INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

3ème Année

Modélisation, Informatique et Communication (MIC)

Théorie de l'Information et Codages

D'après le cours de Pierre PINEL

- Version du 25 février 2018 -

Table des matières

Introduction à la Théorie de l'Information

Introduction à la Théorie de l'Information

Histoire

La Théorie de l'information prend son origine dans l'article publié en 1948 intitulé *A Mathematical Theory of Communication* [1] de Claude Shannon, un ingénieur-chercheur aux Laboratoires Bell et professeur au Massachusetts Institute of Technology.

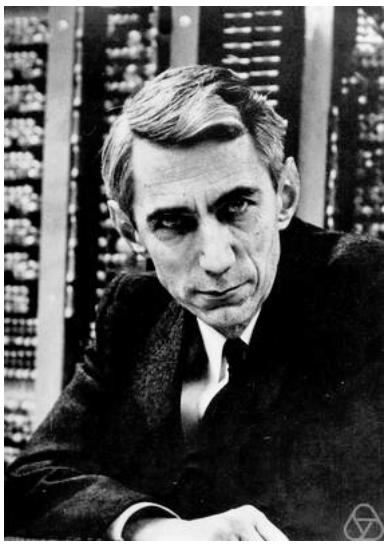

FIGURE 1 – Claude Shannon (1916-2001)

La maxime de Shannon

« L'adversaire connaît le système »
ou bien
« Il n'y a pas de secret sur le cryptosystème »

Cette théorie comporte des applications concrètes en codages, compression de données et en cryptographie.

En 1949 il publie un article intitulé *Communication Theory of Secrecy Systems* [2] qui va poser les bases de la Cryptographie en tant que science. Il définira dans celui-ci le **Théorème de la distance limite** permettant de caractériser la longueur minimale du cryptogramme à intercepter afin de pouvoir décrypter le message.

Dans ce cours, nous aborderons tout d'abord les **bases générales de la Théorie de l'information**, puis les **codages source et canal**, et enfin le **cryptage**.

Chapitre 1

Théorie de l'information

1 Quelques définitions

Tout d'abord définissons quelques notions essentielles à la compréhension de la théorie de l'information :

• **Définition d'un alphabet :**
Un **alphabet** est un jeu de caractères à partir duquel est généré un message.
Par exemple :
— L'alphabet francais courant : `a b c d e f g h i j k l m n o p q r s t u v w x y z`.
— L'alphabet binaire : `0 1`.
— L'alphabet hexadécimal : `0 1 2 3 4 5 6 7 8 9 A B C D E F`.

• **Définition d'un message :**
Un **message** est une combinaison unique de caractères issus d'un même alphabet. La probabilité de réception d'un message m_1 est noté $\mathbb{P}(m_1)$.

Remarque : Il faut bien faire la différence entre le **message** et **l'information** délivrée par celui-ci. Par exemple, le message m_1 : *«Hello»* et le message m_2 : *«Bonjour»* sont deux messages différents. Cependant ces deux messages délivrent la même information.

• **Définition d'un langage :**
Un **langage** est un ensemble de N messages issus d'un même alphabet.

• **Définition d'une source :**
Une **source** est un élément capable d'envoyer des messages dans un langage donné. Une source peut être **bien élevée** (*"well-behaved source"* en anglais) c'est à dire **stationnaire** (les probabilités d'apparition des messages sont indépendantes du temps) et **ergodique** ().

• **Définition d'un destinataire :**
Un **destinataire** est un élément recevant les messages émis par une source.

2 Paradigme de Shannon

Le **Paradigme de Shannon**, issu des travaux de Claude Shannon, modélise la transmission d'une information entre une source et un destinataire via un canal. Une version simplifiée est donnée par le schéma ci dessous :

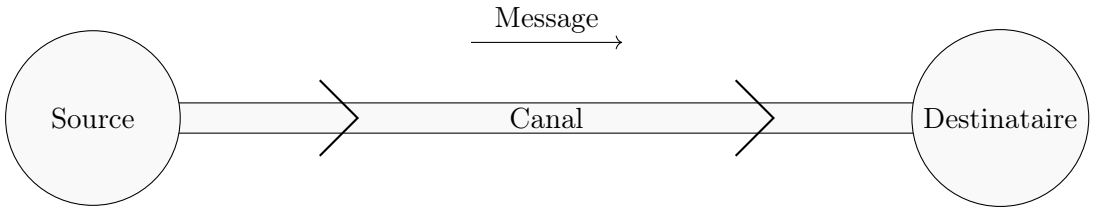

FIGURE 1.1 – Schéma simplifié du Paradigme de Shannon

Le **canal de transmission** entre une source et un destinataire peut revêtir plusieurs formes. Quelques exemples :

— Dans le cas d'un DVD dont on considère l'évolution temporelle, la Source est le DVD à un instant donné, et le destinataire est ce même DVD plus tard dans le temps. Le canal de transmission est ici le temps.
— Dans le cas d'un signal sonore, le canal de transmission sera l'air.

Dans une version plus complète du paradigme, les perturbations sont prises en comptes.
Par hypothèse, elles sont supposées toutes se produire pendant la transmission, au niveau du canal.

— Le **codage source** est le bloc réalisant la compression du message.
— Le **cryptage** permet de s'assurer de la confidentialité et de l'intégrité du message.
— Le **codage canal** permet de détecter et corriger les erreurs survenant durant la transmission.
— Le **canal sûr** est destiné à la transmission des paramètres du cryptage (les clés privées).
— Des **perturbations** (Erreurs de transmission, attaques cryptologiques) peuvent survenir.

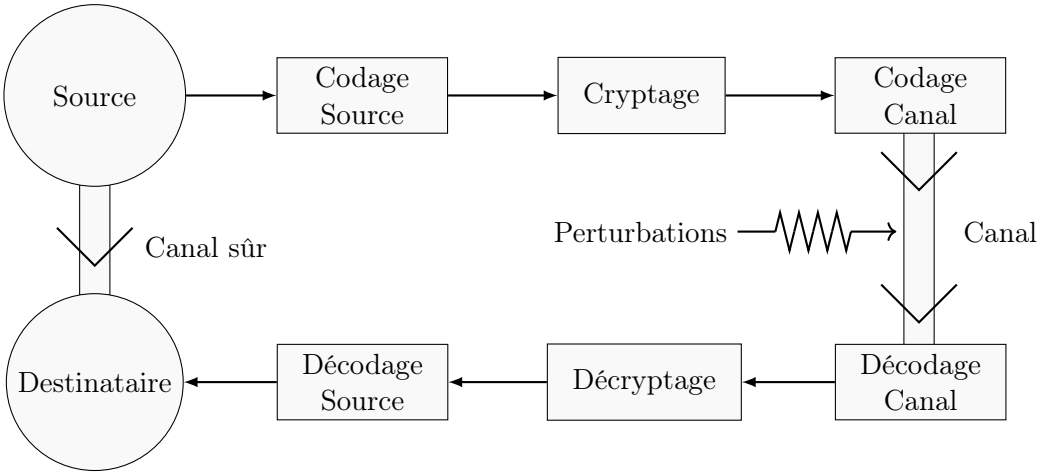

FIGURE 1.2 – Schéma complet du Paradigme de Shannon

Remarque : Il est tout à fait possible inverser l'ordre des blocs codage source et cryptage. Cependant les algorithmes de cryptage étant très souvent chronophages, il est préférable de compresser d'abord les données afin de ne pas crypter des données inutiles.

3 Quantité d'information associée à un message

La théorie de l'information est une théorie quantitative. La problématique réside donc dans la quantité d'information que peut apporter un message. Cette quantité d'information est définie à travers 3 axiomes :

- La quantité d'information $q(m)$ est une fonction décroissante de la probabilité de réception du message contenant cette information $\mathbb{P}(m)$. En effet, plus une information est probable, moins elle aura d'intérêt.
- Si $\mathbb{P}(m) = 1$ alors $q(m) = 0$. Une information certaine n'a pas besoin d'être relayée.
- Soient deux messages m_1 et m_2 indépendants (m_1 et m_2 indépendants $\iff \mathbb{P}(m_2|m_1) = \mathbb{P}(m_2)$). On aura alors $q(m_1, m_2) = q(m_1) + q(m_2)$.

On trouve alors la seule possibilité pour caractériser la quantité d'information transmise par un message m de probabilité de réception $\mathbb{P}(m)$:

$$q(m) = \log\left(\frac{1}{\mathbb{P}(m)}\right) = -\log\left(\mathbb{P}(m)\right)$$

Remarque : On peut utiliser différentes bases pour le logarithme. Ici, le travail sur des infos binaires implique un $\log_2$. L'unité utilisée pour un $\log_2$ est le **Shannon** (noté *Sh*). Par convention, on notera $\mathbb{P}(m) = 0 \Rightarrow q(m) = 0$ (un message impossible n'apporte aucune information).

4 Entropie d'une source

Considérons une source S pouvant émettre N messages issus d'un même un langage.
Hypothèse : la source est "bien élevée" (c'est à dire stationnaire et ergodique).
L'entropie $E(S)$ de la source S représente la quantité moyenne d'information véhiculée par les messages émis par cette source. Soit :

$$E(S) = \frac{\sum \mathbb{P}(m)q(m)}{\sum \mathbb{P}(m)} = \sum_{i=1}^{N} \mathbb{P}(m_i) \log\left(\frac{1}{\mathbb{P}(m_i)}\right) = -\sum_{i=1}^{N} \mathbb{P}(m_i) \log\left(\mathbb{P}(m_i)\right)$$

Remarques :
- Cette formule est une simple moyenne pondérée, les probabilités jouant le rôle des coefficients pondérateurs.
- Les $\mathbb{P}(m)$ étant des probabilités, on a par définition $\sum \mathbb{P}(m) = 1$.
- En logarithme de base 2, ($\log_2$) on aura du Sh/msg.

Pour une source binaire (S_b) on aura $\mathbb{P}(0) = p,\ \mathbb{P}(1) = 1 - p$ soit :

$$E(S_b) = -p \times \log(p) - (1-p) \times \log(1-p)$$

Pour $\mathbb{P}(0) = \mathbb{P}(1) = \frac{1}{2}$ on atteint l'entropie de source maximale $E_{max}(S) = 1\ Sh/msg$ atteinte en . Un bit transporte donc une quantité d'information inférieure ou égale à 1 *Sh*.

