

Théorie des Codes

Christophe Tilmant (tilmant@isima.fr)

ISIMA/Institut Pascal - Université Clermont Auvergne

ZZ3 F5 - Réseaux et Sécurité Informatique



Organisation du cours

Plan du cours : 14 séances de 2h - **Évaluation sur les TP**

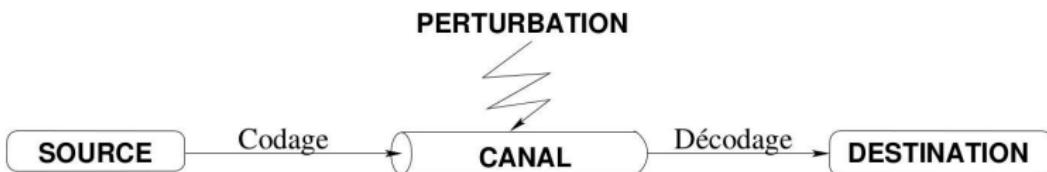
7 cours de 2h (A113) + 7 TP de 2h (A112)

- Efficacité : Codage et Compression - Théorie de l'information ;
 - Intégrité : Détection et correction d'erreurs ;
 - Sécurité : Cryptographie.
- ① Codage entropique : Codage de Huffman ;
 - ② Codage par transformée : Compression par DCT (JPEG) ;
 - ③ Codes linéaires : Code de Hamming ;
 - ④ Codes cycliques : Codes convolutionnels (GSM) ;
 - ⑤ Chiffrement symétrique : Chiffrement de Vigenère ;
 - ⑥ Chiffrement asymétrique : Algorithme RSA ;
 - ⑦ Chiffrement avancé : Algorithme de Shamir.



Introduction

Généralités : Canal de transmission simplifié

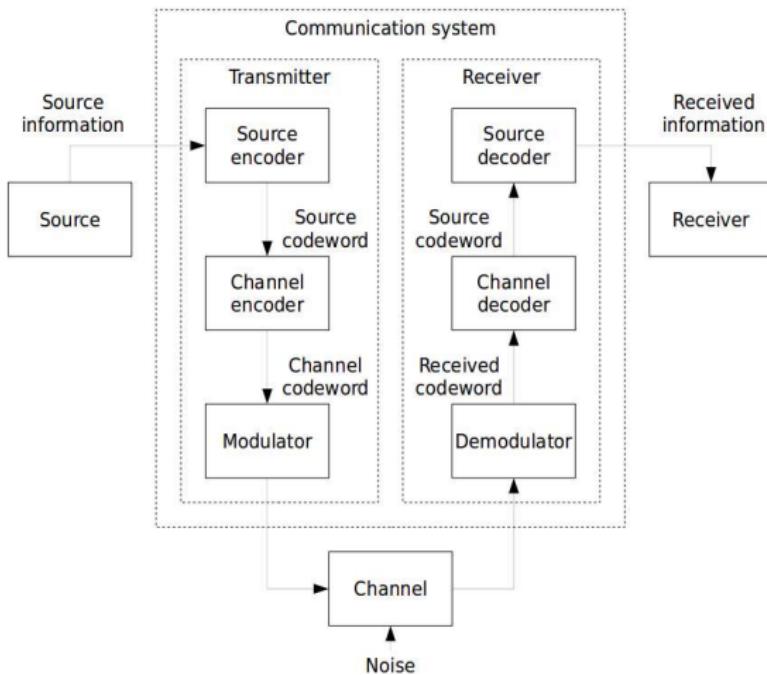


Caractéristiques d'une transmission d'information

- ① Efficacité de la transmission : codage et compression ;
- ② Sécurité de l'information : chiffrement et authentification ;
- ③ Intégrité du message : correction d'erreurs.

Introduction

Généralités : Canal de transmission

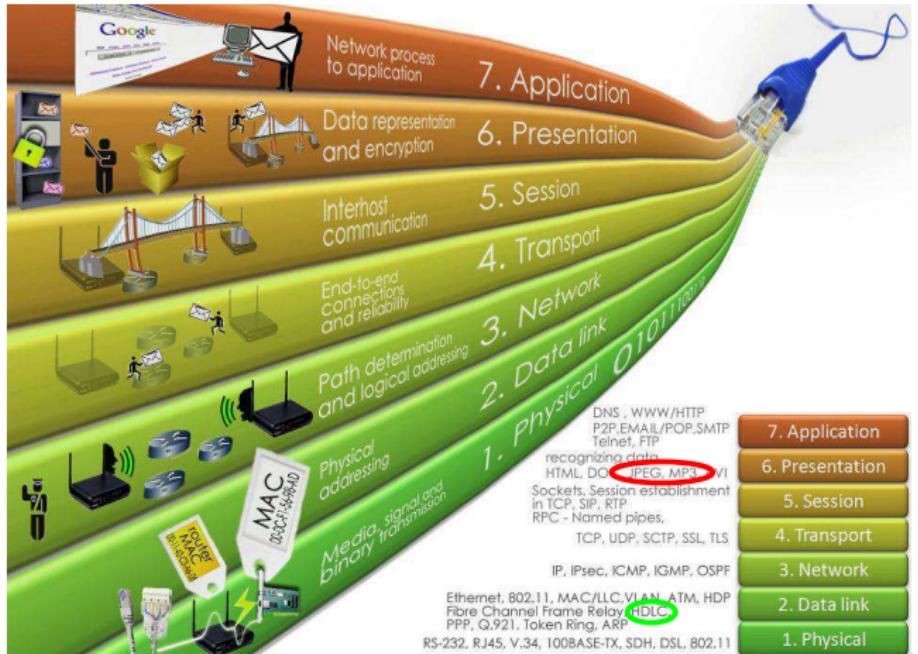


Codages source et canal
Codage source : Codage, compression et chiffrement ;
Codage canal : Codes correcteurs d'erreurs.

Introduction

Généralités : Couche OSI

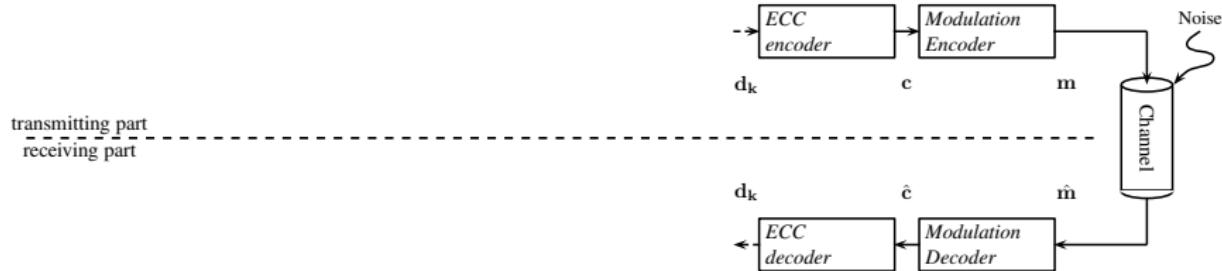
Codage Source ⇒



Codage Canal ⇒

Introduction

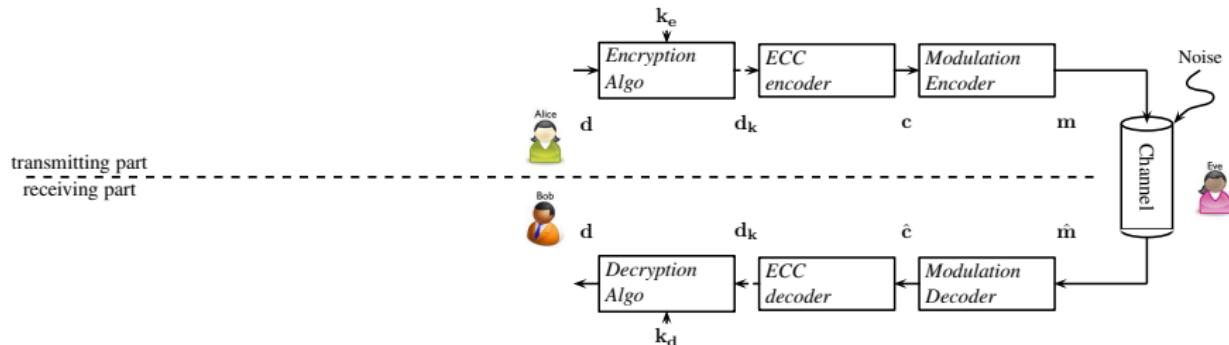
Généralités : Correction, Cryptage, Compression



- Codage canal (couche 2 - Liaison) \Rightarrow ↗ redondance assure l'intégrité : Codes correcteurs d'erreurs ;
- Codage source (couche 6 - Présentation) \Rightarrow
 - ↗ decorrelation assure la confidentialité : Cryptographie ;
 - ↗ redondance assure l'efficacité : Codage et compression

Introduction

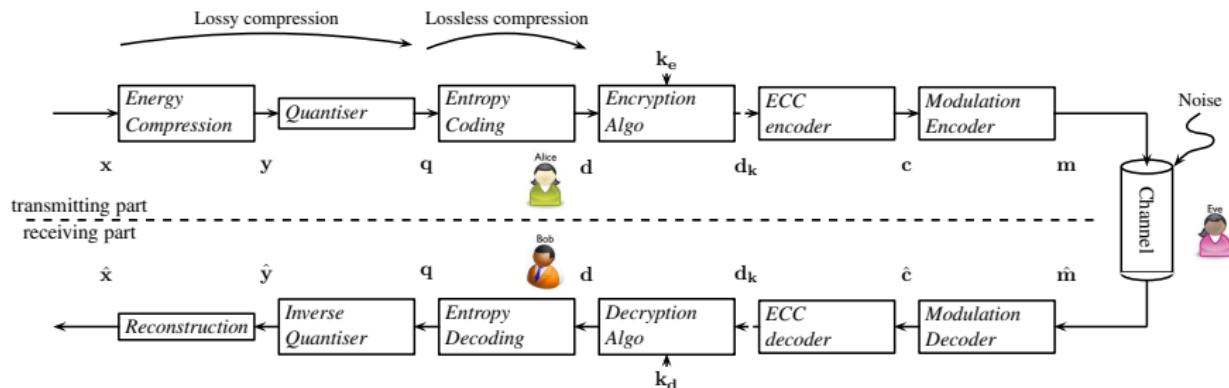
Généralités : Correction, Cryptage, Compression



- Codage canal (couche 2 - Liaison) $\Rightarrow \nearrow$ redondance assure l'intégrité : Codes correcteurs d'erreurs ;
- Codage source (couche 6 - Présentation) \Rightarrow
 - \nearrow décorrélation assure la confidentialité : Cryptographie ;
 - \searrow redondance assure l'efficacité : Codage et compression.

Introduction

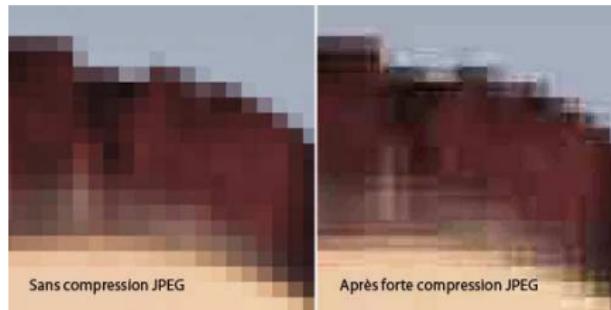
Généralités : Correction, Cryptage, Compression



- Codage canal (couche 2 - Liaison) $\Rightarrow \nearrow$ redondance assure l'intégrité : Codes correcteurs d'erreurs ;
- Codage source (couche 6 - Présentation) \Rightarrow
 - $\bullet \nearrow$ décorrélation assure la confidentialité : Cryptographie ;
 - $\bullet \searrow$ redondance assure l'efficacité : Codage et compression.

