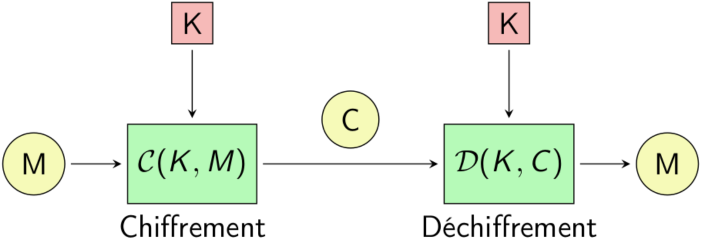
Cryptographie symétrique

Principe

Vue d'ensemble et caractéristiques



- Chiffrement rapide, voire très rapide en implantation matérielle.
- Clés plutôt courtes : 128 - 256 bits (à comparer avec RSA : 1024 - 2048 bits).
- Inconvénient majeur : partage d'un secret (la clé commune K), ce qui est toujours délicat à gérer.

Théorie de Shannon

- En 1949, *Claude Shannon*, dans son fameux article fondateur de la cryptographie moderne (*Communication Theory of Secrecy Systems*), introduit deux propriétés que devrait satisfaire un bon algorithme de chiffrement : la **diffusion** et la **confusion**.
- La propriété de *diffusion* signifie que des changements **minimes** dans les données en entrée se traduisent par des changements **importants** dans les données en sortie.
- La propriété de *confusion* mesure la **complexité** de l'interdépendance entre la *clé*, le *clair* et le *chiffré*. Plus cette complexité est grande, meilleur est l'algorithme.
- En pratique, la diffusion est un processus essentiellement **linéaire** alors qu'au contraire la confusion s'appuie sur des opérateurs **non-linéaires** comme les *S-Boxes*.

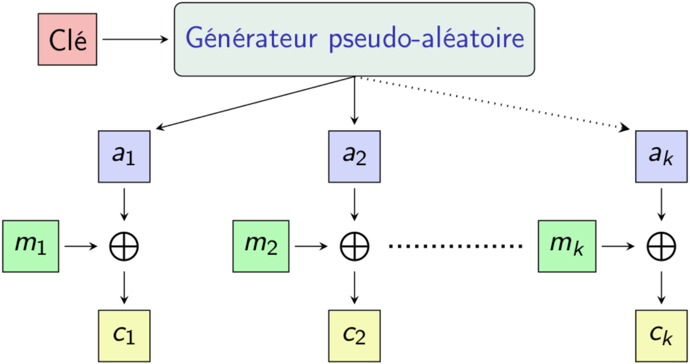
Les deux types de chiffrements symétriques

Il existe deux grandes familles de chiffrements symétriques :

- Les **chiffrements à flot** - on génère à partir de la clé une suite chiffrante pseudo-aléatoire de même longueur que les données. On combine cette suite, par exemple avec un XOR bit-à-bit, avec les données à chiffer.
  - RC4 (Rivest, 1987) : SSL/TLS, WEP, WPA, WPA2,...
  - E0 : Bluetooth
  - A5 : GSM
- Les **chiffrements par bloc** - les données à chiffrer sont découpées en *blocs* de taille fixe (typiquement 64 ou 128 bits). Les blocs sont chiffrés séparément et ensuite combinés selon un *mode opératoire* (ECB, CBC, CTR...).
  - DES (IBM et NSA, 1975) - blocs 64 bits, clés 56 bits
  - IDEA (Lai-Massey, 1992) - blocs 64 bits, clés 128 bits
  - AES/Rijndael (Daemen-Rijmen, 1998) - blocs/clés 128, 192, 256 bits
  - et aussi : RC5, RC6, Camellia,...

Les chiffrements à flots

Schéma de fonctionnement



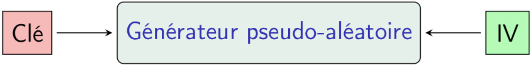
- Des données pseudo-aléatoires, appelées **flux de clé** (*keystream*), sont générées et combinées (le plus souvent avec un XOR) aux données en clair pour produire les données chiffrées.