De façon plus générale, pour un alphabet de N messages, l'entropie de source maximale $E_{max}(S)$ sera atteinte si tous les messages sont équiprobables, c'est à dire : $\forall i, \mathbb{P}(m_i) = \frac{1}{N}$. Ainsi, $E_{max}(S) = \log(N)\ Sh/msg$.

5 Exercice d'application

On dispose de 9 pièces dorées dont seulement 8 sont en or. La neuvième est d'un poids différent (plus lourd ou plus léger).

Est-il possible de déterminer la pièce fausse et son poids en moins de 3 pesées ?

6 Quantité d'information conjointe, quantité d'information conditionnelle

La **quantité d'information conjointe** entre deux messages m_1 et m_2 est donnée par la formule :

$$q(m_1, m_2) = \log\left(\frac{1}{\mathbb{P}(m_1, m_2)}\right)$$

La **quantité d'information conditionnelle** (c'est à dire la quantité d'information contenue par le message m_2 sachant qu'on a reçu le message m_1) est donnée par :

$$q(m_2|m_1) = \log\left(\frac{1}{\mathbb{P}(m_2|m_1)}\right)$$

7 Entropies conjointes et conditionnelles

Les entropies conjointes et conditionnelles correspondent aux moyennes des quantités d'informations conjointes et conditionnelles. Pour deux sources (S_1) et (S_2) on a :

- Entropie conjointe :

$$\mathbb{E}(S_1, S_2) = \frac{\sum_i \sum_j \mathbb{P}(m_i, m_j) q(m_i, m_j)}{\sum_i \sum_j \mathbb{P}(m_i, m_j)} = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \mathbb{P}(m_i, m_j) \log\left(\frac{1}{\mathbb{P}(m_i, m_j)}\right)$$

- Entropie conditionnelle :

$$\mathbb{E}(S_2|S_1) = \frac{\sum_i \sum_j \mathbb{P}(m_i, m_j) q(m_j|m_i)}{\sum_i \sum_j \mathbb{P}(m_i, m_j)} = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \mathbb{P}(m_i, m_j) \log\left(\frac{1}{\mathbb{P}(m_j|m_i)}\right)$$

L'entropie conjointe et l'entropie conditionelle sont liées par la loi des entropies totales :

- Loi des entropies totales :

$$\mathbb{E}(S_1, S_2) = \mathbb{E}(S_1) + \mathbb{E}(S_2|S_1) = \mathbb{E}(S_2) + \mathbb{E}(S_1|S_2)$$

8 Information mutuelle

L'information mutuelle (notée i) est l'information partagée par les deux messages m_1 et m_2. Elle est définie par :

$$i(m_1, m_2) = q(m_1) - q(m_1|m_2) = q(m_1) + q(m_2) - q(m_1, m_2)$$

Remarque : La seconde formule se trouve aisément à partir de la première.

9 Information mutuelle moyenne entre deux sources

Par extension au cas de deux sources de la formule de l'information mutuelle, on obtient l'information mutuelle moyenne (notée I) :

$$I(S_1, S_2) = \frac{\sum_i \sum_j \left[i(m_i, m_j).p(m_i, m_j) \right]}{\sum_i \sum_j p(m_i, m_j)} = \mathbb{E}(S_1) + \mathbb{E}(S_2) - \mathbb{E}(S_1, S_2) = \mathbb{E}(S_1) - \mathbb{E}(S_1|S_2)$$

On peut encadrer l'information mutuelle moyenne : $0 \leqslant I(S_1, S_2) \leqslant \min\left(\mathbb{E}(S_1), \mathbb{E}(S_2)\right)$

10 Lien entre Entropie, Entropie conjointe et Information mutuelle

L'entropie, l'entropie conjointe et l'information mutuelle peuvent être mises en relation grâce à un schéma des flux d'informations partant de la source S_1 :

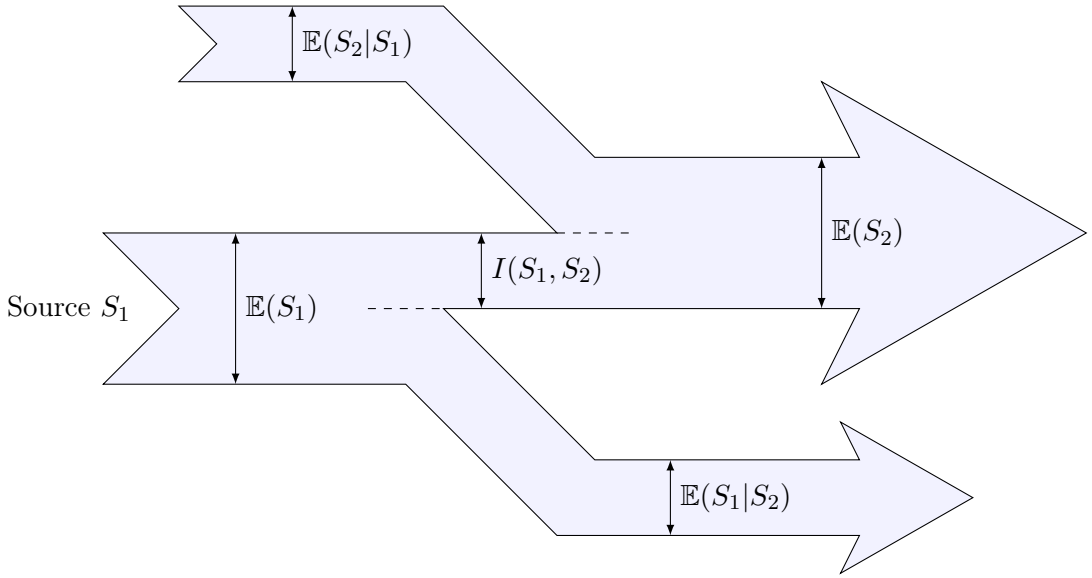

FIGURE 1.3 – Schéma récapitulatif des flux d'informations

11 Exercices d'application

11.1 Météo à Shannonville

La station météorologique de Shannonville a pris à l'essai deux météorologues, Paul et Pierre. Voici les résultats de leurs prévisions sur 16 jours :

Paul		Temps effectif	
		Soleil	Pluie
Prévisions	Soleil	$\frac{10}{16}$	$\frac{1}{16}$
	Pluie	$\frac{3}{16}$	$\frac{2}{16}$

Pierre		Temps effectif	
		Soleil	Pluie
Prévisions	Soleil	$\frac{13}{16}$	$\frac{3}{16}$
	Pluie	0	0

Grâce à la théorie de l'information, dire qui de Pierre ou de Paul devrait être engagé par la station.

Chapitre 2

Codage source

En général le codage source intervient dans les problèmes de compression de données et de minimisation de l'espace occupé par les données compressées.

On considérera dans tout ce chapitre que **le canal de transmission est parfait.**

1 Théorème fondamental du codage source

Théorème (dans le cas d'un alphabet binaire) : On peut s'approcher autant qu'on veut de la limite de 1 Sh/bit en adaptant la représentation des messages.

Théorème (dans le cas d'un alphabet à K caractères) : Quelque soit la source S, il existe une représentation des messages émis par la source S dont **la longueur moyenne** L de représentation de ces messages vérifie la relation :

$$\forall \varepsilon > 0, \quad \frac{\mathbb{E}(S)}{\log_2(K)} \leqslant L \leqslant \frac{\mathbb{E}(S)}{\log_2(K)} + \varepsilon$$

En d'autres termes, $\frac{\mathbb{E}(S)}{\log_2(K)}$ représente **une limite infranchissable** pour représenter des messages d'un alphabet à K caractères sans pertes d'informations.

2 Codage sans perte d'informations

2.1 Codage de Huffman

Exemple : On prend une source S de 4 messages possibles (A, B, C, D) et on souhaite les représenter en codage binaire. On prendra :

$$\mathbb{P}(A) = \frac{1}{2}, \quad \mathbb{P}(B) = \frac{1}{4}, \quad \mathbb{P}(C) = \frac{1}{8}, \quad \mathbb{P}(D) = \frac{1}{8}$$

On calcule l'entropie de la source :

$$\mathbb{E}(S) = \frac{1}{2}\log(2) + \frac{1}{4}\log(4) + \frac{1}{8}\log(8) + \frac{1}{8}\log(8) = 1.75 \ Sh/msg$$

On peut donc coder ces messages de manière optimum à 1.75 bit/msg. On peut utiliser ce codage A : `0`, B : `10`, C : `110`, D : `111`. Le codage étant sur un, deux ou trois bits, le problème du déchiffrage se pose. Le récepteur pourra néanmoins déchiffrer un message car le codage utilisé ne laisse pas la place au doute, aucun message ne constitue la partie gauche d'un autre message (condition suffisante mais non nécessaire). Pour trouver ce codage, on peut utiliser un arbre comme représenté ci dessous :

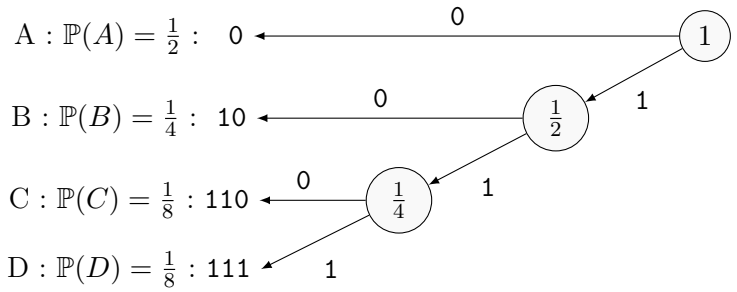

FIGURE 2.1 – Construction de l'arbre de Huffman

Règles de construction :

— On relie à chaque fois les deux nœuds de probabilités les plus faibles, on place le code `0` sur le plus probable.
— Si les deux nœuds sont équiprobables, par convention, on place le code `0` sur la branche supérieure.

L'algorithme de Huffman est **optimal**, c'est à dire qu'il n'est pas possible de réaliser un codage plus compact que le codage de Huffman sans perdre des informations.