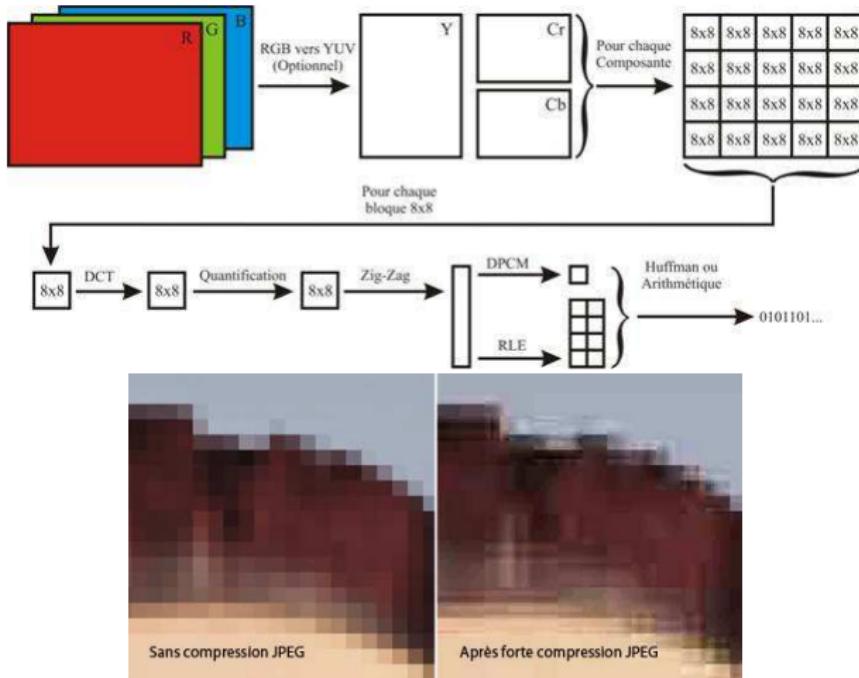
Introduction

Exemple : JPEG → compression



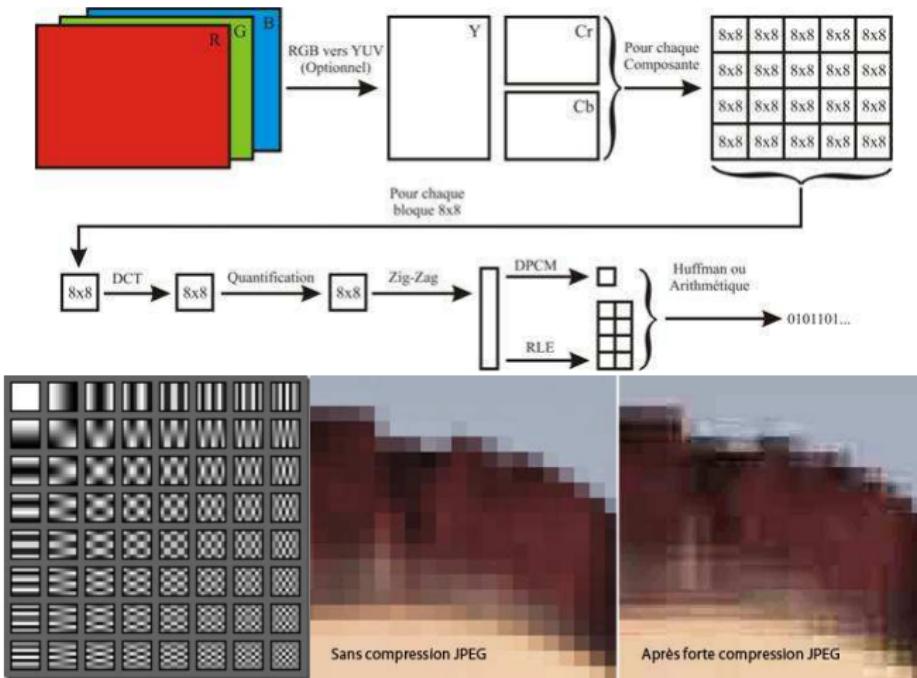
Introduction

Exemple : JPEG → compression



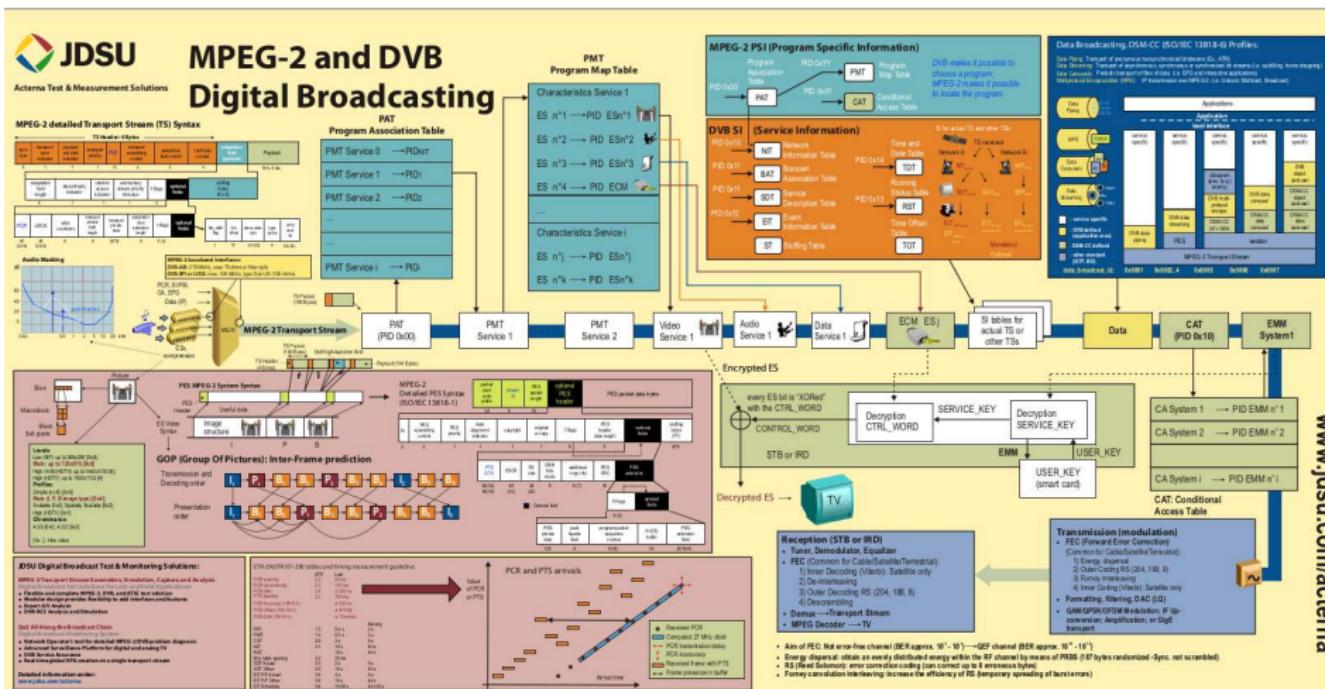
Introduction

Exemple : JPEG → compression



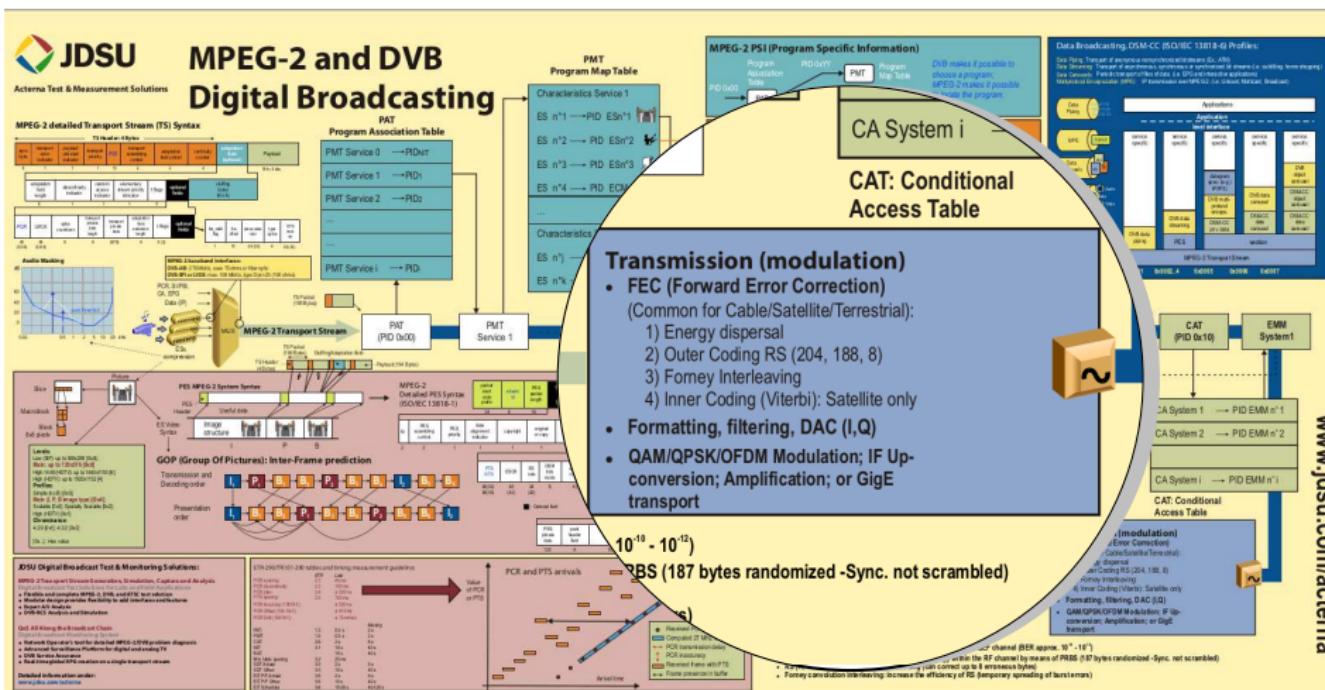
Introduction

Exemple : MPEG-2 → compression, cryptage et correction



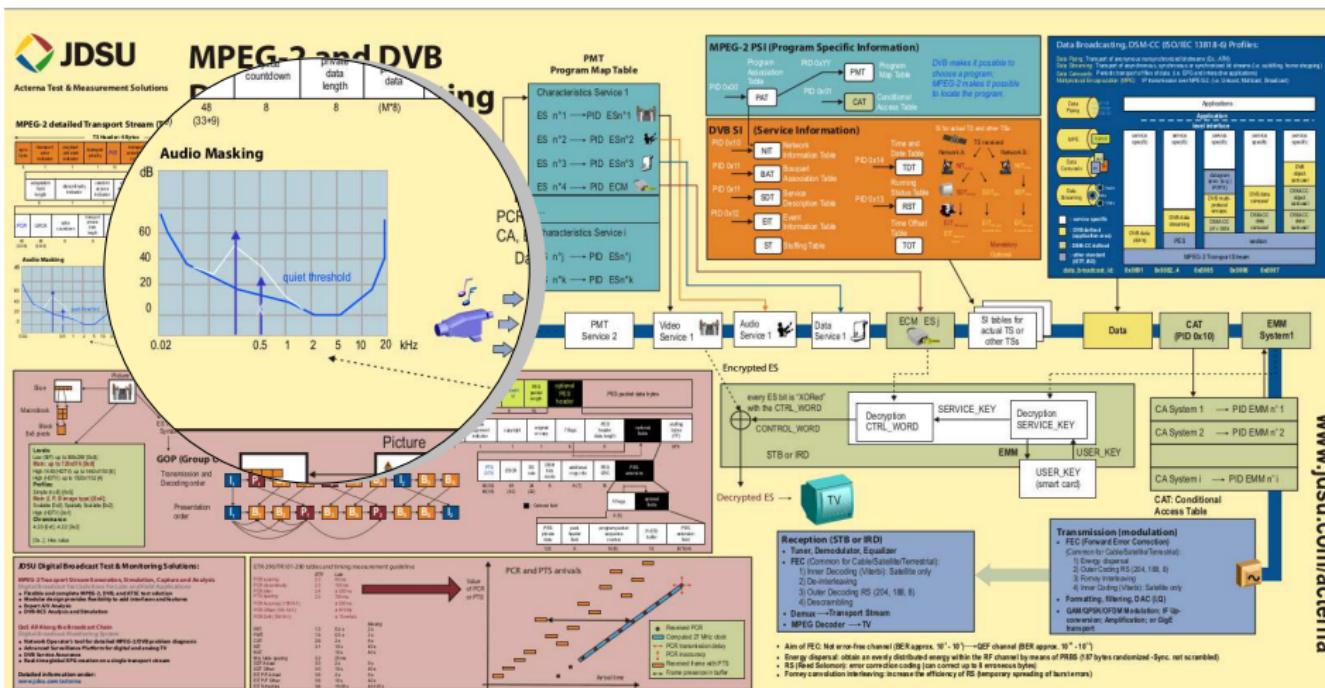
Introduction

Exemple : MPEG-2 → compression, cryptage et correction



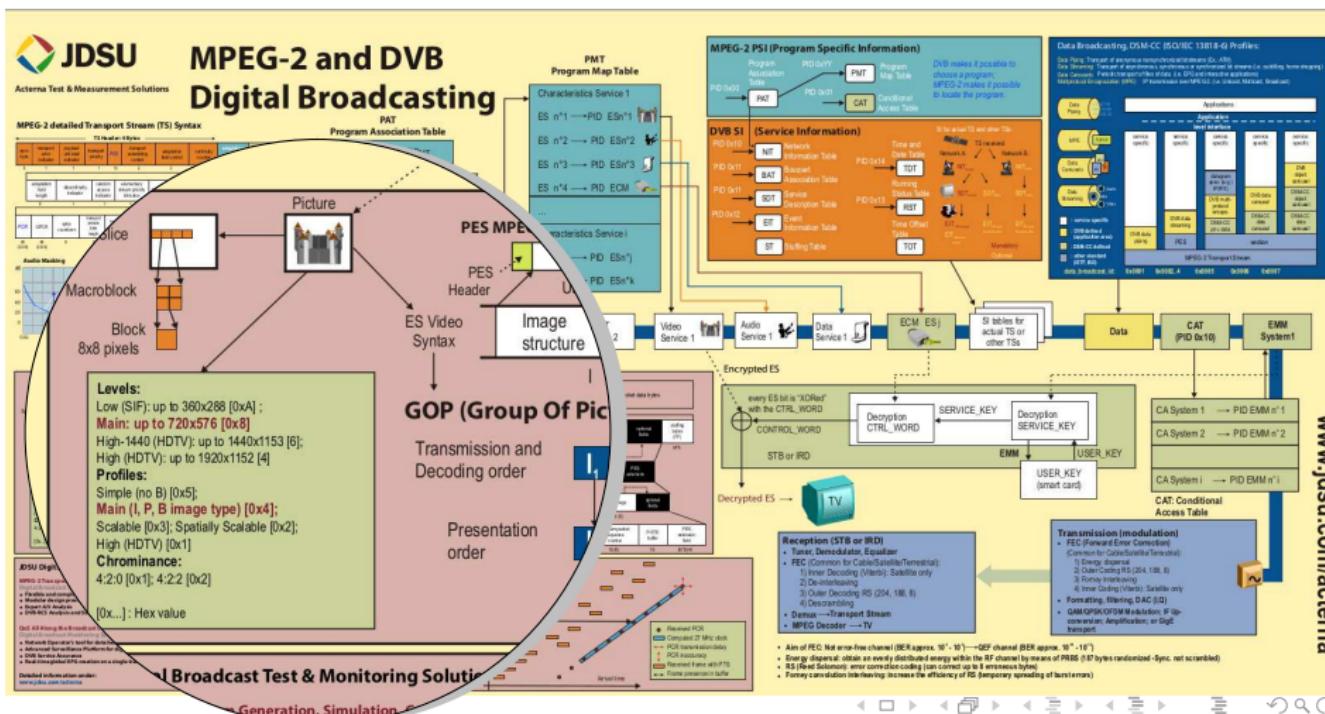
Introduction

Exemple : MPEG-2 → compression, cryptage et correction



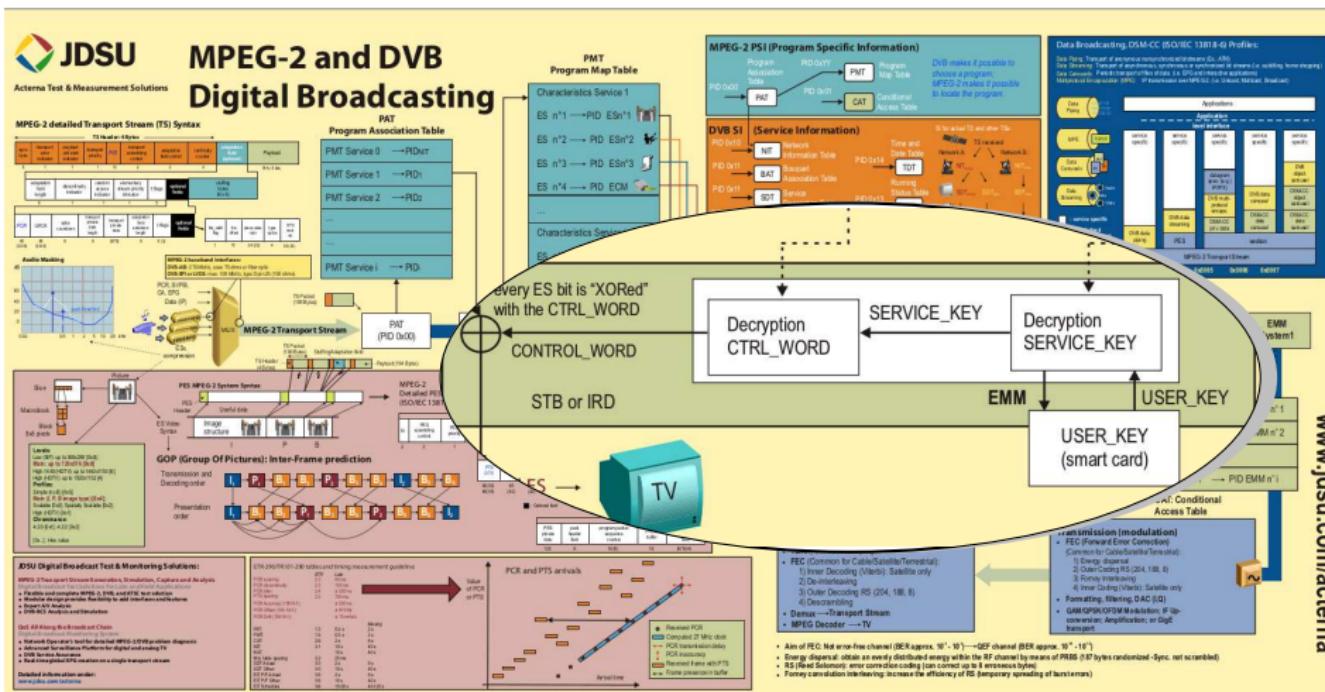
Introduction

Exemple : MPEG-2 → compression, cryptage et correction



Introduction

Exemple : MPEG-2 → compression, cryptage et correction



2 Théorie de l'information et Codage

- Notion de théorie de l'information
- Codage entropique
- Heuristiques de réduction d'entropie

3 Compression par transformée

- Principe d'un codage par transformée
- Choix de la base
- Quantification
- Exemples - Formats standard

4 Détection et correction d'erreurs

- Introduction - Notion de base
- Codes linéaires
- Codes cycliques
- Codes convolutionnels

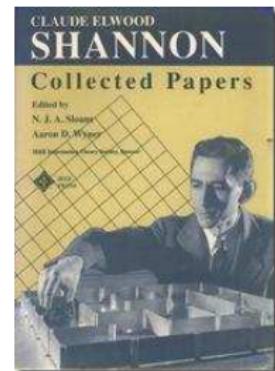
5 Cryptographie

- Les bases de la cryptographie
- Chiffrement à clef secrète
- Chiffrement à clef publique
- Clefs et Cryptanalyse

- 2 Théorie de l'information et Codage
 - Notion de théorie de l'information
 - Codage entropique
 - Heuristiques de réduction d'entropie
- 3 Compression par transformée
- 4 Détection et correction d'erreurs
- 5 Cryptographie

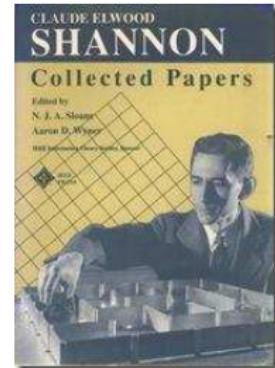
Origines - Objectifs

- Due à Shannon (1948) ;
- Problème de la communication source → récepteur (comment quantifier l'information ?) ;
- Limite du codage ⇒ théorèmes ?
- Comment approcher ces limites ?
- Optimisation codage canal/codage source ?



Origines - Objectifs

- Due à Shannon (1948) ;
- Problème de la communication source → récepteur (comment quantifier l'information ?) ;
- Limite du codage ⇒ théorèmes ?
- Comment approcher ces limites ?
- Optimisation codage canal/codage source ?



La théorie de l'information de Shannon

