Introduction à la Cryptographie

2. Chiffrement symétrique

Cécile Pierrot, Chargée de Recherche INRIA Nancy
cecile.pierrot@inria.fr

Supports de E. Thomé

Telecom Nancy, 2A ISS – 2021

Plan

Penser à l'attaquant

On distingue plusieurs types d'attaque :

- Selon le matériau disponible :
  - attaque à chiffré seul
  - attaque à clair connu
  - attaque à clair choisi (adaptative ou non)
    (toujours possible pour les chiffrements à clé publique)
  - attaque à chiffré choisi (adaptative ou non)
  - attaque à clés liées
- Selon l'accès au matériel chiffrant disponible :
  - cryptanalyse par canaux auxiliaires
  - cryptanalyse par injection de fautes

Qu'est-ce qu'un système sûr ?

Définition (Sécurité inconditionnelle)

Confidentialité parfaite [Shannon, 1949]
La connaissance du message chiffré n'apporte aucune information sur le message clair : la seule attaque possible est la recherche exhaustive.
Pour qu'un chiffrement soit inconditionnellement sûr, il faut que la clé soit aléatoire et aussi longue que le texte clair.

C'est une sort de ≪modèle idéal≫, largement irréaliste.

Sécurité pratique

Les systèmes utilisés dans la pratique sont théoriquement cassables

Définition (Sécurité pratique, ou calculatoire)

La connaissance du message chiffré (et de certains couples clairs-chiffrés) ne permet de retrouver ni la clé ni le message clair en un temps humainement raisonnable.

Complexité d'une attaque : ordres de grandeur

$$\text{Difficulté de l'attaque} = O(\text{Temps} + \text{Mémoire} + \text{Données})$$

- On estime que requérir 2^{128} opérations représente aujourd'hui un niveau raisonnable de sécurité (≪limite de l'infaisable≫)
- Jusqu'au début des années 2000, on fixait plus couramment cette limite à 2^{80}.

Attention à la parallélisation des algorithmes.

Complexité d'une attaque : ordres de grandeur

1000 coeurs à 4GHz pendant 1 an : $\approx 2^{67}$ cycles.

n	2^n	**Exemples**
32	2^{32}	nombre d'êtres humains sur Terre
46	2^{46}	distance Terre - Soleil en millimètres nombre d'opérations effectuées par jour par un ordinateur à 1 GHz
55	2^{55}	nombre d'opérations effectuées par an par un ordinateur à 1 GHz
82	2^{82}	masse de la Terre en kilogrammes
90	2^{90}	nombre d'opérations effectuées en 15 milliards d'années (âge de l'univers) par un ordinateur à 1 GHz
155	2^{155}	nombre de molécules d'eau sur Terre
256	2^{256}	nombre d'électrons dans l'univers

Niveau de sécurité : taille des clés

- Clé symétrique : la taille des clés est souvent de 128 bits (AES)
- Clé asymétrique : la taille des clés est calculée de manière à offrir une sécurité supérieure à 2^{128}
  ex. 3072 bits pour un module RSA
- Problème pratique : plus la taille des clés augmente, plus les algorithmes sont lents surtout en cryptographie asymétrique

Mécanismes de chiffrement

- Chiffrement symétrique
  - chiffrement par flot : A5/1, RC4, Snow3G, etc.
  - chiffrement par bloc : DES, Blowfish, AES, etc.
  - modes de chiffrement : ECB, CBC, CFB, OFB, CTR, etc.
- Chiffrement asymétrique
  - fondé sur la difficulté de la factorisation : RSA
  - fondé sur la difficulté du logarithme discret : ElGamal

Plan

Vernam

Le chiffrement de Vernam, a.k.a. One-Time-Pad (1917).

- Phase initiale. Alice et Bob créent et se partagent une suite aléatoire ≪ infinie ≫ de bits qui constituent la clé.

$$k_0, k_1, \ldots k_N.$$

- Communication. Supposons que les bits de clé $k_0, \ldots k_{\alpha-1}$ ont déjà servi.
  - Alice chiffre son message $m_0, \ldots, m_{b-1}$ de b bits :

$$(c_0, \ldots, c_{b-1}) = (m_0 \oplus k_\alpha, \ldots, m_{b-1} \oplus k_{\alpha+b-1}).$$

  - Bob se rappelle qu'on en était à k_α, et déchiffre :

$$(m_0, \ldots, m_{b-1}) = (c_0 \oplus k_\alpha, \ldots, c_{b-1} \oplus k_{\alpha+b-1}).$$

Vernam

Le chiffrement de Vernam possède un avantage unique.

- Si la suite (k_i) est purement aléatoire, alors il n'y a aucun espoir de retrouver (m_i) à partir de (c_i).
  - Si $c_i = 0$: $\Pr(m_i = 0) = \Pr(k_i = 0) = \frac{1}{2}$,
    et $\Pr(m_i = 1) = \Pr(k_i = 1) = \frac{1}{2}$.
  - Si $c_i = 1$: $\Pr(m_i = 0) = \Pr(k_i = 1) = \frac{1}{2}$,
    et $\Pr(m_i = 1) = \Pr(k_i = 0) = \frac{1}{2}$.
- Attention : ça ne vaut que si chaque bit k_i est utilisé une unique fois. Sinon les probabilités sur k_i ne sont plus indépendantes.

Le chiffrement de Vernam possède un inconvénient unique.

- Pour chiffrer 1Mo de messages, il faut 1Mo de clé.

Plan

Générer de l'aléa

Pour One-Time-Pad, il faut fabriquer de l'aléa. Plusieurs possibilités.

- Croire que c'est facile.
- Mettre un singe devant un clavier.
- Écouter le rayonnement cosmique ou toute autre source physique.

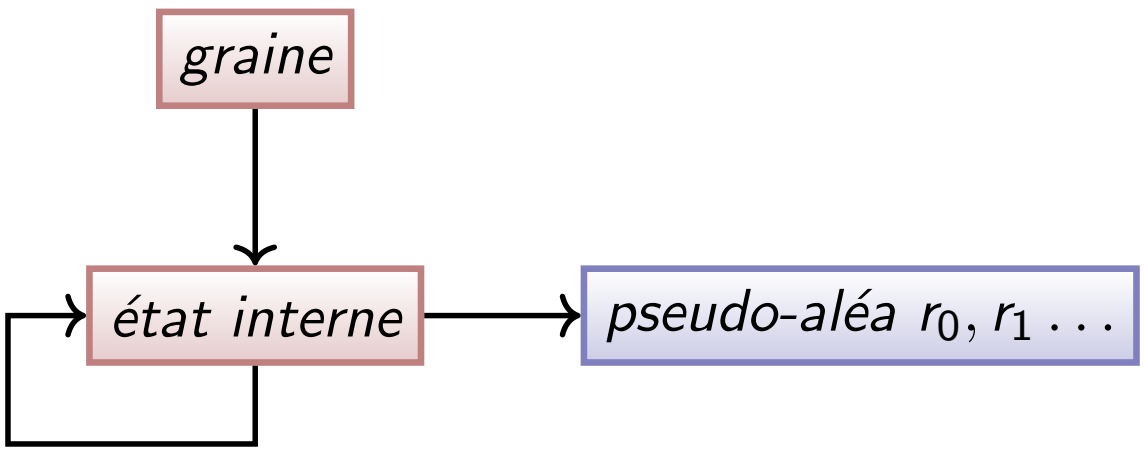
En pratique, un ordinateur est assez déterministe.

Il existe des générateurs aléatoires «matériels», exploitant l'aléa «physique».

Générateurs pseudo-aléatoires

Une autre façon de fabriquer de l'aléa : les générateurs pseudo-aléatoires (PRNG).

- On part d'une graine (aléatoire).
- Cette graine alimente un état interne.
- Le générateur met à jour son état interne après chaque bit produit.



Générateurs pseudo-aléatoires

Une fois la graine spécifiée, le comportement du générateur pseudo-aléatoire est déterministe.

Usages :

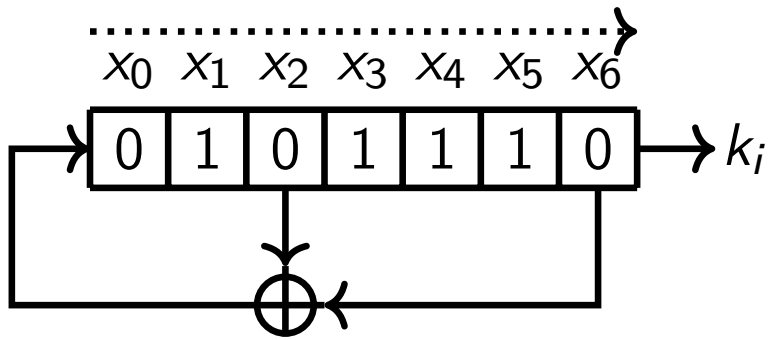
- Aléa dans les ordinateurs :
  - Pas cher à implanter.
  - Modélisation d'événements aléatoires.
  - Aussi : reproductibilité des tests.
- Crypto : utiliser une graine secrète, connue de Alice et Bob, et faire du One-Time-Pad avec.

PRNGs : les LFSRs

LFSR=

- Linear Feedback Shift Register ;
- registre à décalage à rétroaction linéaire.

Le LFSR est un exemple basique et important de PRNG notamment pour son faible coût en matériel.