Dans le cas général on peut **réaliser une extension de source d'ordre** n afin de se rapprocher de l'entropie maximale théorique. Cette opération consiste à considérer non pas un message mais une suite de n messages.

Exemple : On prend la source binaire S pouvant émettre les messages `A` et `B`. Pour une source binaire émettant les messages `A` et `B`, L'extension de source d'ordre 2, que l'on notera S_2 sera : `AA, AB, BA, BB`.

On prendra $\mathbb{P}(A) = 0.9$ et $\mathbb{P}(B) = 0.1$. La source S_2, en tant qu'extension d'ordre 2 de S, envoie les messages `AA, AB, BA, BB`. En calculant les probabilités de réception de chaque message, on en déduit le codage associé grâce au codage de Huffman :

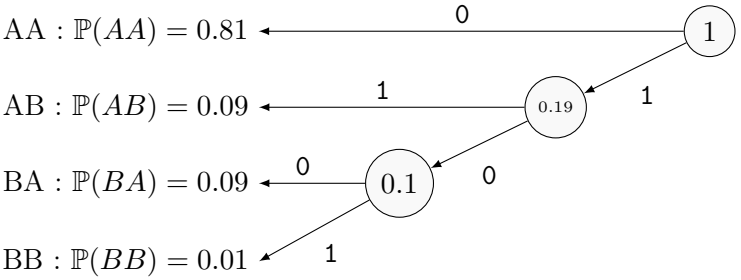

FIGURE 2.2 – Construction de l'arbre de Huffman pour l'exemple

Ce qui nous donne le codage suivant :

Message	AA	AB	BA	BB
Code	0	11	100	101

Ce codage donne une longueur moyenne $L = 1 \times 0.81 + 2 \times 0.09 + 3 \times 0.09 + 3 \times 0.01 = 1.29\ bit/msg$ pour S_2 soit $0.645\ bit/msg$ pour un message de S.

Le codage de Huffman est **adapté à un codage hors-ligne** car il est nécessaire pour sa mise en oeuvre de connaître les probabilités de chaque message à l'avance.

— **Un codage hors ligne** nécessite l'intervention d'une mémoire. (Compression d'une image par exemple).
— **Un codage en ligne** est un processus continu. (Une transmission radio par exemple).

Cet **algorithme de codage est rapide** (constitution d'un arbre). La compression des données va fragiliser le message car l'élimination de la redondance rend **plus difficile la détection d'erreurs de transmission**.

Cet algorithme n'est pas approprié dans tous les cas, par exemple, il existe des cas où le fichier final sera plus gros que l'initial car il y a une nécessité de stockage de la table de codage (ou a minima de la table des probabilités) en amont. Ce sera le cas pour un texte composé de très peu de caractères.

On peut trouver une parade à ces problèmes en gardant les qualités du codage de Huffman : le **"codage auto-adaptatif"**, permettant de faire du codage en ligne (ex : ligne téléphonique) à la volée. Pour cela on initialise des probabilités supposées qui seront ensuite adaptées en fonction des messages transmis.

Exemple : Initialement on prend `a` à `00` (toutes les probabilités sont de $\frac{1}{4}$)
On transmet `a` : l'arbre s'actualise, `a` prend $\frac{2}{5}$ et le reste $\frac{1}{5}$.
On retransmet `a` : l'arbre s'actualise avec `a` $\frac{3}{6}$ et le reste $\frac{1}{6}$.

On perd la rapidité initiale car ici il faut reconstruire la table au fur à mesure (soit plus de calculs) même si après un certain nombre de messages, l'arbre se sera équilibré et ne bougera plus.

2.2 Algorithmes de Lempel-Ziv

Le premier algorithme de Lempel-Ziv date de 1977. Ce sont des algorithmes **par nature auto-adaptatifs** qui utilisent des dictionnaires. Il existe différentes versions de cet algorithme : `LZ77` (1977) `LZ78` (1978), `LZW` (Lempel-Ziv-Welch - 1984) ...

Exemple : *"Si TON TONTON TOND TON TONTON, TON TONTON..."*

— **Initialisation :** on part d'un petit dictionnaire associant chaque caractère à un code (de 1 à 25).
— **Progression :** un curseur se déplace sur le message. On reconnaît la sous-chaîne la plus longue existant déjà dans la table de codage. On transmet alors le code de la chaîne (ici S avec 18) et le caractère d'arrêt (le suivant, ici i). On ajoute le tout au dictionnaire.
— **Transmission :** (18,I). On rajoute dans le dictionnaire Si au code 26.
— En suivant (19,O). On rajoute TO en 27. Et ainsi de suite.

Le dictionnaire n'est pas fixe, il évolue en fonction de la transmission. C'est adapté à du codage en ligne grâce à la propriété d'adaptation de cet algorithme. **Le problème est la gestion de la taille du dictionnaire** qu'on peut limiter en amont par exemple. On notera aussi que plus le dictionnaire est gros, plus le temps d'exploration est long ce qui affecte les performances de l'algorithme.

La structure adaptée pour le dictionnaire est un arbre qui nécessitera plus ou moins de mémoire selon la taille du dictionnaire.

2.3 Codage RLE (Run Lenght Encoding)

On considère une suite de bits, par exemple : `000000011111000000` (18 bits) décomposable en 7 bits `0`, 5 bits `1` et 6 bits `0`. On peut représenter ce type de code en donnant **le nombre de bits dans chaque suite**, ce qui donne `111|101|110` pour (756).

Chaque triplet est appelé un **run** (on utilise des triplets dans cet exemple, mais on peut utiliser des runs de n bits). On ne précise pas de quel type de bits il s'agit à chaque fois, dans ce cas on peut imposer quel est le type du premier run (par convention).

Si le nombre de bits dépasse la taille du run, on le décomposera en plusieurs runs en passant un run `000` sur le bit opposé. Par exemple, la séquence `11111111` se codera `111000001`. Cet exemple montre que cet algorithme n'est adapté qu'à certains types de messages car ici, le code est plus long que le mot d'origine. On retrouvera le même problème avec tous les codes ayant une alternance rapide de caractères.

Exemple de RLE sur une image bitmap monochrome :

Codage RLE de cette image bitmap monochrome de 13×10 pixels avec un run de 3 bits en commençant par le run `0`.

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	1	1	0	1	1	1	0	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	1	0
0	1	0	1	1	1	1	1	1	1	0	1	0
0	1	0	1	0	0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 2.3 – Image Space Invader noir et blanc

Détail des calculs :

Code RLE	111	000	111	000	010	001	101	001	111	000
$n\times$ bit	7×0	0×1	7×0	0×1	2×0	1×1	5×0	1×1	7×0	1×1
Numéro du run	1	2	3	4	5	6	7	8	9	10

Code RLE	011	001	111	111	101	010	001	011	001	010
$n\times$ bit	3×0	1×1	7×0	7×1	5×0	2×1	1×0	3×1	1×0	2×1
Numéro du run	11	12	13	14	15	16	17	18	19	20

Code RLE	011	111	000	100	010	001	001	111	001	001
$n\times$ bit	3×0	7×1	0×0	4×1	2×0	1×1	1×0	7×1	1×0	1×1
Numéro du run	21	22	23	24	25	26	27	28	29	30

Code RLE	010	001	001	001	101	001	001	011	101	010
$n\times$ bit	2×0	1×1	1×0	1×1	5×0	1×1	1×0	1×1	5×0	2×1
Numéro du run	31	32	33	34	35	36	37	38	39	40

Code RLE	001	010	111	000	111	000	011
$n\times$ bit	1×0	2×1	7×0	0×1	7×0	0×1	3×0
Numéro du run	41	42	43	44	45	46	47

Code RLE obtenu au final :

```
11100011100001000110100111100001100111111111010100010110010100111110001000100
01000100111100100101000100100110100100101110101000101011100011100001100
```

Remarquons que le codage RLE sur cette image engendre une expansion de données. Il faut $13\times10 = 130$ bits pour coder l'image brute et $47\times3 = 141$ bits pour coder l'image en RLE. Cela est expliqué par l'alternance de bits à 1 et à 0 plus rapides que les runs, engendrant donc une expansion des données.

3 Codage avec pertes

Dans le codage avec pertes, il faut accepter une non-reconstitution intégrale de l'information. On cherche cependant à ce que l'information perdue soit la plus petite possible, pour une compression des données la plus grande possible.

3.1 Le codage prédictif

Le codage se fait ici par l'écart entre prévision et réalité. Ce qui nécessite une loi de prédiction (simple dans la pratique). Par exemple :

$$\hat{x}(t) = x(t-1)$$

Ou encore dans le cas d'une image, avec $\hat{p}_{(i,j)}$ le pixel prévu aux coordonnées (i, j) :

$$\hat{p}_{(i,j)} = \frac{p_{(i-1,j)} + p_{(i,j-1)}}{2}$$

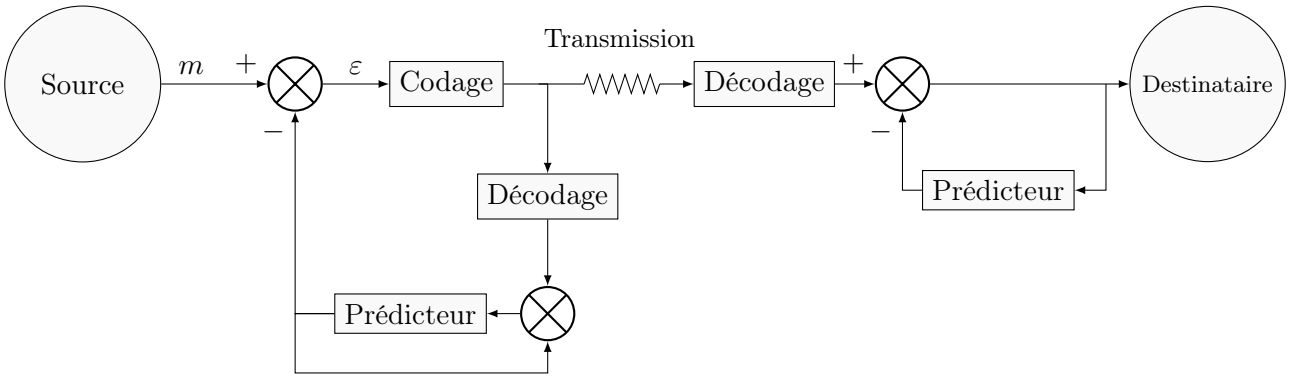

FIGURE 2.4 – Schéma d'une transmission avec codage prédictif

Exemple - Transmission des notes d'un élève :

La transmission des notes d'un élève peut se faire par codage prévisionnel. On prévoit une première note de 12 et une progression de +1 à chaque fois.