C'est une **théorie probabiliste** permettant de quantifier le contenu moyen en information d'un ensemble de messages, dont le codage informatique satisfait une distribution statistique précise.

C'est une approche simple \neq sémantique.

Information combinatoire

Soit une source X qui peut envoyer des messages $x_i \in \Omega$ parmi N_Ω possibilités (ex : lettres d'un alphabet à N lettres). On suppose tous les messages aussi vraisemblables.

Si $N_\Omega = 2^n$ il faut donner n informations élémentaires (oui/non) pour spécifier entièrement un message $x_i \in \Omega$.

Information combinatoire

Soit une source X qui peut envoyer des messages $x_i \in \Omega$ parmi N_Ω possibilités (ex : lettres d'un alphabet à N lettres). On suppose tous les messages aussi vraisemblables.

Si $N_\Omega = 2^n$ il faut donner n informations élémentaires (oui/non) pour spécifier entièrement un message $x_i \in \Omega$.

Définition : Information de x_i relative à Ω

$$i_\Omega(x_i) = \log_2 N_\Omega$$

La quantité d'information décrivant l'événement $x_i \in A \subset \Omega$ vaut ainsi :

$$i_\Omega(A) = \log_2 N_\Omega - \log_2 N_A = \log_2 \frac{N_\Omega}{N_A} = -\log_2 \frac{N_A}{N_\Omega}$$

Cadre probabiliste

Plus un message est rare, plus il porte d'information.

Définition de Shannon : Information de x_i relative à Ω

Soit p_i la probabilité d'occurrence du message x_i

$$i_{\Omega}(x_i) = -\log_2 p_i \text{ (bits)}$$

Remarque 1 : $i_{\Omega}(x_i)$ est aussi une mesure d'incertitude a priori sur la réalisation de x_i (on considère alors X comme une v.a.).

Remarque 2 : Message certain $\Rightarrow i_{\Omega}(x_i) = 0$ car $p_i = 1$.

Remarque 3 : Événements équiprobables :

- $p_i = \frac{1}{N_{\Omega}} \Rightarrow i_{\Omega}(x_i) = \log_2 N_{\Omega}$
- $p_A = \frac{N_A}{N_{\Omega}} \Rightarrow i_{\Omega}(x_i) = -\log_2 \frac{N_A}{N_{\Omega}}$

L'information combinatoire est un cas particulier du cadre probabiliste (messages équiprobables).

Entropie comme mesure d'information

Sources discrètes : Émission de suites de messages appartenant à un alphabet Ω de N_Ω messages possibles : $x = \{x_1, \dots, x_i, \dots\}$
On souhaite calculer l'information moyenne portée par un message issu de la source. Dans un cadre probabiliste, la moyenne correspond à l'espérance mathématique.

