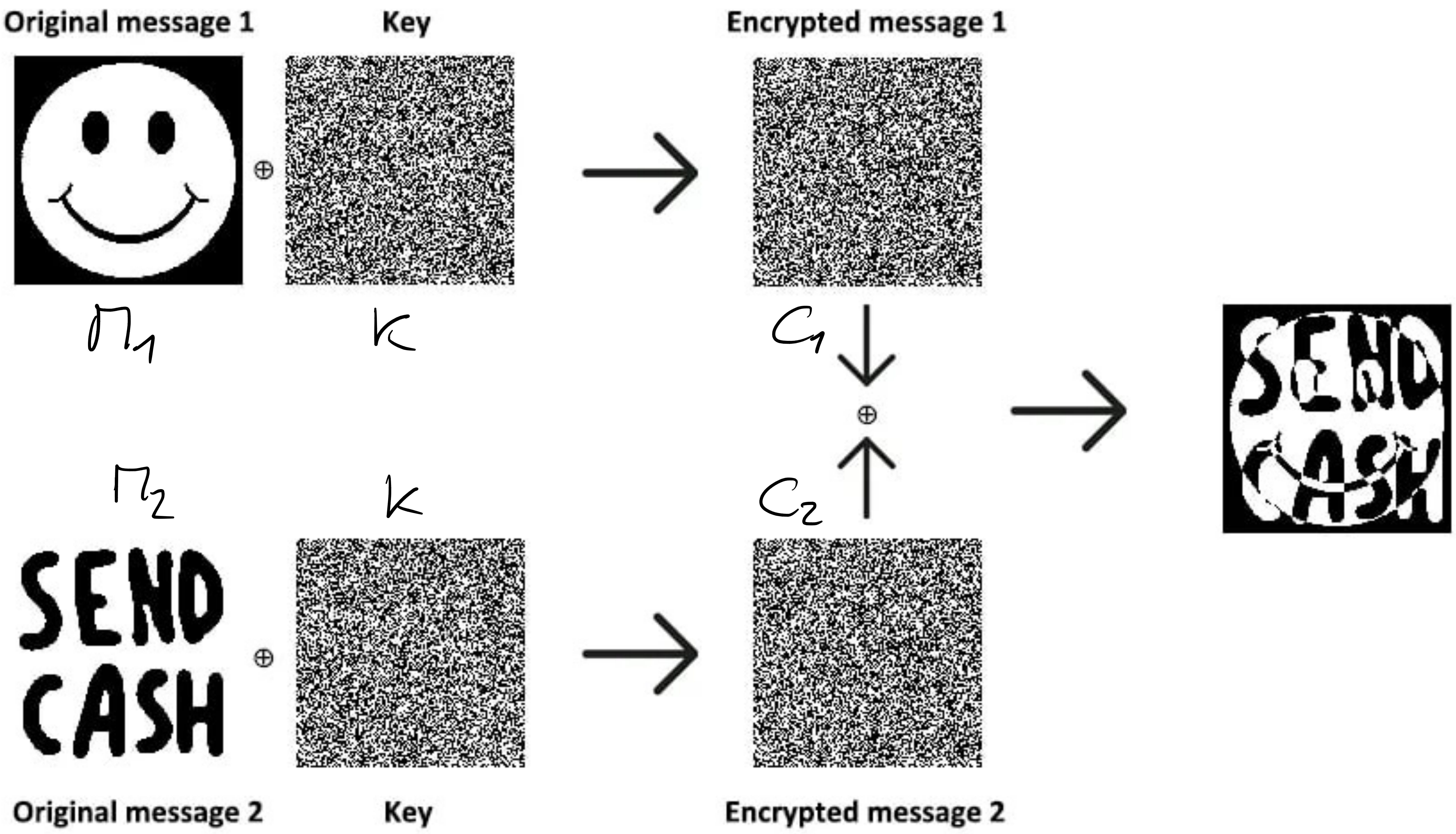
- Pour envoyer un message de longueur n, il faut générer une clé de même longueur, qu'il faut trouver le moyen de partager secrètement avant de communiquer…
- Si la clé n'est pas générée parfaitement aléatoirement, alors le secret n'est plus assuré à 100% (imaginez par exemple que $K = 00000000$, dans le pire des cas…)
- Gare à ne pas réutiliser la même clé K pour envoyer deux messages M_1 et M_2 à la suite avec ce système !