Caractéristiques

- Les chiffrements à flot sont **très rapides**, les implantations matérielles étant particulièrement efficaces.
- On chiffre à la volée sans attendre d'avoir lu, tout ou une partie, des données : bien adapté aux applications **temps réel**.
- Ils sont très utilisés pour la protection des données multimedia.
- Ce chiffrement est adapté du chiffrement de *Vernam* (théoriquement inviolable) sauf qu'ici on génère une suite **pseudo-aléatoire** à partir d'une clé **aléatoire**, généralement de petite taille comparée à la taille des données. Le chiffrement de *Vernam*, quant à lui, utilise une clé aléatoire à **usage unique** de même taille que les données à chiffrer.
- Le chiffrement de *Vernam* est certes sûr mais, en pratique, très difficile à mettre en œuvre. Le chiffrement à flot peut être vu comme un “pseudo-Vernam” adapté à un usage concret.

Sécurité

- Une première difficulté : si on chiffre plusieurs messages avec la **même clé**, on génère le **même aléa**. On ajoute donc une entrée auxiliaire, le vecteur d'initialisation (*Initialization Vector - IV*) :



- Le générateur pseudo-aléatoire doit être de bonne qualité. Idéalement :
  1. chaque bit de sortie vaut 0 ou 1 avec une probabilité $\frac{1}{2}$,
  2. aucun bit de sortie n'est corrélé avec les précédents ou les suivants,
  3. la période du générateur est suffisamment longue.

  Problème : ces trois conditions sont extrêmement difficiles à satisfaire en pratique.

- En l'absence d'une assise théorique solide, la **sécurité** des chiffrements à flot reste **problématique**. Pour preuve : le nombre important d'algorithmes proposés qui ont été cassés...

Linear Feedback Shift Register (LFSR)

Rappel mathématique : Le corps fini à deux éléments $\mathbb{F}_2$ est l'ensemble $\{0, 1\}$ muni des opérations :

- $+$: l'addition modulo 2
- $\times$: la multiplication modulo 2

Remarque : l'addition modulo 2 correspond au *XOR*, et la multiplication modulo 2 correspond au *AND*.

Linear Feedback Shift Register (LFSR)

Définition : Linear Feedback Shift Register

Un **LFSR** de taille n est défini par :

- un **état initial** : $\overrightarrow{\mathbf{r}_0} = (r_{n-1}, r_{n-2}, ..., r_1, r_0) \in \mathbb{F}_2^n$
- un **polynôme de rétroaction** :
  $P(X) = 1 + c_1 X + c_2 X^2 + ... + c_n X^n \in \mathbb{F}_2[X]_{\leq n}$

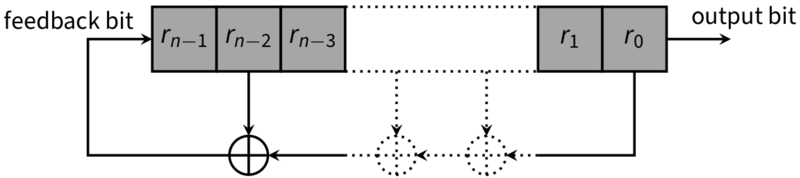
La séquence des registres $(\overrightarrow{\mathbf{r}_t})_{t \geq 0}$ est calculé itérativement. À chaque instant $t \geq 0$, $\overrightarrow{\mathbf{r}_{t+1}} = (r_{t+n}, r_{t+n-1}, ..., r_{t+2}, r_{t+1}) \in \mathbb{F}_2^n$ où

$$r_{t+n} = \sum_{i=1}^{n} c_i \times r_{t+n-i}$$

La suite chiffrante est $(r_t)_{t \geq 0} = (r_0, r_1, r_2, ..., r_t, ...)$

Linear Feedback Shift Register (LFSR)

Les LFSRs sont simples à mettre en oeuvre, notamment grâce à une représentation de ceux-ci qui utilise des circuits logiques :



Linear Feedback Shift Register (LFSR)

Périodicité

On veut éviter de répéter le même motif dans la suite chiffrante.

La valeur du registre d'un LFSR est dans $\mathbb{F}_2^n$ qui est de cardinalité 2^n.
Le nombre de valeur que peut prendre le registre est donc **fini**.

Théorème : Période d'un LFSR

La période maximale d'un LFSR est $2^n - 1$.

De plus, si le polynôme de rétroaction est de degré n et irréductible, alors le LFSR atteint la période maximale $2^n - 1$ pour tout registre initial non nul.