Posons les variables suivantes :

N_k, la $k^{\text{ième}}$ note

P_k, la prévision de la $k^{\text{ième}}$ note

E_k, l'écart de la $k^{\text{ième}}$ note à la prévision.

Calcul de l'écart : $E_k = N_k - P_k$ Fonction de prévision : $P_{k+1} = N_k + 1$

Si la première note est effectivement 12, on va donc transmettre un 0. La note suivante est un 14 alors que la note prévue est 13, on transmet donc 1 etc... Voici un exemple détaillé pour quatre notes :

N_k	$P_k = N_{k-1} + 1$	$E_k = N_k - P_k$	E_k
Note	Prévision	Écart	Transmis
12	12	12 - 12 = 0	0
14	12 + 1 = 13	14 - 13 = +1	+1
13	14 + 1 = 15	13 - 15 = −2	-2
16	13 + 1 = 14	16 - 14 = +2	+2

Remarquons que si les notes de l'élèves sont $\{0, 17, 4, 19, 5\}$, ce codage sera moins intéressant.

Le codage prédictif est **intéressant sur un ensemble de données relativement homogènes**. Il est **intéressant lorsque les écarts à la prévision sont petits** et que la **fonction de prévision à un coût faible à mettre en place**.

3.2 Codage par transformations orthogonales

On transforme le signal en coefficients α_i $S = \sum \alpha_i f_i$ avec f_i les fonctions de base Si les f_i forment une famille orthogonale, le calcul des α_i est simple : $< f_i, f_j >= 0$ ou 1 (1 si i=j, 0 sinon) soit $< S, f_i >= \Sigma \alpha_j < f_j, f_i >= \alpha_i < f_i, f_i >$ avec $< f_i, f_i >= 1$ soit $< S, f_i >= \alpha_i$.

Problème : trouver une famille optimale de décomposition.

DCT : Discrete Cosine Transform : DCT II :

$$S = (x_0, \cdots, x_{n-1}) = \sum_{k=0}^{n-1} \alpha_k f_k$$

$$\text{Avec } f_k = \left(\cos\left(\frac{1}{2}\frac{ki\pi}{n}\right), \cos\left(\frac{3}{2}\frac{ki\pi}{n}\right), \cdots, \cos\left(\frac{2i+1}{2}\frac{ki\pi}{n}\right), \cdots, \cos\left(\frac{2n-1}{2}\frac{ki\pi}{n}\right)\right)$$

Propriétés de la DCT :
- Les α_k "importants" ($> \varepsilon$) sont ceux pour lesquels k est petit.
- Nécessité de symétriser X pour pouvoir l'exprimer comme somme de valeurs à base de cosinus.
- En bidimensionnel : il y a des formules analogues.

4 Exemple récapitulatif : codage MPEG vidéo

Le principe du codage MPEG (Moving Picture Expert Group) réside dans deux actions :
- Éliminer la **redondance intra-image** (sur une image assez grande, il y a de grandes chances qu'un pixel soit le même que celui d'à côté).
- Éliminer la **redondance inter-image** (d'une image à la suivante $\frac{1}{25}^{\text{ème}}$ de seconde plus tard)

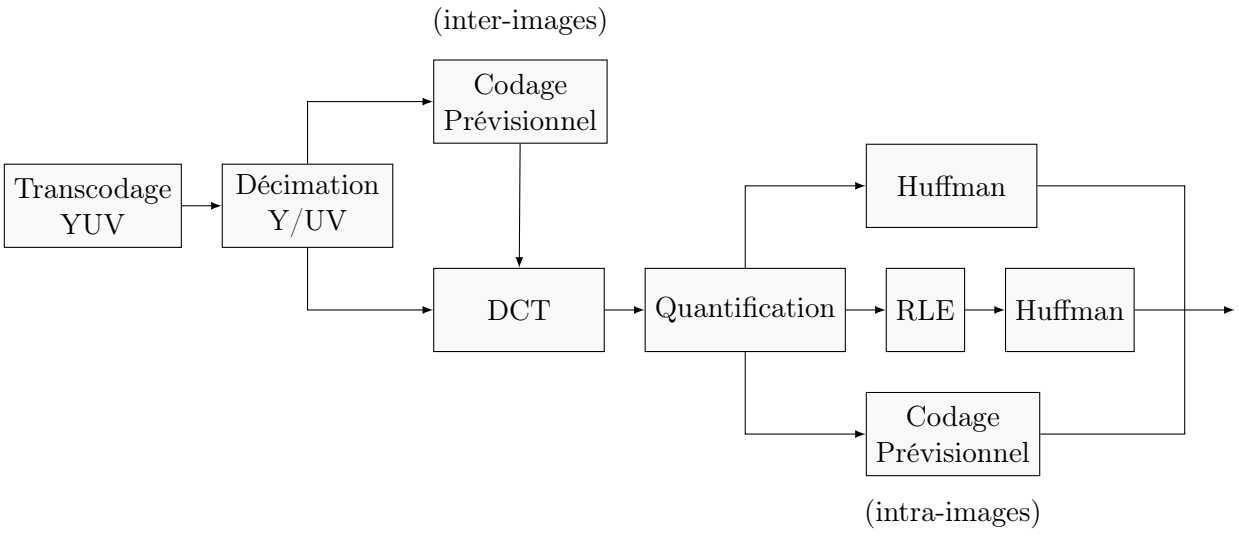

FIGURE 2.5 – Récapitulatif MPEG

Chapitre 3

Codage canal

Le codage canal permet la **détection et la correction d'erreurs**.

Un exemple de codage Canal : Le bit de parité.
Le bit de parité est un bit que l'on rajoute dans le message et qui est le résultat de la somme de tous les bits du message, ce qui permet de détecter un nombre impair d'erreurs dans le message transmis. Un bit de parité ne permet cependant pas de détecter où se situent les erreurs.

1 Théorème fondamental du codage canal

On notera pour le théorème suivant :
— P_e : probabilité d'erreur brute
— P'_e : probabilité d'erreur après correction

$\forall \varepsilon > 0, \forall P_e, \forall P'_e, \exists$ un codage tel que $P'_e < \varepsilon$, à condition que la quantité d'information transmise par message soit inférieure ou égale à une caractéristique du canal qu'on appelle **la capacité du canal**.

Calcul de la capacité du canal : $C = \max\{I(\mathrm{S}, \mathrm{D})\}$

Par exemple pour une transmission électromagnétique
Capacité $= B \times \log_2\left(1 + \frac{S}{N}\right)$
Avec : B la largeur de la bande passante.
S (*Signal* en anglais) l'amplitude du signal (en dB).
N (*Noise* en anglais) l'amplitude du bruit (en dB).

Le codage canal consiste à rajouter une **redondance limitée** et "intelligente" pour pouvoir **détecter et corriger les erreurs**.

2 Les catégories de codage canal

On peut distinguer plusieurs grandes catégories (On notera X le message initial et Y le mot-code)
— Codes instantanés ($Y(t) = f(X(t))$) :
— Codes convolutionnels ($Y(t) = f(X(t), X(t-1), \cdots)$) :

A l'intérieur des codes instantanés, on trouvera :
— Codes "au hasard" (bonnes performances, mais mise en œuvre coûteuse)
— Codes déterministes :
  — Codes linéaires (opérateur $\oplus$ (xor) soit $f(\alpha X \oplus \beta Y) = \alpha f(X) \oplus \beta f(Y))$) :
    — codes cycliques (efficaces contre les erreurs en rafale, qui sont une situation fréquente)
    — les autres

Dans ce chapitre, nous ne traiterons que les codes linéaires.

3 Les codes linéaires

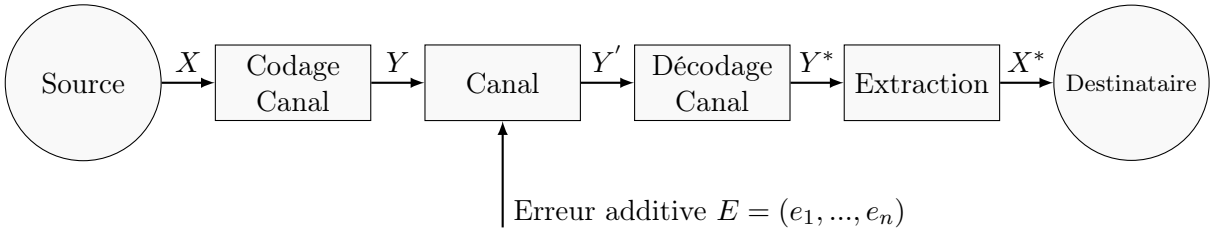

FIGURE 3.1 – Principe du codage canal

$$Y' = Y \oplus E \text{ soit } \forall i, y'_i = y_i \oplus e_i.$$
$$e_i = 1 \Leftrightarrow y'_i \text{ est erroné.}$$

Dimensions :
— $X \in [X]$ espace vectoriel de dimension p.
— $Y \in [Y]$ espace vectoriel de dimension p.
— $\dim[Y'] = \dim[E] = n$.
Passage de X à Y via une matrice $G : Y = X.G = (x_1, ..., x_p).G$
Il vient que G est de dimension $n \times p$. Le code est dit séparable si G permet de recopier X et d'y ajouter les bits de contrôle, on aura alors G de la forme :

$$G = \begin{pmatrix} I_p & A \end{pmatrix} \text{ avec } A \text{ une matrice } (n-p) \times p \text{ qui va définir les bits de contrôle.}$$

Décodage : pour le décodage, on va utiliser une matrice $H_{(n-p)\times n}$ qui va permettre de vérifier si le message a été transformé ou non durant la transmission. Si G est de la forme $\begin{pmatrix} I_p & A \end{pmatrix}$, on pourra prendre H de la forme :

$$H = \begin{pmatrix} A \\ I_{n-p} \end{pmatrix} \text{ avec } A \text{ la même sous-matrice } (n-p) \times p \text{ de la matrice } G.$$

On note S le syndrome. On a alors H la matrice de projection sur $[\bar{Y}]$ telle que $S = Y^{'}H = (Y \oplus E)H$. Il vient alors que si $E = 0$, on aura $S = YH = XGH = 0$ et inversement, si $E \neq 0$ alors $S \neq 0$.