L'état interne à l'étape $i+1$ découle de l'état interne à l'étape i :

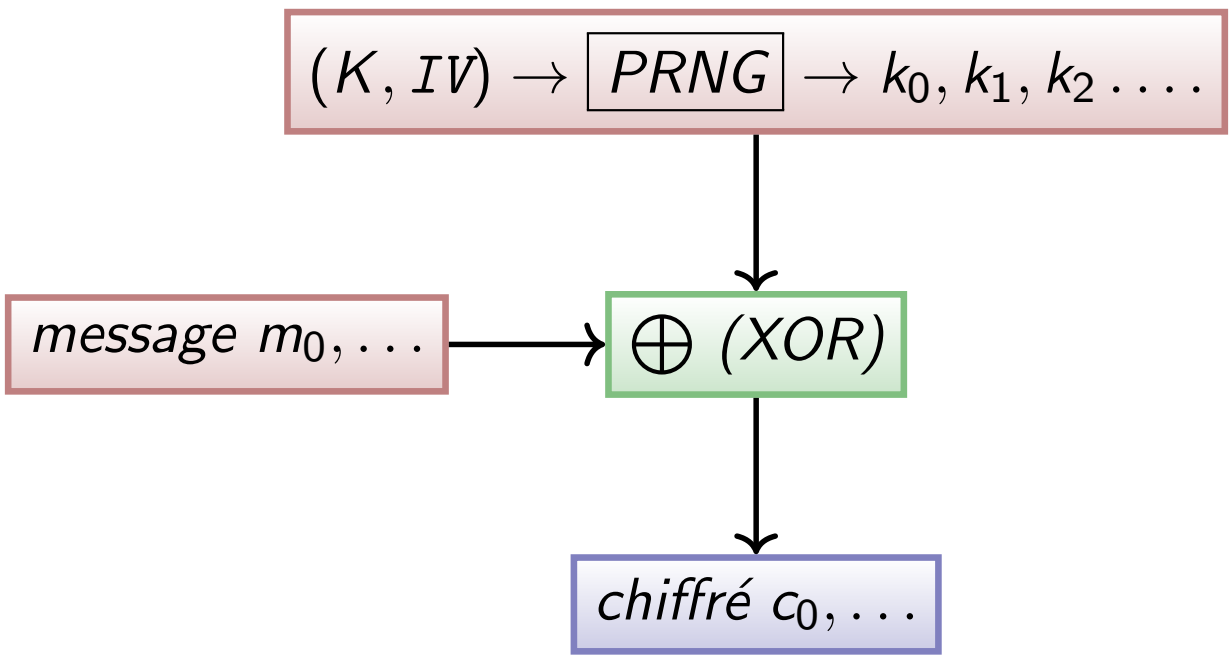
$$x_{k+1}^{i+1} = x_k^i, \quad x_0^{i+1} = x_6^i \oplus x_2^i.$$

Chiffrement à flot

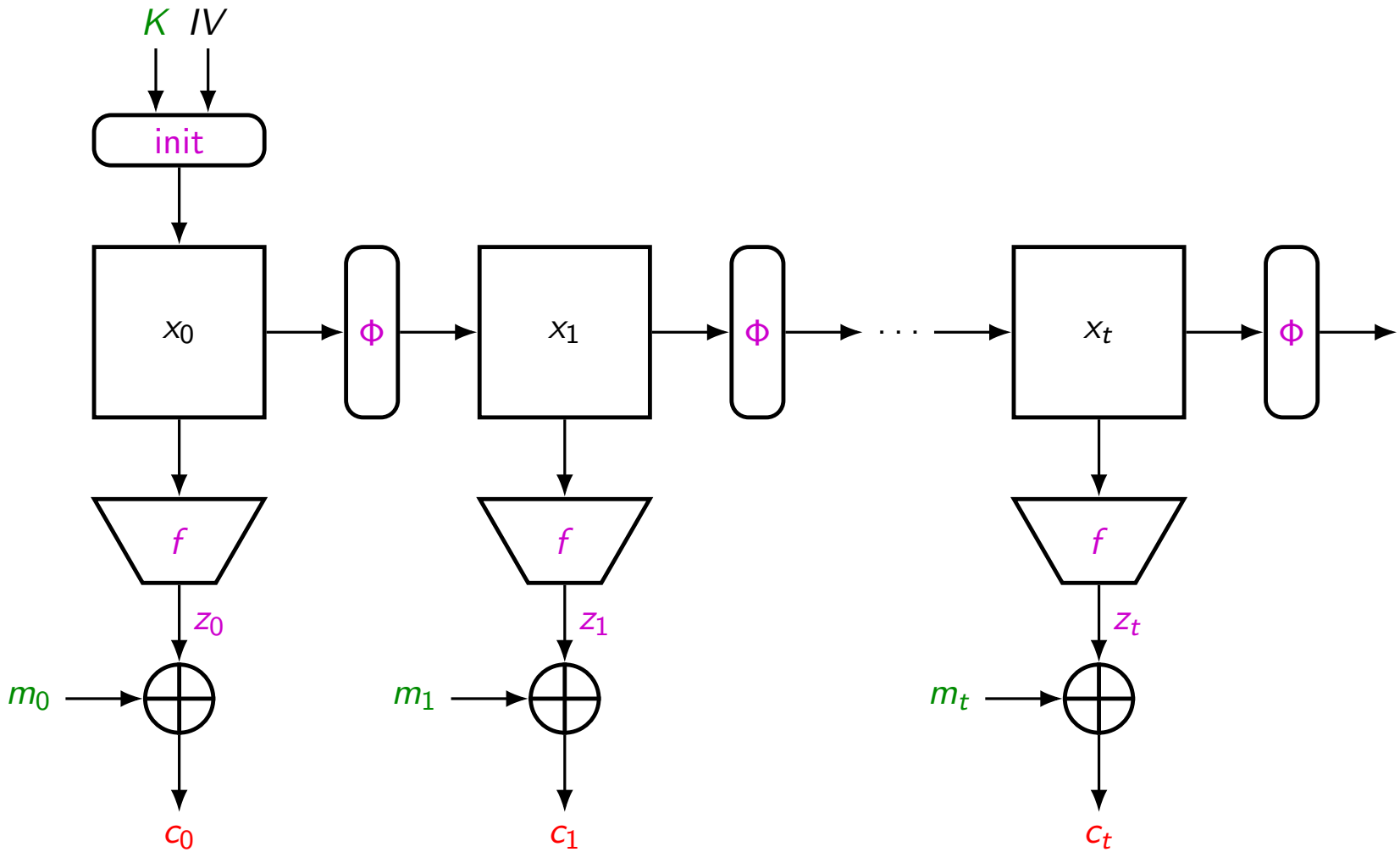
Le point-clé dans One-Time-Pad : la difficulté à générer et partager une volumineuse suite de chiffrement.

Palliatif : utiliser un (PRNG) pour fabriquer une suite chiffrante (k_i). La graine du générateur pseudo-aléatoire sert de clé.

- Chargement de l'état interne à partir de la graine.
- Un état interne $\rightsquigarrow$ bit(s) de clé, et état interne suivant.

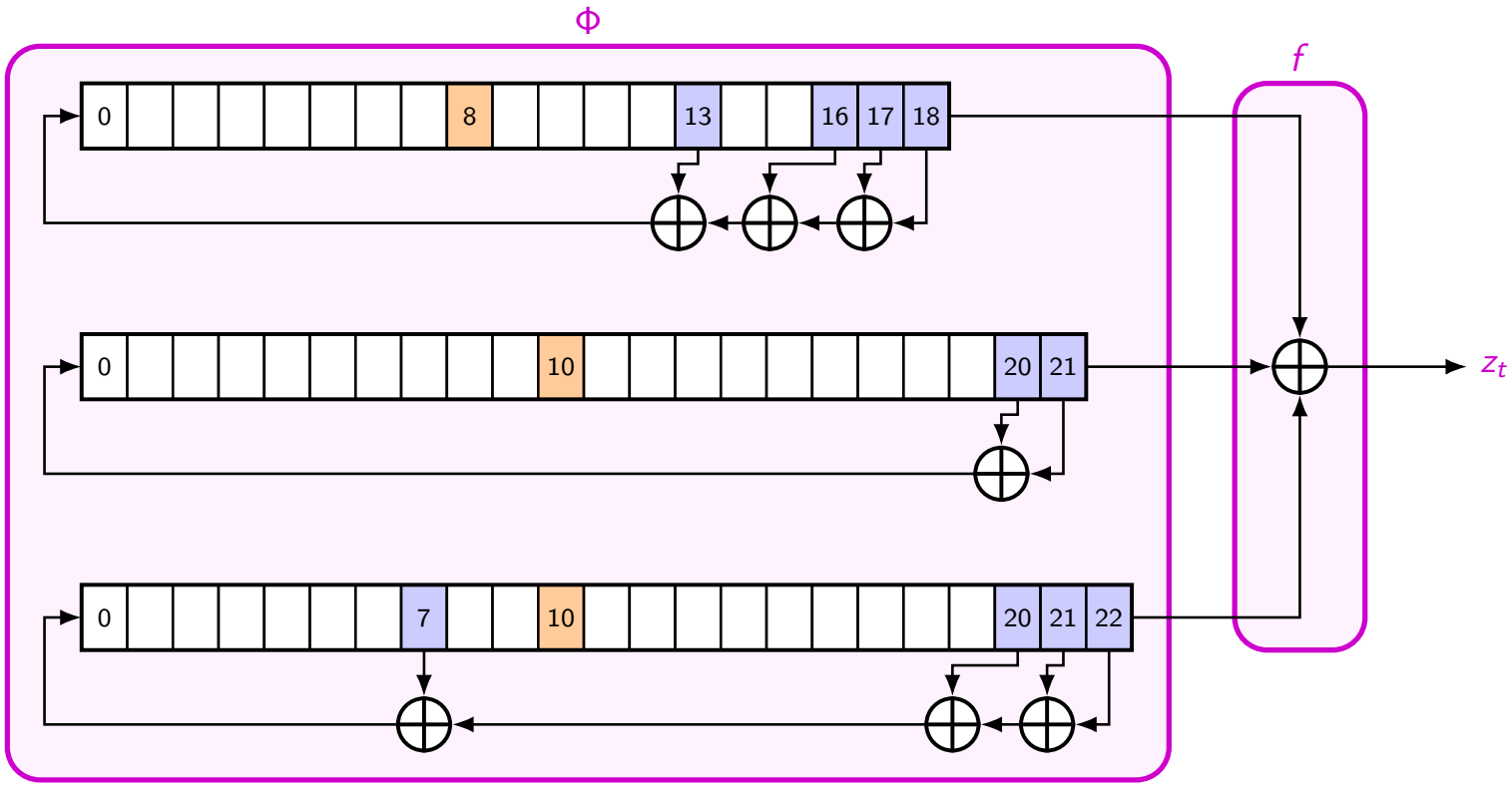


Chiffrement par flot



Notations : x_t=état interne ; f= fonction de filtrage ; Φ= fonction de transition ; IV=valeur initiale.

