Introduction à la Cryptographie

3. Chiffrement symétrique (2)

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

Supports de E. Thomé

Telecom Nancy, 2A ISS – 2021

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

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.

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

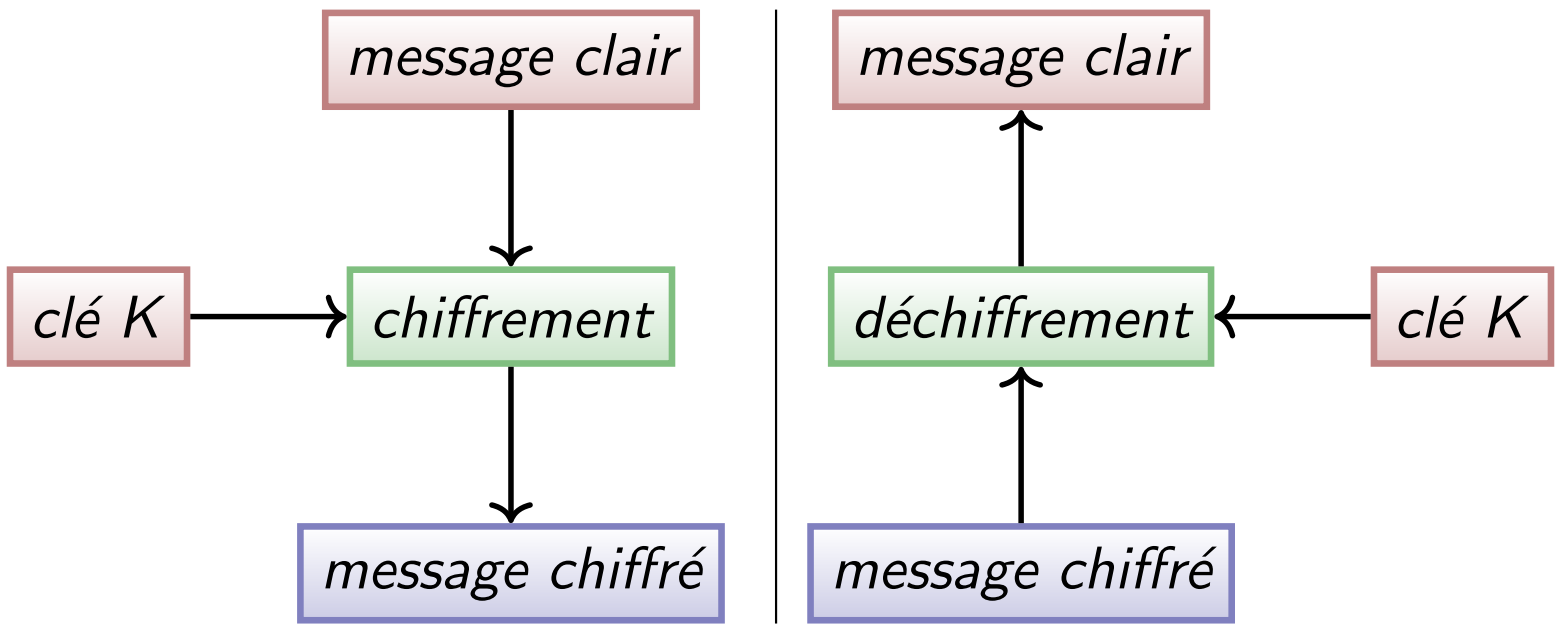
Vu de loin, un chiffrement par bloc se distingue par :

- 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.
- Héritier de Lucifer (1972), tout premier block cipher.
- 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.

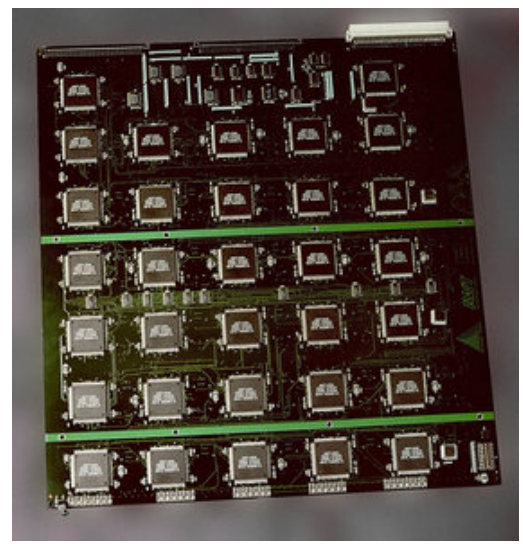
Aujourd'hui DES est «cassé», car 2^{56} est trop petit.

DES

DES (Data Encryption Standard, 1976).

- Créé par IBM et retravaillé par la NSA.
- Héritier de Lucifer (1972), tout premier block cipher.
- 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

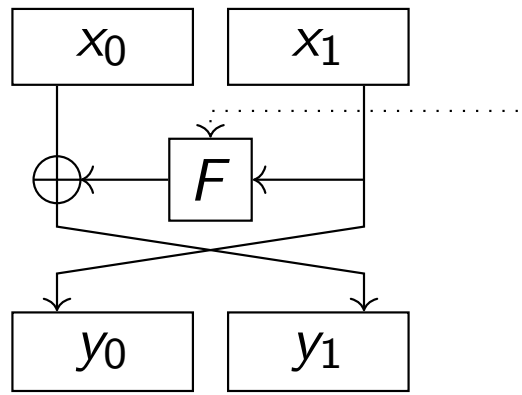
Plusieurs briques élémentaires.

- 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 :

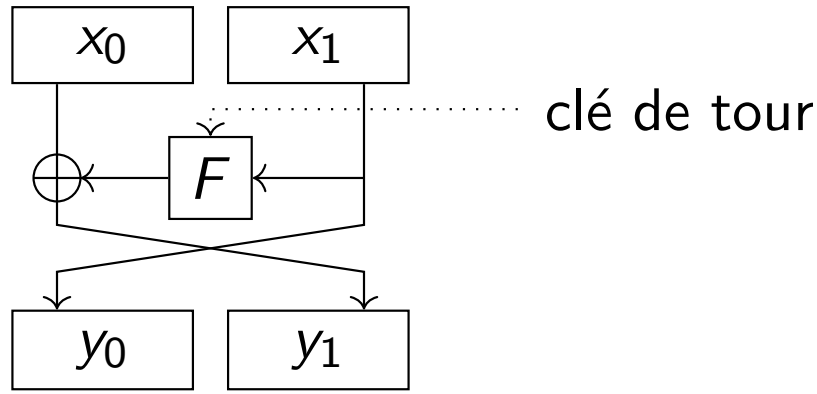


- x_0 et x_1 ont la même largeur en bits (n bits).
- F est une fonction, pas forcément inversible ($n + k$ bits vers n bits).
- On construit ainsi une fonction $\mathcal{R}_F$ telle que :

$$(y_0, y_1) = \mathcal{R}_F(x_0, x_1).$$

En détail : $y_0 = x_1,\ y_1 = x_0 \oplus F(x_1)$.