4 Identification de l'erreur grâce à la valeur de S

$S = Y'H = (Y \oplus E)H = YH \oplus EH = XGH \oplus EH = EH$: point de vue de quelqu'un qui connaît l'erreur.

Si $E = (1000...0)$, $S = H_1$ (la première ligne de H). Inversement, si $S = Y'H = H_1$, le plus probable est que $E = (100...0)$.

Plus généralement, si $\exists(i, j, k, ...)$ uniques tel que $S = H_i \oplus H_j \oplus H_k \oplus ...$, le plus probable est que l'erreur soit de la forme : $E = (0...\underset{i}{1}, ..., \underset{j}{1}, ..., \underset{k}{1}, ...0)$.

Algo de correction
$S = Y'H$
Si $S = 0$ alors : décision : transmission OK. $(E^* = (0...0))$.
Sinon on sait $E \neq 0$
Si $\exists i$ unique tel que $S = H_i$
alors $E^* = (0...\underset{i}{1}0...0)$
Sinon si $\exists i, j$ uniques tel que $S = H_i \oplus H_j$
alors $E^* = (0...\underset{i}{1}...\underset{j}{1}...0)$

etc...
$Y^* = Y' \oplus E^*$ (correction)
extraction de X^* depuis Y^*.

Note : Si $\exists i \neq j$ tel que $S = H_i = H_j$ (cas par exemple du bit de parité, S égal à deux lignes de la matrice). On ne fait pas la correction car on ne sait pas où est l'erreur. On peut essayer de corriger au hasard ou demander une ré-émission.

Idem si $\exists i, j, k, l$ $H_i \oplus H_j = H_k \oplus H_l$ etc...

5 Exemple

On prendra dans cet exemple :

$$X = (x_1, ..., x_4) \quad \text{et} \quad Y = (y_1, ..., y_7)$$

La matrice G sera la suivante :

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} I_4 & | & A \end{pmatrix}$$

On a donc $Y = XG$ soit :

$$\begin{array}{ll} y_1 = x_1 & y_5 = x_2 \oplus x_3 \oplus x_4 \\ y_2 = x_2 & y_6 = x_1 \oplus x_3 \oplus x_4 \\ y_3 = x_3 & y_7 = x_1 \oplus x_2 \oplus x_4 \\ y_4 = x_4 & \end{array}$$

Si y_2 est erroné, on aura une erreur sur y'_5 et y'_7 mais pas sur y'_6 etc...
On trouve la matrice H égale à :

$$H = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Premier Exemple

Prenons un message clair $X = (1010)$, et supposons qu'une erreur de transmission ait frappé le $4^{\text{ème}}$ bit du message Y transmis, $Y = (1010101)$, $E = (0001000)$. On trouve $Y' = Y \oplus E = (1011101)$. On peut donc calculer le syndrome $S = Y'H = (111) = H_4$, on en déduit correctement une erreur sur le $4^{\text{ème}}$ bit.

Deuxième Exemple

Pour une erreur additive égale à $E = (0001001)$, on obtient $Y' = Y \oplus E = (1011100)$. On peut donc calculer le syndrome $S = Y'H = (110) = H_3$ ce qui suggère une correction du $3^{\text{ème}}$ bit. Cependant, cette correction sera erronée car Y' contenait initialement deux erreurs.

6 Performances et distances

Distance entre deux mots-code :

La distance entre deux mots-code correspond au nombre de bits qui diffèrent entre X et Y.

$$d(X,Y) = \text{Card}\{i | x_i \neq y_i\} = \text{Poids}(X \oplus Y)$$

Distance minimale d'un code : $d_{min} = \min_{X \neq Y} \{d(X,Y)\} = \min_{X \neq Y} \{poids(X \oplus Y)\}$.

Pour les codes linéaires, $\exists T = Y \oplus Z$ tel que $d_{min} = \min_{T \neq 0} \{poids(T)\}$

Théorème sur les performances :

Un code de distance minimale d_{min} :
— Détecte toute erreur de poids $\leq d_{min} - 1$
— Corrige (correctement) toute erreur de poids $\leq \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$

Chapitre 4

Cryptographie

1 Introduction

Le premier exemple connu de cryptage (datant d'environ -600 avant J.C.) est un extrait de la bible :

Jérémie XXV, 15 à 26	*A tous les rois du septentrion, Proches ou éloignés,* *Aux uns et aux autres, Et à tous les royaumes du monde* *Qui sont sur la face de la terre.* *Et le roi de* ***Schéschac*** *boira après eux.*

Dans les (versets 15 à 22), Jérémie énumère des villes existantes puis vient le roi de Schéschac, ne correspondant à aucune ville existante de l'époque. Jérémie donne la solution un peu plus loin dans le texte :

Jérémie LI, 41	*Eh quoi ! **Schéschac** est prise !* *Celle dont la gloire remplissait toute la terre est conquise !* *Eh quoi ! Babylone est détruite au milieu des nations !*

Le cryptage utilisé est le **cryptage atbash** réalisant des associations par paires de deux lettres de l'alphabet hébreu.

Alphabet Original		Alphabet Associé	
א	A	ת	T
ב	B	ש	S
ד	K	ל	L
...			
ל	L	ד	K
ש	S	ב	B
ת	T	א	A

Ce qui nous donne :

שֵׁשַׁךְ ⟶ ששד ⟶ בבל ⟶ בָּבֶל

Sheshak ⟶ Sh Sh K ⟶ B B L ⟶ Babel

1.1 Vocabulaire

Le terme **Cryptographie** vient des mots grecs : $\kappa\rho\upsilon\pi\tau\grave{o}\varsigma$(caché) et $\gamma\rho\acute{\alpha}\varphi\varepsilon\iota\nu$ (écrire). Il est différent de 'stéganographie' où le message est caché. En cryptographie le message est codé, mais pas caché. L'attaquant peut trouver le message mais ne doit pas pouvoir le lire.

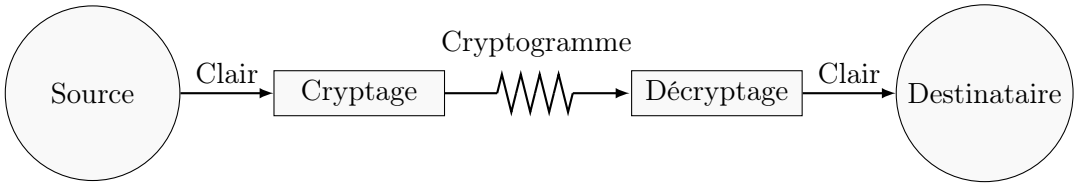

FIGURE 4.1 – Schéma de principe de cryptographie

On notera qu'entre les phases de cryptage et décryptage, des attaques peuvent survenir. Elles sont de deux types, les **attaques actives** (changeant le contenu du message) et les **attaques passives** (écoutant simplement le contenu du message).

But de l'adversaire :
L'adversaire cherche à décrypter un/des cryptogramme(s) et idéalement, reconstituer la clé.

Rappel chapitre 1 : Dans le paradigme de Shannon, il existe un canal sûr permettant l'envoi de la clé de déchiffrement. Un attaquant capable d'écouter ce canal aura accès à la clé et pourra donc décrypter la totalité des messages.

Principes de Kerckhoffs [3] :
— La robustesse peut être pratique ou mathématique.
— Pas de secret sur le cryptosystème, mais seulement sur la clé.

Distance d'unicité : (Shannon)
La **distance d'unicité** (notée D) correspond au nombre de caractères du cryptogramme en dessous duquel il est impossible à un attaquant de décrypter.

$$D = \frac{E(K)}{R.\log_2(N)}$$

Avec :
— N : Nombre de caractères dans l'alphabet utilisé.
— R : redondance du langage du clair. Si $R = 0, E(langage) = \log_2 N$ soit le max). En général $E(langage) = (1 - R)\log_2 N$
— K : Entropie de la source émettant les clés. Si la clé est sur P caractères, $K \leq \log_2 P$

Remarque : Pour rendre un cryptage indéchiffrable, on cherche à se rapprocher de $D = \infty$.
Pour l'obtenir :
— Effectuer un codage source sur le message pour supprimer les redondances ($R = 0$).
— Obtenir une clé de longueur infinie. (cf plus loin)

2 Cryptographie classique

Il existe deux grandes techniques :
— Substitution : Changer un élément par un autre.
— Transposition : Garder tous les caractères du message mais changer leur ordre.

2.1 Substitution monogrammatiques

• Cryptage de Jules CÉSAR

Jules CÉSAR cryptait ses correspondances militaires avec ses généraux en décalant l'alphabet par permutations circulaires de 3 lettres :

$$\mathrm{A} \to \mathrm{D},\ \mathrm{B} \to \mathrm{E},\ \mathrm{C} \to \mathrm{F},\ \cdots,\ \mathrm{Z} \to \mathrm{C}$$

Ce cryptage ne respecte pas le second principe de Kerckhoffs selon lequel il n'y a **pas de secret sur le cryptosystème, mais seulement sur la clé**. Or ce système est un système fixe **sans clé**.

• Cryptage par substitution de type Jules CÉSAR

Ce cryptage s'appuie sur le cryptage de Jules CÉSAR mais en introduisant une clé d que l'on appellera **décalage fixe**, correspondant au nombre de caractères duquel on décale le caractère clair pour obtenir le caractère crypté. De manière évidente, $d \in [0, 25]$ et un cryptage avec $d = 0$ correspondra au message initial.

$$\mathrm{A} \to \mathrm{A} + d,\ \mathrm{B} \to \mathrm{B} + d,\ \mathrm{C} \to \mathrm{C} + d,\ \cdots,\ \mathrm{Z} \to \mathrm{Z} + d$$

Le **décryptage par brute force est réalisable** sur ce cryptage. Dans le meilleur des cas, la première tentative de décryptage sera la bonne. Et dans le pire des cas il faudra 26 essais pour trouver le clair. Il faudra donc en moyenne $\left\lfloor \frac{26-1}{2} \right\rfloor = 13$ essais.