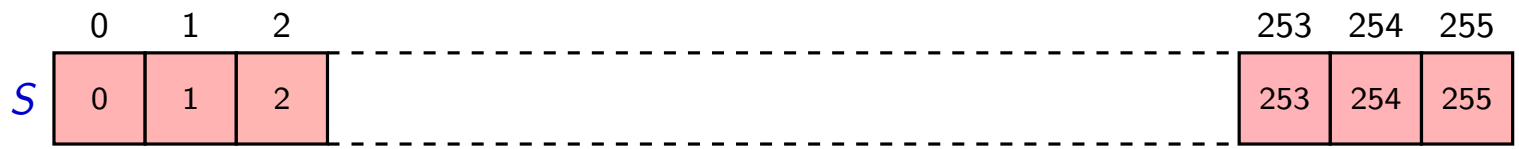
Exemple : A5/1 [1987]



- Clé secrète de 64 bits, IV de 22 bits
- Cryptanalyse en quelques minutes [Nohl & Paget, 2009]
- Snowden : *NSA «can process encrypted A5/1 in real time»*
- Toujours utilisé dans des milliards de téléphones portables (GSM)

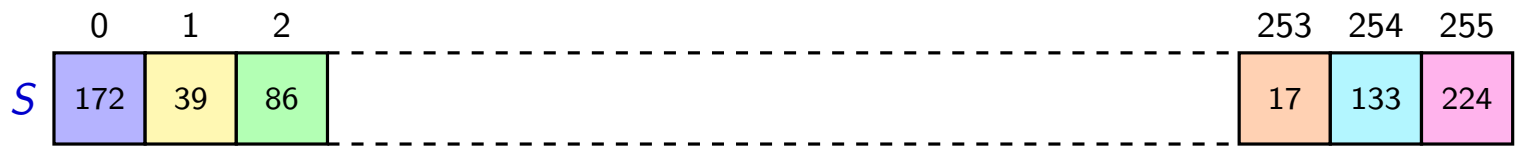
Exemple : RC4 [Rivest, 1987]

- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



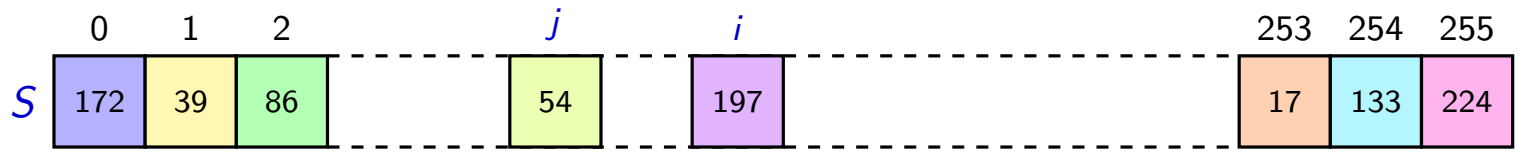
Exemple : RC4 [Rivest, 1987]

- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



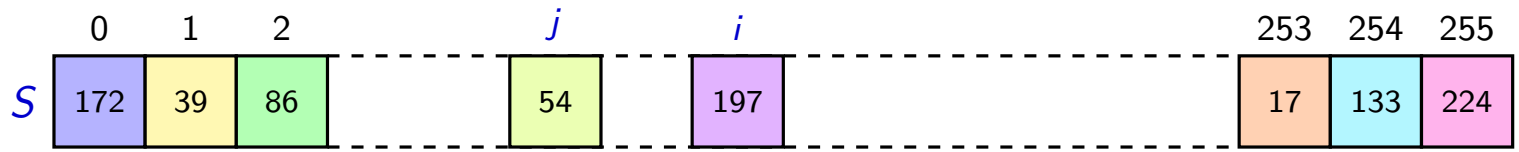
Exemple : RC4 [Rivest, 1987]

- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



Exemple : RC4 [Rivest, 1987]

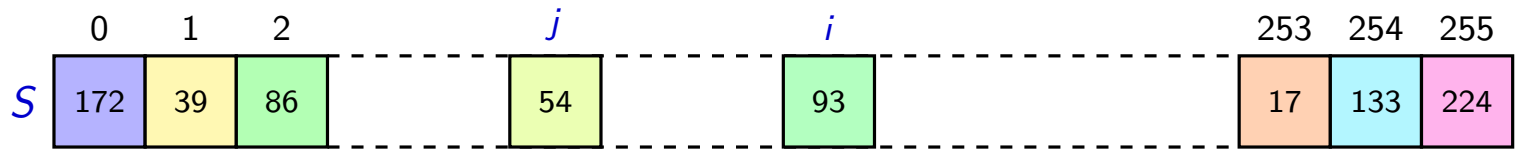
- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



- Fonction de transition Φ :
  - $i \leftarrow (i + 1) \bmod 256$
  - $j \leftarrow (j + S[i]) \bmod 256$
  - échanger les valeurs de $S[i]$ et $S[j]$

Exemple : RC4 [Rivest, 1987]

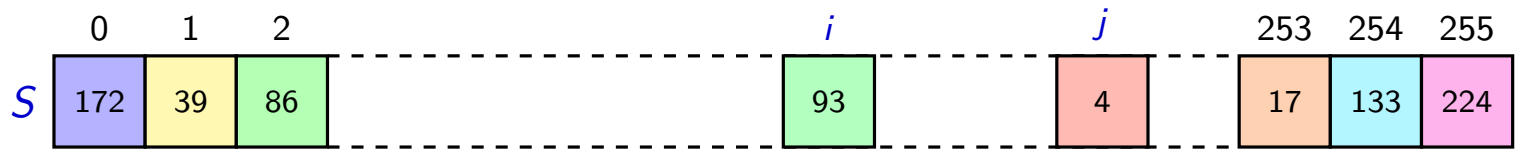
- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



- Fonction de transition Φ :
  - $i \leftarrow (i + 1) \bmod 256$
  - $j \leftarrow (j + S[i]) \bmod 256$
  - échanger les valeurs de $S[i]$ et $S[j]$

Exemple : RC4 [Rivest, 1987]

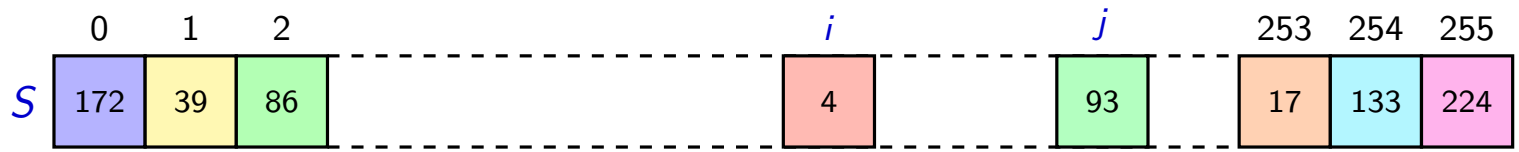
- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



- Fonction de transition Φ :
  - $i \leftarrow (i + 1) \bmod 256$
  - $j \leftarrow (j + S[i]) \bmod 256$
  - échanger les valeurs de $S[i]$ et $S[j]$

Exemple : RC4 [Rivest, 1987]

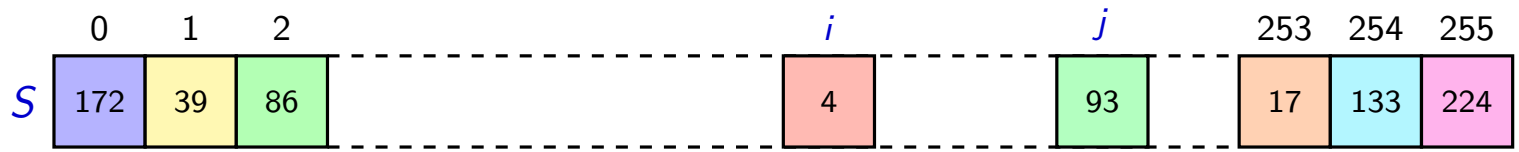
- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



- Fonction de transition Φ :
  - $i \leftarrow (i + 1) \bmod 256$
  - $j \leftarrow (j + S[i]) \bmod 256$
  - échanger les valeurs de $S[i]$ et $S[j]$

Exemple : RC4 [Rivest, 1987]

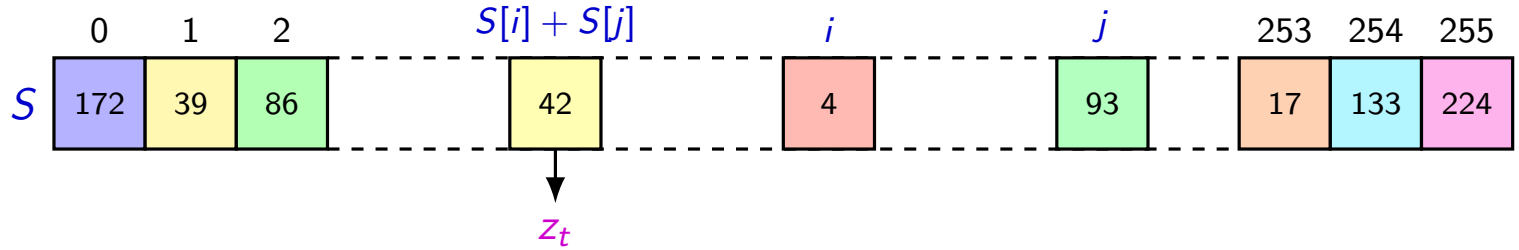
- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



- Fonction de transition Φ :
  - $i \leftarrow (i + 1) \bmod 256$
  - $j \leftarrow (j + S[i]) \bmod 256$
  - échanger les valeurs de $S[i]$ et $S[j]$
- Fonction de filtrage f : $z_t \leftarrow S[(S[i] + S[j]) \bmod 256]$

Exemple : RC4 [Rivest, 1987]

- Utilisé dans SSL/TLS, SSH, WEP, WPA, etc.
- Très efficace en logiciel, mais nombreux biais statistiques sur Z
- Clé secrète de 40 à 2048 bits, pas d'IV en tant que tel
- État interne :
  - tableau S de 256 octets : permutation de $\{0, 1, \ldots, 255\}$
  - indices i et $j \in \{0, 1, \ldots, 255\}$



- Fonction de transition Φ :
  - $i \leftarrow (i + 1) \bmod 256$
  - $j \leftarrow (j + S[i]) \bmod 256$
  - échanger les valeurs de $S[i]$ et $S[j]$
- Fonction de filtrage f : $z_t \leftarrow S[(S[i] + S[j]) \bmod 256]$

Plan

Chiffrement par bloc vs. chiffrement à flot

La problématique du chiffrement à flot : chiffrer une donnée de longueur très grande.

Le chiffrement par bloc est différent. On souhaite chiffrer des données de taille fixe.

