Cours 4 - Cryptanalyse de Modes d'Opération et Rappels sur l'AES

Maxime Bombar

Rappels de la Semaine Dernière

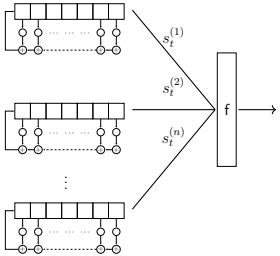
LFSR : Avantages et Inconvénients

- Ils sont extrêmement rapides et peu gourmands en ressources (implémentation matérielle, peu de mémoire).
- La suite produite possèdes de bonnes propriétés statistiques.
- La suite produite a une très grosse période.

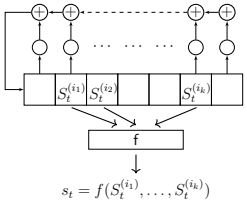
Leur complexité linéaire est logarithmique en la période...
$\Rightarrow$ Application directe de l'algorithme de Berlekamp-Massey.

Augmenter la Complexité Linéaire

On a vu deux façons d'augmenter la complexité linéaire :



LFSR Combinés



LFSR Filtrés

Il faut tenir compte de nouvelles gammes d'attaques (e.g., correlations, algébriques...)

Modes de Chiffrement

Modes d'opération

Contrairement aux chiffrements par flots, les chiffrements par blocs chiffrent des blocs de taille fixée (*e.g.* 128 bits).

Comment chiffrer un message plus gros que la taille d'un bloc ?

Rappels de constructions

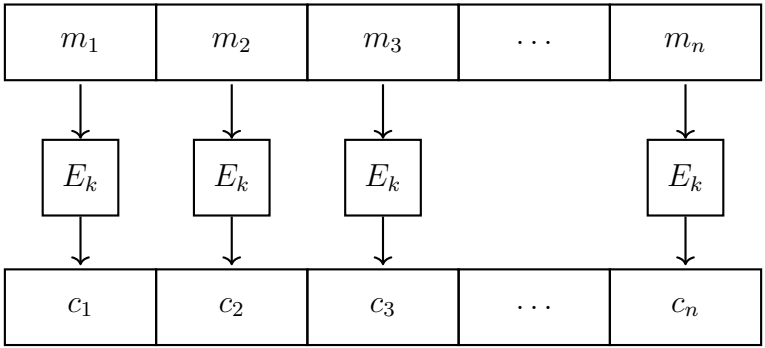
Un schéma cryptographique symétrique est composé de deux objets :

- Une **primitive**, qui va fonctionner sur des objets de taille fixée (*e.g.* chiffrement par blocs, fonction de compression pour faire du hachage)
- Un **mode d'opération**, qui étend la primitive sur des messages de taille arbitraire, en offrant éventuellement des fonctionalités supplémentaires (*e.g.* authentification).

Les modes peuvent être **prouvés** sûrs sous certaines hypothèses sur la primitive sous-jacente (*e.g.* qu'elle est **indistinguable** d'une permutation aléatoire).

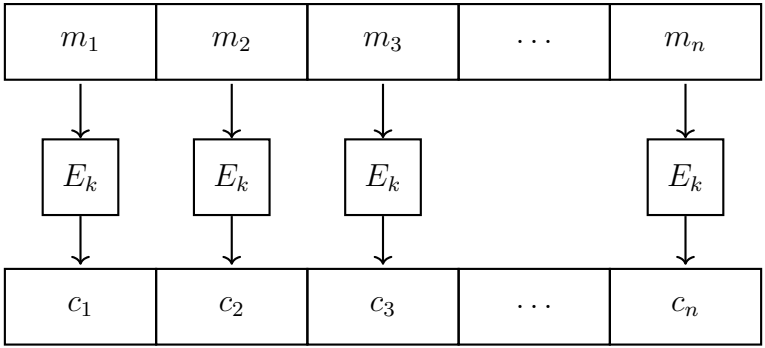
- C'est pour ça que la cryptanalyse se focalise plutôt sur les primitives.
- Attention au bon usage des modes (on verra le cas de la construction de Merkle-Damgard pour les fonctions de hachage).

