INTRODUCTION DE LA CRYPTOGRAPHIE ET D'AES

La cryptographie est un système de chiffrement de l'information, dans l'objectif d'un transfert sécurisé de données ou d'informations privées entre généralement deux individus. L'idée est la suivante : un individu A désire envoyer un message à un individu B et uniquement lisible par l'individu B. Au lieu d'envoyer son message en français ou en anglais par exemple, l'individu A pourrait envoyer son message en chinois si l'individu B comprend aussi cette langue, cependant si le message tombe entre les mains d'un individu comprenant le chinois, le changement de langue n'aura servi à rien !

La cryptographie va plus loin que ça, en effet la cryptographie basique utilise un système de clés. Une clé est utilisée par l'individu A pour coder son message et l'individu B a besoin de cette même clé pour décoder le message. Le message sera illisible par tout autre individu ne possédant pas la clé.
Par analogie, considérons une simple boîte ouvrable que par une seule clé, l'individu A peut mettre son message dans la boîte et fermer le tout à clé, seul l'individu B ayant la même clé pourra accéder au message.
Par exemple, pour coder son message, l'individu A peut utiliser une clé numérique. Considérons la clé $c = 1$ et associons chaque lettre de l'alphabet à un nombre suivant l'ordre alphabétique $(A = 1, B = 2, \ldots, Z = 26)$. Considérons le message "salut", en associant chaque lettre à son nombre, on obtient la suite de nombres suivante $(19,\ 1,\ 12,\ 21,\ 10)$, l'individu A décide d'augmenter de c chaque nombre:$(19 + c,\ 1 + c,\ 12 + c,\ 21 + c,\ 10 + c)$. Ici la clé vaut 1, obtient ainsi : $(20,\ 2,\ 13,\ 22,\ 11)$, ce qui donne le message "tbmvu". C'est ce message que recevra l'individu B, et pour le déchiffrer il aura besoin de la clé c. Il lui suffira alors de faire l'opération inverse pour obtenir le message $(20 - c,\ 2 - c,\ 13 - c,\ 22 - c,\ 11 - c)$, ce qui lui donnera $(19,\ 1,\ 12,\ 21,\ 10)$ et donc le message "salut" !

Voici donc le principe de la cryptographie (un message avec une clé pour le chiffrer/déchiffrer). Nous étudierons cependant un système de chiffrement plus complexe : l'AES (Advanced Encryption Standard).

Auparavant, on utilisait des algorithmes de chiffrements comme le DES (Data Encryption Standard) ou le 3DES. Cependant, ces deux algorithmes de chiffrement possédaient quelques inconvénients : l'algorithme DES n'était pas assez adapté pour certaines applications et le 3DES était trois fois plus lent que l'AES. De plus, le chiffrement était adapté à des blocs de taille 8 octets, jugés trop petits, et donc le niveau de sécurité risquait de ne plus être très efficace après quelques décennies.

Toutes ces considérations ont conduit le NIST (National Institute of Standards and Technology) à la conclusion qu'un nouveau système de chiffrement par bloc était nécessaire pour remplacer le chiffrement DES. C'est ainsi qu'en 1997, le NIST a lancé un appel à propositions pour un nouveau système de chiffrement.
Le NIST et la communauté scientifique internationale ont discuté des avantages et des inconvénients des chiffrements soumis et ont réduit au fur et à mesure le nombre de candidats. Finalement, en 2001, le NIST a déclaré le chiffrement par bloc Rijndael comme le nouvel AES et l'a publié en tant que norme finale.
Le chiffrement AES est tellement efficace qu'il est devenu obligatoire dans plusieurs normes industrielles et est utilisé dans plusieurs systèmes commerciaux (sécurité internet, cryptage Wi-Fi, réseau Skype, etc…). D'ailleurs il a été approuvé par le gouvernement fédéral américain, et la NSA (National Security Agency) des Etats-unis a annoncé qu'elle autorisait l'AES à chiffrer les documents classifiés.

On va donc se pencher sur la manière dont sont chiffrées les données, de sorte à effectuer des échanges sécurisés d'informations, en étudiant le fonctionnement de l'algorithme AES, ses différentes étapes, et ses différents ciphers.

DÉFINITION DES CORPS FINIS

Théorème :
Les corps finis existent ssi ils ont p^m éléments, avec $m \in \mathbb{N}$, $p \in \mathbb{P}$. On les note $GF(p^m)$. (GF pour Galois Field qui est le terme anglais pour corps fini)

Par exemple, $GF(256) = GF(2^8)$ est un corps fini, et en particulier celui qui nous intéresse pour l'AES.