Linear Feedback Shift Register (LFSR)

Attaque

Berlekamp-Massey

L'**algorithme de Berlekamp-Massey** peut retrouver le polynôme de rétroaction en connaissant seulement $2n$ bits de la suite chiffrante.

Conséquences : Attaques dans les modèles KPA, CPA, CCA ou même tout autre modèle d'attaquant où une paire texte en clair/texte chiffré est connue.

Exemple du chiffrement à flots : A5/1

A5/1 est un algorithme de chiffrement standard du GSM. Même s'il est aujourd'hui cassé, il est toujours utilisé notamment en Europe et en Afrique.

A5/1 combine la sortie de 3 LFSRs $\mathcal{L}_1$, $\mathcal{L}_2$ et $\mathcal{L}_3$ ayant respectivement les polynômes de rétroaction suivants :

$$
\begin{aligned}
P_1(X) &= 1 + X^{14} + X^{17} + X^{18} + X^{19} \\
P_2(X) &= 1 + X^{21} + X^{22} \\
P_3(X) &= 1 + X^{8} + X^{21} + X^{22} + X^{23}
\end{aligned}
$$

Exemple du chiffrement à flots : A5/1

Les LFSRs ne sont **pas incrémentés de manière synchrone**.
Un bit par registre permet de savoir si un LFSR doit être incrémenté ou non à un instant t.

Ces bits sont appelés *clock bits* et sont :

- h_1 : le 9^e bit du registre de $\mathcal{L}_1$;
- h_2 : le 11^e bit du registre de $\mathcal{L}_2$;
- h_3 : le 11^e bit du registre de $\mathcal{L}_3$;

$$majority(h_1, h_2, h3) = \begin{cases} 1 & \text{si } h_1 + h_2 + h3 \geq 2 \\ 0 & \text{si } h_1 + h_2 + h3 < 2 \end{cases}$$

Si $h_i = majority(h_1, h_2, h3)$ alors le registre de $\mathcal{L}_i$ sera mis à jour à l'instant suivant. Sinon, il reste inchangé.

Les chiffrements par bloc

Les chiffrements par bloc : principe général

- L'idée maitresse : une fonction de chiffrement f_K (K étant la clé secrète) est construite par **itérations successives** d'une fonction simple g_K : $f_K = g_K^d = \underbrace{g_K \circ \cdots \circ g_K}_{d \text{ fois}}$.
- On sait depuis l'étude des *systèmes dynamiques* que le comportement de g^d peut être **imprévisible** (pour d assez grand), même si g est très simple.
- Plus précisément, on dérive $K_1, \ldots, K_d$ sous-clés de la clé principale K et $f_K = g_{K_d} \circ \cdots \circ g_{K_1}$ (algorithme de dérivation de sous-clés).
- Chaque fonction de tour g_{K_i} est optimisée : opérations simples.
- Les algorithmes de chiffrement par bloc sont **performants** et leur **sécurité bien étudiée**.
- Inconvénient : nécessité d'avoir recours à l'utilisation de **modes opératoires**.

Réseau de Feistel - Définitions et propriétés (1)

- Un réseau de Feistel, du nom de son inventeur *Horst Feistel* cryptologue chez IBM, est un dispositif général de chiffrement par blocs au cœur de nombre d'algorithmes symétriques comme : DES, 3DES, Blowfish, Twofish, Camellia, SEED, RC5, OAEP,...
- Soit $\mathcal{M} = \{0,1\}^{2n}$, l'ensemble des messages à chiffrer. On va définir une fonction de chiffrement de $\mathcal{M}$ vers l'ensemble des messages chiffrés $\mathcal{C} = \mathcal{M}$. La clé secrète est un ensemble $K = \{K_1, \ldots, K_d\}$ avec $K_i \in \{0,1\}^k$. On se donne également une fonction quelconque $F : \{0,1\}^{k+n} \to \{0,1\}^n$.
- Soit $M \in \mathcal{M}$ un message en clair. On le découpe en deux blocs de même longueur (il existe des variantes avec des longueurs différentes) $M = (L, R)$ avec $L, R \in \{0,1\}^n$. On définit par récurrence la suite finie $(L_i, R_i)_{0 \leq i \leq d}$ comme suit :