Mode ECB (*Electronic Code-Book*)



Voyez-vous un potentiel problème ?

Mode ECB (*Electronic Code-Book*)

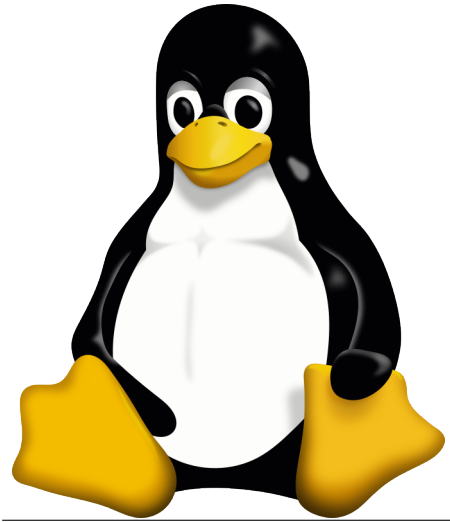


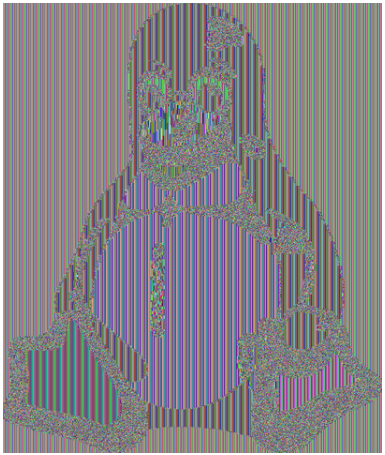
Deux blocs identiques (m, m) vont produire deux blocs identiques (c, c) dans le chiffré

$\rightarrow$ on peut reconnaître des motifs dans le chiffré.

$\rightarrow$ à n'utiliser que sur des données aléatoires....

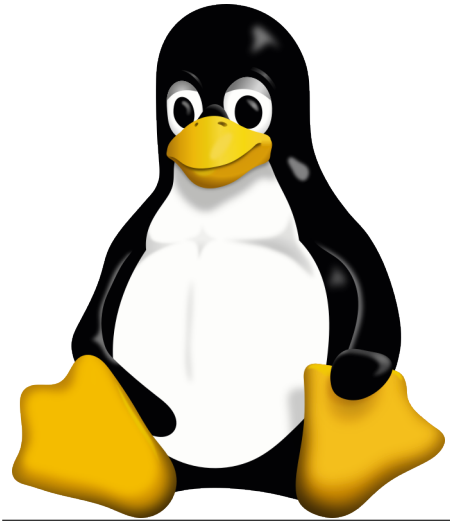
Chiffrement de Tux (Mode ECB) [1]