Structure du cryptosystème

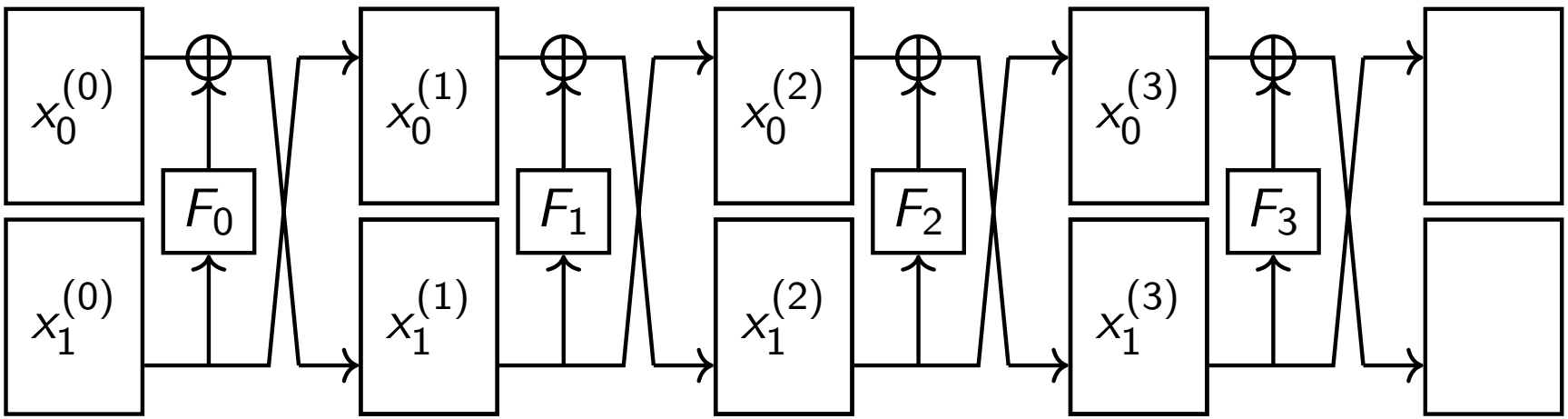


Un ≪tour≫ donne une fonction inversible de $2n$ bits vers $2n$ bits :

- 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 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-clé pour la sécurité : le bon choix des clés de tour à partir de la clé maîtresse.

Feistel $\rightarrow$ 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, $F_i(t) = f_i(t \oplus k_i)$.

On verra que ce cadencement de clé ne doit pas être négligé.

Boites S

Supposons que :

- chaque fonction F_i, paramétrée par k_i, s'écrive :

$$F_i(t) = f_i(t \oplus k_i);$$

- et les f_i soient linéaires dans l'espace vectoriel $(\mathbb{Z}/2\mathbb{Z})^n$.

Alors :

Boites S

Supposons que :

- chaque fonction F_i, paramétrée par k_i, s'écrive :

$$F_i(t) = f_i(t \oplus k_i);$$

- et les f_i soient linéaires dans l'espace vectoriel $(\mathbb{Z}/2\mathbb{Z})^n$.

Alors :

- F_i est affine dans $(\mathbb{Z}/2\mathbb{Z})^n$.

Boites S

Supposons que :

- chaque fonction F_i, paramétrée par k_i, s'écrive :

$$F_i(t) = f_i(t \oplus k_i);$$

- et les f_i soient linéaires dans l'espace vectoriel $(\mathbb{Z}/2\mathbb{Z})^n$.

Alors :

- F_i est affine dans $(\mathbb{Z}/2\mathbb{Z})^n$.
- $\mathcal{R}_{F_i}$ est affine dans $(\mathbb{Z}/2\mathbb{Z})^{2n}$.

Boites S

Supposons que :

- chaque fonction F_i, paramétrée par k_i, s'écrive :

$$F_i(t) = f_i(t \oplus k_i);$$

- et les f_i soient linéaires dans l'espace vectoriel $(\mathbb{Z}/2\mathbb{Z})^n$.

Alors :

- F_i est affine dans $(\mathbb{Z}/2\mathbb{Z})^n$.
- $\mathcal{R}_{F_i}$ est affine dans $(\mathbb{Z}/2\mathbb{Z})^{2n}$.
- DES_K est affine dans $(\mathbb{Z}/2\mathbb{Z})^{2n}$.

Boites S

Supposons que :

- chaque fonction F_i, paramétrée par k_i, s'écrive :

$$F_i(t) = f_i(t \oplus k_i);$$

- et les f_i soient linéaires dans l'espace vectoriel $(\mathbb{Z}/2\mathbb{Z})^n$.

Alors :

- F_i est affine dans $(\mathbb{Z}/2\mathbb{Z})^n$.
- $\mathcal{R}_{F_i}$ est affine dans $(\mathbb{Z}/2\mathbb{Z})^{2n}$.
- DES_K est affine dans $(\mathbb{Z}/2\mathbb{Z})^{2n}$.
- Donc ce n'est pas sûr du tout.

DES spécifie des fonctions f_i particulières, fabriquées pour être non linéaires : ce sont les fameuses boites S (S comme substitution).

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.

Plan

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.

Vu autrement, c'est un key schedule un peu simpliste sur un Feistel à 32 tours, qui fait qu'on peut écrire $E_K = \mathrm{DES}_{K_2} \circ \mathrm{DES}_{K_1}$.

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.