Types de corps finis :
- Si $m = 1$, $GF(p)$ est un corps premier
- Si $m \geq 1$, $GF(p^m)$ est un corps étendu

I. Arithmétique dans les corps premiers $GF(p)$

Les éléments de $GF(p)$ sont les entiers compris entre 0 et $p - 1$.

a) Addition, soustraction, multiplication

Soit $a, b \in GF(p)$,

$$a + b \equiv c \mod p$$
$$a - b \equiv d \mod p$$
$$a \times b \equiv e \mod p$$

b) Inversion

Soit $a \in GF(p)$, son inverse a^{-1} satisfait $a \times a^{-1} \equiv 1 \mod p$
Il peut être déterminé à l'aide de l'algorithme d'Euclide étendu.

II. Arithmétique dans les corps étendus $GF(2^m)$

a) Représentation des éléments

Les éléments de $GF(2^m)$ sont des polynômes de la forme $A(X) = a_{m-1}X^{m-1} + \ldots + a_1X + a_0$, avec $a_i \in GF(2)$.

Exemple avec $GF(8) = GF(2^3)$:
Les éléments sont de la forme $A(X) = a_2X^2 + a_1X + a_0 = (a_2, a_1, a_0)$
On a donc un ensemble $GF(2^3) = \{0,1,X, X+1, X^2, X^2+1, X^2+X, X^2+X+1\}$.

b) Addition et soustraction

Soit $A, B \in GF(2^m)$,

$$C(X) = A(X) + B(X) = \sum_{k=0}^{m-1} c_kX^k \text{ avec } c_k = a_k + b_k \mod 2$$

$$D(X) = A(X) - B(X) = \sum_{k=0}^{m-1} d_k X^k \text{ avec } d_k = a_k - b_k \mod 2 = a_k + b_k \mod 2$$

c) Multiplication

Soit $A, B \in GF(2^m)$,

$$E(X) = A(X) \times B(X) = (\sum_{k=0}^{2m-2} e_k X^K) \mod P(X) \text{ avec :}$$

$$\begin{cases} e_k = \sum_{i+j=k} a_i \times b_j \mod 2 \\ P(X) \text{ un polynôme irréductible dans } GF(2^m) \end{cases}$$

Pour $GF(256) = GF(2^8)$, le polynôme irréductible standard d'AES est $P(X) = X^8 + X^4 + X^3 + X + 1$.

d) Inversion

Soit $A(X) \in GF(2^m)$, son inverse $A(X)^{-1}$ satisfait $A(X) \times A(X)^{-1} \equiv 1 \mod P(X)$
Elle peut aussi être déterminé à l'aide de l'algorithme d'Euclide étendu.

STRUCTURE DE L'ALGORITHME

Soit une matrice carrée d'ordre 4 :

$$I = \begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix} \text{ avec } b_i \in GF(2^8)$$

Chaque éléments de cette matrice représente un octet, et l'algorithme s'exécute avec cette matrice en entrée, par bloc de 16 octets (ou 128 bits, puisque 1 octet = 8 bits).
Une série d'opération est effectuée sur cette matrice pour obtenir la sortie, chiffrée par l'algorithme.

On peut représenter tous les entiers entre 0 et 255 par un élément de $GF(2^8) = GF(256)$. Par exemple, 10 s'écrit en binaire $(1010)_2$, et sera représenté dans $GF(2^8)$ par $X^3 + X$.

I. Répétition des étapes

L'algorithme est composé de 4 étapes, qui constituent un tour : substitution, décalage, mixage et ajout de la clé.
Selon la taille de la clé, le nombre de tour change : pour une clé de 128 bits, il y a 10 tours, pour une clé de 192 bits, 12 tours, et pour une clé de 256 bits, 14 tours.
Avant de commencer ces tours, l'étape d'ajout de la clé est effectué une première fois, et lors du dernier tour, le mixage n'est pas effectué.
Après ces tours effectués, on obtient une matrice à la sortie, qui contient 16 octets, correspondants au chiffrement des 16 octets fournis à l'entrée.

II. Substitution

La substitution consiste à appliquer une transformation à chaque élément de la matrice.
Si on note S cette transformation, on a :

$$\begin{aligned} S : GF(2^8) &\to GF(2^8) \\ a &\mapsto S(a) \end{aligned}$$

Cette transformation est bijective, de sorte à associer à chaque élément un unique autre élément du même ensemble, tout en évitant les points fixes. Elle permet d'éviter une quelconque linéarité dans le chiffrement.