  A partir de $C_1 = M_1 \oplus K$ et $C_2 = M_2 \oplus K$, Eve peut effectuer :

$$C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = (M_1 \oplus M_2) \oplus (K \oplus K) = M_1 \oplus M_2$$

$\rightarrow$ plus aucune garantie de sécurité !

Défauts de ce système



2 points de vue:

1°) on veut réutiliser une clé K

pour envoyer plusieurs messages $M_1, M_2, \ldots$

2°) on a une clé K d'une certain longueur (56 bits)

on veut envoyer un message M d'160.

Premier essai pour réutiliser la clé k :

Comme de tout le monde !

$$\left\{\begin{array}{l} \text{Soit } f : \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n \\ \qquad\qquad k \quad , \quad M \quad \longmapsto f(k,M) \\ \qquad \text{"non-linéaire"} \end{array}\right.$$

Alice calcule $C = M \oplus f(k,M)$

$\rightarrow$ envoie $C_1 = M_1 \oplus f(k,M_1)$, $C_2 = M_2 \oplus f(k,M_2)$

si Eve intercepte $C_1, C_2 \Longrightarrow$? mais pas ok pour Bob non plus !

Un essai pour réutiliser la clé K :

Data Encryption Standard *(DES)*, 1976

Principe (et seulement le principe, pas tout le système) :

On suppose que le message M et la clé K sont l'un et l'autre de longueur $2n$ bits et on décompose ceux-ci en :

$$M = (\underbrace{M_a}_{n\text{ bits}}, \underbrace{M_b}_{n\text{ bits}}), \quad K = (\underbrace{K_a}_{n\text{ bits}}, \underbrace{K_b}_{n\text{ bits}})$$

(NB: en pratique $n = 32$ bits)

Soit f: $\{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n$ une fonction **non-linéaire**.

$$K \quad , \quad M \quad \longmapsto f(K, M)$$

Note : Si on utilisait $f(K, M) = K$, le système décrit à la page suivante ne serait rien d'autre que le «one-time pad» décrit avant.

Data Encryption Standard

Eve → ?

- Alice calcule ***successivement*** :

$$C_a = M_a \oplus f(K_a, M_b) \quad \text{puis} \quad C_b = M_b \oplus f(K_b, C_a)$$

et envoie $C = (C_a, C_b)$.

Bob calcule à son tour : $\begin{cases} D_b = C_b \oplus f(K_b, C_a) \\ D_a = C_a \oplus f(K_a, D_b) \end{cases}$

$$D_b = (M_b \oplus f(K_b, C_a)) \oplus f(K_b, C_a) = M_b \oplus (\underbrace{f(K_b, C_a) \oplus f(K_b, C_a)}_{=0}) = M_b$$

$$D_a = C_a \oplus f(K_a, M_b) = (M_a \oplus \underbrace{f(K_a, M_b)) \oplus f(K_a, M_b)}_{=0} = M_a$$

Data Encryption Standard

- Alice calcule ***successivement*** :

$$C_a = M_a \oplus f(K_a, M_b) \quad \text{puis} \quad C_b = M_b \oplus f(K_b, C_a)$$

$$\text{et envoie} \quad C = (C_a, C_b).$$

- Comment Bob peut-il déchiffrer ce message ? Tout simplement, en fait ! Il refait **les mêmes opérations dans l'ordre inverse** :

$$D_b = C_b \oplus f(K_b, C_a) \quad \text{puis} \quad D_a = C_a \oplus f(K_a, D_b)$$