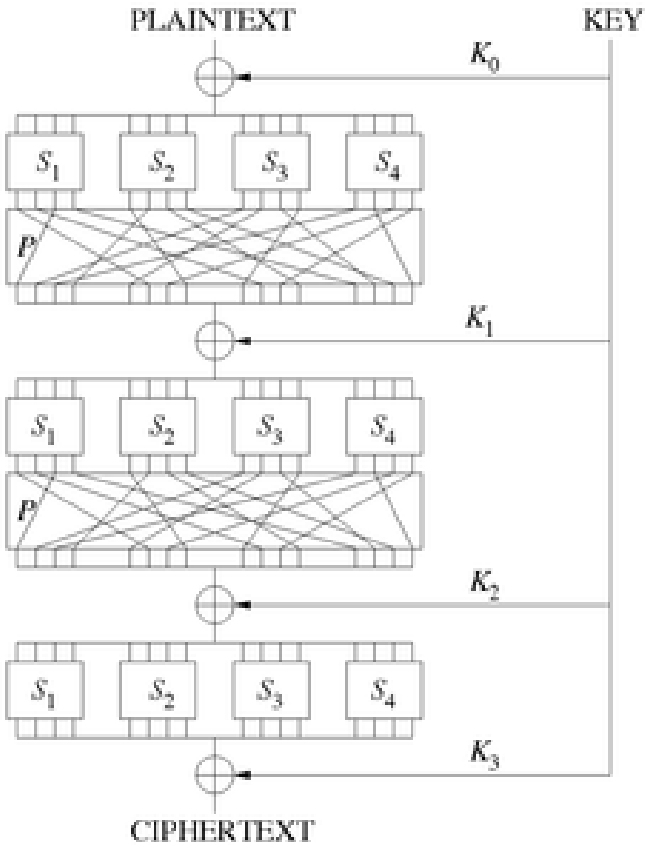
15 propositions, 5 finalistes.
Auteurs de l'algo choisi : Rijmen, Daemen (Leuven, Belgique).

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).



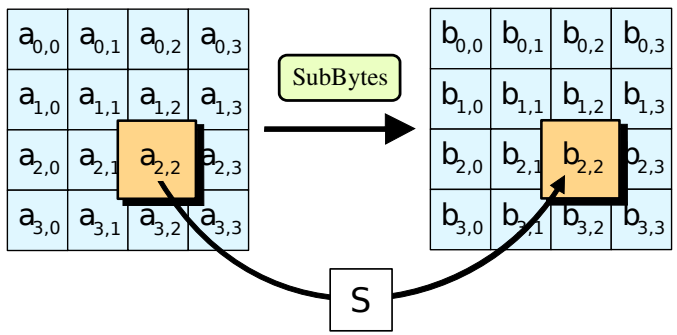
- Comme pour DES, il y a des tours et donc un cadencement de clé.
- À chaque étape la clé de tour est ajoutée (XOR) à la valeur courante.
- Il y a encore des boîtes S qui apportent la non-linéarité.

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.

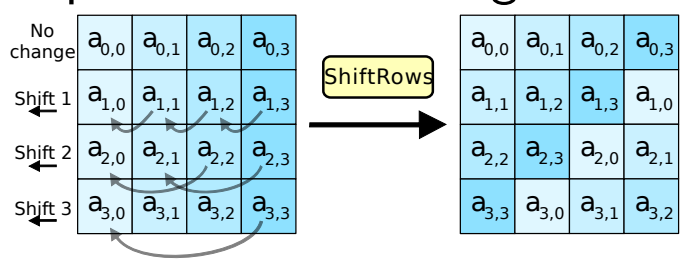


AES

- 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.

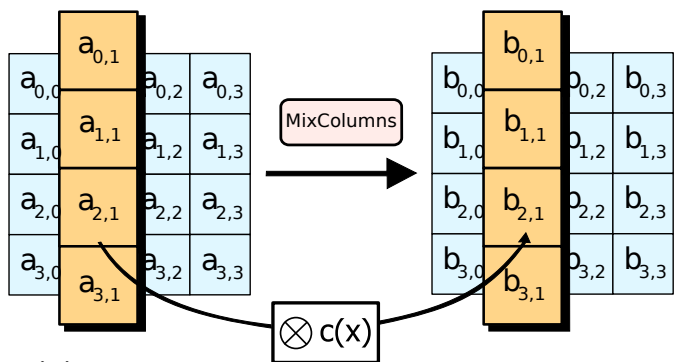


AES

- 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.

Implantation d'AES

AES a une structure très algébrique :

- Octets = éléments dans $\mathbb{F}_{2^8}$.
- `MixColumns` : une opération sur des polynômes à coefficients dans $\mathbb{F}_{2^8}$.

L'ensemble peut être vu comme une (grosse) fraction rationnelle à coefficients dans $\mathbb{F}_{2^8}$.

$\Rightarrow$ implantation facile.

Depuis 2010 les processeurs incluent des instructions spécifiques, destinées à aider AES.

- Instruction hardware pour faire un tour d'AES.
- Instruction hardware pour multiplier deux polynômes à coefficients dans $\mathbb{F}_2$.

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.

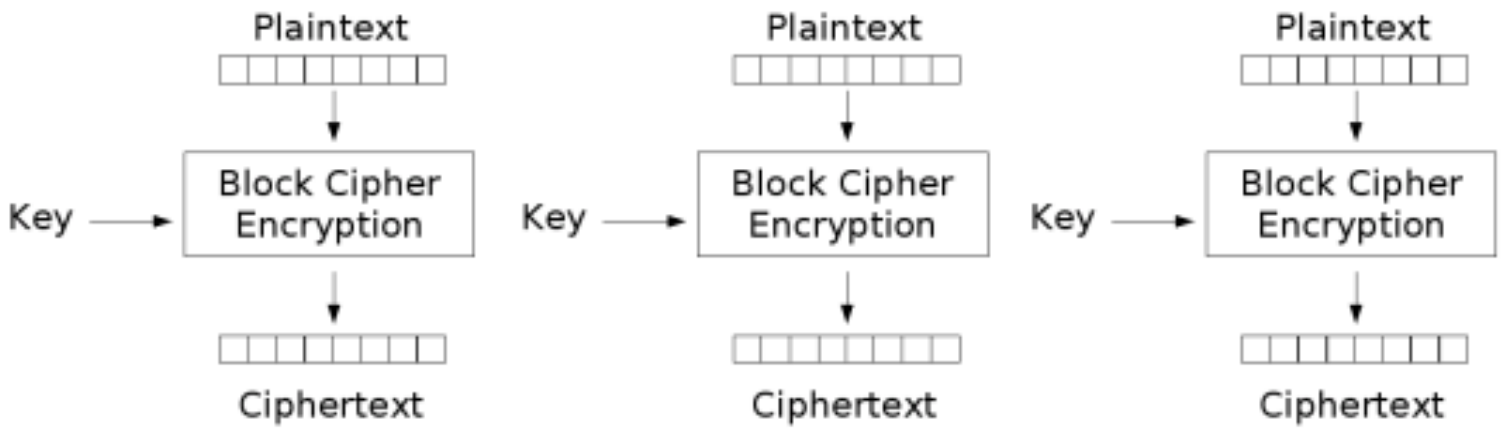
Plan

Modes opératoires : ECB

Pour chiffrer un texte de taille variable avec un block cipher :

Idée immédiate : chiffrer chaque bloc du texte indépendamment.