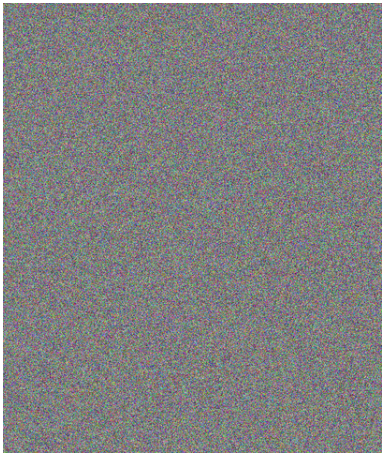




1. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

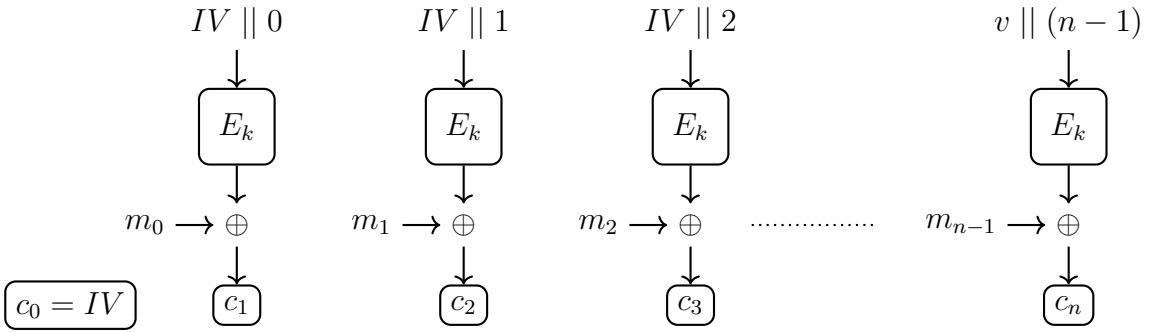
Chiffrement de Tux (Mode CTR) [1]





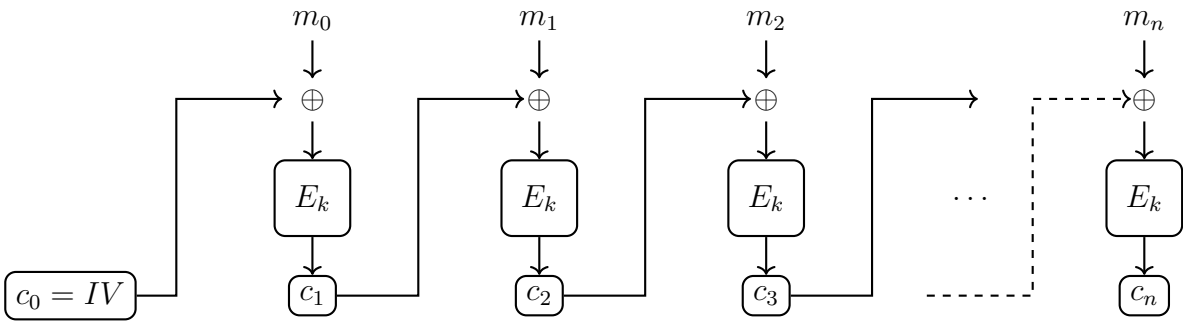
1. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Mode CTR (*Counter Mode*)



On chiffre une valeur initiale IV et un compteur pour obtenir une **suite chiffrante**, à la manière d'un chiffrement par flot.

Mode CBC (*Cipher Block Chaining*)



IV doit être différente pour chaque message, et doit-être **non-prédictible** (cf attaque BEAST [a] sur TLS1.0).

a. Browser Exploit Against SSL/TLS

Attaque par Collision

Imaginons qu'on chiffre N blocs avec la même clé.

On obtient $c_1 = E_k(m_0 \oplus IV)$ et $c_i = E_k(m_{i-1} \oplus c_{i-1})$.

Supposons qu'il existe $\{i, j\}$ tels que $c_i = c_j$. Que se passe-t-il ?

Attaque par Collision

Imaginons qu'on chiffre N blocs avec la même clé.

On obtient $c_1 = E_k(m_0 \oplus IV)$ et $c_i = E_k(m_{i-1} \oplus c_{i-1})$.

Supposons qu'il existe $\{i, j\}$ tels que $c_i = c_j$. Que se passe-t-il ?

$$c_i = c_j \iff E_k(m_{i-1} \oplus c_{i-1}) = E_k(m_{j-1} \oplus c_{j-1}) \iff m_{i-1} \oplus c_{i-1} = m_{j-1} \oplus c_{j-1} \iff c_{i-1} \oplus c_{j-1} = m_{i-1} \oplus m_{j-1}.$$

On obtient le xor entre deux blocs. Applications concrètes lorsque :

- On envoie un même secret de nombreuses fois
- Une partie du clair est connue

e.g. un cookie de session TLS envoyé dans un header connu.

Nombre de chiffrés avant collision ?

Quel est le nombre N de blocs à n bits à chiffrer avant d'espérer observer une collision ?