- Ainsi, Bob retrouve M, car :

$$\begin{cases} D_b = C_b \oplus f(K_b, C_a) = M_b \oplus f(K_b, C_a) \oplus f(K_b, C_a) = M_b \\ D_a = C_a \oplus f(K_a, \underbrace{D_b}_{= M_b}) = M_a \oplus f(K_a, M_b) \oplus f(K_a, M_b) = M_a \end{cases}$$

En pratique

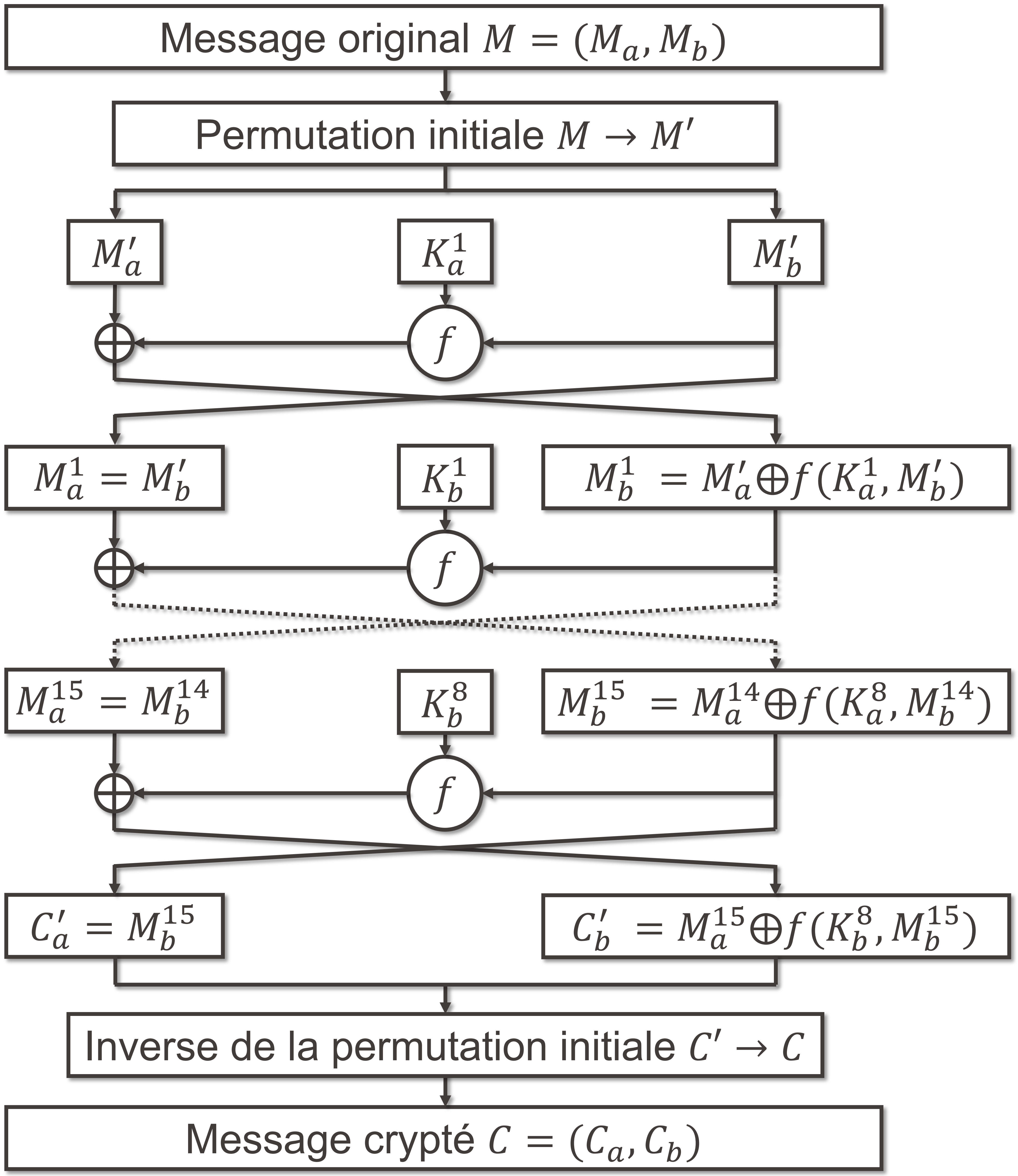
- Le système DES applique le principe ci-dessus **8 fois de suite** (avec la même clé K réutilisée 8 fois, mais avec des décalages) pour chiffrer un message, avec en plus une permutation du message à l'entrée et à la sortie (voir illustration page suivante).