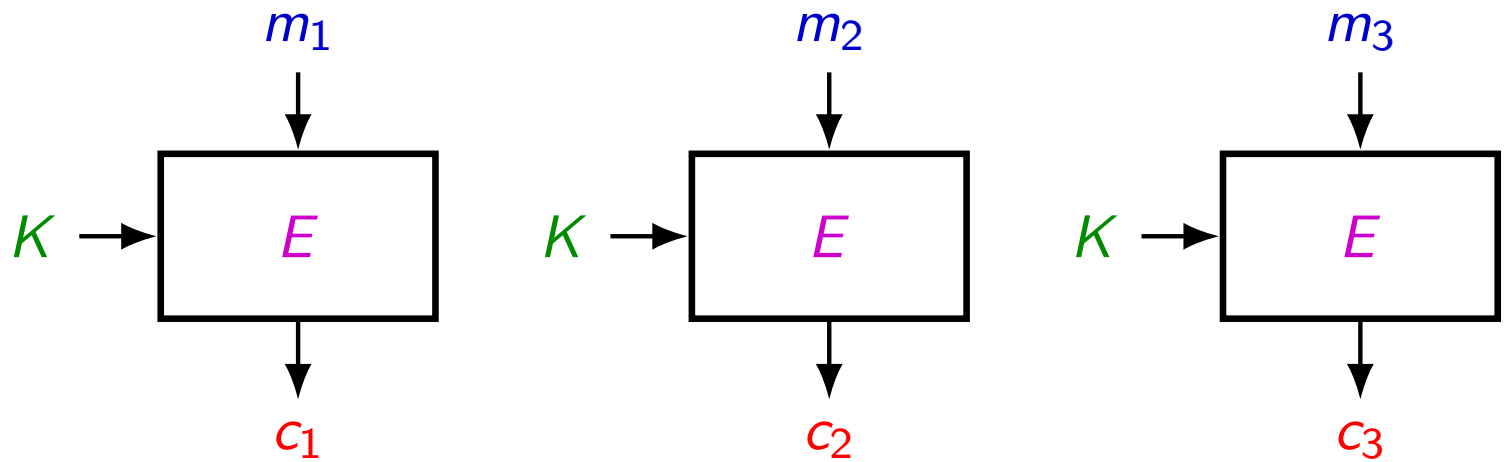
$$\text{message} = m_0 m_1 \ldots,$$
$$\text{chiffré} = c_0 c_1 \ldots, \text{ avec } c_i = E_K(m_i).$$



Electronic Codebook (ECB) mode encryption

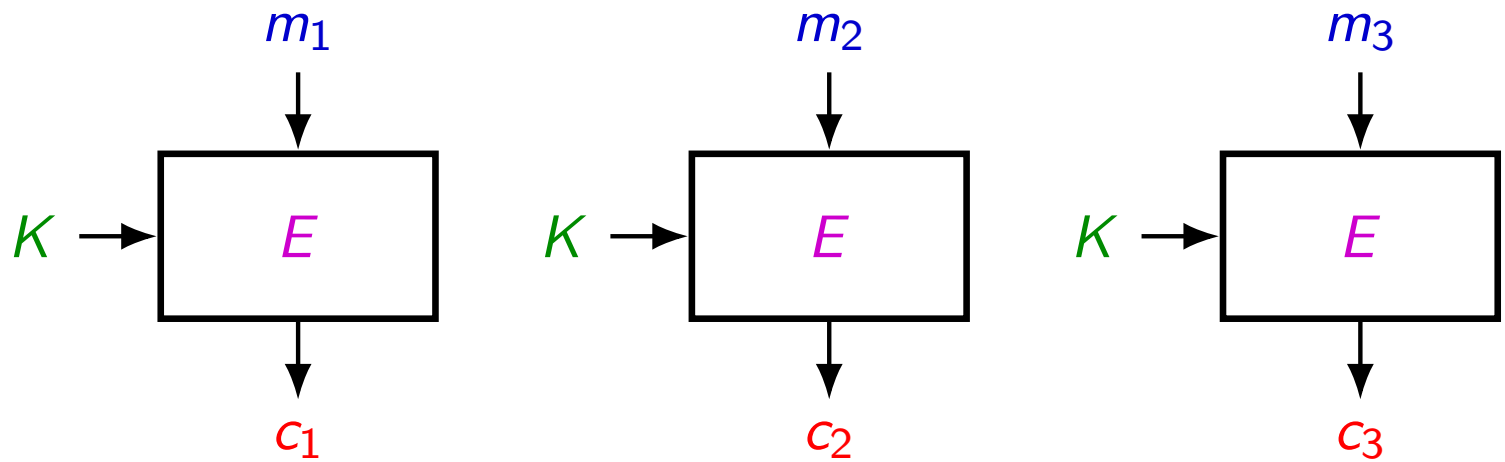
Mode de chiffrement ECB

- Chiffrement :