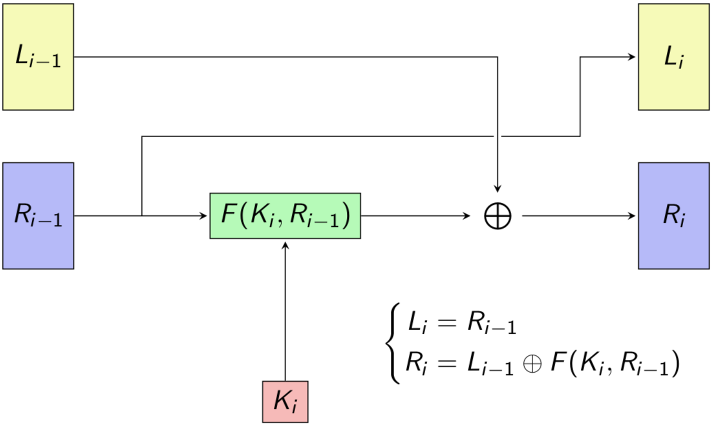
$$\begin{cases} (L_0, R_0) & = & (L, R), \\ (L_i, R_i) & = & (R_{i-1}, L_{i-1} \oplus F(K_i, R_{i-1})), \text{ pour } 1 \leq i \leq d. \end{cases}$$

Réseau de Feistel - Définitions et propriétés (2)

- Le chiffré du message $M = (L, R)$ est alors $C = (L_d, R_d)$.
- L'avantage principal d'un réseau de Feistel est que, pour déchiffrer un message, il n'est pas nécessaire de supposer inversibles les fonctions $F_i : X \to F(K_i, X)$, pour $1 \leq i \leq d$, et de savoir les inverser.
- En effet, on montre aisément que : $(R_{i-1}, L_{i-1}) = (L_i, R_i \oplus F(K_i, L_i))$.
- Un exemple : le DES (Digital Encryption Standard). Ici on a $n = 32$, $d = 16$ et $k = 48$. La clé secrète $K = \{K_1, \ldots, K_{16}\}$ est déduite d'une clé maître de 56 bits (cassable aujourd'hui en moins de 24h).
- Enfin on a : $F_i(X) = P(S(L(X)) \oplus K_i)$ où :
  1. L est une application linéaire de $\mathbb{F}_2^{32}$ dans $\mathbb{F}_2$,
  2. S est une application non linéaire (*S-Box*) de $\{0,1\}^{48}$ dans $\{0,1\}^{32}$,
  3. P une permutation de $\{0,1\}^{32}$.

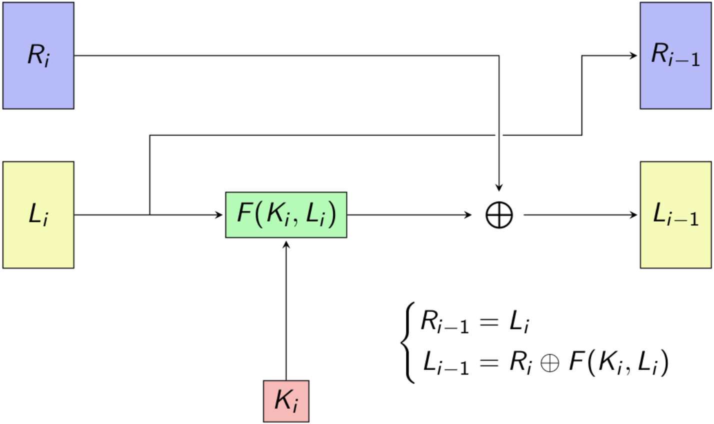
CY TECH

Schéma d'un tour - Chiffrement



$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus F(K_i, R_{i-1}) \end{cases}$$

Schéma d'un tour - Déchiffrement



$$\begin{cases} R_{i-1} = L_i \\ L_{i-1} = R_i \oplus F(K_i, L_i) \end{cases}$$

Le triple DES

- L'algorithme DES, standardisé en 1976, n'est plus utilisé aujourd'hui, non pas en raison d'une faiblesse structurelle (jamais découverte à ce jour), mais à cause de sa clé trop courte (56 bits, longueur imposée par la NSA, alors qu'initialement elle était de 64 bits).
- Son successeur est le triple DES ou 3DES (clé de 3×56 bits). Il existe plusieurs variantes dont celle recommandée par le NIST et répandue dans le monde bancaire. Elle nécessite trois (ou deux) clés k_1, k_2 et k_3 ($k_3 = k_1$ dans le cas de deux clés) :

$$\longrightarrow \boxed{\mathrm{DES}_{k_1}} \longrightarrow \boxed{\mathrm{DES}_{k_2}^{-1}} \longrightarrow \boxed{\mathrm{DES}_{k_3}} \longrightarrow$$

- Inconvénient : 3DES est trois plus lent que DES.
- Alors pourquoi pas simplement un double DES ? Parce qu'il existe une attaque permettant de retrouver les deux clés secrètes, de complexité en temps 2^{57} au lieu de 2^{112} attendu (force brute).