C'est éventuellement une brique de base.

$$\boxed{\text{chiffrement par bloc}} + \boxed{\text{mode opératoire}}$$
$$=$$
$$\boxed{\text{chiffrement d'une donnée longueur arbitraire}}$$

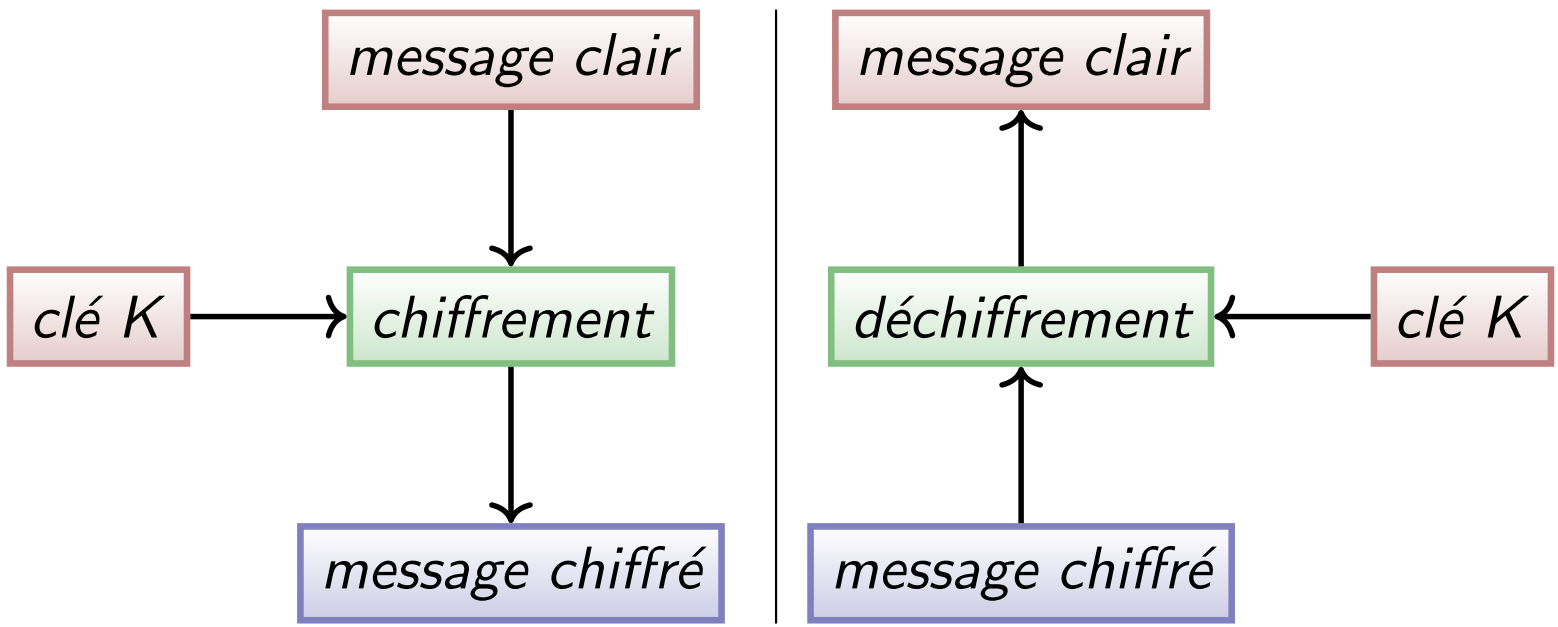
Vu de loin, un chiffrement par bloc se distingue par :

- Sa longueur de bloc.
- Sa longueur de clé.

Chiffrement par bloc : schéma

Autre concept : le chiffrement par bloc.

- Messages clair et message chiffrés sont des bloc de n bits.
- Le système prescrit la taille n des bloc.
- La clé est aussi constituée d'une suite de bits (n_K bits).



Plan

DES

DES (Data Encryption Standard, 1976).

- Créé par IBM et retravaillé par la NSA.
- DES en avance sur son temps. Exploite des idées inconnues du monde académique à l'époque.
- Taille de bloc : 64 bits. Taille de clé : 56 bits.

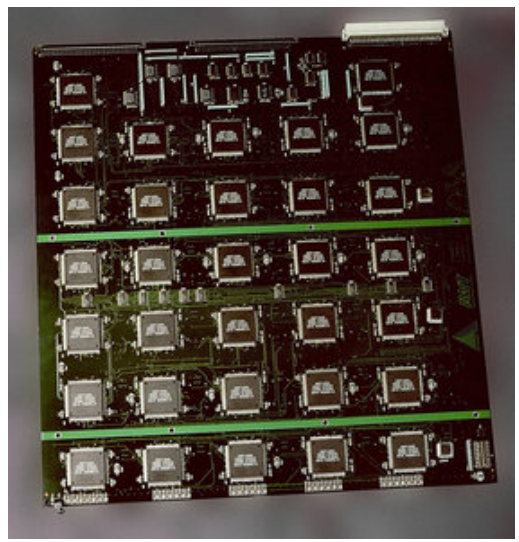
Aujourd'hui DES est «cassé», car 2^{56} est trop petit.

DES

DES (Data Encryption Standard, 1976).

- Créé par IBM et retravaillé par la NSA.
- DES en avance sur son temps. Exploite des idées inconnues du monde académique à l'époque.
- Taille de bloc : 64 bits. Taille de clé : 56 bits.

Aujourd'hui DES est «cassé», car 2^{56} est trop petit.



EFF Deep Crack (1999)
matériel dédié
casser une clé $\sim$ 1 jour.

Structure du cryptosystème

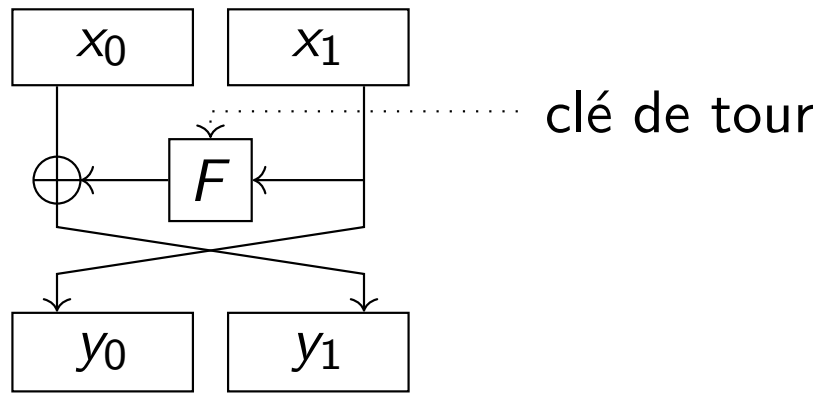
Plusieurs briques élémentaires.

- Réseau de Feistel
- Cadencement de clefs (Key schedule)
- Boites S

Réseau de Feistel (1)

La fonction de chiffrement du DES est construite comme un réseau de Feistel. Un tel réseau est une suite de plusieurs tours.

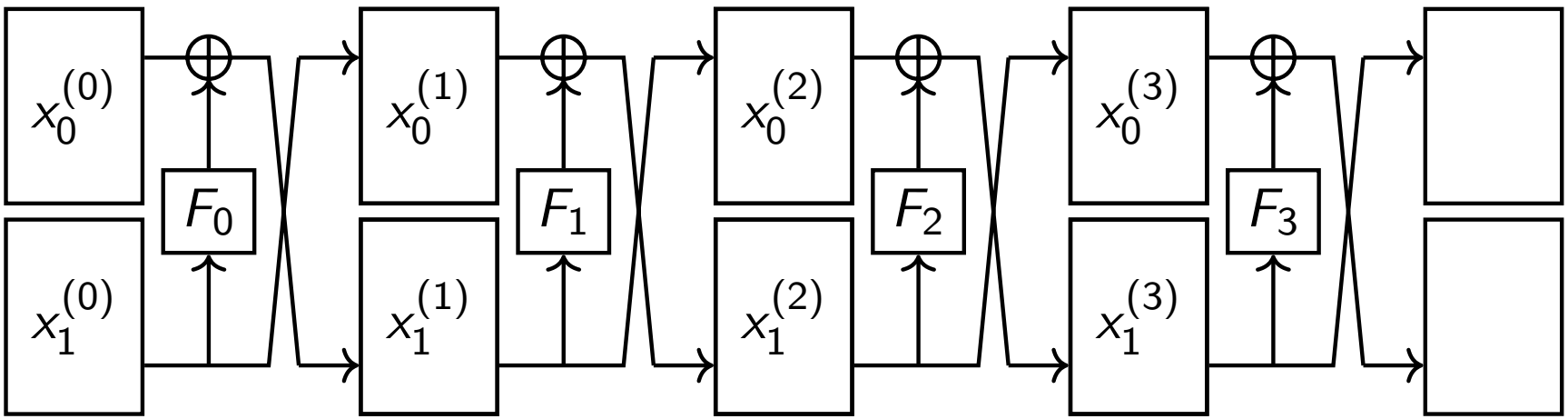
Un ≪tour≫ d'un réseau de Feistel est une fonction $\mathcal{R}_F$ inversible de $2n$ bits vers $2n$ bits :



- La donnée d'entrée x est sur $2n$ bits.
- Les $2n$ bits sont séparés en n bits pour x_0, n bits pour x_1.
- Chaque fonction F (pas forcément inversible) va de $n + k$ bits vers n bits.
- Élément clé : l'inverse de $\mathcal{R}_F$ est facile à déterminer.

Réseau de Feistel (2)

Il est possible d'enchaîner plusieurs tours.



- Les fonctions F_i sont là pour mettre du désordre ;
- La structure garantit qu'on peut fabriquer une inverse, et donc déchiffrer.

Point crucial : le bon choix des clés de tour à partir de la clé maîtresse.

Feistel → chiffrement

- DES est paramétré par une clé maitresse K de 56 bits.
- DES utilise un réseau de Feistel à 16 tours.
- DES spécifie un algorithme de dérivation = le cadencement de clé (key schedule) à partir de K, de 16 clés de tours $(k_0, \ldots, k_{15})$ de 48 bits chacune.
- Les clés k_i ne sont pas indépendantes. En moyenne chaque bit de K est utilisé dans 14 des 16 clés k_i.
- La clé k_i est utilisée pour paramétrer la fonction F_i : moralement, $F_i(t) = f_i(t \oplus k_i)$. DES spécifie des fonctions f_i particulières, fabriquées pour être non linéaires : ce sont les boites S (S comme substitution).

Ce cadencement de clé ne doit pas être négligé.

Résumé sur DES

DES a été la cible de nombreuses attaques.

- Cryptanalyse différentielle (1990) : DES résiste très bien.
- Cryptanalyse linéaire (1994) : presque.
- La force brute (1997) : ça a finalement marché.

DES a rempli ses objectifs initiaux.

- Simple à implémenter (sur le matériel d'époque !).
- Résistant à de nombreuses cryptanalyses. Essentiellement, l'attaquant n'a qu'une option : essayer toutes les clés.

Evolution : DES avec une clé plus longue ?

L'idée d'étendre DES pour obtenir une clé plus longue est ancienne.

Suggestion :

- considérer une clé K de 112 bits.
- écrire $K = (K_1, K_2)$ où K_1 et K_2 ont 56 bits.
- algorithme de chiffrement qui réutilise le DES :

$$E_K(m) = \mathrm{DES}_{K_2}(\mathrm{DES}_{K_1}(m)).$$

  - car on a assez confiance en la résistance de DES à l'analyse.
  - logique économique : il existe plein de puces implantant DES.
- Le cryptographe espère que l'attaquant n'a pas d'autre option que d'essayer les 2^{112} possibilités.

Meet-in-the-middle

Si $\mathcal{C}$ est un système de chiffrement paramétré par une clé K à n bits, alors le système $\mathcal{C}_{K_2} \circ \mathcal{C}_{K_1}$ (qui a pour clé de $2n$ bits le couple (K_1, K_2)) est attaquable avec quelques couples clair/chiffré :

Algorithme **meet-in-the-middle** pour retrouver (K_1, K_2)

- On suppose que l'on connait un couple clair / chiffré (x,y), i.e. $y = \mathcal{C}_{K_2}(\mathcal{C}_{K_1}(x))$.
- Calculer $\mathcal{L}_1 = \{\mathcal{C}_t(x),\ t \in [0, 2^n[\}$. Trier $\mathcal{L}_1$.
- Pour chaque $u \in [0, 2^n[$, calculer $\mathcal{C}_u^{-1}(y)$.
  - S'il existe t tel que $\mathcal{C}_t(x) = \mathcal{C}_u^{-1}(y)$, alors (t, u) est un candidat pour la clé(K_1, K_2).
- Avec plusieurs couples clair/chiffré, on filtre les candidats.