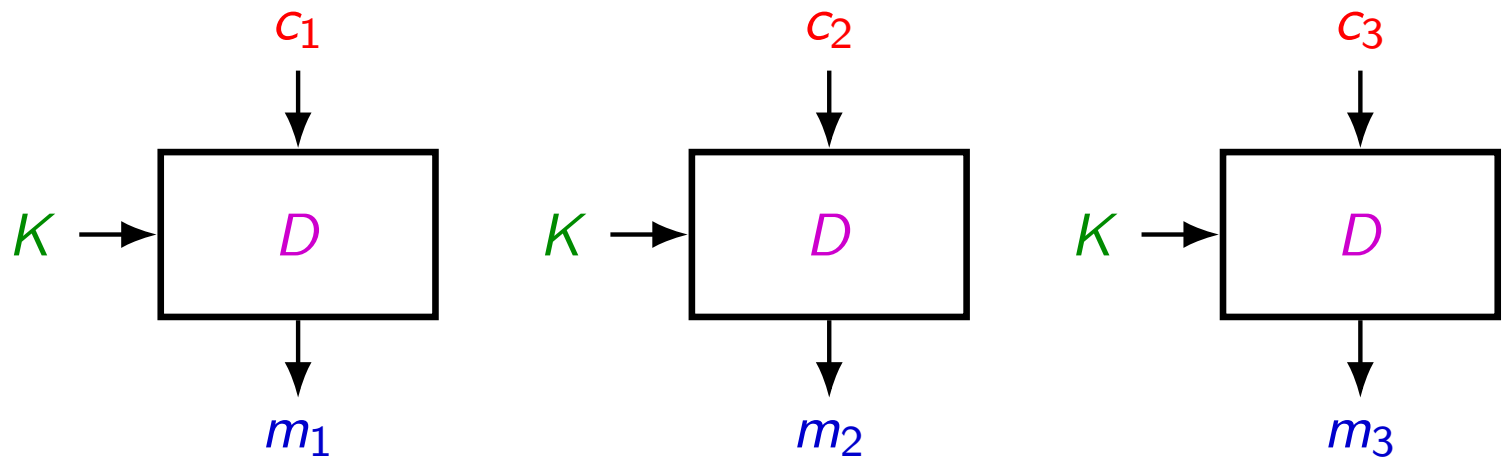


Mode de chiffrement ECB

- 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
```

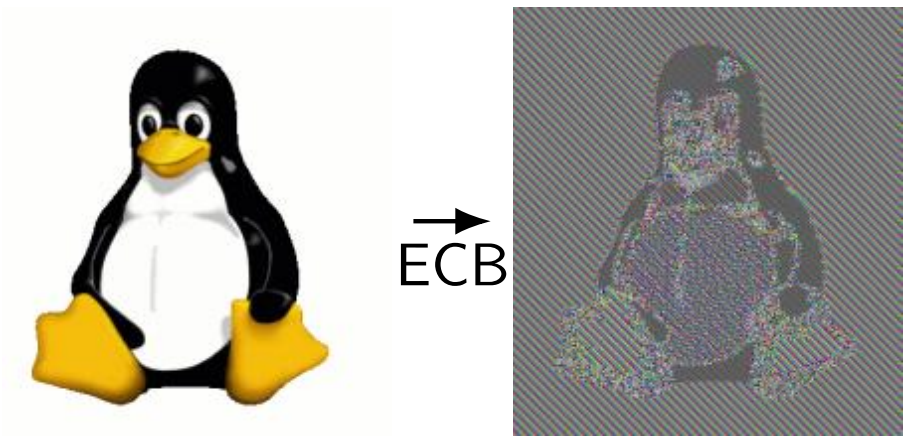
De la sorte, on fait fuiter de l'information.

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