Entropie comme mesure d'information

Sources discrètes : Émission de suites de messages appartenant à un alphabet Ω de N_Ω messages possibles : $x = \{x_1, \dots, x_i, \dots\}$
On souhaite calculer l'information moyenne portée par un message issu de la source. Dans un cadre probabiliste, la moyenne correspond à l'espérance mathématique.

Entropie d'une source simple

L'entropie d'une source simple est l'espérance mathématique de la variable aléatoire $i(x)$, $x \in \Omega$. Elle est notée $S(X)$ et correspond à une information moyenne par message source :

$$S(X) = - \sum_{i=1}^{N_\Omega} p_i \log_2 p_i \text{ (bits)}$$

Entropie comme mesure d'information

Remarque : On dit aussi "entropie de la loi p_i ". C'est une mesure de dispersion :

- $S(X)$ est maximale si les x_i sont équiprobables, alors $S(X) = \log_2 N_\Omega$.
- si $\exists x_0 \neq p(x_0) = 1$ alors $S(X) = 0$.

Exemple : Entropie de la **loi de Bernoulli**. Deux valeurs avec des proba p et $(1 - p)$.

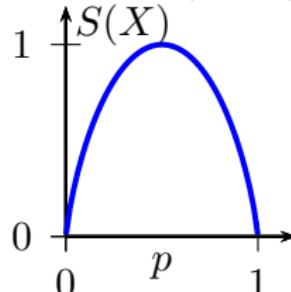
Entropie comme mesure d'information

Remarque : On dit aussi "entropie de la loi p_i ". C'est une mesure de dispersion :

- $S(X)$ est maximale si les x_i sont équiprobables, alors $S(X) = \log_2 N_\Omega$.
- si $\exists x_0 \neq p(x_0) = 1$ alors $S(X) = 0$.

Exemple : Entropie de la **loi de Bernoulli**. Deux valeurs avec des proba p et $(1 - p)$.

$$S(X) = -p \log_2 p - (1 - p) \log_2(1 - p)$$



Principe du codage

Le **codage** consiste à associer à chaque message de X (c'est-à-dire les x_i) un code binaire (mot), noté w_i , de longueur l_i .

Principe du codage

Le **codage** consiste à associer à chaque message de X (c'est-à-dire les x_i) un code binaire (mot), noté w_i , de longueur l_i .

Longueur moyenne d'un code

$$\bar{l}_X = \sum_{i=1}^{N_\Omega} p_i l_i \text{ (bits/symbole)}$$

Principe du codage

Le **codage** consiste à associer à chaque message de X (c'est-à-dire les x_i) un code binaire (mot), noté w_i , de longueur l_i .

Longueur moyenne d'un code

$$\bar{l}_X = \sum_{i=1}^{N_\Omega} p_i l_i \text{ (bits/symbole)}$$

Codage entropique

Le codage entropique consiste à optimiser le choix des $\{w_i\}_{1 \leq i \leq N_\Omega}$ du code pour minimiser \bar{l}_X .

On peut tout coder sur $\lceil \log_2 N_\Omega \rceil$ bits mais on peut réduire \bar{l}_X en utilisant des w_i plus courts pour les x_i les plus fréquents et plus long pour les x_i les plus rares.

Décoder sans ambiguïté : Codes préfixes

Des codes avec des mots de longueur variable \Rightarrow pas toujours déchiffrables de manière unique.

Exemple : $\{x_i\}_{1 \leq i \leq 4} \Rightarrow \{w_1 = 0, w_2 = 10, w_3 = 110, w_4 = 101\}$.
Le message 1010 \Rightarrow w_2w_2 ou w_4w_1 ?? Problème.

Code préfixe

Un code préfixe est tel qu'aucun mot ne soit le préfixe d'un autre.
Tout code possédant la propriété du préfixe est déchiffrable.

Propriété : code préfixe = arbre binaire (feuille)

- En effet, w_k préfixe de $w_j \Leftrightarrow w_k$ ancêtre de w_j .
- Optimiser un code préfixe \Leftrightarrow construire un arbre optimal.

Exemple : $\{w_1 = 0, w_2 = 10, w_3 = 110, w_4 = 111\}$ est code préfixe \Rightarrow déchiffrable de manière unique.

Théorème de Shannon

Théorème de Shannon ou Théorème du codage sans bruit

Soit X une source dont les symboles $\{x_i\}_{1 \leq i \leq N_\Omega}$ ont des probabilités d'occurrences $\{p_i\}_{1 \leq i \leq N_\Omega}$. Le nombre de bits moyen d'un code préfixe vérifie :

$$\boxed{\bar{l}_X \geq S(X)} = - \sum_{i=1}^{N_\Omega} p_i \log_2 p_i$$

De plus il existe un code préfixe tel que $\boxed{\bar{l}_X \leq S(X) + 1}$

Le théorème de Shannon nous explique qu'il existe un code déchiffrable pour s'approcher de l'entropie de la source, mais sans nous donner ce codage.

Principe du codage par blocs

Le codage associe à un message un mot binaire de longueurs l_i entiers. L'entropie $\Rightarrow l_i$ optimal \neq entier.

On peut réduire le "coût" l_i en codant les symboles par blocs de taille K .

Soit la source $\vec{X} = (X_1, \dots, X_K)$ où $X_i \in \Omega$ et $\vec{X} \in \Omega^K$. Le **codage par blocs** consiste à associer à chaque message de \vec{X} un code binaire, noté $w(\vec{x})$, de longueur $l(\vec{x})$.

Principe du codage par blocs

Le codage associe à un message un mot binaire de longueurs $\underline{l_i}$ entiers. L'entropie $\Rightarrow \underline{l_i}$ optimal \neq entier.

On peut réduire le "coût" $\underline{l_i}$ en codant les symboles par blocs de taille K .

Soit la source $\vec{X} = (X_1, \dots, X_K)$ où $X_i \in \Omega$ et $\vec{X} \in \Omega^K$. Le **codage par blocs** consiste à associer à chaque message de \vec{X} un code binaire, noté $w(\vec{x})$, de longueur $l(\vec{x})$.

Longueur moyenne d'un code par blocs

Le nombre moyen de bits par symbole vaut :

$$\bar{l}_X = \frac{1}{K} \bar{l}_{\vec{X}} = \frac{1}{K} \sum_{\vec{x} \in \Omega^K} p(\vec{x}) l(\vec{x})$$

Théorème de Shannon

Théorème de Shannon

Soit X une source d'entropie $S(X)$. Tout code déchiffrable de \vec{X} de longueur moyenne $\bar{l}_{\vec{X}}$ et de longueur moyenne par symbole \bar{l}_X , vérifie :

$$S(\vec{X}) \leq \bar{l}_{\vec{X}} \leq S(\vec{X}) + 1 \Rightarrow \boxed{S(X) \leq \bar{l}_X \leq S(X) + \frac{1}{K}}$$

Théorème

Pour toute source stationnaire d'entropie $S(X)$, il existe un procédé de codage binaire déchiffrable de longueur \bar{l}_X aussi proche que l'on veut de la borne inférieure $S(X)$ ($K \rightarrow +\infty$).

- 2 Théorie de l'information et Codage
 - Notion de théorie de l'information
 - Codage entropique
 - Heuristiques de réduction d'entropie
- 3 Compression par transformée
- 4 Détection et correction d'erreurs
- 5 Cryptographie

Codage de Huffman

Algorithme

On considère N_Ω symboles dont les probabilités d'occurrence sont classées par ordre croissant $p_k \leq p_{k+1}$:

$$\{(x_1, p_1), \dots, (x_i, p_i), \dots, (x_{N_\Omega}, p_{N_\Omega})\}.$$

On agrège les deux symboles de probabilités minimale x_1 et x_2 en un seul symbole $x_{1,2}$ de probabilité $p_{1,2} = p_1 + p_2$.

Un arbre de code préfixe optimal pour les N_Ω symboles s'obtient en construisant un arbre de code préfixe optimal pour les $N_\Omega - 1$ symboles $\{(x_{1,2}, p_{1,2}), (x_3, p_3), \dots, (x_{N_\Omega}, p_{N_\Omega})\}$ et en divisant les feuilles $x_{1,2}$ en deux nœuds fils correspondant à x_1 et x_2 .

C'est une construction récursive, dont le résultat n'est pas forcément unique.

Codage de Huffman : Exemple

Soit la source X constituée des $N = 6$ symboles suivants $\{x_i\}_{1 \leq i \leq N}$ avec leur probabilité d'occurrence $\{p_i\}_{1 \leq i \leq N}$:

Symbol	Probabilité
$x_1 = a$	$p_1 = 0,35$
$x_2 = b$	$p_2 = 0,10$
$x_3 = c$	$p_3 = 0,19$
$x_4 = d$	$p_4 = 0,25$
$x_5 = e$	$p_5 = 0,06$
$x_6 = f$	$p_6 = 0,05$

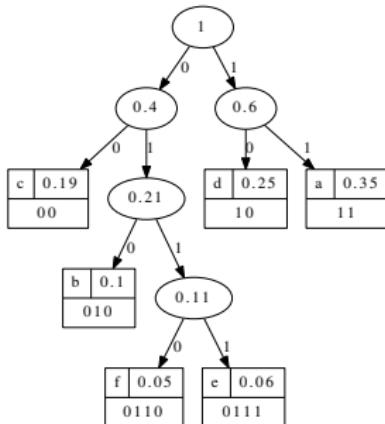
- ① Calculer $S(X)$;
- ② Construire un code de Huffman pour X ;
- ③ Comparer \bar{l}_X et $S(X)$.

Codage de Huffman : Solution

- ① $S(X) = 2,277$ bits, on remarque que $\lceil \log_2 N \rceil = 3$ bits ;

Codage de Huffman : Solution

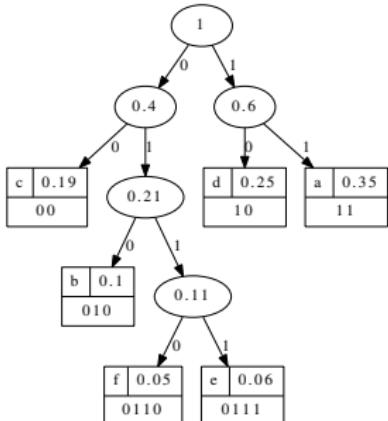
- ① $S(X) = 2,277$ bits, on remarque que $\lceil \log_2 N \rceil = 3$ bits ;
- ② Arbre de Huffman :



Symbole	Probabilité	Mot
$x_1 = a$	$p_1 = 0,35$	11
$x_2 = b$	$p_2 = 0,10$	010
$x_3 = c$	$p_3 = 0,19$	00
$x_4 = d$	$p_4 = 0,25$	10
$x_5 = e$	$p_5 = 0,06$	0111
$x_6 = f$	$p_6 = 0,05$	0110

Codage de Huffman : Solution

- ① $S(X) = 2,277$ bits, on remarque que $\lceil \log_2 N \rceil = 3$ bits ;
- ② Arbre de Huffman :



Symbole	Probabilité	Mot
$x_1 = a$	$p_1 = 0,35$	11
$x_2 = b$	$p_2 = 0,10$	010
$x_3 = c$	$p_3 = 0,19$	00
$x_4 = d$	$p_4 = 0,25$	10
$x_5 = e$	$p_5 = 0,06$	0111
$x_6 = f$	$p_6 = 0,05$	0110

- ③ $\bar{l}_X = 0,35 \times 2 + 0,10 \times 3 + 0,19 \times 2 + 0,25 \times 2 + 0,06 \times 4 + 0,05 \times 4 = 2,32$ bits et $\bar{l}_X \geq S(X)$.

Codage de Huffman

Théorème

Un code issu de l'algorithme de Huffman est optimal parmi tous les codes préfixes.

Attention : Huffman peut coûter 1 bit de plus que l'entropie car l_k est entier. Si un symbole a une probabilité d'occurrence proche de 1 \Rightarrow Huffman est proche de l'entropie + 1 ($S(X) \leq \bar{l}_X \leq S(X) + 1$).