III. Décalage

Cette étape va simplement échanger la place des éléments de la matrice. La première ligne est inchangée. La deuxième ligne est décalée d'un élément vers la gauche, la troisième de deux éléments vers la gauche et la quatrième de trois éléments vers la gauche (ou un vers la droite, c'est la même chose).
Si on note D cette transformation, on a :

$$D : \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix} \mapsto \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_5 & a_9 & a_{13} & a_1 \\ a_{10} & a_{14} & a_2 & a_6 \\ a_{15} & a_3 & a_7 & a_{11} \end{bmatrix}$$

Elle permet d'éviter que les colonnes soient chiffrées de manière indépendante les unes des autres.

IV. Mixage

Le mixage s'applique sur une colonne et permet d'avoir en sortie une nouvelle colonne dans laquelle chaque élément dépend de tous les éléments de la colonne d'entrée.
Si on note M cette transformation s'appliquant sur une colonne, on a :

$$M : \begin{bmatrix} a_i \\ a_{i+1} \\ a_{i+2} \\ a_{i+3} \end{bmatrix} \mapsto \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} a_i \\ a_{i+1} \\ a_{i+2} \\ a_{i+3} \end{bmatrix} \text{ avec } i \in \{0,4,8,12\}$$

Elle permet encore une fois d'éviter que les lignes soient chiffrées de manière indépendantes.

V. Ajout de la clé

Pour chaque tour, on ajout la partie de la clé qui correspond au tour.
Si on note C cette transformation, on a :

$$C : \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix} \mapsto \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix} + \begin{bmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{bmatrix}$$

C'est cette étape qui fait que le chiffrement est unique pour chaque clé utilisée.

VI. Extension de la clé

Pour simplifier l'étude, on va uniquement utiliser une clé de taille 128 bits (16 octets), avec donc 10 tours.

La clé initiale est étendue en sous-clés pour chaque tour de l'algorithme.
La sous-clé ajoutée au début de l'algorithme K_0 est la clé elle même.

$$K_0 = \begin{bmatrix} k_{0,0} & k_{0,4} & k_{0,8} & k_{0,12} \\ k_{0,1} & k_{0,5} & k_{0,9} & k_{0,13} \\ k_{0,2} & k_{0,6} & k_{0,10} & k_{0,14} \\ k_{0,3} & k_{0,7} & k_{0,11} & k_{0,15} \end{bmatrix}$$

Les sous-clés des tours sont calculées de manière récursive, c'est à dire que pour obtenir la sous-clé d'un tour, il faut connaitre celle du tour précédent (et donc la clé initiale pour le premier tour).

Pour les 4 premiers coefficients (première colonne) :

$$\begin{cases} k_{n+1,0} = k_{n,0} + S(k_{n,13}) + RC_{n+1} \\ k_{n+1,1} = k_{n,1} + S(k_{n,14}) \\ k_{n+1,2} = k_{n,2} + S(k_{n,15}) \\ k_{n+1,3} = k_{n,3} + S(k_{n,12}) \end{cases}$$

Et pour les 12 autres coefficients :

$$\begin{cases} k_{n+1,4} = k_{n,4} + k_{n+1,0} \\ \dots \\ k_{n+1,15} = k_{n,15} + k_{n+1,11} \end{cases}$$

Avec S la substitution, et RC_{n+1} le coefficient du tour $n+1$, tel que $RC_{n+1} = X^n \in GF(2^8)$

DÉCHIFFREMENT ET PREUVE DE L'ALGORITHME

Chaque étape correspond à une application bijective sur la matrice de données, faisant de l'algorithme une bijection, composée des bijections de chaque étape. Il existe donc une bijection réciproque, composée des bijections réciproques des étapes, dans le sens opposé : on va donc effectuer le même nombre de tours, avec pour chaque tour, dans l'ordre : l'addition de la clé, le mixage, le décalage et la substitution, en enlevant le mixage lors du premier tour et en effectuant un ajout de la clé supplémentaire à la fin, afin de correspondre parfaitement à l'ordre inversé des étapes de l'algorithme de chiffrement.