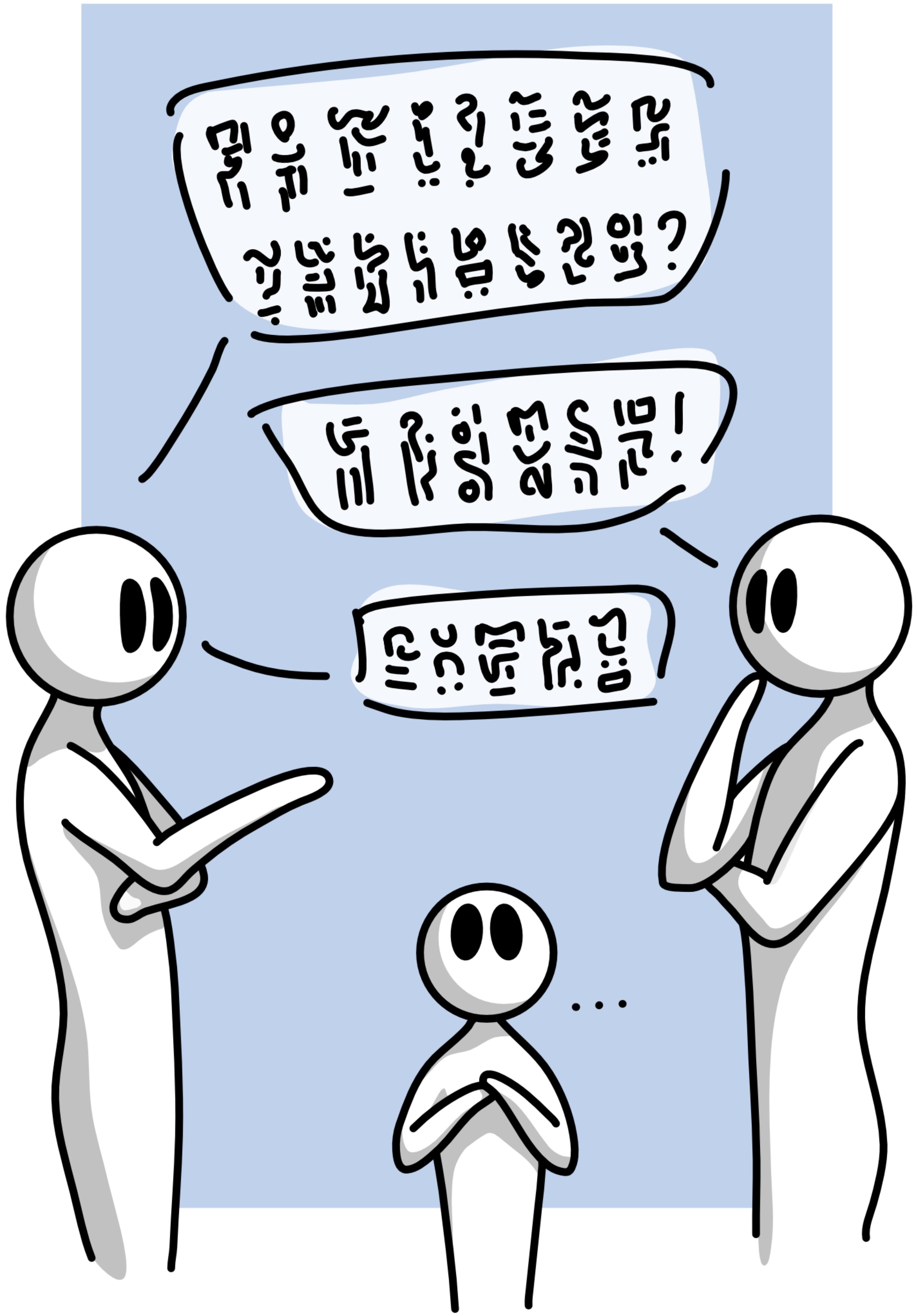
- Ainsi, Alice peut chiffrer plusieurs messages $M_1, M_2, \ldots$ avec la même clé K et espérer que même si Eve intercepte $C_1, C_2, \ldots$ elle ne sera pas capable de retrouver les messages $M_1, M_2, \ldots$ sans la clé K.