Code de Huffman pour des blocs : code coûteux en temps de calcul

pour estimer les probabilités d'occurrences des vecteurs $\vec{X} \in \Omega^K$.

Codage arithmétique

Le codage arithmétique est une méthode statistique souvent meilleure que le code de Huffman ! En codage arithmétique, un symbole peut être codé par un nombre non entier de bits.

Codage par blocs \Rightarrow intervalles emboîtés.

Soit un bloc de symbole $\vec{x} = (x_1, \dots, x_K)$ produit par un vecteur aléatoire $\vec{X} = (X_1, \dots, X_K)$ de K variables aléatoires indépendantes. Chaque X_k a la même densité de probabilité $p(x)$ avec $p(x_k) = p_k$.

Codage arithmétique

Un code arithmétique représente un bloc de symbole \vec{x} par un intervalle $[a_K; a_K + b_K]$ inclus dans $[0; 1]$ avec b_K égale à la probabilité d'occurrence de la séquence : $b_K = \prod_{k=1}^K p(x_k)$

Codage arithmétique

Algorithme du codage arithmétique

- $a_0 = 0$ et $a_1 = 1$ (Initialisation \Rightarrow codage de 0 symbole)
- Soit $[a_i; a_i + b_i]$ l'intervalle correspondant aux i premiers symboles x_1, \dots, x_i
- Le nouvel intervalle $[a_{i+1}; a_{i+1} + b_{i+1}]$ est un sous-intervalle de $[a_i; a_i + b_i]$ avec une longueur réduite de $p(x_{i+1})$:

$$b_{i+1} = b_i \times p(x_{i+1}) \text{ et } a_{i+1} = a_i + b_i \times \prod_{k=1}^i p(x_k)$$

- L'intervalle final $[a_K; a_K + b_K]$ représente la séquence x_1, \dots, x_K où \overrightarrow{x} est codé par un nombre binaire $c_n \in [a_K; a_K + b_K]$

Codage arithmétique : Exemple

Soit la source X constituée des $N = 6$ symboles suivants $\{x_i\}_{1 \leq i \leq N}$ avec leur probabilité d'occurrence $\{p_i\}_{1 \leq i \leq N}$:

Symbol	Probabilité
$x_1 = a$	$p_1 = 0,1$
$x_2 = b$	$p_2 = 0,1$
$x_3 = c$	$p_3 = 0,1$
$x_4 = d$	$p_4 = 0,2$
$x_5 = e$	$p_5 = 0,4$
$x_6 = f$	$p_6 = 0,1$

- ➊ Réaliser un codage arithmétique de la séquence "bebecafdead".
- ➋ Comment réaliser le décodage ?

Codage arithmétique : Solution

① Découpage de l'intervalle :

Symbol	$x_1=a$	$x_2=b$	$x_3=c$	$x_4=d$	$x_5=e$	$x_6=f$
Probabilité	$p_1=0,1$	$p_2=0,1$	$p_3=0,1$	$p_4=0,2$	$p_5=0,4$	$p_6=0,1$
Intervalle	$[0;0,1[$	$[0,1;0,2[$	$[0,2;0,3[$	$[0,3;0,5[$	$[0,5;0,9[$	$[0,9;1[$

Codage arithmétique : Solution

① Découpage de l'intervalle :

Symbol	$x_1=a$	$x_2=b$	$x_3=c$	$x_4=d$	$x_5=e$	$x_6=f$
Probabilité	$p_1=0,1$	$p_2=0,1$	$p_3=0,1$	$p_4=0,2$	$p_5=0,4$	$p_6=0,1$
Intervalle	$[0;0,1[$	$[0,1;0,2[$	$[0,2;0,3[$	$[0,3;0,5[$	$[0,5;0,9[$	$[0,9;1[$

② Compression et décompression :

Symbol	Borne inférieure	Borne supérieure	Réel	Intervalle	Symbol	Taille
b	0.1	0.2	0.15633504500	[0,1,0,2[b	0,1
e	0.15	0.19	0.5633504500	[0,5,0,9[e	0,4
b	0.154	0.158	0.158376125	[0,1,0,2[b	0,1
e	0.1560	0.1576	0.58376125	[0,5,0,9[e	0,4
c	0.15632	0.15648	0.209403125	[0,2,0,3[c	0,1
a	0.156320	0.156336	0.09403125	[0,0,0,1[a	0,1
f	0.1563344	0.1563360	0.9403125	[0,9,1,0[f	0,1
d	0.15633488	0.15633520	0.403125	[0,3,0,5[d	0,2
e	0.156335040	0.156335168	0.515625	[0,5,0,9[e	0,4
a	0.156335040	0.1563350528	0.0390625	[0,0,0,1[a	0,1
d	0.15633504384	0.15633504640	0.390625	[0,3,0,5[d	0,2



Codage Adaptatifs : Algorithme de Huffman Dynamique

Compression d'un flot à la volée en faisant une seule lecture.

Algorithme Algorithme de Huffman dynamique : compression.

Soit $nb(c)$, le nombre d'occurrence d'un caractère c

Initialiser l'arbre de Huffman (AH) avec le caractère @, $nb(@) \leftarrow 1$

Tant que on n'est pas à la fin de la source **Faire**

 Lire un caractère c de la source

Si c'est la première occurrence de c **Alors**

$nb(c) \leftarrow 0$

$nb(@) \leftarrow nb(@) + 1$

 Afficher en sortie le code de @ dans AH suivi de c .

Sinon

 Afficher le code de c dans AH

Fin Si

$nb(c) \leftarrow nb(c) + 1$

 Mettre à jour AH avec les nouvelles fréquences

Fin Tant que

Codage Adaptatifs : Algorithme de Huffman Dynamique

Algorithm Algorithme de Huffman dynamique : décompression.

Soit $nb(c)$, le nombre d'occurrence d'un caractère c

Initialiser l'arbre de Huffman (AH) avec le caractère @, $nb(@) \leftarrow 1$

Tant que on n'est pas à la fin du message codé **Faire**

 Lire un mot de code c du message (jusqu'à une feuille de AH)

Si $c = @$ **Alors**

$nb(@) \leftarrow nb(@) + 1$

 Lire dans c les k bits du message et les afficher en sortie.

$nb(c) \leftarrow 0$

Sinon

 Afficher le caractère correspondant à c dans AH

Fin Si

$nb(c) \leftarrow nb(c) + 1$

 Mettre à jour AH avec les nouvelles fréquences

Fin Tant que

- 2 Théorie de l'information et Codage
 - Notion de théorie de l'information
 - Codage entropique
 - Heuristiques de réduction d'entropie
- 3 Compression par transformée
- 4 Détection et correction d'erreurs
- 5 Cryptographie

Introduction

Heuristiques de réduction d'entropie

Le codage statistique tire parti des caractères apparaissant souvent (p forte). Les codages entropiques (codage de Huffman par exemple) présentent des résultats théoriques intéressants.

Ils sont souvent utilisés avec d'autres techniques de codage. Ces autres procédés ne sont pas toujours optimaux en théorie mais ils permettent en amont de **diminuer l'entropie de la source**, de ce fait le codage entropique s'en trouve plus efficace.

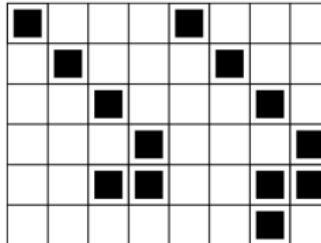
Run-length encoding (RLE)

Le codage RLE lit chaque caractère à la suite mais, lorsque 3 caractères identiques successifs sont rencontrés, il utilise un code spécial de répétition (@ par exemple) suivi du caractère à répéter et du nombre de répétitions.

Exemple : "aabbbcdfffff@eeee"

Exercice :

- Quel serait "un" codage RLE de l'image suivante ?



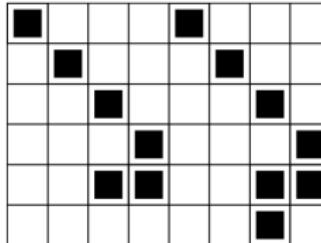
Run-length encoding (RLE)

Le codage RLE lit chaque caractère à la suite mais, lorsque 3 caractères identiques successifs sont rencontrés, il utilise un code spécial de répétition (@ par exemple) suivi du caractère à répéter et du nombre de répétitions.

Exemple : "aabbbcdffff@eeee" \Rightarrow "aa@b3c@f5@@1@e4"

Exercice :

- Quel serait "un" codage RLE de l'image suivante ?



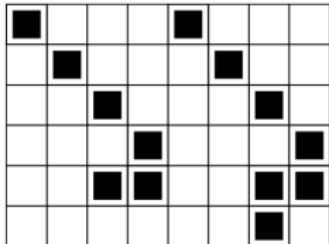
Run-length encoding (RLE)

Le codage RLE lit chaque caractère à la suite mais, lorsque 3 caractères identiques successifs sont rencontrés, il utilise un code spécial de répétition (@ par exemple) suivi du caractère à répéter et du nombre de répétitions.

Exemple : "aabbbcdffff@eeee" \Rightarrow "aa@b3c@f5@@1@e4"

Exercice :

- Quel serait "un" codage RLE de l'image suivante ?



\Rightarrow 681314131413141312222611

Move to Front (MTF)

L'algorithme MTF est un système de transformation de flot, il consiste à remplacer chaque caractère par un indice, donné par un tableau évoluant de manière dynamique. Dès qu'un caractère apparaît il est d'abord codé par sa valeur, puis il passe en début de liste. Cette technique est notamment utilisable en conjonction avec la transformée de Burrows-Wheeler (BTW).

Le décodage est tout aussi simple : à partir du même tableau initial, il suffit d'émettre le caractère correspondant à l'indice et de ranger le tableau en passant ce caractère en premier. Le tableau évolue exactement comme pendant la phase de codage.

Move to Front : Exemple

Itération	Séquence	Liste
bananaaa	1	(a b cdefghijklmnopqrstuvwxyz) ↑
bananaaa	1,1	(b a cdefghijklmnopqrstuvwxyz) ↑
ba n anaaa	1,1,13	(abcde fghijklm n opqrstuvwxyz) ↑
bananaaa	1,1,13,1	(nabcde fghijklmopqrstuvwxyz)
bananaaa	1,1,13,1,1	(anbcde fghijklmopqrstuvwxyz)
bananaaa	1,1,13,1,1,1	(nabcde fghijklmopqrstuvwxyz)
bananaaa	1,1,13,1,1,1,0	(anbcde fghijklmopqrstuvwxyz)
bananaaa	1,1,13,1,1,1,0,0	(anbcde fghijklmopqrstuvwxyz)
Finale	1,1,13,1,1,1,0,0	(anbcde fghijklmopqrstuvwxyz)

Burrows-Wheeler Transform (BWT)

L'idée de Burrows et Wheeler est de trier les caractères (n'apporte aucun gain de compression immédiat) afin que le MTF et le RLE soient les plus efficaces possibles.

BWT

- La chaîne de caractères = première ligne d'un tableau ;
- Nouvelle ligne = chaîne d'un caractère décalée vers la gauche ;
- Ces lignes sont ensuite classées par ordre alphabétique ;
- On extrait la dernière colonne et le rang de la première lettre.