Advanced Encryption Standard (AES) - Historique et Description

- AES est un algorithme de **chiffrement symétrique par bloc**. Il a remporté en 2000 le concours lancé par le NIST en 1997.
- Il y avait 15 participants à ce concours et c'est la proposition *Rijndael*, du nom de ses concepteurs belges *Joan Daemen* et *Vincent Rijmen*, qui a été retenue pour succéder au DES.
- AES est un sous-ensemble de la proposition initiale *Rijndael* : la taille des blocs est **fixée à 128 bits** (au lieu d'une taille variable multiple de 32 bits et comprise entre 128 bits et 256 bits).
- AES se décline en trois versions :
  - **AES-128** $\Rightarrow$ clé de 128 bits, 10 tours
  - **AES-192** $\Rightarrow$ clé de 192 bits, 12 tours
  - **AES-256** $\Rightarrow$ clé de 256 bits, 14 tours
- L'originalité d'AES est que la plupart des opérations se font dans le corps fini $\mathbb{F}_{2^8}$

Représentation d'un bloc

- Chaque bloc de 128 bits est découpé en 16 octets $b_0 b_1 \cdots b_{15}$.
- Les 16 octets sont ensuite placés de la manière suivante dans une matrice 4×4 $(a_{i,j})_{0 \leq i,j \leq 3}$:

$$\begin{array}{|c|c|c|c|} \hline b_0 & b_4 & b_8 & b_{12} \\ \hline b_1 & b_5 & b_9 & b_{13} \\ \hline b_2 & b_6 & b_{10} & b_{14} \\ \hline b_3 & b_7 & b_{11} & b_{15} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array}$$

Représentation d'un bloc

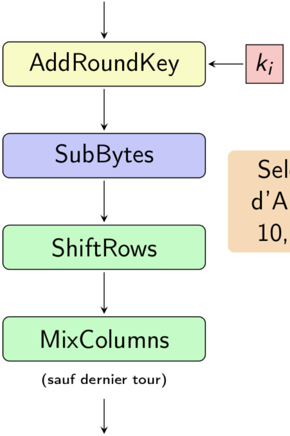
A chaque tour, on effectue quatre opérations sur la matrice 4×4 :

XOR du bloc entrant avec la sous-clé k_i

Substitution (*S-Box*) (*confusion*)

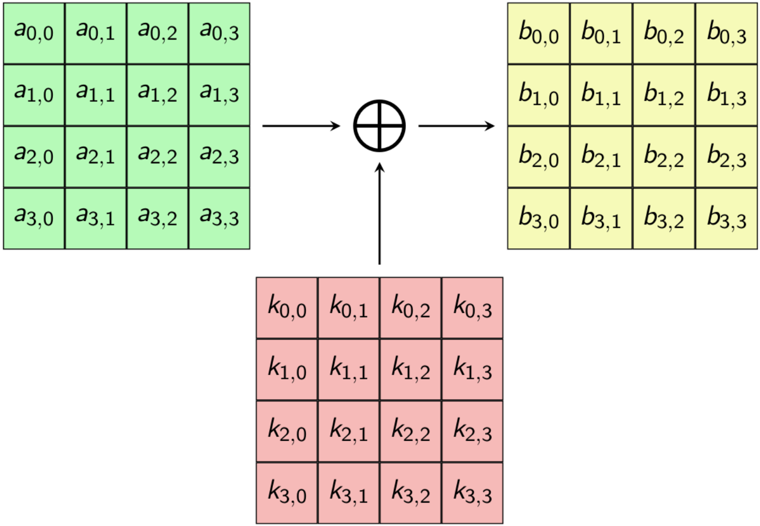
Décalage de lignes (*diffusion*)