Le paradoxe des anniversaire

Combien faut-il de personnes dans une salle pour qu'au moins deux personnes aient la même date d'anniversaire ?

Le paradoxe des anniversaire

Combien faut-il de personnes dans une salle pour qu'au moins deux personnes aient la même date d'anniversaire ?

Réponse : $\approx \sqrt{\frac{\pi}{2}} \times \sqrt{366} \approx 24.$

Le paradoxe des anniversaire

Combien faut-il de personnes dans une salle pour qu'au moins deux personnes aient la même date d'anniversaire ?

Réponse : $\approx \sqrt{\frac{\pi}{2}} \times \sqrt{366} \approx 24$.

Il suffit de chiffrer $\approx \sqrt{\frac{\pi}{2}} 2^{n/2}$ blocs pour obtenir une collision.

Par exemple, pour $n = 64$ (*e.g.* 3-DES), il nous faut environ 1.25×2^{32} blocs, soit 64Go de données.

Rappel de proba

Soit X une variable aléatoire à valeurs entières. Alors,

$$\mathbb{E}(X) = \sum_{\ell \geqslant 0} \mathbb{P}(X > \ell).$$

$$\begin{aligned}
\mathbb{E}(X) &= \sum_{k=1}^{\infty} k \cdot \mathbb{P}(X = k) = \sum_{k=1}^{\infty} \sum_{\ell=0}^{k-1} \mathbb{P}(X = k) = \sum_{k=1}^{\infty} \sum_{\ell=0}^{\infty} \mathbb{P}(X = k) \mathbb{1}_{\ell < k} \\
&= \sum_{\ell=0}^{\infty} \sum_{k=1}^{\infty} \mathbb{P}(X = k) \mathbb{1}_{k > \ell} \quad \text{(Tous les termes sont positifs.)} \\
&= \sum_{\ell=0}^{\infty} \sum_{k > \ell}^{\infty} \mathbb{P}(X = k) = \sum_{\ell=0}^{\infty} \mathbb{P}(X > \ell).
\end{aligned}$$

Analyse de l'attaque sur CBC

On modélise notre chiffrement sur un bloc comme une permutation aléatoire de $\{0,1\}^n$, de sorte que les chiffrés sont vus comme des variables aléatoires uniformes et indépendantes sur $\{0,1\}^n$.

Soit X le nombre de chiffrements à faire avant d'observer une collision. Alors, on a

$$\mathbb{E}(X) = \sum_{\ell \geqslant 0} \mathbb{P}(X > \ell).$$

Analyse (Suite)

On veut estimer $\mathbb{P}(X > \ell)$ pour $\ell \geqslant 0$. Soit $N \stackrel{\text{def}}{=} 2^n$.

$$\begin{aligned}
\mathbb{P}(X > \ell) &= \frac{|\{\ell - \text{uplets de } \{0,1\}^n \text{ sans collision}\}|}{N^\ell} \\
&= \frac{N \times (N-1) \times \cdots \times (N-\ell+1)}{N^\ell} \\
&= \prod_{k=1}^{\ell-1} \left(1 - \frac{k}{N}\right) \approx \prod_{k=1}^{\ell-1} e^{-k/N} \\
&= e^{-\frac{\ell(\ell-1)}{2N}}
\end{aligned}$$

Analyse (Fin)

Le nombre moyen de chiffrés avant d'obtenir une collision est

$$\mathbb{E}(X) \approx \sqrt{\pi/2} \cdot 2^{n/2}.$$

Preuve :

$$\mathbb{E}(X) = \sum_{\ell>0} \mathbb{P}(X > \ell) \approx \sum_{\ell>0} e^{-\ell^2 \cdot 2^{-n-1}} \approx \int_0^\infty e^{-x^2 \cdot 2^{-n-1}} dx = \sqrt{\pi/2} \cdot 2^{n/2}.$$

Conséquences, et plus encore

- Cette attaque a été démontrée en pratique en 2016 (SWEET32).
- Il faut changer la clé bien avant avoir chiffré $2^{n/2}$ blocs.
- Il faut utiliser des chiffrements sur des blocs d'au moins 128 bits.
- **Remarque :** ce type d'« attaque des anniversaires » s'adapte à de nombreux modes.