Chaque dernière lettre de chaque ligne précède la première lettre de la même ligne (sauf pour la ligne originale). Les lignes étant rangées par ordre alphabétique, on peut retrouver la première colonne du tableau grâce à la dernière colonne

Burrows-Wheeler Transform (BWT) : Exemple

Ici on code "compresso" par "4ESROCMPSE"

	D ₀									D ₀	F	L
D ₀	C	O	M	P	R	E ₁	S ₁	S ₂	E ₂		C	E ₂
D ₁	O	M	P	R	E	S	S	E	C		E ₂	S ₂
D ₂	M	P	R	E	S	S	E	C	O		E ₁	R
D ₃	P	R	E	S	S	E	C	O	M	tri	D ₂	M
D ₄	R	E	S	S	E	C	O	M	P	→	D ₁	O
D ₅	E ₁	S	S	E	C	O	M	P	R		D ₃	P
D ₆	S ₁	S	E	C	O	M	P	R	E ₁		D ₄	R
D ₇	S ₂	E	C	O	M	P	R	E	S ₁		D ₇	S ₂
D ₈	E ₂	C	O	M	P	R	E	S	S ₂		D ₆	S ₁

Remarque : Texte relativement long (plusieurs fois les mêmes mots), le texte codé comportera de nombreuses répétitions de caractères. Un exemple est la transformation de "TEXTUELTEXTUEL" qui donne : "7UUTTEELLXXTTEE"

Burrows-Wheeler Transform (BWT) : Décompression

Algorithm 1 Reciproque de la BWT

- 1: On dispose de la chaîne **L** et de l'index primaire *index*
 - 2: **F** \leftarrow tri de **L**
 - 3: Calculer le vecteur de transformation **H** tel que $\forall j \quad L[H[j]] = F[j]$
 - 4: **for** *i* de 0 à Taille de **L** **do**
 - 5: Afficher **L**[*index*]
 - 6: *index* \leftarrow **H**[*index*]
 - 7: **end for**
-

Algorithme de Lempel-Ziv

C'est un algorithme de compression par dictionnaire (on code l'indice dans un dictionnaire des symboles). Il y a plusieurs variantes de cet algorithme (LZ77, LZ78, LZMA, LZW, ...)

Algorithme LZW : compression.

Soit $chaine \leftarrow \emptyset$ la chaîne courante

Tant que on n'est pas à la fin de la source **Faire**

Lire un caractère c de la source

Si $chaine|c$ est dans le dictionnaire **Alors**

$chaine \leftarrow chaine|c$

Sinon

Afficher le code de $chaine$

Ajouter $chaine|c$ au Dictionnaire

$chaine \leftarrow c$

Fin Si

Fin Tant que

Algorithme LZW : décompression.

Lire un octet $prec$ du message codé

Tant que on n'est pas à la fin du message codé **Faire**

Lire un octet $cour$ du message codé

Si $cour$ est dans le dictionnaire **Alors**

$chaine \leftarrow$ la traduction de $cour$ dans le dictionnaire

Sinon

$chaine \leftarrow$ la traduction de $prec$ dans le dictionnaire

$chaine \leftarrow chaine|cour$

Fin Si

Afficher $chaine$

Soit c le premier caractère de $chaine$

Soit mot la traduction de $prec$ dans le dictionnaire

Ajouter $mot|c$ au dictionnaire

$prec \leftarrow cour$

Fin Tant que

LZW : Exemple

Exercice En utilisant la table ASCII hexadécimale (7bits) ci-dessous comme dictionnaire initial, compresser et décompresser la chaîne “BLEBLBL-BA” avec LZW.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	\\$	%	&	,	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	©	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	-
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

LZW : Solution

Exercice

La compression de “BLEBLBLBA” par LZW donne donc le résultat suivant :

chaîne	Affichage	Dictionnaire
B	~~ 42	BL ↔ 80
L	~~ 4C	LE ↔ 81
E	~~ 45	EB ↔ 82
BL	~~ 80	BLB ↔ 83
BLB	~~ 83	BLBA ↔ 84
A	~~ 41	

Si l'on décomprime cette sortie sans traiter le cas particulier, on obtient à cause du décalage de mise en place du dictionnaire la sortie suivante :

Code	Affichage	Dictionnaire
42	~~ B	
4C	~~ L	BL ↔ 80
45	~~ E	LE ↔ 81
80	~~ BL	EB ↔ 82
83	~~ ???	

Puisque que le code 83 n'est pas connu c'est que l'on est précisément dans le cas particulier et que le prochain groupe est donc forcément la répétition du précédent (*BL*) augmenté de la première lettre du groupe, soit *BLB*. On voit donc bien ici, que le traitement du cas particulier permet de mettre à jour le dictionnaire une itération plus tôt. Ainsi, toute chaîne répétée deux fois de suite peut être traitée correctement à la décompression.

Exemples

- Commande "pack" de Unix (obsolète) \Rightarrow Huffman Dynamique.
- bzip2



- Commande "compress" de Unix implémente LZ77.
- Commande "gzip" est basé sur l'algorithme *deflate*, qui est une combinaison des algorithmes LZ77 et Huffman (brevet de LZW).
- Format gif (LZW) \Rightarrow png (équivalent avec LZ77)

Comparaison des algorithmes de compression

Algorithme	Fichier compressé	Taux	codage	décodage
7-Zip-4.42 (LZMA+?)	5.96 Mo	62.57%	23.93s	6.27s
RAR-3.51 (?)	6.20 Mo	61.07%	14.51s	0.46s
rzip-2.1 -9 (LZ77+Go)	6.20 Mo	61.09%	9.26s	2.94s
ppmd-9.1 (Prédicatif)	6.26 Mo	60.71%	11.26s	12.57s
bzip2-1.0.3 (BWT)	6.69 Mo	57.96%	7.41s	3.16s
gzip-1.3.5 -9 (LZ77)	7.68 Mo	51.77%	2.00s	0.34s
gzip-1.3.5 -2 (LZ77)	8.28 Mo	47.99%	1.14s	0.34s
WinZip-9.0 (LZW+?)	7.72 Mo	51.55%	5s	5s
compress (LZW)	9.34 Mo	41.31%	1.11s	0.32s
lzop-1.01 -9 (LZW+?)	9.35 Mo	41.32%	6.73s	0.09s
lzop-1.01 -2 (LZW+?)	10.74 Mo	32.54%	0.45s	0.09s
pack (Huffman)	11.11 Mo	30.21%	0.33s	0.27s

2 Théorie de l'information et Codage

3 Compression par transformée

- Principe d'un codage par transformée
- Choix de la base
- Quantification
- Exemples - Formats standard

4 Détection et correction d'erreurs

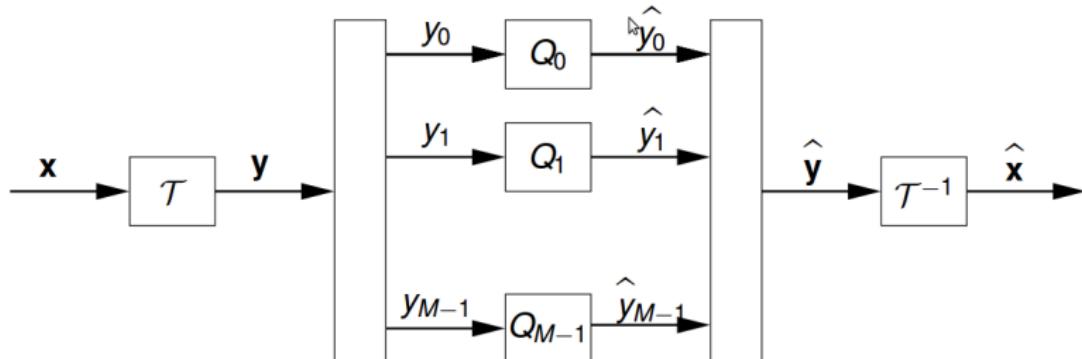
5 Cryptographie

Principe du codage par transformée

Introduction

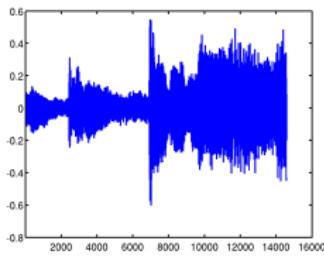
Transformation linéaire

- Changement de base \Rightarrow Représentation alternative du signal ;
- Concentration de l'énergie \Leftrightarrow représentation parcimonieuse ;
- Réduire la corrélation (\searrow l'entropie de la source).



Principe d'un codage par transformée

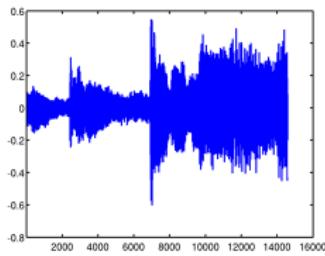
Signal, Image, ...



Principe d'un codage par transformée

Signal, Image, ...

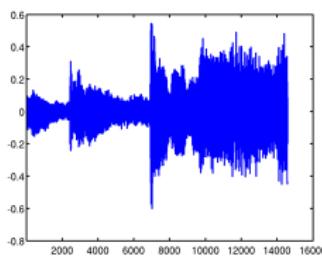
$$\left. \begin{array}{l} \text{Signal : } f \in L^2([0, 1]) \\ \text{Image : } f \in L^2([0, 1]^2) \end{array} \right\}$$



Principe d'un codage par transformée

Signal, Image, ...

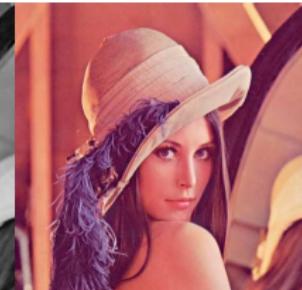
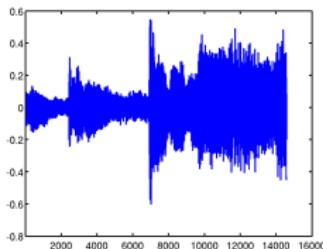
$$\left. \begin{array}{l} \text{Signal : } f \in L^2([0, 1]) \\ \text{Image : } f \in L^2([0, 1]^2) \end{array} \right\}$$



Principe d'un codage par transformée

Signal, Image, ...

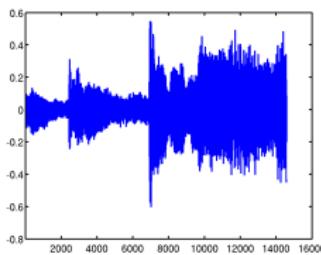
$$\left. \begin{array}{l} \text{Signal : } f \in L^2([0, 1]) \\ \text{Image : } f \in L^2([0, 1]^2) \end{array} \right\}$$



Principe d'un codage par transformée

Signal, Image, ...

$$\left. \begin{array}{l} \text{Signal : } f \in L^2([0, 1]) \\ \text{Image : } f \in L^2([0, 1]^2) \end{array} \right\} \Rightarrow \begin{array}{l} \text{Cas général : } f : [0, 1]^d \rightarrow \mathbb{R}^c \\ d : \text{dimensions, } c : \text{canaux.} \end{array}$$



Principe d'un codage par transformée

Décomposition orthogonale

Considérons un signal continu $f \in L^2([0, 1]^d)$ et une base orthogonale $\mathcal{B} = \{\psi_m\}_m$ de $L^2([0, 1]^d)$.