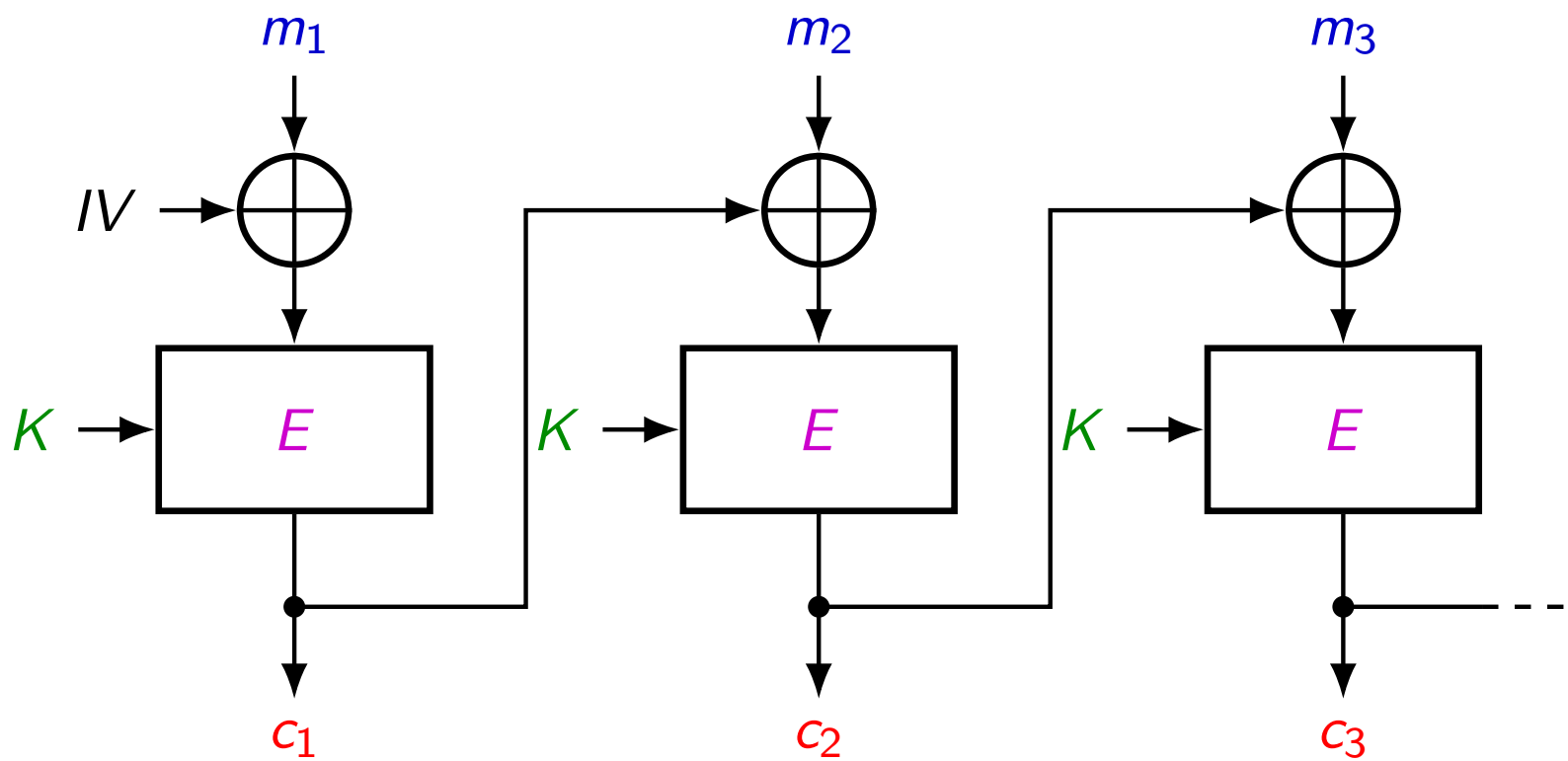
L'information qui fuite avec ECB n'est pas acceptable.

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

- Chaînage ;
- Paramétrisation de chaque bloc.

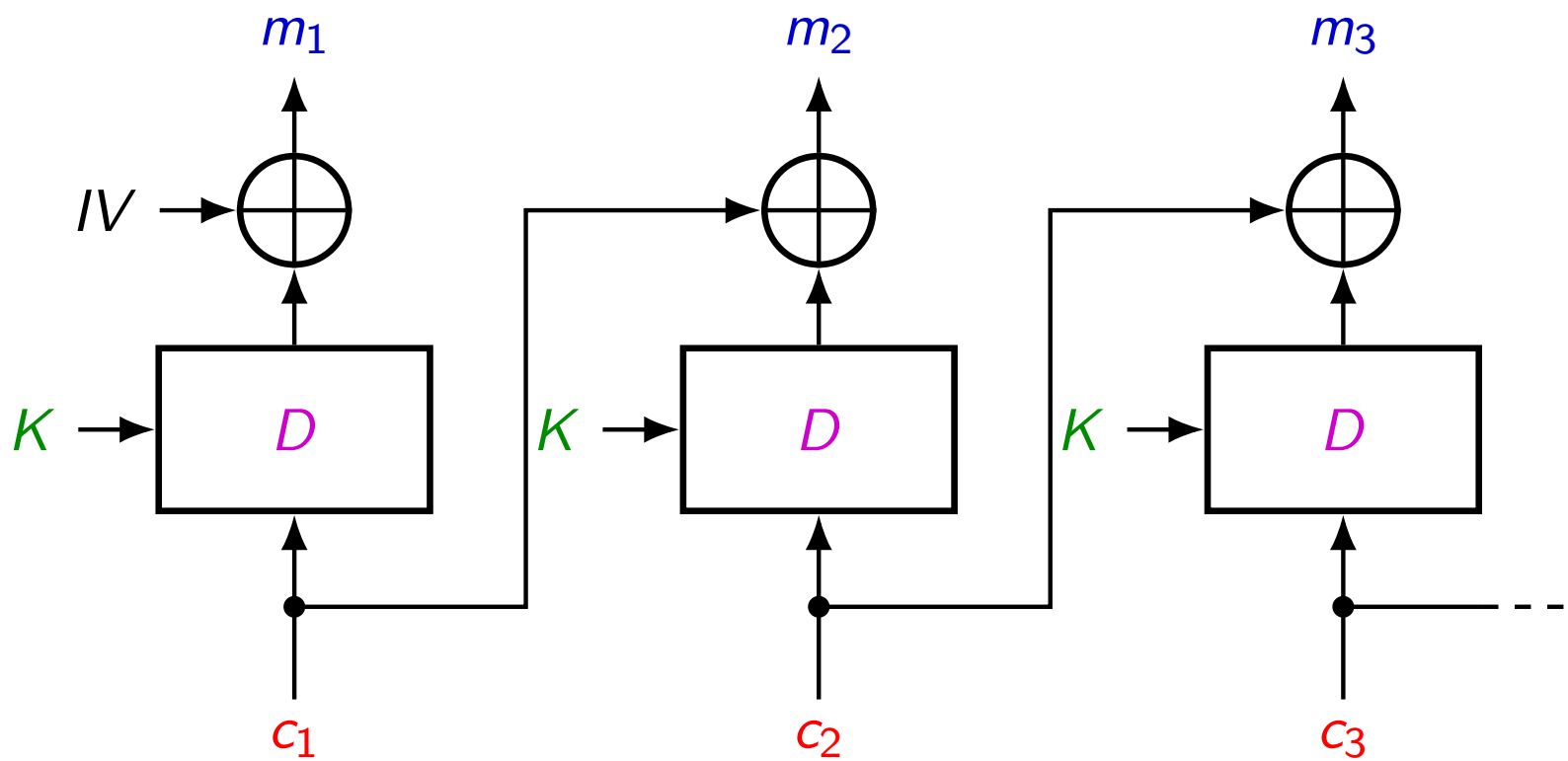
Mode de chiffrement CBC

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