Mélange des colonnes (*diffusion*)

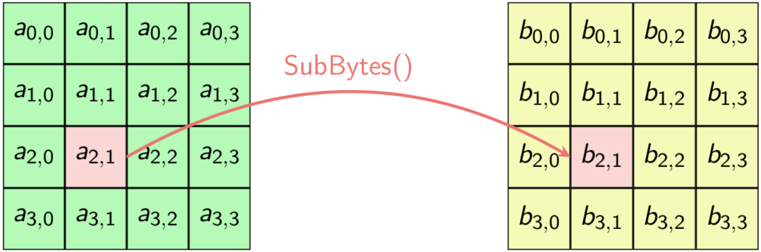


Selon les variantes d'AES, on enchaîne 10, 12 ou 14 tours

L'opérateur *AddRoundKey*



L'opérateur *SubBytes*



- *SubBytes* opère **indépendamment** sur chaque octet. C'est donc une application de $\{0, \ldots, 255\}$ dans lui-même.
- C'est un opérateur non-linéaire $\Rightarrow$ *confusion*.

L'opérateur *ShiftRows*

- Cet opérateur décale cycliquement les lignes d'un bloc. Chaque ligne est décalée d'une valeur différente suivant la taille du bloc sauf la ligne 0 qui reste toujours inchangée. C'est une opération de *diffusion*.
- Dans le cas du chiffrement AES-128, la ligne i est décalée de i octets vers la gauche pour $i = 1, 2, 3$:

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$\longrightarrow$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

L'opérateur *MixColumns*

Cet opérateur est une transformation linéaire (*diffusion*) agissant sur les colonnes de la matrice 4×4 :

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Cette transformation s'écrit (multiplication dans $\mathbb{F}_{2^8}$) :

$$\begin{pmatrix} b_{0,2} \\ b_{1,2} \\ b_{2,2} \\ b_{3,2} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 3 & 1 & 1 & 2 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{pmatrix}$$

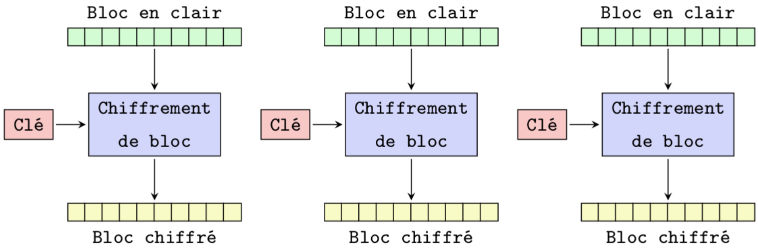
Performance et Sécurité

- AES-128 est presque aussi rapide que DES. Précisément, AES-128 est 2.7 fois plus rapide que 3DES, lui-même 3 fois plus lent que DES.
- AES est donc **très rapide** et en outre **peu gourmand en mémoire**. Cela lui permet d'être efficace sur une grande variété de matériels.
- AES a été conçu pour résister aux attaques classiques comme la cryptanalyse *linéaire* ou *différentielle*.
- La meilleure **attaque**, due à des chercheurs de Microsoft en 2011, ne permet de gagner que 2 bits, soit 2^{126} opérations au lieu de 2^{128} pour une attaque par force brute. Cette attaque est donc **impraticable**.
- Attention : comme pour tout algorithme de chiffrement, AES n'est pas immunisé contre les attaques par canal auxiliaire (*side channel attack*).

Mode opératoire de chiffrement par bloc

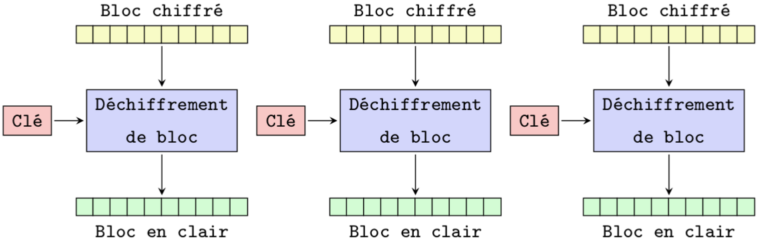
- Jusqu'à présent, on ne s'est préoccupé que du chiffrement d'un bloc. Or un message à chiffrer comporte généralement plusieurs blocs.
- Un mode opératoire est un algorithme, reposant sur un algorithme de chiffrement par bloc (3DES, AES,...), permettant de chiffrer des **données de taille arbitraire**.
- L'idée naïve, consistant à chiffrer chaque bloc indépendamment (mode *Electronic Codebook* - ECB) n'offre pas, comme nous le verrons, suffisamment de garantie quant à la confidentialité.
- Des dizaines de modes ont été proposés depuis la fin des années 70. Pour n'en citer que quelques uns parmi eux : ECB, *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), *Output Feedback* (OFB) et *Counter* (CTR), ce dernier étant dû à Diffie-Hellman (1979).