Décomposition du signal sur une base orthonormée

$$f = \sum_m f_m^{\mathcal{B}} \psi_m \text{ où } f_m^{\mathcal{B}} = \langle f \mid \psi_m \rangle = \int f(\mathbf{x}) \overline{\psi_m}(\mathbf{x}) d\mathbf{x}$$

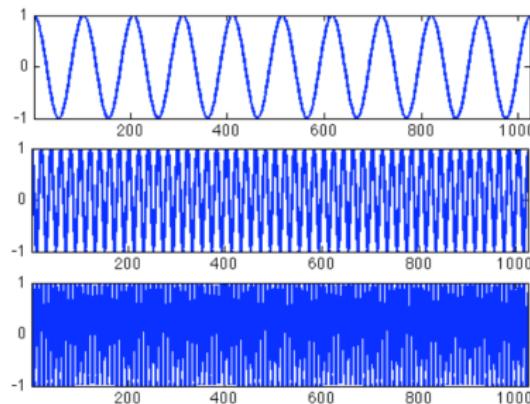
L'énergie se conserve :

$$\|f\|^2 = \int |f(\mathbf{x})|^2 d\mathbf{x} = \sum_m |f_m^{\mathcal{B}}|^2$$

Principe d'un codage par transformée

Décomposition orthogonale : exemples en 1D (son)

$$\text{Fourier (1D)} : \psi_m(x) = e^{j2\pi mx} = \cos(2\pi mx) + j \sin(2\pi mx);$$

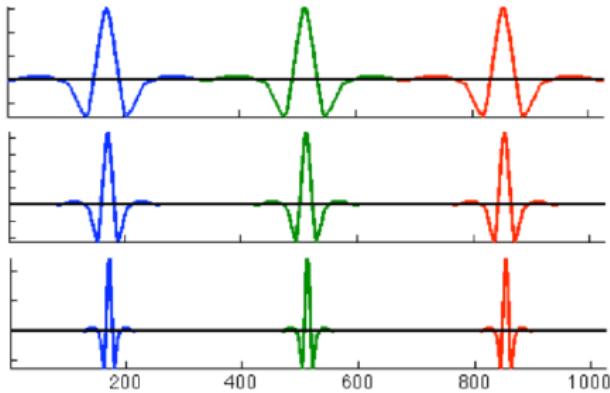
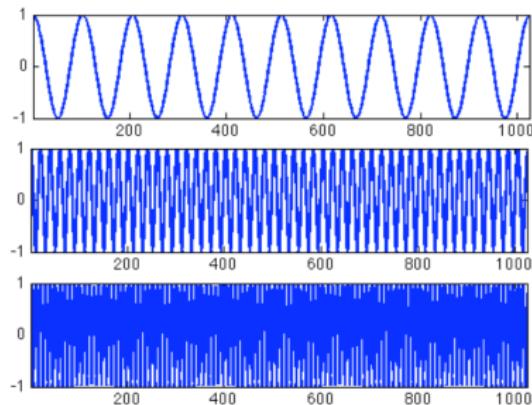


Principe d'un codage par transformée

Décomposition orthogonale : exemples en 1D (son)

Fourier (1D) : $\psi_m(x) = e^{j2\pi mx} = \cos(2\pi mx) + j \sin(2\pi mx)$;

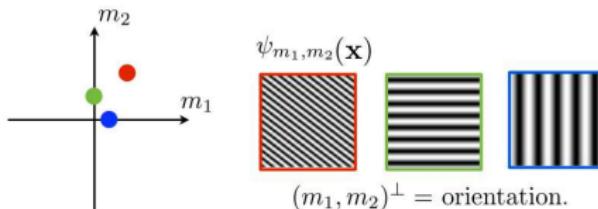
Ondelette (1D) : $\psi_{m=\{s,x_0\}}(x) = \frac{1}{\sqrt{s}}\psi\left(\frac{x-x_0}{s}\right)$.



Principe d'un codage par transformée

Décomposition orthogonale : exemples en 2D (image + vidéo)

$$\text{Fourier (2D)} : \psi_{m=\{m_1, m_2\}}(\mathbf{x}) = e^{j2\pi(m_1x+m_2y)} ;$$



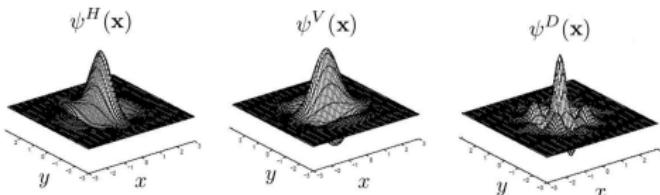
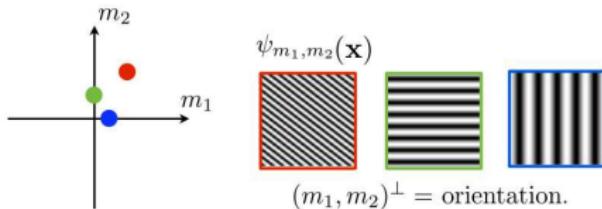
Principe d'un codage par transformée

Décomposition orthogonale : exemples en 2D (image + vidéo)

Fourier (2D) : $\psi_{m=\{m_1, m_2\}}(\mathbf{x}) = e^{j2\pi(m_1x+m_2y)}$;

Ondelette (2D) : $\{\psi_{m=\{s, x_0, y_0\}}^\omega(\mathbf{x})\}_{\omega=H,V,D}$

où $\psi_{m=\{s, x_0, y_0\}}^\omega(\mathbf{x}) = \frac{1}{s}\psi^\omega\left(\frac{x-x_0}{s}, \frac{y-y_0}{s}\right)$.



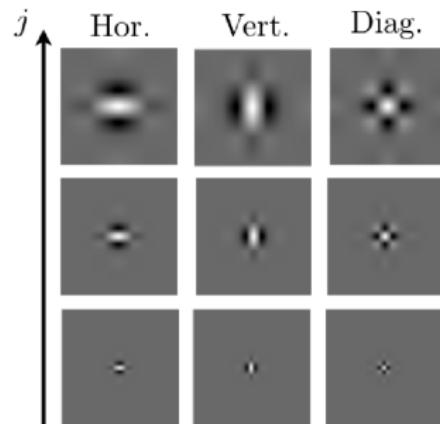
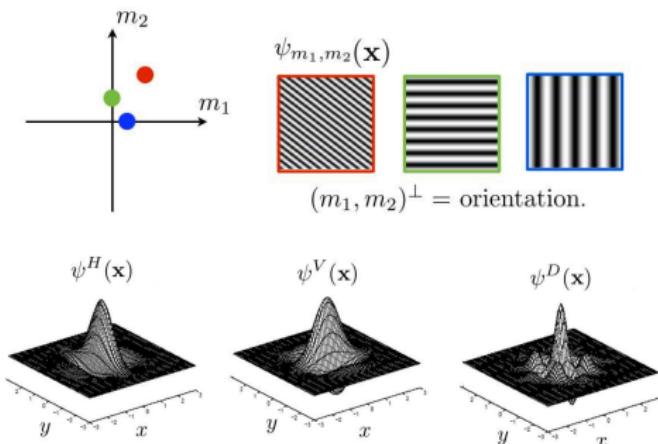
Principe d'un codage par transformée

Décomposition orthogonale : exemples en 2D (image + vidéo)

Fourier (2D) : $\psi_{m=\{m_1, m_2\}}(\mathbf{x}) = e^{j2\pi(m_1x+m_2y)}$;

Ondelette (2D) : $\{\psi_{m=\{s, x_0, y_0\}}^\omega(\mathbf{x})\}_{\omega=H,V,D}$

où $\psi_{m=\{s, x_0, y_0\}}^\omega(\mathbf{x}) = \frac{1}{s}\psi^\omega\left(\frac{x-x_0}{s}, \frac{y-y_0}{s}\right)$.



Principe d'un codage par transformée

Décomposition orthogonale discrète

Considérons un signal discret $f \in \mathbb{C}^N$ et une base orthogonale $\mathcal{B} = \{\psi_m\}_m \text{de } \mathbb{C}^N$.

- Signal 1D : $f \in \mathbb{C}^N$, $N = \#$ échantillons
- Signal 2D : $f \in \mathbb{C}^N = \mathbb{C}^{N_0 \times N_0} = \#$ pixels

Décomposition du signal sur une base orthonormée discrète

$$f = \sum_m f_m^{\mathcal{B}} \psi_m \text{ où } f_m^{\mathcal{B}} = \langle f \mid \psi_m \rangle = \sum_{n=0}^{N-1} f[n] \overline{\psi_m}[n]$$

L'énergie se conserve :

$$\|f\|^2 = \sum_{n=0}^{N-1} |f[n]|^2 dx = \sum_m |f_m^{\mathcal{B}}|^2$$

Principe d'un codage par transformée

Décomposition orthogonale discrète : Exemples 1D et 2D

- Transformée de Fourier discrète - TFD (1D) :

$$\psi_m[n] = e^{2j\pi m \frac{n}{N}}$$

- Transformée de Fourier discrète - TFD (2D) :

$$\psi_{m=\{u,v\}}[k, l] = e^{j \frac{2\pi}{N} (uk + vl)}$$

- Transformée en cosinus discret - TCD (1D) :

$$\psi_m[n] = \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) m \right]$$

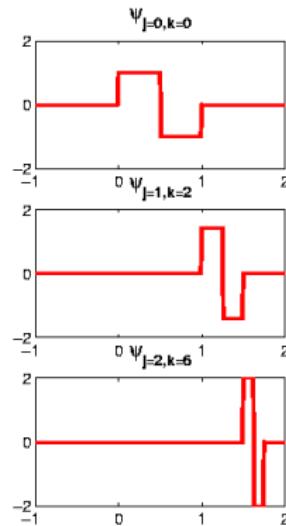
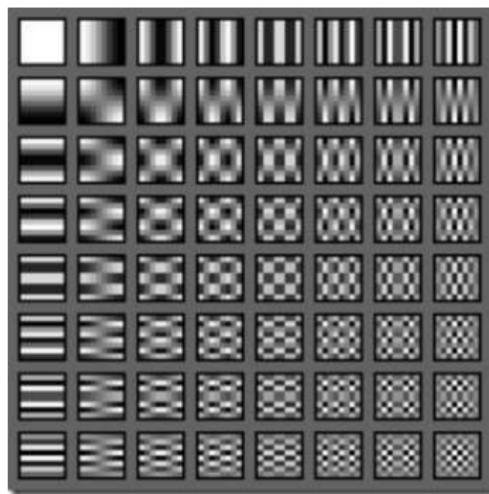
- Transformée en ondelette dyadique - TOD (1D) :

$$\psi_{m=\{j,n_0\}}[n] = 2^{-j/2} \psi(2^{-s}n - n_0)$$

Il existe des algorithmes rapides : FFT ($O(N \log N)$), FWT ($O(N)$)).

Principe d'un codage par transformée

Décomposition orthogonale discrète : Exemples 1D et 2D



Principe d'un codage par transformée

Compression par transformée

- Quantification : $f_m^{\mathcal{B}} \longrightarrow \tilde{f}_m^{\mathcal{B}} = Q(f_m^{\mathcal{B}})$
- Reconstruction : $\tilde{f} = \sum_m \tilde{f}_m^{\mathcal{B}} \psi_m \neq \sum_m f_m^{\mathcal{B}} \psi_m = f$



Principe d'un codage par transformée

Compression par transformée

- Quantification : $f_m^{\mathcal{B}} \longrightarrow \tilde{f}_m^{\mathcal{B}} = Q(f_m^{\mathcal{B}})$
- Reconstruction : $\tilde{f} = \sum_m \tilde{f}_m^{\mathcal{B}} \psi_m \neq \sum_m f_m^{\mathcal{B}} \psi_m = f$



Principe d'un codage par transformée

Compression par transformée

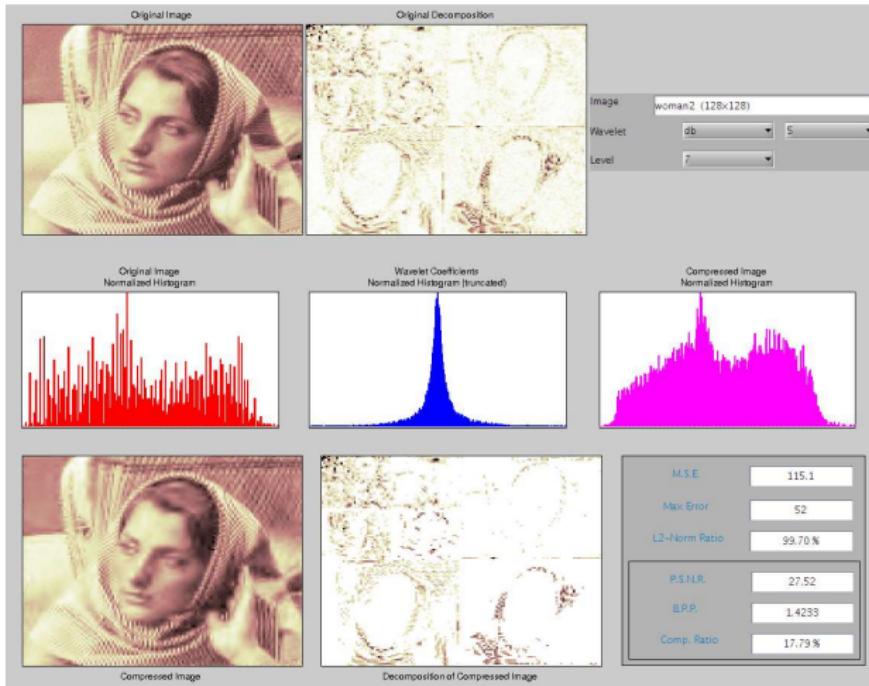
- Quantification : $f_m^{\mathcal{B}} \longrightarrow \tilde{f}_m^{\mathcal{B}} = Q(f_m^{\mathcal{B}})$
- Reconstruction : $\tilde{f} = \sum_m \tilde{f}_m^{\mathcal{B}} \psi_m \neq \sum_m f_m^{\mathcal{B}} \psi_m = f$



Comment choisir Q et ψ_m ? \Leftrightarrow minimiser $\{S(\tilde{f}_m^{\mathcal{B}})\}_m$

Principe d'un codage par transformée

Compression par transformée : Exemple



Principe d'un codage par transformée

Compression par transformée : Optimiser la distorsion

Comment mesurer la détérioration des signaux ? \Rightarrow erreur quadratique (mais n'est pas l'idéal)

Taux de distorsion

$$D = \mathbb{E} \left[\|f - \tilde{f}\|^2 \right] = \sum_m \mathbb{E} \left[|f_m^{\mathcal{B}} - \tilde{f}_m^{\mathcal{B}}|^2 \right]$$

Soit R_m le nombre de bits alloué au codage de $f_m^{\mathcal{B}}$.