I. Ajout de la clé

L'addition étant sa propre opération réciproque dans $GF(2^8)$, cette étape est donc la même pour le déchiffrement que pour le chiffrement :

$C^{-1} = C$, et $C(C(x)) = x$

II. Mixage inverse

Pour inverser le mixage des colonnes, il suffit de multiplier chaque colonne par la matrice inverse de celle utilisée dans le mixage lors du chiffrement :

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On définit alors M^{-1}, tel que $M^{-1}(M(x)) = x$, par :

$$M^{-1} : \begin{bmatrix} a_i \\ a_{i+1} \\ a_{i+2} \\ a_{i+3} \end{bmatrix} \mapsto \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} a_i \\ a_{i+1} \\ a_{i+2} \\ a_{i+3} \end{bmatrix} \text{ avec } i \in \{0,4,8,12\}$$

Les coefficients de cette matrice inverse sont bien des éléments de $GF(2^8)$, ici écrits avec leur représentation en hexadécimal. Par exemple, $(0B)_{16} = (00001011)_2 = X^3 + X + 1 \in GF(2^8)$.

III. Décalage inverse

Pour inverser le décalage, il suffit donc de décaler les éléments vers la droite, de sorte à ce qu'ils reviennent à leur position initiale. On définit donc D^{-1}, tel que $D^{-1}(D(x)) = x$, par :

$$D^{-1} : \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix} \mapsto \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_{13} & a_1 & a_5 & a_9 \\ a_{10} & a_{14} & a_2 & a_6 \\ a_7 & a_{11} & a_{15} & a_3 \end{bmatrix}$$

IV. Substitution inverse

Pour le chiffrement, on a une bijection S tel que, pour chaque élément de la matrice, on remplace le coefficient a par $S(a)$. On a donc également une bijection réciproque S^{-1} tel que $S^{-1}(S(a)) = a$, ce qui permet d'inverser l'étape de substitution.

V. Preuve de l'algorithme

Si l'on note e_k le chiffrement et e_k^{-1} le déchiffrement, ces deux applications étant deux bijections réciproques, on a bien $e_k^{-1}(e_k(x)) = x$

UTILISATION DE CIPHERS

Le but d'un cipher et de permettre de chiffrer une série de données (et pas seulement 16 octets), comme par exemple un email ou un fichier.
Il existe différents ciphers : ECB, CBC, OFB, et CFB, et chacun d'entre eux à un fonctionnement différent.

La série de données à chiffrer est séparée en morceaux $x_1, \ldots, x_n$ de 16 octets chacun, appelés blocs, qui sont transmis avec la clé k au cipher pour être chiffrer les uns après les autres, pour donner une série de blocs chiffrés $y_1, \ldots, y_k$.
Dans le cas où la taille de la série de données à chiffrer n'est pas un multiple de 16 octets, on lui ajoute une marge à la fin (par exemple un 1 suivit de plusieurs 0 jusqu'à obtenir un multiple de 16 octets), de sorte à bien obtenir des blocs faisant tous la bonne taille.

Certains ciphers utilisent aussi des paramètres supplémentaires comme un vecteur d'initialisation, noté IV (pour Initialization Vector), étant un bloc de 16 octets choisi à chaque série de données dans le but de donner une dimension d'aléatoire à chaque fois.

Dans la suite de notre document, on va noter e_k la fonction représentant le chiffrement AES avec la clé k, et e_k^{-1} le déchiffrement avec la clé k.

I. Electronic Codebook Mode (ECB)

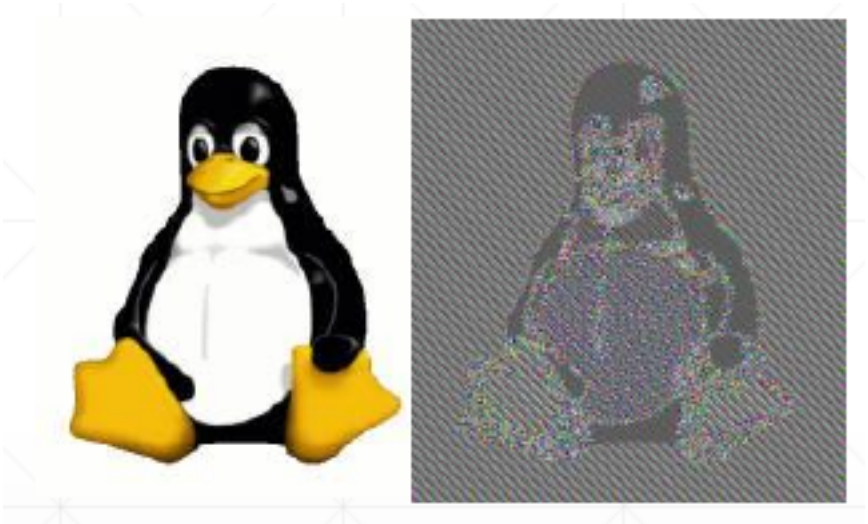
L'ECB est le mode le plus direct pour chiffrer des données, chaque bloc est passé directement dans l'algorithme, accompagné de la clé :

$$\begin{cases} y_i = e_k(x_i) \\ x_i = e_k^{-1}(y_i) = e_k^{-1}(e_k(x_i)) \end{cases} \text{avec } i \geq 1$$

Cependant, ce mode possède plusieurs faiblesses, à cause de sa simplicité : si on chiffre plusieurs fois le même bloc avec ce mode, on obtient à chaque fois le même bloc chiffré.