Vers crypto avancée

On peut démontrer que le mode CBC est IND-CPA si la primitive de chiffrement par blocs est une permutation aléatoire, et qu'on ne chiffre qu'au plus $2^{n/2}$ blocs.

Attention : ça ne dit rien contre les problèmes d'**authentification** ou d'**intégrité** !

Cryptanalyse du mode CBC par canaux auxiliaires

En TD : Attaque de Vaudenay (S. Vaudenay 2002)

Vous implémenterez une attaque dévastatrice contre le mode CBC s'il est utilisé avec un mauvais système de padding. Il s'agira de l'une des seules attaques par canaux auxilaires que l'on verra dans ce cours.

Modes d'authentification et d'intégrité

Message Authentication Code (MAC)

Def : On appelle MAC (parfois aussi appelé tag) une petite quantité de données utilisée pour assurer l'intégrité d'un message, et vérifier qu'il n'a subit aucune modification après transmission.

Remarque : On peut construire des MAC à l'aide de fonctions de hachage (on en reparlera), mais aussi comme un mode de chiffrement par blocs.

La cryptanalyse d'un MAC consiste à produire un tag t valide pour un nouveau message m n'ayant jamais été précédemment envoyé.

MAC (formel)

Formellement, un MAC est un triplet d'algorithmes `(Gen, Mac, Verif)` tels que

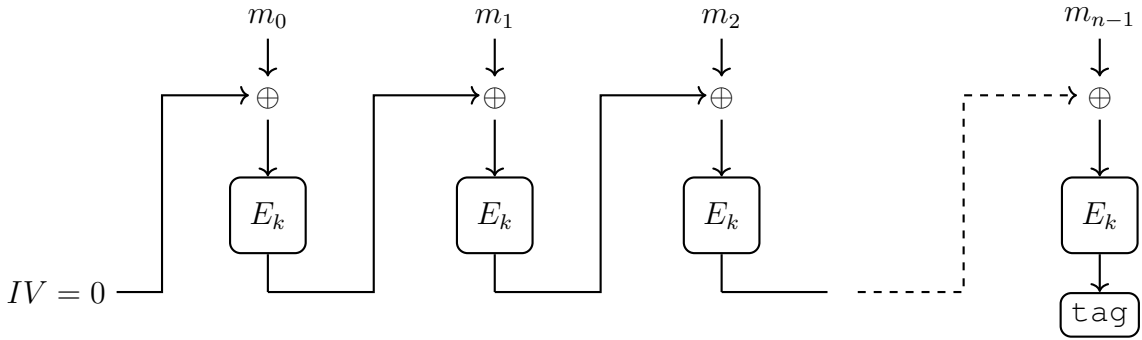
- `Gen` génère une clé K.
- `Mac` prend en entrées un message $m \in \{0,1\}^*$ et une clé K et renvoie un tag t (éventuellement randomisé).
- `Verif` prend en entrées un message $m \in \{0,1\}^*$, une clé K et un tag t, et renvoie un booléen vérifiant la validité du tag sur le message.

Pour chaque clé K générée par `Gen`, et pour chaque message m, il faut que

$$\texttt{Verif}(m, K, \texttt{Mac}(m, K)) = \texttt{True}$$

et ce, quelle que soit la partie aléatoire de `Mac`.