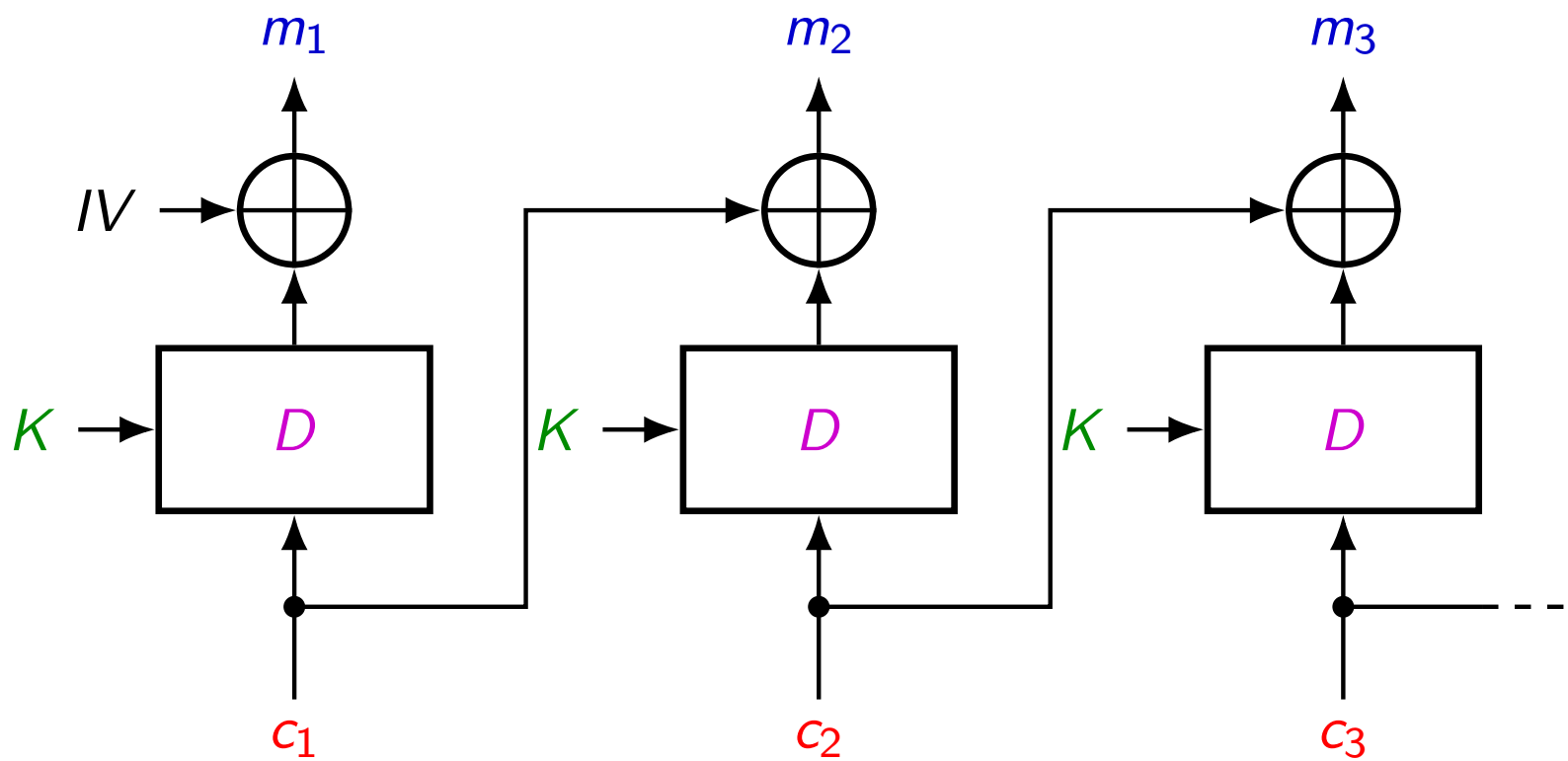
Mode de chiffrement CBC

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



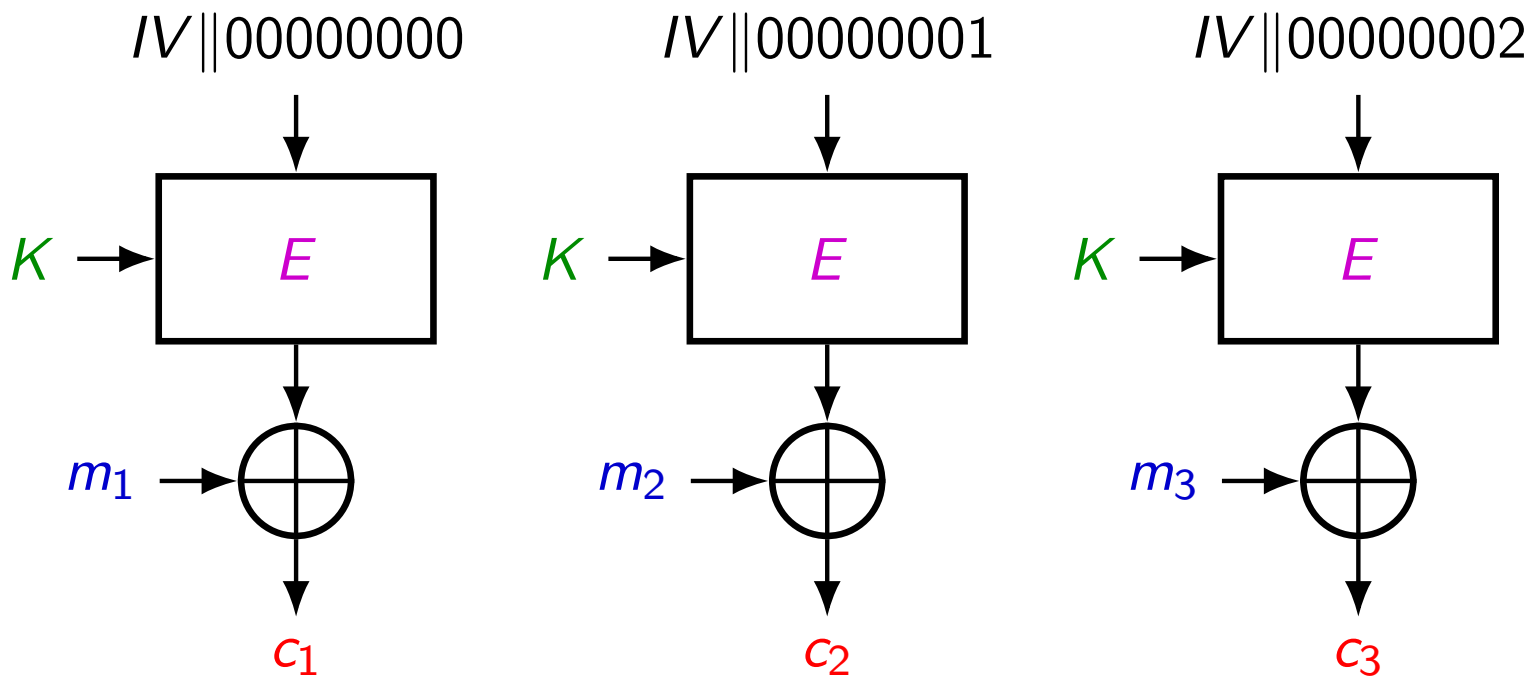
Mode de chiffrement CBC

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