- Pourtant, ce système de chiffrement a été « cassé » pour la première fois en 1999 et remplacé depuis par le système AES (Advanced Encryption Standard).

Système DES complet

(Notez que le chiffrement et le déchiffrement s’effectuent à nouveau avec les mêmes opérations !)





Information, Calcul et Communication

Cryptographie à clé publique

Olivier Lévêque

École polytechnique fédérale de Lausanne

Introduction

Alice et Bob désirent échanger des informations de manière confidentielle (et/ou envoyer des messages authentifiés) **sans** disposer d'une clé secrète K échangée au préalable.

(Comment) est-ce possible ?

- **Réponse 1**

  Ça n'est pas possible !

- **Réponse 2**

  C'est possible en pratique grâce aux **opérations difficilement inversibles** ou « **opérations à sens unique** ».

Protocole: version simplifiée

Exemple : Supposons qu'il soit facile de multiplier mais difficile de diviser !

Alice

N_1

Bob

N_2

$N_3 = 426$

$N_4 = 36210$

$N_5 = 52398$

$K = 4453830$

Protocole: version simplifiée

Exemple : Supposons qu'il soit facile de multiplier mais difficile de diviser !

- Alice choisit un nombre N_1
  Bob choisit un nombre N_2
- Ils se mettent d'accord sur un 3ème nombre M (public)
- Alice calcule $N_3 = N_1 \cdot M$ de son côté et envoie N_3 à Bob
  Bob calcule $N_4 = N_2 \cdot M$ de son côté et envoie N_4 à Alice
- Alice calcule ensuite $N_1 \cdot N_4 = N_1 \cdot N_2 \cdot M$
  Et Bob calcule $N_2 \cdot N_3 = N_2 \cdot N_1 \cdot M$