Complexité ? Tps ??? , Mémoire ??? .

Meet-in-the-middle

Si $\mathcal{C}$ est un système de chiffrement paramétré par une clé K à n bits, alors le système $\mathcal{C}_{K_2} \circ \mathcal{C}_{K_1}$ (qui a pour clé de $2n$ bits le couple (K_1, K_2)) est attaquable avec quelques couples clair/chiffré :

Algorithme **meet-in-the-middle** pour retrouver (K_1, K_2)

- On suppose que l'on connait un couple clair / chiffré (x,y), i.e. $y = \mathcal{C}_{K_2}(\mathcal{C}_{K_1}(x))$.
- Calculer $\mathcal{L}_1 = \{\mathcal{C}_t(x),\ t \in [0, 2^n[\}$. Trier $\mathcal{L}_1$.
- Pour chaque $u \in [0, 2^n[$, calculer $\mathcal{C}_u^{-1}(y)$.
  - S'il existe t tel que $\mathcal{C}_t(x) = \mathcal{C}_u^{-1}(y)$, alors (t, u) est un candidat pour la clé(K_1, K_2).
- Avec plusieurs couples clair/chiffré, on filtre les candidats.

Complexité ? Tps $O(2^n)$, Mémoire $O(2^n)$. Compromis tps-mémoire.

3DES

Qu'est-ce que l'on fait pour sauver le DES ? On complique encore d'un niveau. Triple-DES (1998).

$$3\mathrm{DES}_{K_1K_2}(m) = \mathrm{DES}_{K_1}(\mathrm{DES}^{-1}_{K_2}(\mathrm{DES}_{K_1}(m))).$$

La clé est de 112 bits.

- meet-in-the-middle pas possible.
- Feature : si $K_1 = K_2$, alors on retombe sur DES.
- Évidemment 3 fois plus lent que DES.

Mis en place comme une transition avant de trouver mieux. Encore très utilisé, même si depuis, on a mieux.

Plan

AES

Fin des années 1990 :

- constat que DES est en fin de vie.
- appel à contributions NIST pour proposer un successeur.

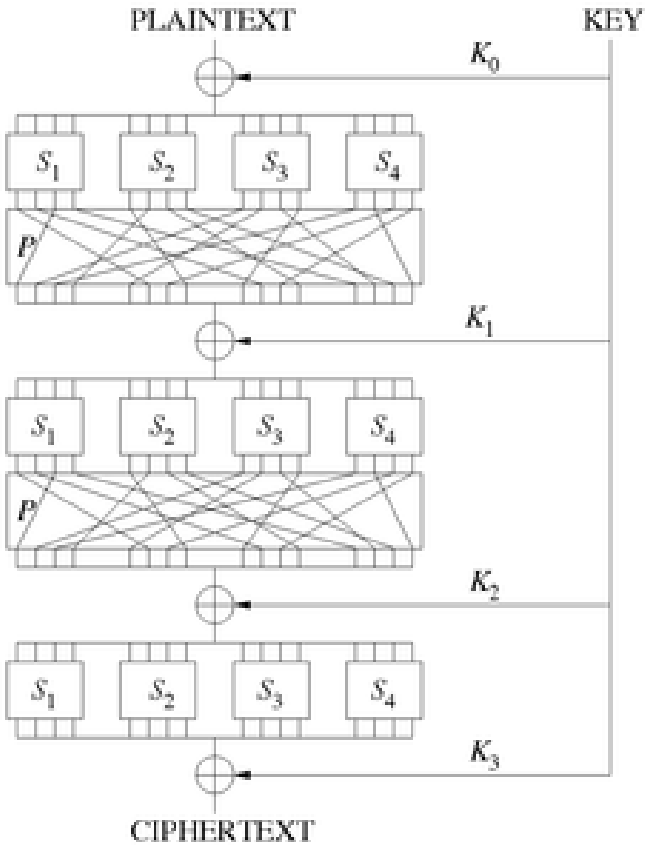
15 propositions, 5 finalistes.
Auteurs de l'algo choisi : Rijmen, Daemen (Leuven, Belgique).

Cahier des charges :

- Plusieurs tailles de clé possibles.
- Taille de bloc 128 bits.
- Rapide en logiciel et en matériel.
- Solide !

Structure de l'AES

- AES n'est pas un réseau de Feistel.
- AES est un réseau de substitutions-permutations (SPN).



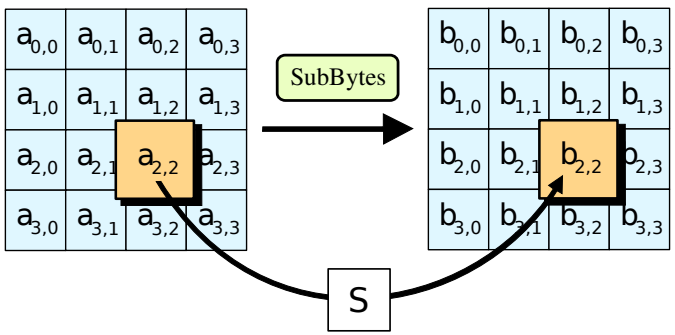
- Comme pour DES, il y a des tours et donc un cadencement de clé.
- À chaque étape la clé de tour est ajoutée (XOR) à la valeur courante.
- Il y a encore des boîtes S qui apportent la non-linéarité.

AES

- AES opère sur des données de 128 bits. = 16 octets.
- On représente l'état interne par un carré 4×4.
- Chaque octet est vu comme un élément du corps $\mathbb{F}_{2^8}$.

Les étapes de chaque tour sont :

- Boite S (Substitution) : `SubBytes` = inverse dans $\mathbb{F}_{2^8}$ de chq octet.

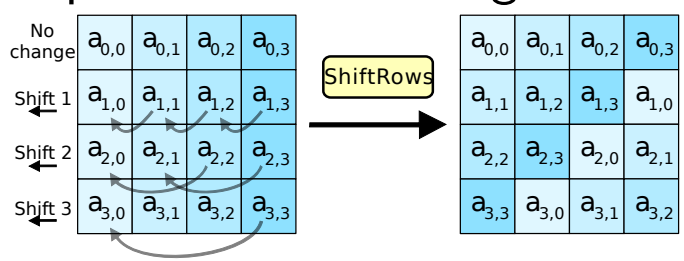


AES

- AES opère sur des données de 128 bits. = 16 octets.
- On représente l'état interne par un carré 4×4.
- Chaque octet est vu comme un élément du corps $\mathbb{F}_{2^8}$.

Les étapes de chaque tour sont :

- Boite S (Substitution) : `SubBytes` = inverse dans $\mathbb{F}_{2^8}$ de chq octet.
- Permutation :
  - `ShiftRows` : Chaque octet se déplace sur sa ligne vers la droite d'autant de case que l'indice de sa ligne.

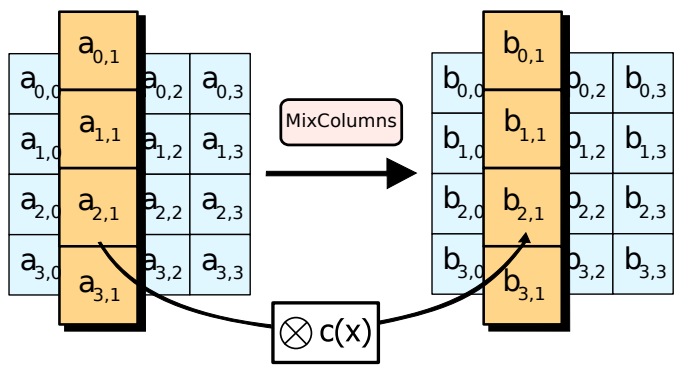


AES

- AES opère sur des données de 128 bits. = 16 octets.
- On représente l'état interne par un carré 4×4.
- Chaque octet est vu comme un élément du corps $\mathbb{F}_{2^8}$.

Les étapes de chaque tour sont :

- Boite S (Substitution) : `SubBytes` = inverse dans $\mathbb{F}_{2^8}$ de chq octet.
- Permutation :
  - `ShiftRows` : Chaque octet se déplace sur sa ligne vers la droite d'autant de case que l'indice de sa ligne.
  - `MixColumns` : Chaque colonne est transformée par une opération linéaire (multiplication par une matrice donnée).



Nombre de tours

AES se décline en plusieurs versions.

- Taille de clé de 128 bits.
- Taille de clé de 192 bits.
- Taille de clé de 256 bits.

NB : La taille de bloc est inchangée (128 bits), seul le cadencement de clés et le nombre de tours sont modifiés : 10, 12, 14.

Attaques sur AES

Le graal de nombreuses personnes.

En pratique : rien d'effectif à ce jour.

- Sur un nombre réduit de tours.
  - AES-128 (10 tours) : Une attaque par texte clair choisi casse 7 tours de AES-128 (Ferguson et al, 2000).
  - AES-192 (12 tours) : Une attaque par texte clair choisi casse 8 tours de AES-192 et 256.
  - AES-256 (14 tours) : Une attaque par clé apparentée casse 9 tours de AES-256.
- Attaque complète sur les 10 tours de l'AES-128 ? Attaque Microsoft, 2011, en $2^{126,1}$ ($< 2^{128}$ par force brute). Non pratique.

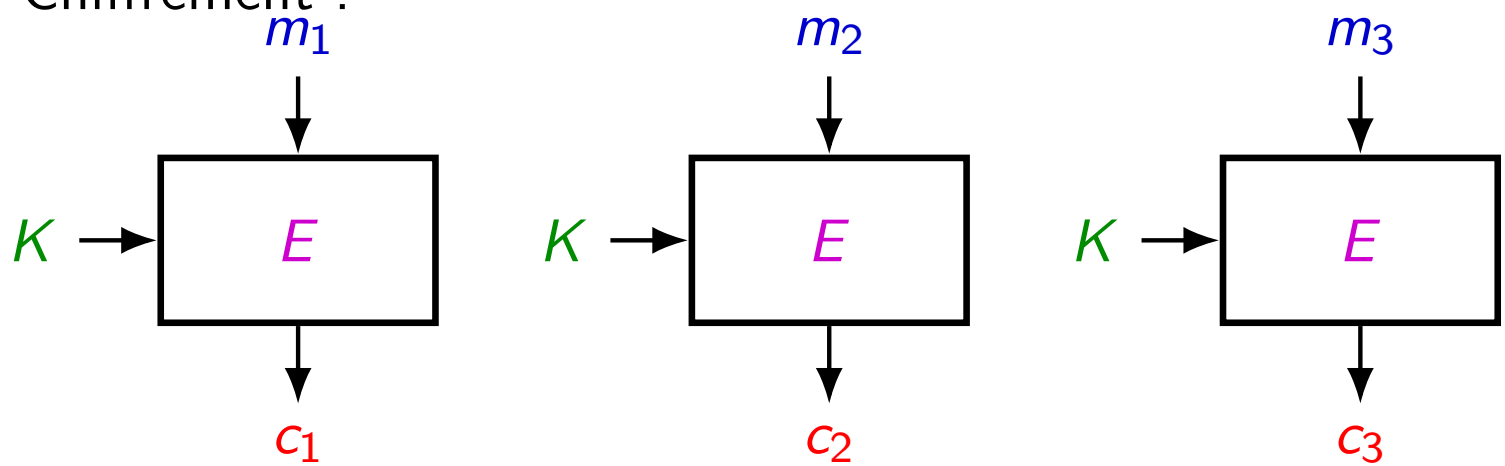
Plan

Mode opératoire ECB

Pour chiffrer un texte de grande taille avec un chiff. par bloc :

Idée naïve : chiffrer chaque bloc du texte indépendamment.