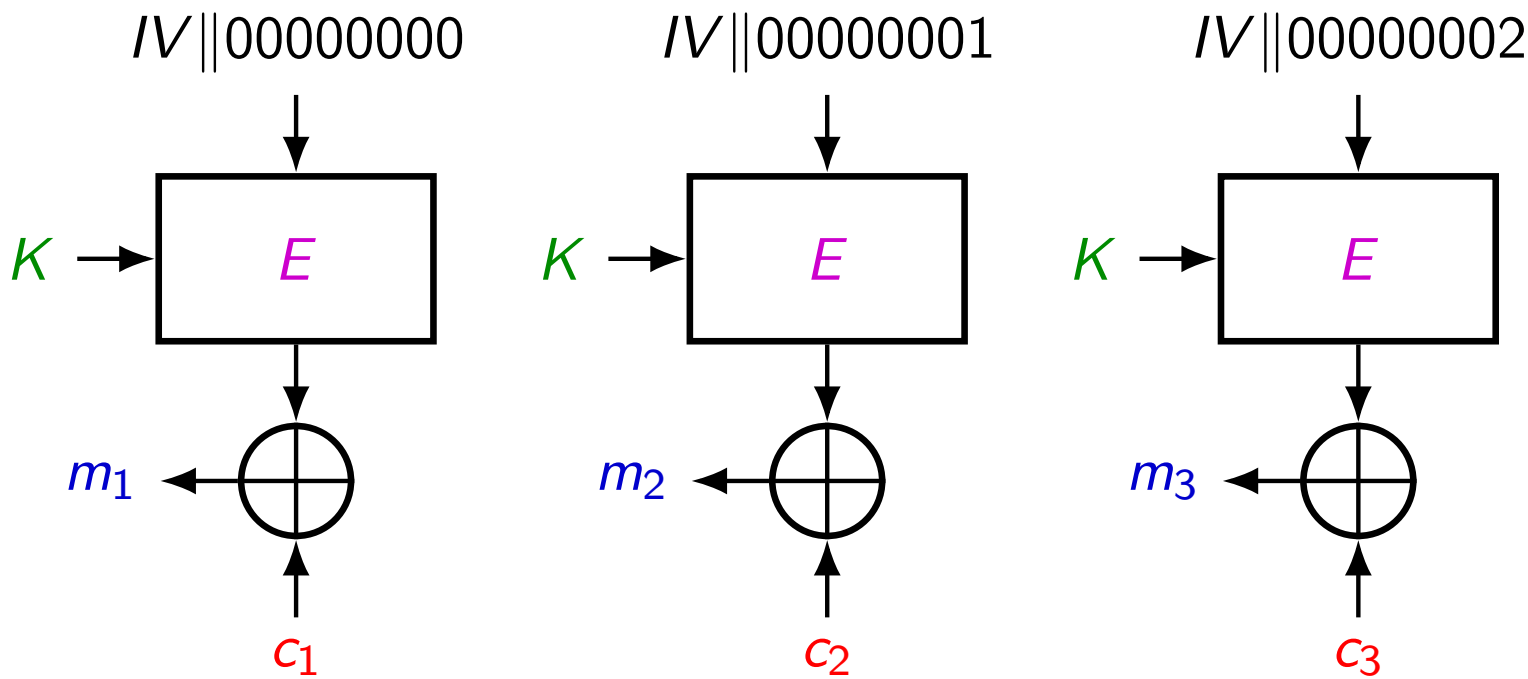
Mode de chiffrement CTR

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



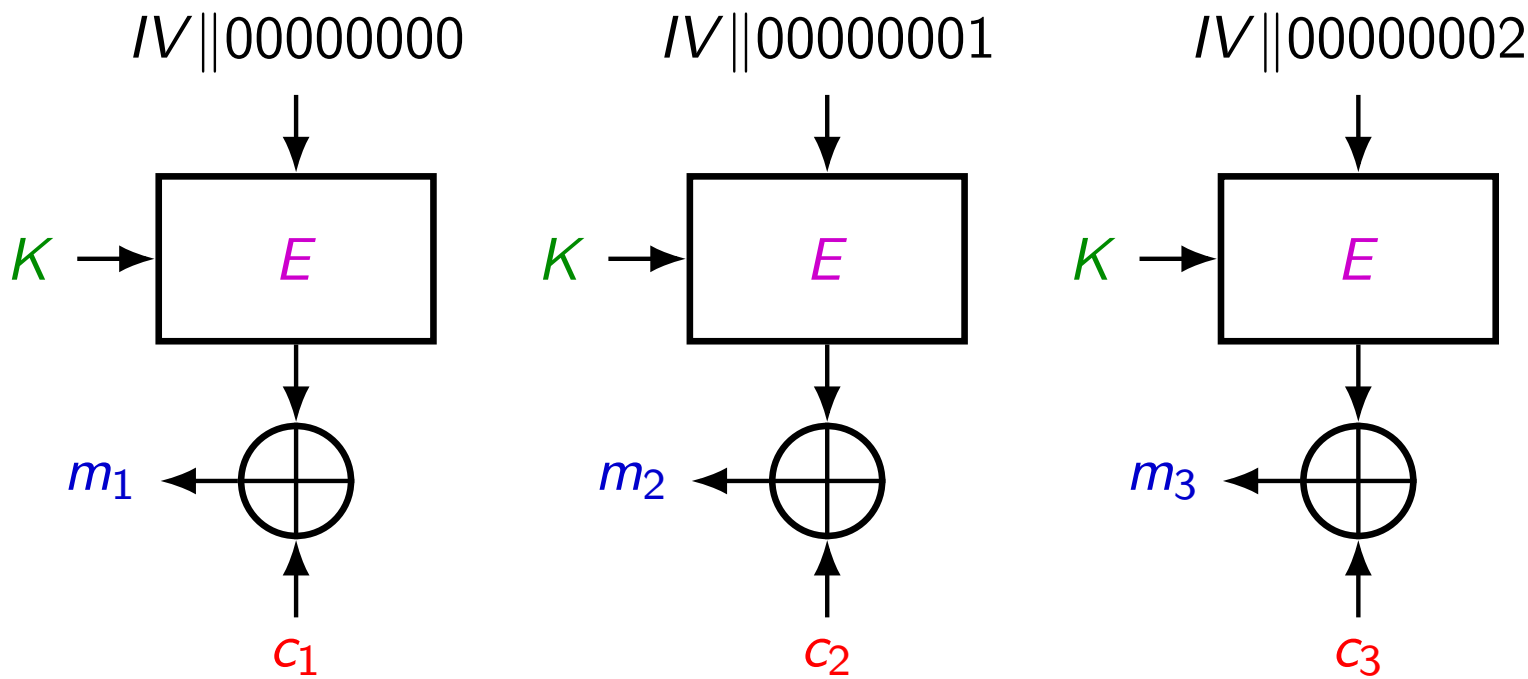
Mode de chiffrement CTR

- 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 : 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.