• Substitutions monogrammatiques à alphabet désordonné

Le principe de la substitution à alphabet désordonné repose sur l'association aléatoire de deux lettres de l'alphabet ensemble. Ainsi, il est possible de générer 26! combinaisons (et donc clés) différentes.

$$\mathrm{A} \to \mathrm{D},\ \mathrm{B} \to \mathrm{E},\ \mathrm{C} \to \mathrm{F},\ \cdots,\ \mathrm{Z} \to \mathrm{C}$$

Le **décryptage par brute force n'est pas réalisable** ici, étant donné qu'il y a $26! \simeq 4.10^{26}$ clés possibles. Même en effectuant un essai par nanoseconde (10^{-9} seconde), le temps moyen de décryptage sera de 2.10^{17} secondes soit environ 5 milliards d'années. Cependant, un **décryptage par analyse fréquentielle** peut être mis en place, à condition de connaître la langue dans lequel le message clair a été écrit.

Le décryptage par analyse fréquentielle consiste en une **analyse statistique de la fréquence d'apparition des caractères dans le cryptogramme**. Ensuite, par comparaison avec une table de fréquences d'apparition des lettres de la langue du message clair, il sera possible d'associer chaque caractère du cryptogramme avec le caractère clair correspondant.

Voici les fréquences d'occurrence des 10 lettres les plus utilisées en anglais et en français :

Alphabet Français			
E	17.3%	T	7.1%
A	8.4%	R	6.6%
S	8.1%	L	6.0%
I	7.3%	U	5.7%
N	7.1%	O	5.3%

Alphabet Anglais			
E	12.70%	N	6.75%
T	9.05%	S	6.33%
A	8.17%	H	6.09%
O	7.51%	R	5.99%
I	6.97%	D	4.25%

• **Substitution à alphabets multiples (polyalphabétiques)**

Le chiffre de Vigenère Le chiffre de Vigenère a été exposé par Blaise DE VIGENÈRE dans son ouvrage *Traicté des chiffres, ou Secrètes manières d'escrire* [4] en 1586. Le principe de ce chiffre repose dans **l'association d'une permutation alphabétique à un caractère de la clé**.

Dans la première version de ce chiffre, chaque permutation associé à un caractère de la clé est un cryptage de type Jules CÉSAR. Il existe différentes versions de ce chiffre (notamment en associant un caractère de la clé à un alphabet désordonné), mais nous ne traiterons que la version originale associant chaque caractère de la clé à un cryptage de type Jules CÉSAR. Ce cryptage utilise un alphabet différent pour chaque caractère crypté, en fonction d'une clé (Type Jules César mais le décalage dépend de la clé, décalage à récupérer dans le tableau)

En relevant le caractère présent à l'intersection de la $k^{\text{ème}}$ lettre de la clé avec la $k^{\text{ème}}$ du clair dans le tableau, on forme le cryptogramme.

	A	B	C	$\cdots$	Y	Z
A	A	B	C	$\cdots$	Y	Z
B	B	C	D	$\cdots$	Z	A
C	C	D	E	$\cdots$	A	B
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Y	Y	Z	A	$\cdots$	W	X
Z	Z	A	B	$\cdots$	X	Y

Exemple de clair : `BONJOUR`
Exemple de clé : `BAC`

En utilisant le tableau de cryptage ci-contre, on obtient le cryptogramme : `COPKOWS`

Le mot-clé sera répété autant de fois que nécessaire pour arriver au bout du message. Par exemple pour le mot clé `BAC` et le clair `BONJOUR`, la clé sera : `BACBACB`

Méthode de décryptage

Afin de pouvoir décrypter un cryptogramme issu du chiffre de Vigenère, il est nécessaire de connaître la longueur du mot-clé (qui sera ensuite répété pour former la clé complète).

Décryptage : Il "suffit" de connaître la longueur de la clé pour faire une analyse de fréquence. Ici la clé étant de taille 3, les lettres 1,4,7 du cryptogramme vont être traitées de la même façon, de même pour 2,5 etc...

Pour connaître la longueur de la clé, il faut se baser sur des répétitions dans le cryptogramme qui vont être créées par une redondance de la langue et la répétition de la clé au même endroit.

La méthode est la suivante :

— Relever les répétitions dans le cryptogramme
— En déduire la longueur probable de la clé (via un PGCD).
— Avec les valeurs d'intervalles suivantes $\{21, 35, 70, 61, 14\}$ on trouvera 7 en éliminant la valeur 61 sûrement due au hasard.

La clé peut-être un mot-clé, une phrase-clé, un texte-clé... ou même le clair lui même (on parlera de procédé autoclave) en utilisant une amorce. Exemple : clair = "BONJOUR...", clé = "BACBONJOUR..." (avec BAC l'amorce)

• **Cryptage par clé aléatoire**

Ex : Téléphone rouge

— Clé tirée physiquement au hasard et placé sur une bande magnétique.
— Cryptage du style de Vigenère (clair + clé aléatoire = cryptogramme).
— Deux problèmes : source et destinataire doivent avoir la même clé pour pouvoir déchiffrer et problème de correction d'erreur qui peuvent survenir. (Protocole complexe de transmission des bandes via avion dans le cas du téléphone rouge).

Ce cryptage est strictement indéchiffrable à partir du moment où la clé est tirée au hasard.

2.2 Substitutions polygrammatiques

Le chiffre Playfair, en réalité inventé par Charles WHEATSTONE en 1854, est popularisé par le diplomate anglais Lyon PLAYFAIR au service de la reine Victoria.

Le principe de ce chiffre repose dans une **substitution bigrammatique**. Tout d'abord, l'alphabet est placé dans un tableau carré de 5×5 (en confondant les lettres I et J) ou bien dans un tableau de 6×6 si l'on veut également crypter les chiffres de 0 à 9.

Pour plus de solidité sur ce chiffre il est possible de remplir le tableau à partir d'un mot ou d'une phrase clé. Un exemple avec la clé `PRIX NOBEL` est donné ci-contre.

P	R	I.J	X	N
O	B	E	L	A
C	D	F	G	H
K	M	Q	S	T
U	V	W	Y	Z

Méthode de cryptage :

Pour chaque bigramme du clair, il faut identifier le rectangle formé par les deux coins correspondant à ses lettres. En suite, chaque lettre du clair est cryptée par la lettre du coin opposé du rectangle situé sur la même ligne.

Prenons par exemple le bigramme `AU` du clair `CH|AU|SX|SE|TX|TE`, les lettres `U` et `U` forment deux coins d'un rectangle dans le tableau. On les remplace par les deux lettres par les deux autres coins du rectangle.

O	B	E	L	A
C	D	F	G	H
K	M	Q	S	T
U	V	W	Y	Z

Ainsi le bigramme `AU` est crypté par `OZ`.

Bigramme particulier : Deux lettres identiques.

Si les deux lettres du bigramme sont identiques, une **lettre nulle** (par exemple `X`) est insérée entre ces deux lettres. La lettre nulle doit être assez différente du clair pour que le destinataire puisse comprendre que c'est une lettre nulle. Prenons par exemple le clair `CHAUSSETTE`. Une fois celui-ci séparé en bigrammes on obtient `CH|AU|SS|ET|TE`. Le troisième bigramme étant une lettre double, il faut donc lui rajouter une lettre nulle. On obtient : `CH|AU|SX|SE|TX|TE` (Le rajout de la lettre nulle décale les caractères, il nous faut donc répéter la même opération sur `TT`).

Bigramme particulier : Deux lettres sur la même ligne ou la même colonne.

Si les deux lettres du clair se trouvent sur la **même ligne du tableau**, chaque lettre sera codée par la **lettre à sa droite**. Par exemple (avec la clé PRIX NOBEL) le bigramme `DG` sera codé par `FH`.

De même, si les deux lettres du clair se trouvent sur la **même colonne du tableau**, chaque lettre sera codée par la **lettre du dessous**. Par exemple (avec la clé `PRIX NOBEL`) le bigramme `EQ` sera codé par `FW`.

NOTE : Notons aussi que **les lignes et les colonnes sont circulaires**. Si un décalage à droite sur une ligne engendre une sortie du tableau, il suffit de revenir au début de cette ligne, à gauche. De même pour les colonnes si le décalage vers le bas engendre une sortie du tableau, il suffit de revenir au début de la colonne, en haut.

Cryptanalyse du chiffre de Playfair

par analyse de fréquence des bigrammes

Utilisation de "chiffres" (dictionnaire de cryptage)

— Ex : Reine : 1234 | Roi : 486 | Prince : 132 | ...

— Le 486 a dit à la 1234 que le 132 etc..

— Nécessité d'avoir quand même des cryptogrammes pour les lettres prises isolément

2.3 Transpositions

Le cryptage par transposition est une méthode procédant par simple mélange des unités de cryptage. Nous étudierons ici la transposition à Croix Grecque et la transposition à tableau complet.

• Transposition à Croix Grecque

La transposition à Croix Grecque est nommée ainsi en référence à l'agencement des lettres utilisé dans cette transposition, rappelant la forme de la Croix Grecque.

On place les lettres du clair dans un tableau de 3×3 cases dans l'ordre ci contre.

	1	
2		3
	4	

A chaque fois qu'un tableau de quatre lettres est rempli, on rajoute un autre tableau de quatre lettres à sa droite. Lorsque tous les tableaux sont écrits, il suffit de lire les lettres lignes par lignes pour former le cryptogramme.

Ainsi, le clair : `LA VIE EST BELLE LES OISEAUX CHANTENT` est représenté par :

	L			E			B			E			O			A			H			E	
A		V	E		S	E		L	L		E	I		S	U		X	A		N	N		T
	I			T			L			S			E			C			T			.	

Et le cryptogramme obtenu est : `LEBEO AHEAV ESELL EISUX ANNTI TLSEC T.`

Cryptanalyse :
La fréquence de chaque caractère du clair est respectée de façon exacte dans le cryptogramme. Si l'on connaît la langue dans laquelle le clair est écrit, il est possible d'effectuer une analyse fréquentielle des caractères du cryptogramme pour retrouver le clair. Ce cryptage ne dispose pas de clé. Il est donc faible.