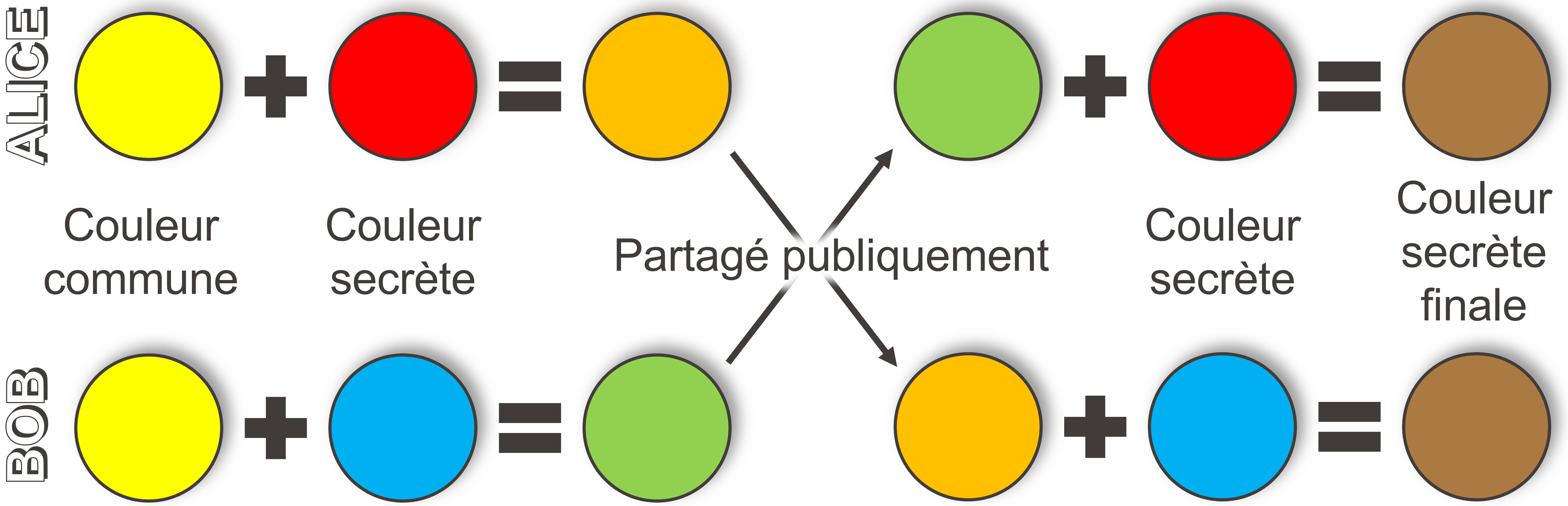
  égaux = secret partagé

Alice et Bob sont ainsi tombés d'accord sur un même nombre.

- Si Eve intercepte N_3 ou N_4 (ou même les deux nombres) et qu'elle ne sait pas diviser, elle ne peut pas retrouver le secret.

La même chose avec des couleurs

- Les couleurs sont facilement mélangeables, mais difficilement séparables



Arithmétique modulaire

- Bien sûr, diviser est une opération facile en réalité :

Eve peut donc effectuer N_3/M et retrouver N_1, de même que N_2.
Par conséquent elle peut calculer le « secret » $N_1 \cdot N_2 \cdot M$.

Il faut donc trouver autre chose !

- Soit P un **grand** nombre premier. Sur l'ensemble $\{0,1,\ldots,P-1\}$, on définit l'addition, la multiplication et l'exponentiation **modulo P** :
  (toutes des opérations faciles à exécuter)

$$N_1 + N_2 \ (mod\ P) \qquad N_1 \cdot N_2 \ (mod\ P) \qquad N_1^{N_2} \ (mod\ P)$$

Exemples : $4 + 3 \ (mod\ 5) = 2$ $\quad 4 \cdot 3 \ (mod\ 5) = 2$ $\quad 4^3 \ (mod\ 5) = 4$
(avec $P = 5$)

raisonnablement
facile à faire

Arithmétique modulaire

P un grand nombre premier

- Il se trouve que l'opération $N_1^{N_2}(mod\ P) = N_3$ est difficile à **inverser** (i.e., si on nous donne P, N_1 et N_3, il est difficile de retrouver N_2).

Voilà donc **l'opération à sens unique** que nous allons utiliser.

Si P a n chiffres, on ne connaît pas d'algorithme de complexité polynomiale en n qui résout ce problème.

Arithmétique modulaire

tester toutes les possibilités
mais il y a 10^n possibilités

- Il se trouve que l'opération $N_1^{N_2}(mod\ P) = N_3$ est difficile à **inverser** (i.e., si on nous donne P, N_1 et N_3, il est difficile de retrouver N_2).

Voilà donc **l'opération à sens unique** que nous allons utiliser.