- Chiffrement :

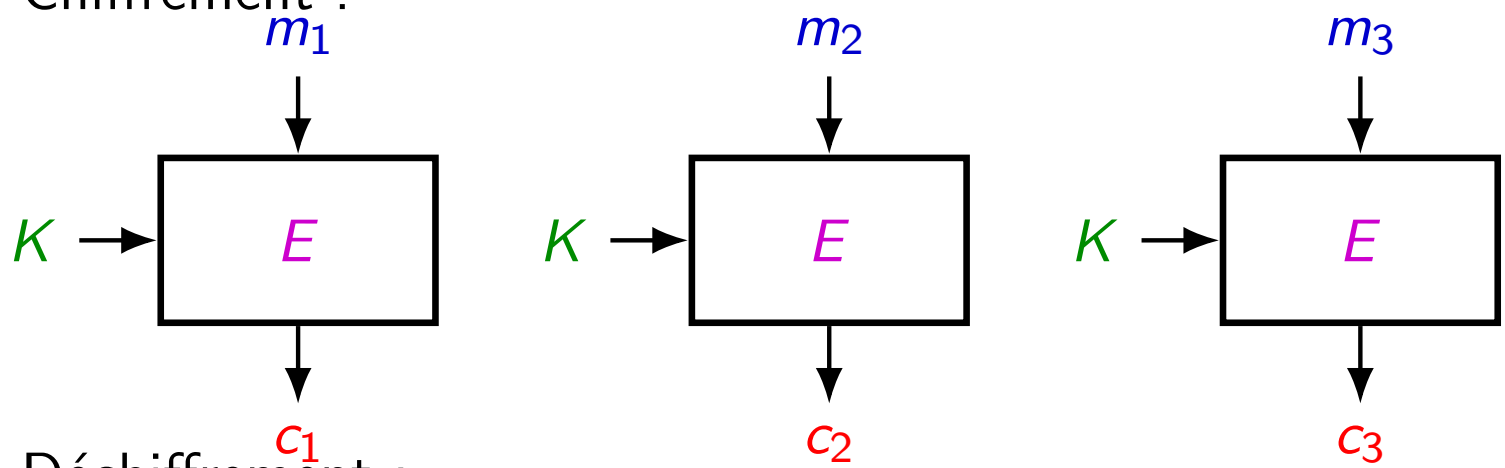


Mode opératoire ECB

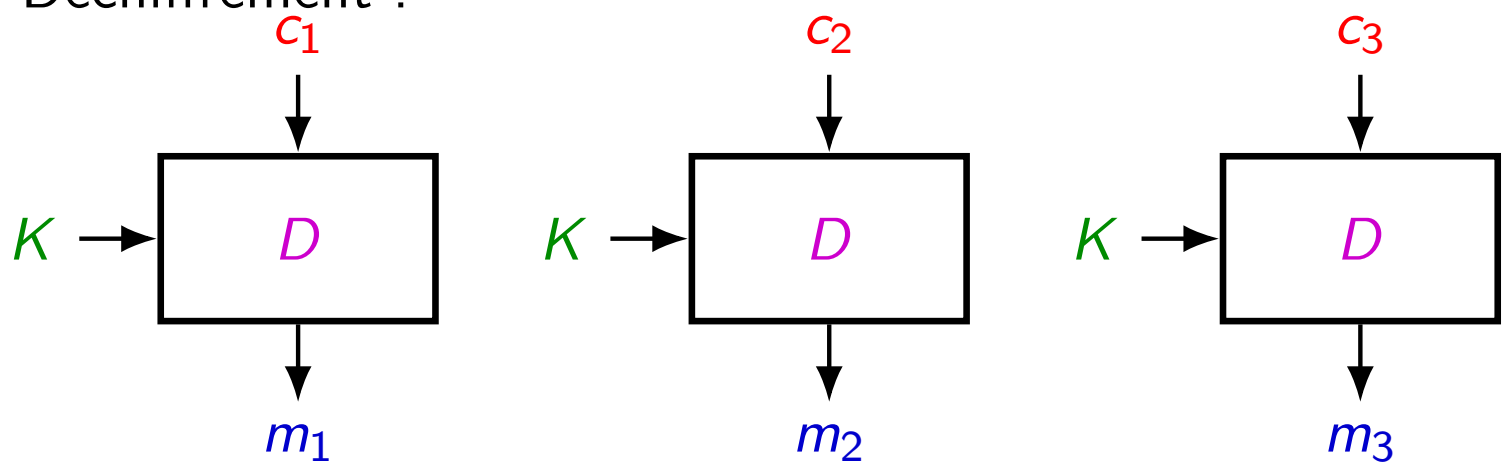
Pour chiffrer un texte de grande taille avec un chiff. par bloc :

Idée naïve : chiffrer chaque bloc du texte indépendamment.

- Chiffrement :



- Déchiffrement :



Problème avec ECB

Les bloc identiques sont chiffrés pareil :

```
0   TWELVE V   2c 47 3e ef d1 60 96 e5
1   IRTUAL C   db a5 a6 f9 dd 90 da 7b
2    IRCUITS   b2 ca 22 14 c0 60 94 cb
3   AND/OR V   29 9a a9 e7 26 97 a2 bf
4   IRTUAL C   db a5 a6 f9 dd 90 da 7b
5     ALLS     25 35 cb 12 8c 13 d6 a8
```

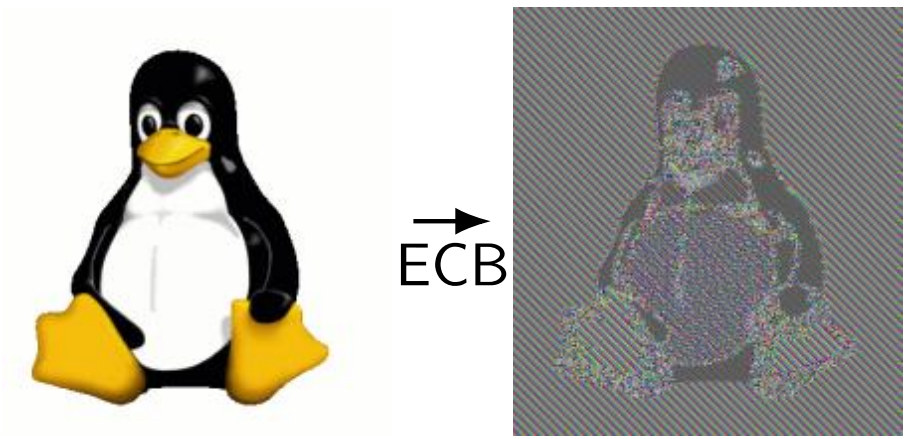
De la sorte, on fait fuiter de l'information.

Problème avec ECB

Les bloc identiques sont chiffrés pareil :

```
0   TWELVE V   2c 47 3e ef d1 60 96 e5
1   IRTUAL C   db a5 a6 f9 dd 90 da 7b
2    IRCUITS   b2 ca 22 14 c0 60 94 cb
3   AND/OR V   29 9a a9 e7 26 97 a2 bf
4   IRTUAL C   db a5 a6 f9 dd 90 da 7b
5     ALLS     25 35 cb 12 8c 13 d6 a8
```

De la sorte, on fait fuiter de l'information.



ECB = mauvaise idée

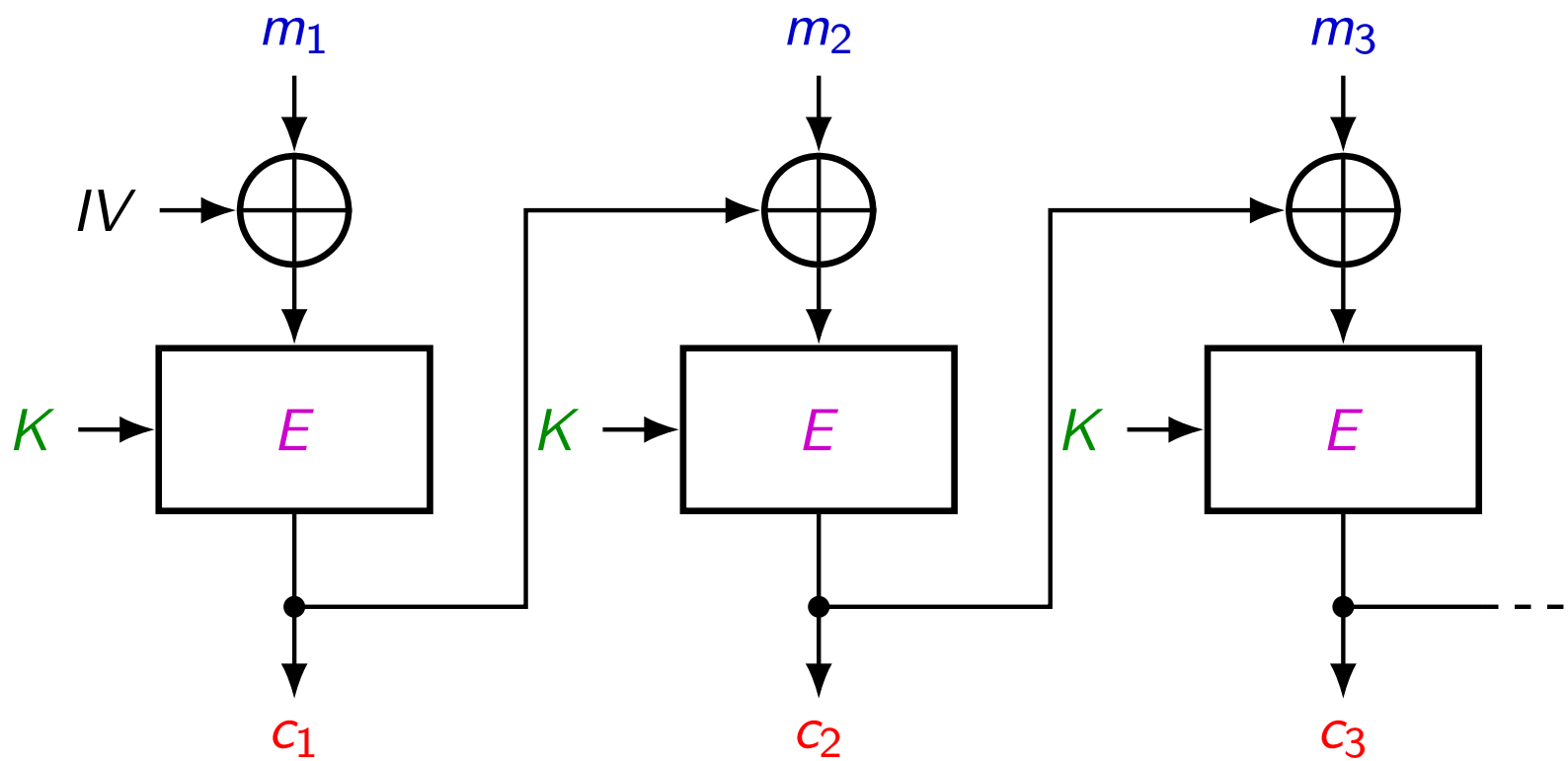
L'information qui fuite avec ECB n'est pas acceptable.