• Transposition à tableau complet

Le cryptage par transposition à tableau complet repose sur l'utilisation d'un mot clé. Ce mot clé est placé horizontalement dans un tableau de manière à ce que chaque lettre de la clé corresponde à une colonne du tableau. Ensuite le clair est écrit de gauche à droite ligne par ligne dans le tableau, sous le mot clé.

Pour obtenir le cryptogramme, on relève simplement les colonnes dans l'ordre alphabétique des lettres de la clé. Si on a deux fois la même lettre, on prend celle la plus à gauche en premier. Voici un exemple :

T	O	U	L	O	U	S	E
L	A	V	I	E	E	S	T
B	E	L	L	E	L	E	S
O	I	S	E	A	U	X	C
H	A	N	T	E	N	T	X

Clair : `LA VIE EST BELLE LES OISEAUX CHANTENT`
Clé : `TOULOUSE`
Cryptogramme : `TSCXI LETAE IAEEA ESEXT LBOHV LSNEL UN`

Dans le mot clé `TOULOUSE`, par ordre alphabétique on va d'abord prendre la lettre `E`, puis `L` puis le premier `O` puis le second `O`, puis `S`, puis `T`, puis le premier `U` et enfin le second `U`.

2.4 Surchiffrement

Le surchiffrement est une opération consistant à crypter un cryptogramme intermédiaire. Nous prendrons l'exemple du système ADFGVX réalisant deux opérations de cryptage, une première par substitution et une seconde par transposition.

• Cryptosystème ADFGVX

Le cryptosystème ADFGVX a été inventé en 1918 par le lieutenant allemand Fritz NEBEL (1891–1977). Il a été utilisé à partir du 5 mars 1918 afin de préparer l'offensive allemande sur Paris. On doit la cryptanalyse de ce chiffre au lieutenant français Georges PAINVIN ayant permis de contrer l'offensive allemande sur Paris. Le nom ADFGVX provient des lettres avec lesquelles le cryptogramme est formé. Ces lettres ont été choisies pour leurs différences dans leur représentation en morse les rendant simples à détecter et évitant les erreurs. (En morse : A ·−, D −··, F ··−·, G −−·, V ···−, X −··−)

Opération de Substitution

Les opérations de substitutions sont réalisées dans un tableau de 6×6 cases (contenant les 26 lettres de l'alphabet et les chiffres de 0 à 9) dont voici un exemple ci-contre.

	A	D	F	G	V	X
A	0	A	B	7	W	4
D	6	N	3	K	J	X
F	Q	D	M	9	C	8
G	O	1	L	U	I	V
V	S	E	Y	5	T	P
X	Z	F	G	H	2	R

Le tableau est remplit pour réaliser des associations d'une lettre du clair vers deux lettres du cryptogramme (correspondant aux coordonnées de la lettre du clair dans le tableau.) Par exemple ici,

`(Z : XA)`, `(F : XD)`, `(M : FF)`, `(4 : AX)`

En prenant par exemple le clair : `LA VIE EST BELLE`

On obtient le cryptogramme : `GF AD GX GV VD VD VA VV AF VD GF GF VD`

Opération de Transposition

L'opération de transposition consiste à écrire une clé dans un tableau et remplir le cryptogramme (issu de l'opération de substitution) ligne par ligne sous la clé. Ensuite, on ordonne les colonnes suivant l'ordre alphabétique (ou l'ordre contre-alphabétique en fonction de ce qui a été décidé) et on relève ligne par ligne les lignes ainsi obtenues et on obtient le cryptogramme.

Reprenons le clair `LA VIE EST BELLE` et effectuons l'opération de transposition avec la clé `CODE`

Clé	C	O	D	E
Message codé	G	F	A	D
	G	X	G	V
	V	D	V	D
	V	A	V	V
	A	F	V	D
	G	F	G	F
	V	D	.	.

Clé Ordonnée	C	D	E	O
Message codé	G	A	D	F
	G	G	V	X
	V	V	D	D
	V	V	V	A
	A	V	D	F
	G	G	F	F
	V	.	.	D

Cryptogramme : `GA DF GG VX VV DD VV VA AV DF GG FF VD`

Cryptanalyse

La seule faiblesse est le facteur humain. En effet il faut une première clé de 36 caractères pour remplir le tableau et une seconde clé pour l'opération de transposition, et il est nécessaire de les noter pour s'en souvenir. L'information peut donc facilement tomber entre les mains de l'ennemi.

3 Cryptographie contemporaine

3.1 Cryptographie à clé secrète

La cryptographie à clé secrète repose sur le même principe que la cryptographie historique en utilisant des mécanismes de substitution et de transposition et une clé partagée entre la source et le destinataire. Ce type de cryptage est basé sur l'utilisation de l'électronique et de l'informatique. Nous traiterons ici de la machine LUCIFER et du Data Encryption Standard (DES).

• La machine LUCIFER

La machine sur laquelle travaillait Horst FEISTEL chez IBM dans les années 1970 s'appellait `Demonstration`. Cependant, le système d'exploitation ne pouvant pas supporter l'usage de noms aussi longs, le nom `Demonstration` fut raccourci en `Demon` qui fut transformé en `Lucifer`.

La machine LUCIFER est basée sur deux blocs, les `P_BOX` et les `S_BOX` :

— **Les** `P_BOX` : Elles réalisent une **permutation** fixe (suivant un câblage physique interne) des bits du mot d'entrée pour créer le mot de sortie.

— **Les** `S_BOX` : Elles réalisent une **substitution** d'un mot de 4 bits par un autre mot de 4 bits. Généralement les `S_BOX` sont des mémoires ROM dans laquelle chaque adresse est pré-remplie par un mot aléatoire de 4 bits. Toutes les valeurs doivent être atteintes (de `0000` à `1111`) par le tirage aléatoire de telle sorte à ce qu'il n'y ai pas de redondance. Ainsi, le mot d'entrée correspond à une adresse mémoire de la `S_BOX` et le mot de sortie correspond au contenu de cette adresse.

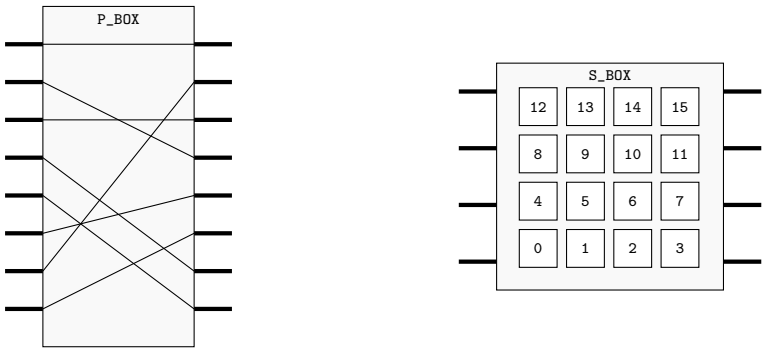

FIGURE 4.2 – Principe des `P_BOX` et `S_BOX`

Fabrication et mise en oeuvre

Le facteur limitant est le nombre de pattes (de l'ordre de 512 à 1024). Le composant réalisant la substitution (la `S_BOX`) est une simple ROM (ou RAM) qui sera limitée à 32 ou 64 bits, soit un nombre très inférieur à la `P_BOX`. Il faudra donc plusieurs `S_BOX` pour une `P_BOX`.

La machine LUCIFER consiste en une juxtaposition alternée de `P_BOX` et `S_BOX` visant à réaliser à la chaîne des **transpositions et des substitutions sur le message d'origine**.

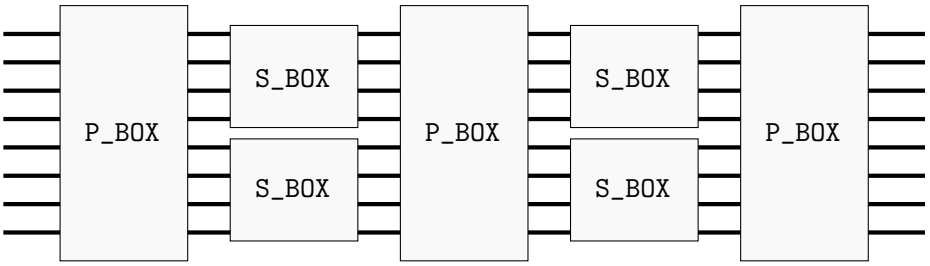

FIGURE 4.3 – Schéma de principe de la Machine LUCIFER

Cryptanalyse de la machine de LUCIFER

— La `P_BOX` (n pattes) est facile à cracker (en n essais) en envoyant des messages composés d'un 1 et de zéros et en observant la sortie. Technologie très fragile, même avec un nombre de pattes important car la `P_BOX` est linéaire.
— La `S_BOX` (p pattes) nécessitera 2^p essais ce qui représente un temps non négligeable et en fait donc un système robuste de part sa réponse non linéaire.

Une très grande faiblesse de la machine LUCIFER est que celle-ci **ne possède pas de clé**. En effet, si une machine tombe entre les mains de l'adversaire, il pourra décrypter l'intégralité des cryptogrammes transmis. Il sera donc nécessaire de réaliser une nouvelle machine, ce qui résulte en un coût logistique très important.