Remarques

- La difficulté de l'opération d'inversion dépend des nombres P et N_1 choisis (c'est un long chapitre : celui du **logarithme discret**).

classiquement : $N_1^{N_2} = N_3 \longrightarrow N_2 = \log_{N_1}(N_3) = \frac{\log_2(N_3)}{\log_2(N_1)}$

Arithmétique modulaire

- Il se trouve que l'opération $N_1^{N_2} (mod\ P) = N_3$ est difficile à **inverser** (i.e., si on nous donne P, N_1 et N_3, il est difficile de retrouver N_2).

Voilà donc **l'opération à sens unique** que nous allons utiliser.

Remarques

- La difficulté de l'opération d'inversion dépend des nombres P et N_1 choisis (c'est un long chapitre : celui du **logarithme discret**).
- Il existe par contre des algorithmes efficaces pour trouver de grands nombres premiers et donc réaliser concrètement ce qui va suivre !

Protocole de Diffie-Hellman : (nbs connus de tous)

publique

1°) Alice et Bob se mettent d'accord sur un grand nombre premier P et $1 < Q < P-1$
(à n chiffres, avec $n = 100$)

secret

2°) Alice choisit N_1 un nombre à n chiffres

Bob choisit N_2 un autre nombre à n chiffres

3°) Alice calcule $N_3 = Q^{N_1} \pmod{P}$ et transmet N_3 à Bob

Bob calcule $N_4 = Q^{N_2} \pmod{P}$ et transmet N_4 à Alice

publique

4°) ~~Bob calcule $N_3 \cdot N_2$ et Alice calcule $N_4 \cdot N_1$~~

~~Pb : $N_3 \cdot N_2 \neq N_4 \cdot N_1$~~

Bob calcule $N_3^{N_2} \pmod{P}$

Alice calcule $N_4^{N_1} \pmod{P}$

<u>En effet</u> :

- $N_3^{N_2} \pmod{P} = \left(Q^{N_1} \pmod{P}\right)^{N_2} \pmod{P}$
  $= \left(Q^{N_1}\right)^{N_2} \pmod{P} = Q^{N_1 N_2} \pmod{P} = K$
- $N_4^{N_1} \pmod{P} = \left(Q^{N_2} \pmod{P}\right)^{N_1} \pmod{P}$
  $= \left(Q^{N_2}\right)^{N_1} \pmod{P} = Q^{N_2 N_1} \pmod{P} = K$

Protocole d'échange de clé de Diffie-Hellman

Utilisons cette opération à sens unique pour l'échange d'une clé secrète entre Alice et Bob :

- Alice et Bob choisissent d'abord ensemble un grand nombre premier P à n chiffres, ainsi qu'un autre nombre Q entre 1 et $P-1$.
  (P et Q sont donc publics)
- Alice choisit un nombre N_1 entre 1 et $P-1$
  Bob choisit un nombre N_2 entre 1 et $P-1$
  secrètement
- Alice effectue $N_3 = Q^{N_1}(mod\ P)$ et publie N_3
  Bob effectue $N_4 = Q^{N_2}(mod\ P)$ et publie N_4
- Alice effectue ensuite $N_4^{N_1}(mod\ P) = (Q^{N_2})^{N_1}(mod\ P) = Q^{N_1 \cdot N_2}\ (mod\ P) = K$
  Bob effectue quant à lui $N_3^{N_2}(mod\ P) = (Q^{N_1})^{N_2}(mod\ P) = Q^{N_1 \cdot N_2}\ (mod\ P) = K$

Conclusion

- Alice et Bob ont ainsi trouvé une clé secrète commune K (= un grand nombre = une longue suite de bits).

- Notez également qu'Alice ne connaît toujours pas N_2, ni Bob ne connaît N_1.

- A partir de P, Q, N_3 et N_4, il est difficile pour Eve de retrouver N_1 ou N_2, car il faudrait inverser

$$N_3 = Q^{N_1}(mod\ P) \text{ ou } N_4 = Q^{N_2}(mod\ P)$$

  et donc Eve n'a pas non plus accès au secret K.