Il est nécessaire d'introduire une variabilité du chiffrement des bloc.

- Chaînage ;
- Paramétrisation de chaque bloc.

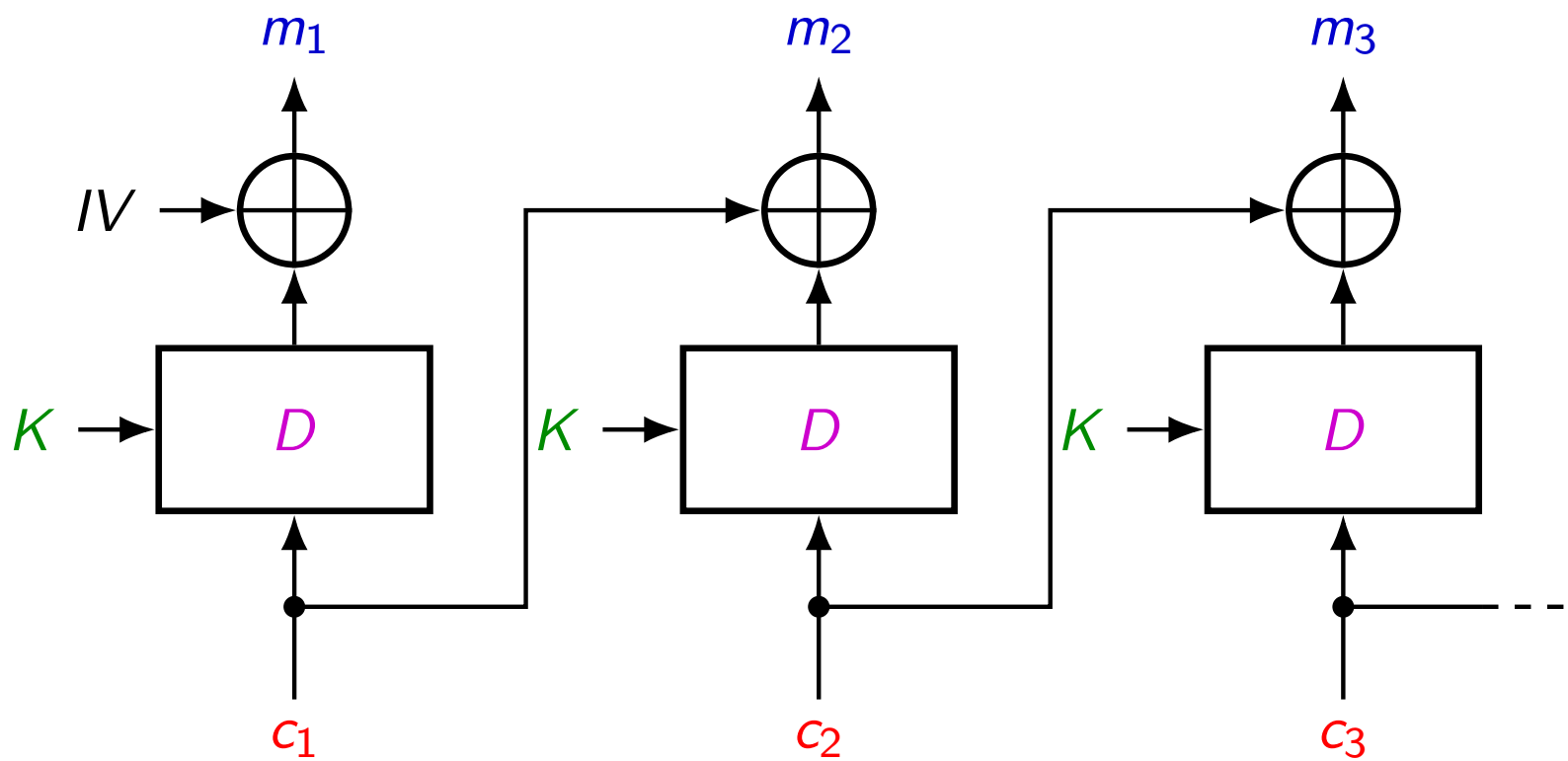
Mode de chiffrement CBC

- Mode CBC pour *Cipher Block Chaining*
  - chiffrement séquentiel



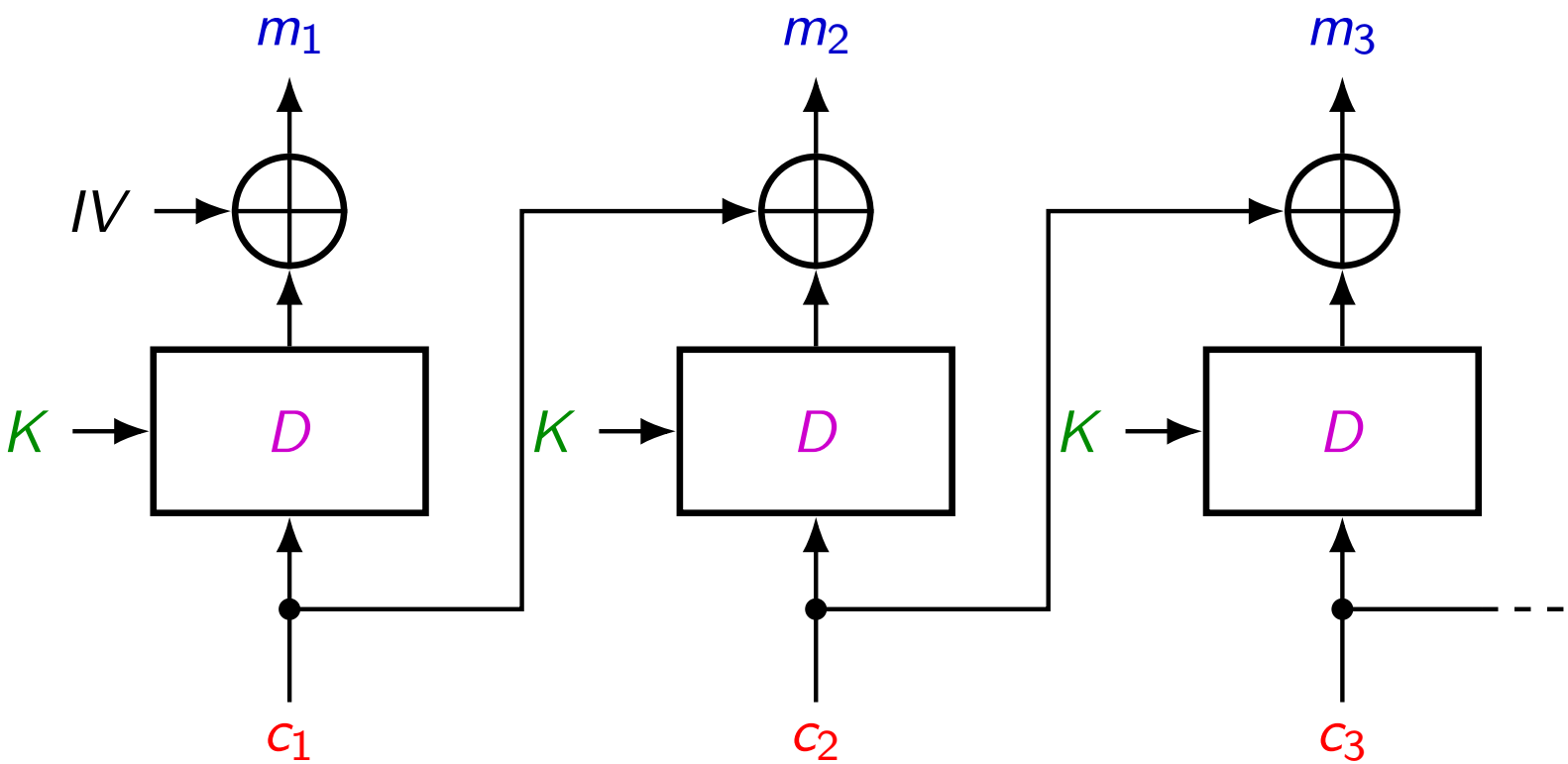
Mode de chiffrement CBC

- Mode CBC pour *Cipher Block Chaining*
  - chiffrement séquentiel
  - déchiffrement parallèle