• Data Encryption Standard (DES)

Le cryptage Data Encryption Standard (DES), est basé sur un Schéma de Feistel. Il chiffre des messages clairs de 64 bits en cryptogrammes de 64 bits. La clé utilisée n'est longue que de 56 bits, elle est ainsi vulnérable aux attaques à clair connu, hypothétiquement par volonté de la NSA lors du développement de ce chiffre [5]. Il se déroule en 16 étapes, appelées "*round*" :

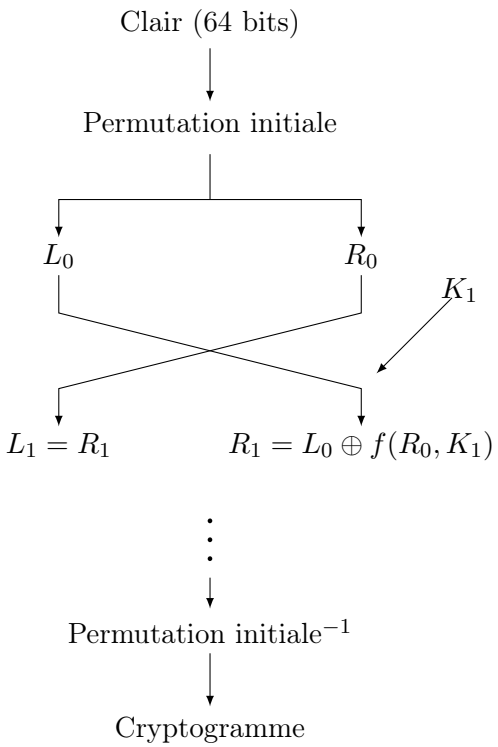

FIGURE 4.4 – DES

Pourquoi le schéma est-il (malgré les apparences) inversible ?

$$L_{i+1} \leftarrow R_i$$
$$R_{i+1} \leftarrow L_i \oplus f(R_i, K_{i+1})$$

On trouve :

$$L_i \leftarrow R_{i+1} \oplus f(L_{i+1}, K_{i+1})$$
$$R_i \leftarrow L_{i+1}$$

Le DES n'est plus normalisé depuis 2002 et a été remplacé par l'AES (Advanced Encryption Standard) même si le DES est toujours très utilisé, notamment dans les banques.

3.2 Cryptographie à clés publiques

Le principe de ce type de chiffrement repose sur un couple (clé privée, clé publique) possédé par chaque participant. Nous verrons ici trois cryptages à clés publiques : Le Protocole de Diffie Hellman (utilisé pour générer les clés du DES), le Cryptage d'ElGamal et le Chiffrement RSA.

• Protocole de Diffie-Hellman

Ce protocole, introduit en 1976 par DIFFIE et HELLMAN [6], repose sur la **construction d'un secret commun** à l'aide d'une **clé privée et d'une clé publique**. Ce système servira de base pour la génération des clés du cryptage DES.

Le protocole de Diffie-Hellman repose sur l'utilisation des "fonctions trappe" (*trap-door one way functions*) dont le calcul direct est réalisable numériquement mais dont le calcul de l'inverse est infaisable numériquement. **Ici la fonction trappe est l'exponentielle.** Le calcul du logarithme discret est infaisable numériquement, alors que le calcul direct de l'exponentielle peut se faire par un **algorithme d'exponentiation rapide**.

Protocole :

Alice choisit un nombre premier p_A, et un nombre entier g_A générateur d'un groupe multiplicatif d'entiers. Alice transmet ensuite sa clé publique, le couple (p_A, g_A), à Bob et reçoit en échange la clé publique (p_B, g_B) de Bob. Alice et Bob établissent alors chacun de leur côté une clé privée, ($\texttt{Priv}_\texttt{A}$ pour Alice et $\texttt{Priv}_\texttt{B}$ pour Bob) et calculent leur clé publique grâce à la formule $\texttt{Pub} = a^{\texttt{Priv}} \mod p$:

Clé publique d'Alice	Clé publique de Bob
$\texttt{Pub}_\texttt{A} = g_A^{\texttt{Priv}_\texttt{A}} \mod p_A$	$\texttt{Pub}_\texttt{B} = g_B^{\texttt{Priv}_\texttt{B}} \mod p_B$

Alice et Bob calculent ensuite le secret commun chacun de leur coté :

Calcul du secret commun par Alice	Calcul du secret commun par Bob
$S_A = \texttt{Pub}_\texttt{B}^{\texttt{Priv}_\texttt{A}} \mod p_A$	$S_B = \texttt{Pub}_\texttt{A}^{\texttt{Priv}_\texttt{B}} \mod p_B$

Preuve de fonctionnement :

$$S_A = \texttt{Pub}_\texttt{A}^{\texttt{Priv}_\texttt{B}} = (a^{\texttt{Priv}_\texttt{A}})^{\texttt{Priv}_\texttt{B}} = a^{\texttt{Priv}_\texttt{B}\texttt{Priv}_\texttt{A}} = \texttt{Pub}_\texttt{B}^{\texttt{Priv}_\texttt{A}} = S_B$$

Ainsi seul Alice et Bob peuvent établir leur secret commun.

• Cryptage d'ElGamal

Ce cryptage à clé publique fut inventé en 1984 par Taher ELGAMAL lorsqu'il travaillait dans les laboratoires Hewlett-Packard. [7]

Aya choisit trois nombres : un nombre quelconque a, un nombre premier p, et une clé privée $\texttt{Priv}_\texttt{A}$. Elle calcule ensuite la valeur de sa clé publique telle que $\texttt{Pub}_A = a^{\texttt{Priv}_\texttt{A}} \mod p$.

Pendant ce temps là, Bachir choisit lui aussi une clé privée $\texttt{Priv}_\texttt{B}$, et calcule la valeur de sa clé publique $\texttt{Pub}_\texttt{B} = a^{\texttt{Priv}_\texttt{B}} \mod p$. Bachir veut envoyer le message M, crypté, à Aya. Il tire $r \in [\![1, p-2]\!]$ au hasard et calcule :

L'indice	La charge utile	Le cryptogramme.
$I = a^r \mod p$	$C_U = M.\texttt{Pub}_\texttt{A}^{\,r} \mod p$	$C = (I, C_U)$

Il envoie ensuite à Aya le cryptogramme constitué de la concaténation de l'indice et de la charge utile.

Pour reconstituer le message à partir du cryptogramme $C = (I, C_U)$ Aya calcule la valeur de M à l'aide du petit théorème de Fermat :

$$M = C_U \times I^{-\mathtt{Priv_A}} \mod p$$

Elle obtient alors le message envoyé par Bachir.

Petit Théorème de Fermat

Si p est un nombre premier et si a est un entier non divisible par p, alors $a^{p-1} - 1$ est un multiple de p. Énoncé mathématique :

$$a^{p-1} - 1 \equiv 0 \mod p$$

Il en découle le calcul de a^{-1} :

$$a^{-1} = a^{p-2} \mod p$$

Démonstration :

$$\forall a,\ a^{p-1} - 1 \equiv 0 \mod p \Longleftrightarrow a^{p-1} = 1 \mod p$$

En multipliant par a^{-1} :

$$\Longrightarrow a^{-1} = a^{-1} \times a^{p-1} = a^{p-2} \mod p$$

• Chiffrement RSA

Le Chiffrement RSA a été inventé en 1977 par Ronald RIVEST, Adi SHAMIR et Leonard ADLEMAN (dont les initiales ont donné le nom du chiffrement : R.S.A.). Il est le cryptage le plus utilisé actuellement parmi les **chiffrements à clés publiques**.

Alice choisit deux nombres premiers p, q et calcule $n = p \times q$, et $\Phi(n) = (p-1)(q-1)$. Les nombres p, q et $\Phi(n)$ sont privés et seul n est public. $\Phi(n)$ est appelé totient (ou indicatrice d'Euler) et est le nombre de nombres (inférieurs à n) premiers avec n. (On rappelle que deux nombres sont premiers entre eux si et seulement si ils ne possèdent qu'un seul diviseur commun = 1)

Alice choisit sa clé privée $\mathtt{Priv_A}$, et elle calcule la valeur de sa clé publique :

$$\mathtt{Pub_A} = \mathtt{Priv_A}^{-1} \mod \Phi(n)$$

Bob veut crypter le message M pour Alice. Il constitue alors le cryptogramme qu'il envoie ensuite à Alice :

$$C = M^{\mathtt{Pub_A}} \mod n$$

Quand Alice reçoit le cryptogramme, elle calcule $C^{\mathtt{Priv_A}} \mod n$ et retrouve le message M que Bob souhaitait lui transmettre.

Exemple de totient avec $\Phi(15)$:

Il y a 8 nombres premiers avec 15 : {1, 2, 4, 7, 8, 11, 13, 14}. Et 15 se décompose en facteurs premiers de la manière suivante $15 = 3 \times 5$. Donc :

$$\Phi(15) = (5-1) \times (3-1) = 8$$

En hommage à Alexander d'Agapayeff (un cryptographe anglais d'origine russe), voici un cryptogramme qu'il proposa en tant qu'exercice à la fin de la première édition de son manuel de cryptographie "Codes and Ciphers" [8] publié en 1939. À ce jour, personne n'a réussi à le décrypter.

Here is a cryptogram upon wich the reader is invited to test his skills :

75628	28591	62916	48164	91748	58464	74748	28483	81638	18174
74826	26475	83828	49175	74658	37575	75936	36565	81638	17585
75756	46282	92857	46382	75748	38165	81848	56485	64858	56382
72628	36281	81728	16463	75828	16483	63828	58163	63630	47481
91918	46385	84656	48565	62946	26285	91859	17491	72756	46575
71658	36264	74818	28462	82649	18193	65626	48484	91838	57491
81657	27483	83858	28364	62726	26562	83759	27263	82827	27283
82858	47582	81837	28462	82837	58164	75748	58162	92000	

Bibliographie

[1] Claude SHANNON. *A Mathematical Theory of Communication*, 1948.

[2] Claude SHANNON. *Communication Theory of Secrecy Systems*, 1949.

[3] Auguste KERCKHOFFS. Desiderata de la cryptographie militaire. *Journal des sciences militaires*, pages 5–38, 161–191, Janvier, Février 1883 (vol. IX).

[4] Blaise DE VIGENÈRE. *Traicté des chiffres, ou Secrètes manières d'escrire.* Abel L'Anglelier, 1586.

[5] Senate State Committee on Intelligence. *Unclassified Summary : Involvement of NSA in the development of the Data Encryption Standard*, April 1978.

[6] Whitfield DIFFIE and Martin E. HELLMAN. *New Directions in Cryptography*, 1976.

[7] Taher ELGAMAL. *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, 1984.

[8] Alexander D'AGAPAYEFF. *Codes and Ciphers.* Oxford University Press, 1939.

Rédaction et mise en forme par Rémi GASCOU (3 MIC 2017-2018)

Avec les contributions de :
François MANACH (3 MIC 2014-2015) - Version 1
Barbara JOANNES - Erwan BEGUIN (3 MIC 2016-2017)
Samy BARRECH - Anaïs RABARY (3 MIC 2016-2017)

- Version du 25 février 2018 -