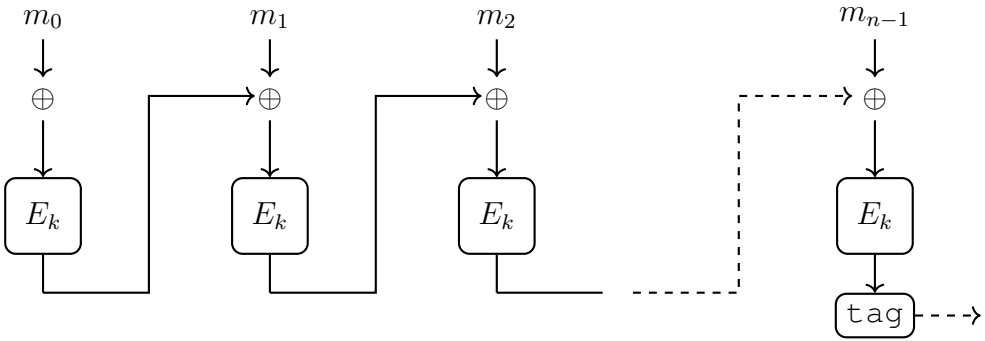
Première Construction CBC-MAC



Voyez-vous un problème ?

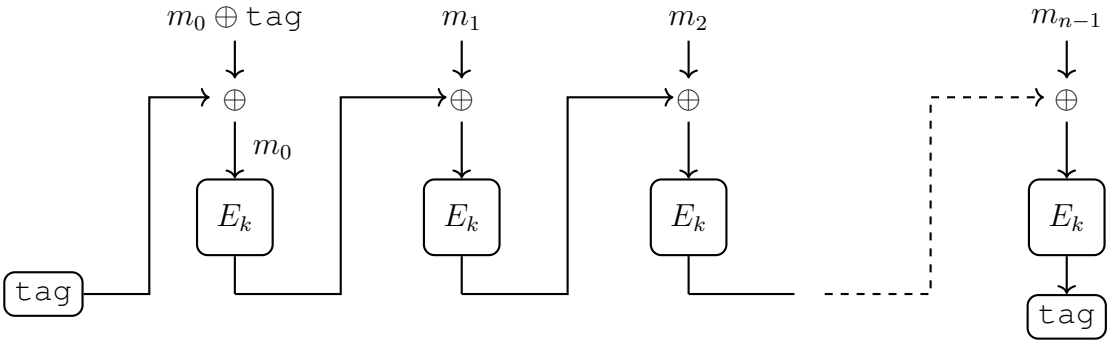
Indication : si $\texttt{tag} = \texttt{Mac}(m)$, que vaut $\texttt{Mac}(\texttt{tag}||m \oplus \texttt{tag})$ (où on a complété par des zeros pour avoir les bonnes tailles).

Attaque sur weak CBC-MAC



$$E_k(m) = \texttt{tag}$$

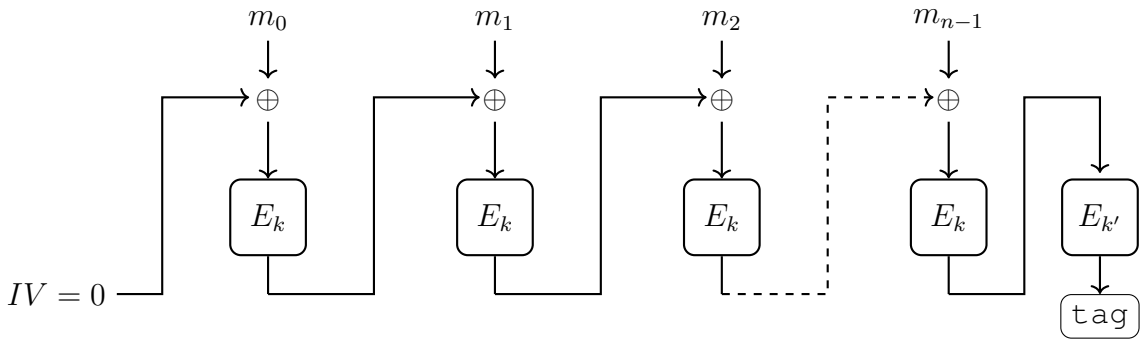
Attaque sur weak CBC-MAC



$$E_k(m||m \oplus \texttt{tag}) = \texttt{tag}!$$

$\Longrightarrow$ On peut donc forger un MAC sur un nouveau message (particulier) !