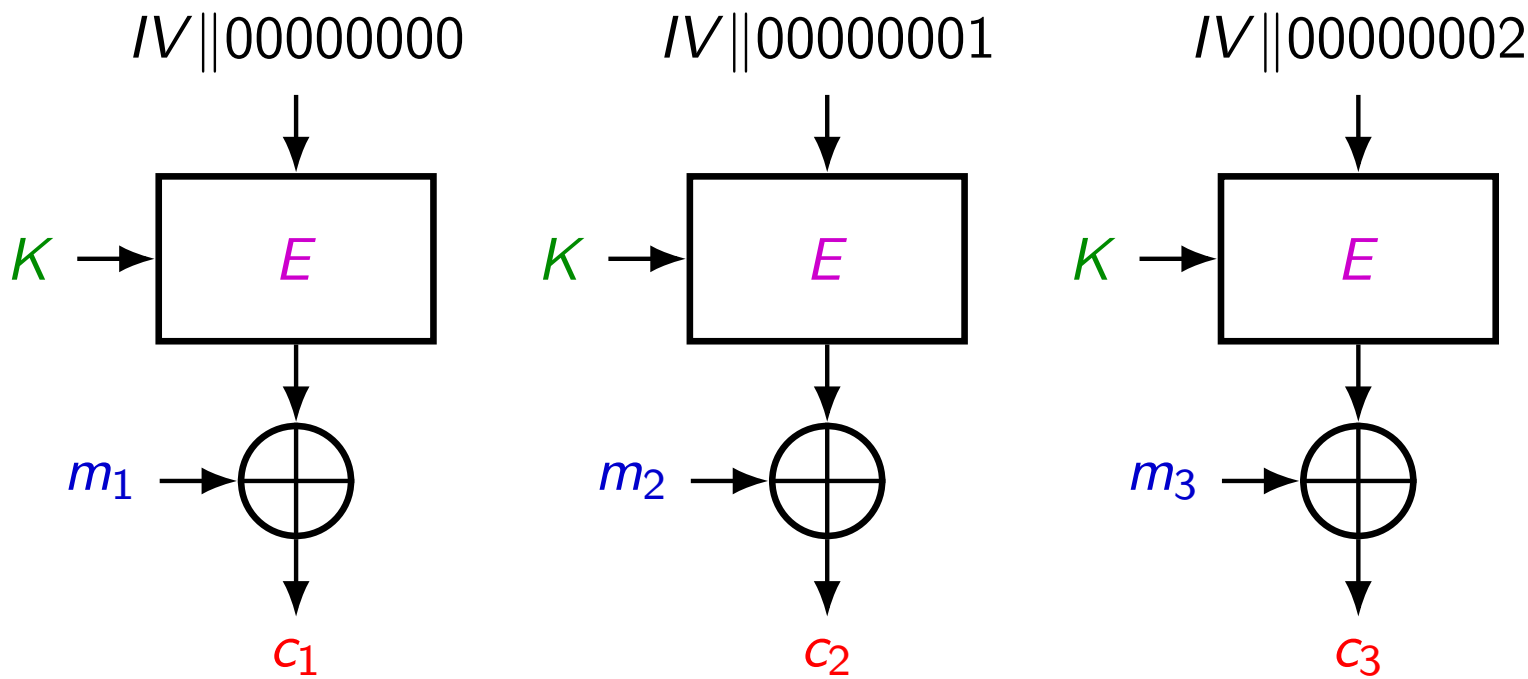
Mode de chiffrement CBC

- Mode CBC pour *Cipher Block Chaining*
  - chiffrement séquentiel
  - déchiffrement parallèle
  - vulnérable à des attaques sur le *padding*



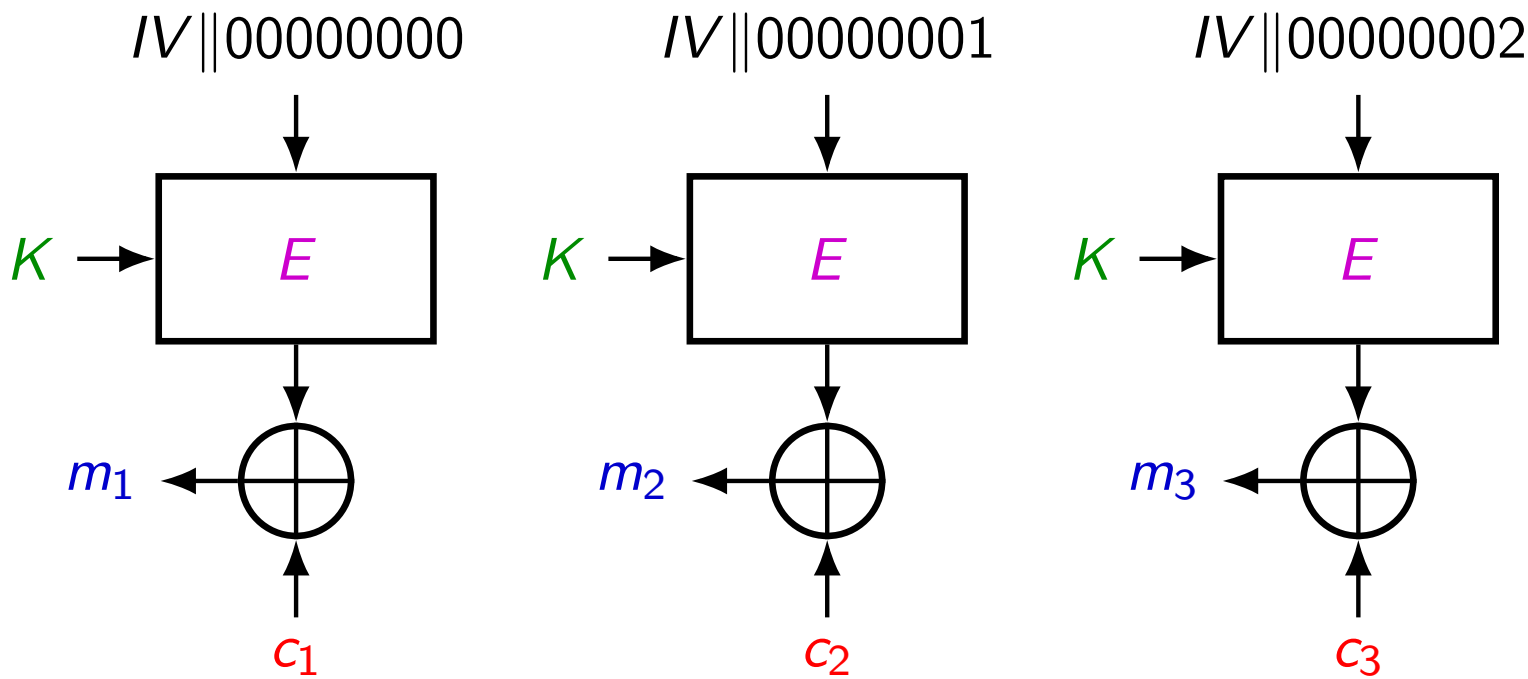
Mode de chiffrement CTR

- Mode CTR (pour *Counter*)
  - transforme un chiffrement par bloc en chiffrement par flot
  - chiffrement parallèle



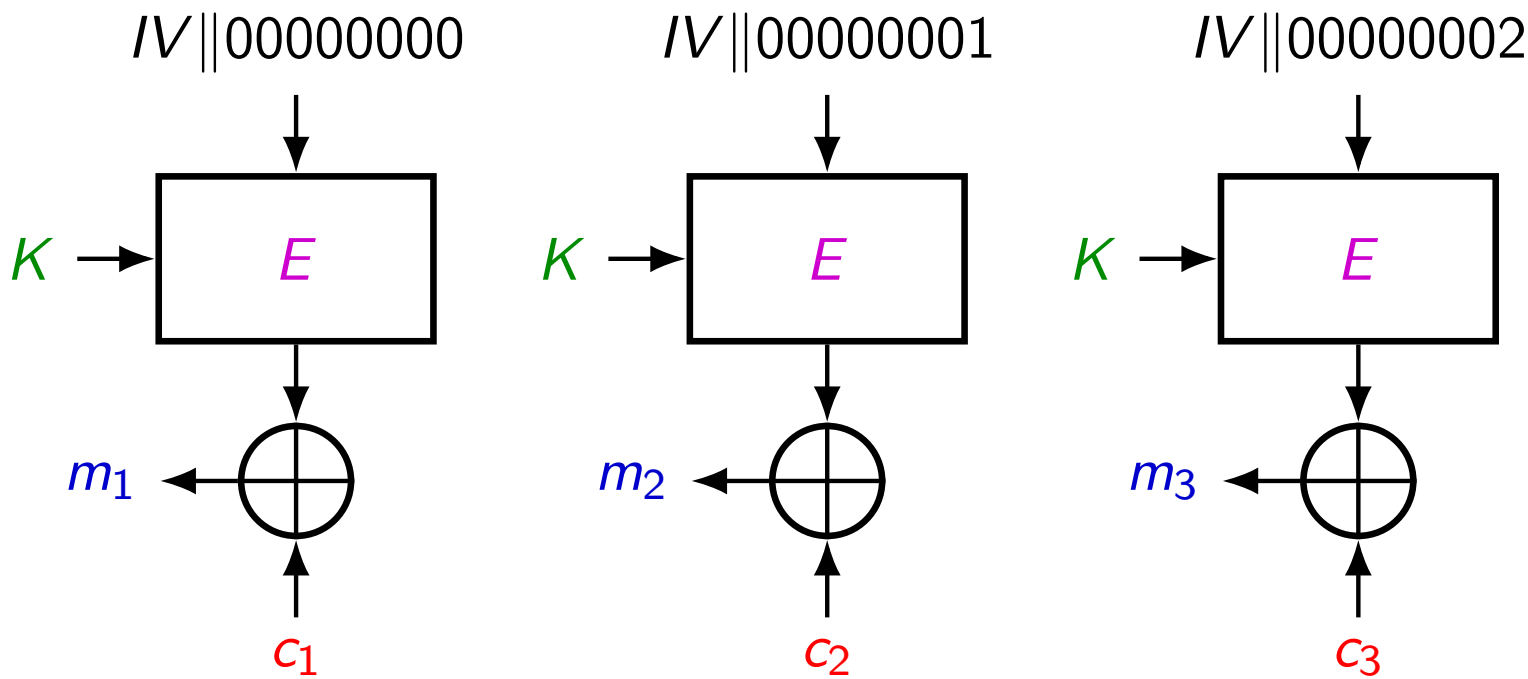
Mode de chiffrement CTR

- Mode CTR (pour *Counter*)
  - transforme un chiffrement par bloc en chiffrement par flot
  - chiffrement parallèle
  - déchiffrement parallèle



Mode de chiffrement CTR

- Mode CTR (pour *Counter*)
  - transforme un chiffrement par bloc en chiffrement par flot
  - chiffrement parallèle
  - déchiffrement parallèle
  - vraiment très simple et sans problème majeur



Modes opératoires plus avancés

En pratique, on peut souhaiter que le mode opératoire transporte également des informations d'authentification, puisse contrôler l'intégrité.

- C'est un problème commun. Vouloir le résoudre avec des modes simples n'est pas facile, il y a de nombreux pièges.
- Des modes opératoires plus adaptés existent.
  - Nombreux mais beaucoup sont brevetés (donc inutiles).
  - La bonne réponse : Galois Counter Mode (GCM).

Conclusion sur le chiffrement symétrique

Le chiffrement symétrique est :

- Rapide (en soft, en hard).
- Les bons algorithmes sont solides, et implantés partout. [1]

Tendance dangereuse : ≪ plus on mélange, plus on gagne en sécurité ≫.

- DES, AES sont bâtis sur des structures simples, l'objectif étant de prouver certaines propriétés des systèmes.
- Rien n'est pire qu'un protocole "fait maison". Les outils répondent aux problèmes posés, il faut les utiliser.

Deux problèmes persistants :

- Quid du premier échange de la clé.
- Impossible d'obtenir des signatures.

1. Il y a même des instructions pour AES sur les CPUs Intel.