Il existe donc des attaques possibles en utilisant ce mode, notamment une attaque par substitution : sans même déchiffrer le contenu, il est possible d'échanger des blocs sans casser le message (étant donné que le déchiffrement de chaque bloc est indépendant). Avec un peu de repérage de l'attaquant, il peut donc modifier le contenu facilement à sa guise.

De plus, comme chaque bloc identique est toujours chiffré de la même manière, tant que la clé reste la même, on peut donc reconnaitre un paterne dans le résultat, ce qui est remarquable avec l'exemple de cette image et sa version chiffrée avec le mode ECB :



C'est pourquoi, en pratique, ce mode est peu utilisé, et des modes plus complexes ont été inventés.

II. Cipher Block Chaining Mode (CBC)

Contrairement aux blocs chiffrés avec ECB, un bloc chiffré avec CBC (mais également les autres modes que nous verrons ensuite) ne dépend pas uniquement du bloc d'entrée, mais aussi de tous les blocs chiffrés avant celui ci, et d'un vecteur d'initialisation,
Pour chiffrer un bloc, on lui ajoute d'abord le résultat du chiffrement du bloc précédent (par addition au sens de $GF(2^8)$), ou le vecteur d'initialisation dans le cas du premier bloc. De cette manière, y_1 dépend de x_1 et de l'IV, y_2 dépend de x_2, x_1 et de l'IV, …

On peut noter le chiffrement et déchiffrement du premier bloc de cette manière :

$$\begin{cases} y_1 = e_k(x_1 + IV) \\ x_1 = e_k^{-1}(y_1) + IV \end{cases}$$

Et pour tous les blocs suivants ($i \geq 2$) :

$$\begin{cases} y_i = e_k(x_i + y_{i-1}) \\ x_i = e_k^{-1}(y_i) + y_{i-1} \end{cases}$$

De cette manière, en choisissant des IVs différents à chaque chiffrement, la même série de données chiffrée plusieurs fois donnera des résultats différents. On ne peut donc pas reconnaître un bloc, car même si un même bloc est chiffré plusieurs fois, il donnera un résultat différent à chaque fois, et une substitution de bloc dans la chaine casserait tous les blocs suivants, ce qui rend donc cette attaque impossible.

III. Output Feedback Mode (OFB)

Dans ce mode, l'IV est chiffré successivement, et ajouté à chaque bloc. Le chiffrement et le déchiffrement se fait donc de la même manière, comme ajouté deux fois la même chose revient à ne rien ajouter.

On peut donc poser une suite $(s_i)_{i\geq 1}$ définie par :

$$\begin{cases} s_1 = e_k(IV) \\ s_{i+1} = e_k(s_i) \end{cases}$$

Et on peut chiffrer et déchiffrer un bloc comme ceci :

$$\begin{cases} y_i = s_i + x_i \\ x_i = s_i + y_i \end{cases}$$

L'avantage de ce mode est que, bien qu'il utilise un IV, le chiffrement d'un bloc n'est pas déterminant pour le chiffrement des blocs suivants. De ce fait, on peut calculer en avance les éléments de $(s_i)_{i\geq 1}$ et gagner du temps pour chiffrer/déchiffrer des blocs.

IV. Cipher Feedback Mode (CFB)

Le CFB est un mélange du CBC et de l'OFB : Le chiffrement et le déchiffrement se fait avec la même opération, à savoir l'addition avec l'IV puis les blocs précédents.

On définit donc le chiffrement du premier bloc de cette manière :

$$\begin{cases} y_1 = e_k(IV) + x_1 \\ x_1 = e_k(IV) + y_1 \end{cases}$$

Et pour tous les blocs suivants ($i \geq 2$) :

$$\begin{cases} y_i = e_k(y_{i-1}) + x_i \\ x_i = e_k(y_{i-1}) + y_i \end{cases}$$

BIBLIOGRAPHIE

[1] Christof Paar, Jan Pelzl : *Understanding Cryptography*. Springer. 2009.

LIENS UTILES

Cahier de TIPE virtuel : https://trello.com/b/lKhSvegX

Dépôt GitHub : https://github.com/NAS-TIPE/TIPE