Réparation : (E)CBC-MAC



- On rajoute une étape de chiffrement avec une seconde clé.
- Ce mode est alors prouvé sûr (EUF-CMA) si E_k est une fonction aléatoire.

Chiffrement Authentifié (AE)

- On prend un chiffrement (*i.e.* une primitive, et un mode) sûr (IND-CPA).
- On complète avec un MAC.

Attention : A priori, il y a 3 possibilités naturelles, mais seule la **dernière** est à utiliser.

- **Encrypt-and-MAC** : $c = (E_k(m), \mathtt{MAC}_{K'}(m))$. À proscrire !
- **MAC-then-Encrypt** : $\mathtt{tag} = \mathtt{MAC}_{K'}(m)$ et $c = E_k(m||\mathtt{tag})$ (utilisé dans TLS jusqu'à version 1.2). Attention, certaines vulnérabilités.
- **Encrypt-then-MAC** : $c_1 = E_k(m), \mathtt{tag} = \mathtt{MAC}_{K'}(c_1)$. Le chiffré final est $c = (c_1, \mathtt{tag})$. Ici, on assure l'**authenticité du chiffré**. Pour déchiffrer, on vérifie le $\mathtt{tag}$, puis on déchiffre si valide. **Utilisation moderne**.

Quelques Exemples

- CCM (*Counter with CBC-MAC*) : Counter mode pour chiffrement, CBC-MAC pour authenticité.
- GCM (*Galois Counter Mode*) : Counter mode pour chiffrement, un MAC particulièrement efficace. Utilisé dans TLS1.2 et TLS1.3, SSH, ...

Ces modes assurent aussi l'intégrité de **données additionnelles** (AD) non secrètes (headers...). On parle d'*Authenticated Encryption with Associated Data* (AEAD).

Le chiffrement AES

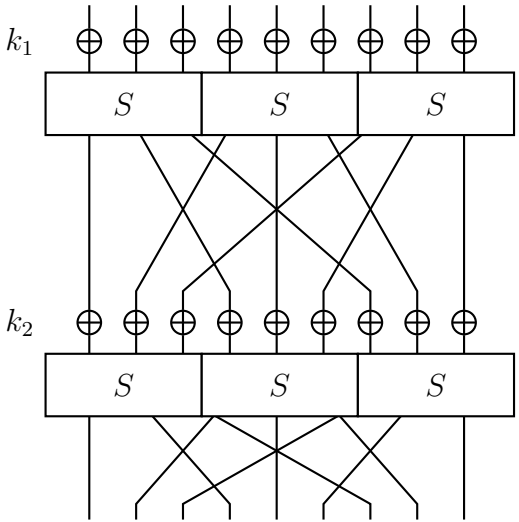
AES : *Advanced Encryption Standard*

- Créé par Daemen et Rijmen en 1997 pour une compétition organisée par le NIST.
- Standardisé en 2001, et **très largement** utilisé aujourd'hui.
- Nom original : *Rijndael Cipher*.

AES est un SPN qui opère sur des blocs de 128 bits, et trois variantes sont possibles :

- AES-128 : Clés de 128 bits, 10 tours.
- AES-192 : Clés de 192 bits, 12 tours.
- AES-256 : Clés de 256 bits, 14 tours.

Rappel : Réseaux de Substitution Permutation (SPN)



- Alternance de couches non linéaires (boîtes S) et de couches linéaires (applications linéaires, **dont** permutations bit à bit et addition de clés).
- Les boîtes S sont des fonctions booléennes sur un petit nombre de bits pour pouvoir être tabulées.

Structure de l'AES

- L'AES opère sur des blocs de 128 bits, *i.e.* 16 octets, qu'on représente en général par une matrice 4×4 d'octets numérotés de 0 à 15.
- Un algorithme de *key schedule* transforme la clé maître en des clés de tour de 128 bits.
- On note w_i la matrice du bloc à la fin du tour i (w_0 est le bloc de clair représenté sous cette forme).