Mode *Electronic Codebook* (ECB) - Chiffrement



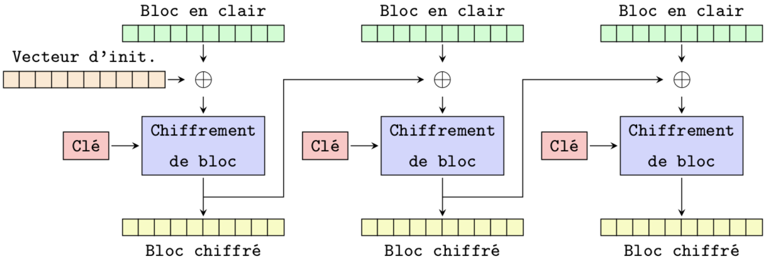
- Chaque bloc est (dé)chiffré **indépendamment** des autres blocs. Avantage : le (dé)chiffrement est parallélisable.
- Inconvénient majeur : deux blocs en clair identiques seront chiffrés de manière identique. Fortement déconseillé pour les applications cryptographiques.

Mode *Electronic Codebook* (ECB) - Déchiffrement



- Le déchiffrement, en ce qui concerne le mode opératoire, est identique au chiffrement.

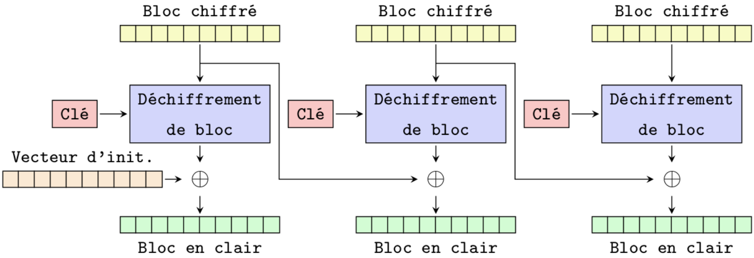
Mode *Cipher Block Chaining* (CBC) - Chiffrement



- Si $(B_i)_{i\geq 1}$, $(C_i)_{i\geq 1}$, ε_K et IV désignent respectivement, la suite des blocs en clair, la suite des blocs chiffrés, la fonction de chiffrement et le vecteur d'initialisation, on a :

$$\begin{cases} C_0 & = & IV, \\ C_i & = & \varepsilon_K(B_i) \oplus C_{i-1}, \text{ pour } i \geq 1. \end{cases}$$

Mode *Cipher Block Chaining* (CBC) - Déchiffrement

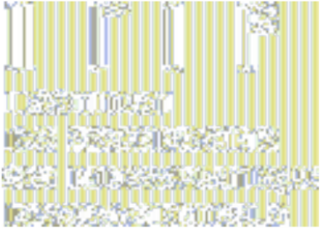


- Si $(B_i)_{i\geq 1}$, $(C_i)_{i\geq 1}$, ε_K^{-1} et IV désignent respectivement, la suite des blocs en clair, la suite des blocs chiffrés, la fonction de déchiffrement et le vecteur d'initialisation, on a :

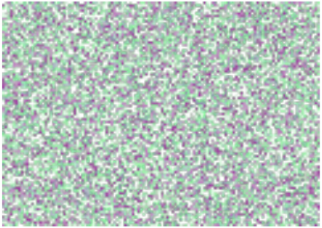
$$\begin{cases} C_0 & = & IV, \\ B_i & = & \varepsilon_K(C_i)^{-1} \oplus C_{i-1}, \text{ pour } i \geq 1. \end{cases}$$

Différents chiffrements du logo du laboratoire IRIF

`logo original`



`chiffré AES-128-ECB`



`chiffré AES-128-CBC`

`chiffré AES-128-CTR`