$w_i =$

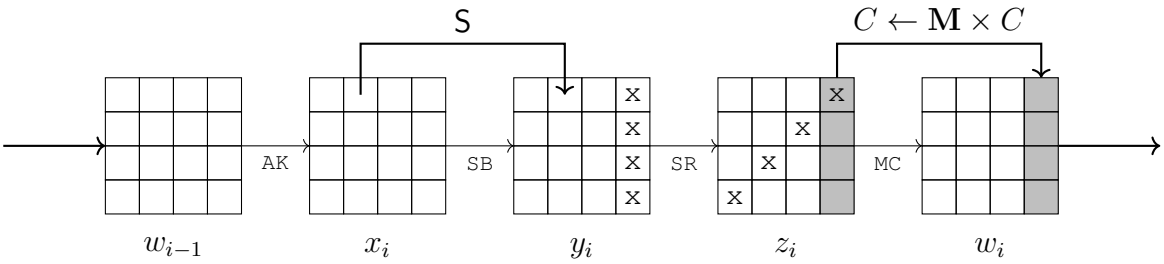
0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Fonction de tour $\longrightarrow$

$w_{i+1} =$

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

La fonction de tour



`AddRoundKey` Ajoute la clé K_{i-1}
`SubBytes` applique sur chaque octet la boîte S (**non-linéaire**)
`ShiftRows` rotation vers la gauche de chaque ligne (permutation **linéaire**)
`MixColumns` Multiplie chaque colonne $C \in (\mathbb{F}_{2^8})^4$ par une matrice $\mathbf{M}$ standardisée

Boîte S de l'AES

On représente les octets comme des éléments de $\mathbb{F}_{2^8} = \mathbb{F}_{256}$ via l'isomorphisme (d'espaces vectoriels)

$$\Psi\colon (x_0, x_1, \ldots, x_7) \in \mathbb{F}_2^8 \longmapsto \sum_{i=0}^{7} x_i X^i,$$

où $\mathbb{F}_{256} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1) = \mathbb{F}_2[\alpha]$.

La boîte S de l'AES correspond alors à la composition $\varphi \circ (\Psi^{-1} \circ \mathbf{I} \circ \Psi)$ où

$$\mathbf{I}\colon x \in \mathbb{F}_{256} \mapsto x^{254} = \begin{cases} x^{-1} & \text{si } x \neq 0 \\ 0 & \end{cases}$$

et φ est une opération affine sur $\mathbb{F}_2^8$ de la forme $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{b}$ où $\mathbf{A}, \mathbf{b}$ sont standardisées.

Couche linéaire : MixRows et MixColumns

- La diffusion est assurée par les opérations linéaires `MixRows` et `MixColumns`.
- `MixRows` effectue une permutation circulaire décalée, de chaque ligne.
- `MixColumns` est une permutation linéaire de 32 bits qui s'applique en parallèle sur chaque colonne de l'état interne (qui est donc un élément de $\mathbb{F}_{2^8}^4$) :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \alpha & \alpha+1 & 1 & 1 \\ 1 & \alpha & \alpha+1 & 1 \\ 1 & 1 & \alpha & \alpha+1 \\ \alpha+1 & 1 & 1 & \alpha \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

où α est une racine de $X^8 + X^4 + X^3 + X + 1$

Le choix de toutes ces opérations n'est pas anodin, et est dicté par la cryptanalyse ! !

Attaques sur AES, en 2025

AES avec $\leqslant 6$ tours : Cassé « en pratique ».

Exploiter l'algorithme de diversification des clés

G. Leurent, C. Pernot - *New Representations of the AES Key Schedule* - EUROCRYPT 2021.

Dans les prochaines séances

- Cryptanalyse Différentielle
- Cryptanalyse Linéaire
- Cryptanalyse Algébrique