$R = \sum_m R_m$ est le nombre total de bits pour coder le signal.

Il faut trouver une base et un quantificateur permettant à R fixé de minimiser D .

2 Théorie de l'information et Codage

3 Compression par transformée

- Principe d'un codage par transformée
- Choix de la base
- Quantification
- Exemples - Formats standard

4 Détection et correction d'erreurs

5 Cryptographie

Choix de la base

fonction des signaux à compresser : Pour un taux de compression élevé, le codage par transformée doit produire le + de coefficients quantifiés NULS possibles, avec des positions codées efficacement.

- Processus Gaussiens : base de Karhunen-Loeve est optimale (signaux audio \sim processus gaussiens). Mais le temps de calcul est trop coûteux. On utilise une approximation par une base plus structurée (cosinus locaux) ;
- Processus non-gaussiens : Pas de solutions théoriques, le problème est difficile en général.

Choix de la base

Processus Gaussiens : Base de Karhunen-Loeve

Base de Karhunen-Loeve (KL)

Si le signal f est un processus gaussien. La base de KL au optimale par rapport au taux de distorsion.

On considère la matrice de covariance du signal :

$R[m, n] = \text{cov}(f[n], f[m]) = \mathbb{E} [(f[n] - \mathbb{E}[f[n]])(f[m] - \mathbb{E}[f[m]])]$
de taille $N \times N$. Elle est symétrique et définie positive.

La base KL est
l'ensemble des vecteurs
propres ψ_m de R rangés
par valeurs propres
décroissantes σ_m^2 .

$$f = \sum_{m=0}^{N-1} f_m^B \psi_m \rightarrow \tilde{f} = \sum_{m=0}^{M-1} f_m^B \psi_m \text{ où } M < N$$

$$D = \sum_{m=M}^{N-1} \mathbb{E} [|f_m^B|^2] = \sum_{m=M}^{N-1} \sigma_m^2$$

Choix de la base

Bases d'atomes temps-fréquence

Le choix de la base s'oriente vers une transformée temps-fréquence linéaire, qui corrèle le signal avec une famille de fonctions bien concentrées en temps et en fréquence

⇒ **atomes temps-fréquence**

Théorème de Parseval

$$f_m^{\mathcal{B}} = \langle f | \psi_m \rangle = \int f(x) \overline{\psi_m}(x) dx = \int \widehat{f}(\nu) \overline{\widehat{\psi}_m}(\nu) d\nu$$

où

$$f(x) \xrightarrow{T\!F} \widehat{f}(\nu), \quad \psi_m(x) \xrightarrow{T\!F} \widehat{\psi}_m(\nu)$$

Choix de la base

Bases d'atomes temps-fréquence : Boite d'Heisenberg

- Si $\psi_m(x) \approx 0$ pour x en dehors du voisinage de x_0
- Si $\widehat{\psi}_m(\nu) \approx 0$ pour ν en dehors du voisinage de ν_0

alors

- $\langle f | \psi_m \rangle$ ne dépend que des valeurs de f au voisinage de x_0 et des valeurs de \widehat{f} au voisinage de ν_0 .

Boites d'Heisenberg

La "tranche" d'information contenu dans $\langle f | \psi_m \rangle$ est représentée dans le plan temps-fréquence dans une région, appelé boite d'Heisenberg, dont la position et la taille dépend de l'étalement de $\psi_m(x)$ en temps et de $\widehat{\psi}_m(\nu)$ en fréquence.

Choix de la base

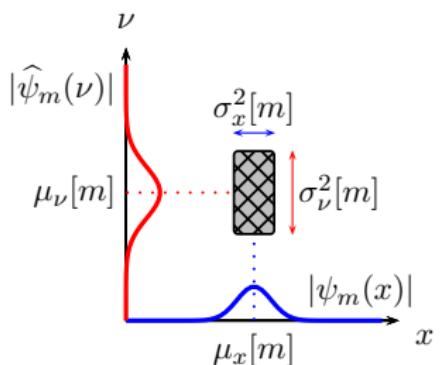
Bases d'atomes temps-fréquence : Boîte d'Heisenberg

La base étant normée : $\|\psi_m(x)\|^2 = 1$.

Théorème de Plancherel $\Rightarrow \|\psi_m(x)\|^2 = \|\widehat{\psi}_m(\nu)\|^2 = 1$

Position et Étalement temps-fréquence :

$$\begin{cases} \mu_x[m] = \int_{-\infty}^{+\infty} x |\psi_m(x)|^2 dx \\ \mu_\nu[m] = \int_{-\infty}^{+\infty} \nu |\widehat{\psi}_m(\nu)|^2 d\nu \\ \sigma_x^2[m] = \int_{-\infty}^{+\infty} (x - \mu_x[m])^2 |\psi_m(x)|^2 dx \\ \sigma_\nu^2[m] = \int_{-\infty}^{+\infty} (\nu - \mu_\nu[m])^2 |\widehat{\psi}_m(\nu)|^2 d\nu \end{cases}$$



Choix de la base

Bases d'atomes temps-fréquence : Théorème de Gabor-Heisenberg

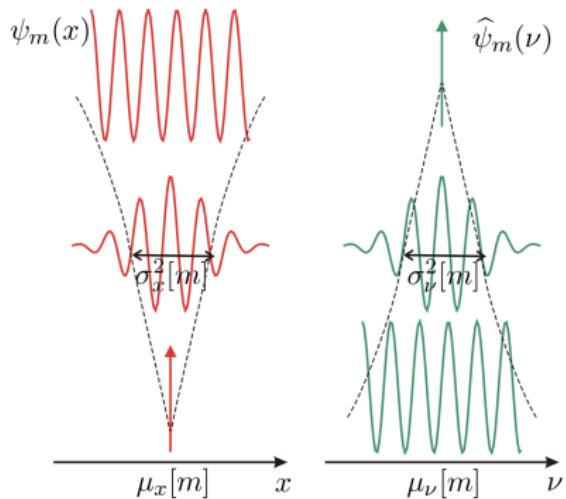
Théorème de Gabor-Heisenberg

$$\sigma_x^2 \times \sigma_\nu^2 \geq \frac{1}{4}$$

Il n'existe pas d'atomes temps-fréquence entièrement concentré en un temps x_0 et à une fréquence ν_0 .

⇒ Compromis à trouver sur la résolution temps-fréquence.

Remarque : L'inégalité devient une égalité (meilleur compromis) pour la base de Gabor.

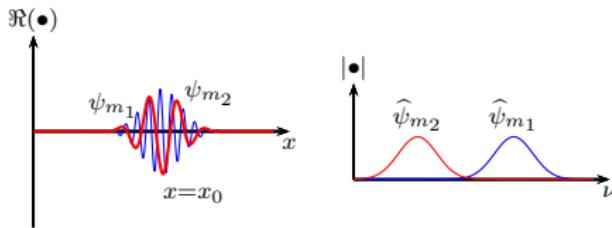
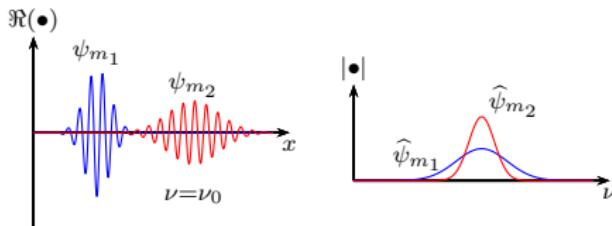


Choix de la base

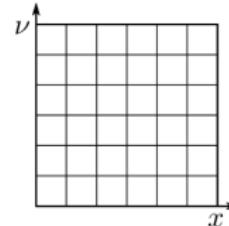
Transformée de Fourier à fenêtre

$$\psi_m(x) = e^{j2\pi\nu_0x} \times g(x - x_0) \text{ ici } m = \{x_0, \nu_0\}$$

$$\widehat{\psi}_m(\nu) = e^{-j2\pi(\nu-\nu_0)x_0} \times \widehat{g}(\nu - \nu_0)$$



Si g est une gaussienne
on est dans la meilleure
résolution $\sigma_x^2 \times \sigma_\nu^2 = \frac{1}{4}$.
(Transformée de Gabor)



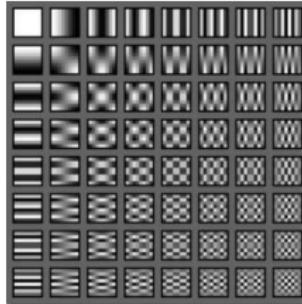
Choix de la base

Transformée en cosinus locaux

En pratique, l'utilisation d'une transformée à valeur réelle est utilisée : transformée en cosinus locaux. La version discrète s'appelle la transformée en cosinus discret ([en] : DCT).

$$\psi_m[n] = \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \times g[n - n_0] \text{ ici } m = \{n_0, k\}$$

Dans le cas 2D avec un voisinage de 8 pixels par 8 pixels (JPEG - MPEG) :



Choix de la base

Transformée en ondelette

Idée de base : ψ_m contient toujours le même nombre d'oscillations.

- ψ_m + court ($\searrow \sigma_x^2[m]$) \Rightarrow oscille + vite ($\nearrow \sigma_\nu^2[m]$ et $\nearrow \nu_0$);
- ψ_m + long ($\nearrow \sigma_x^2[m]$) \Rightarrow oscille - vite ($\searrow \sigma_\nu^2[m]$ et $\searrow \nu_0$).

Base d'ondelette en 1D

Soit $\Psi \in L^2([0, 1])$ une ondelette mère et de moyenne nulle ($\int \Psi(x) dx = 0 = \widehat{\Psi}(0)$). On définit une famille d'ondelettes à partir de Ψ par dilatation et translation :

$$\psi_m(x) = \frac{1}{\sqrt{s}} \Psi \left(\frac{x - x_0}{s} \right) \text{ ici } m = \{s, x_0\}$$

Choix de la base

Transformée en ondelette

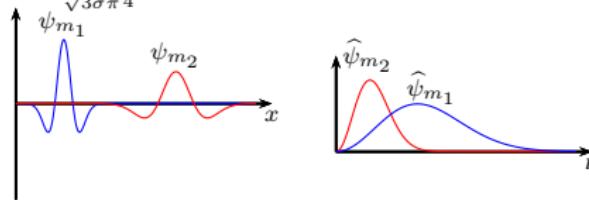
$$\psi_m(x) = \frac{1}{\sqrt{s}} \Psi \left(\frac{x - x_0}{s} \right)$$

Quand le facteur d'échelle $s \nearrow \Rightarrow \sigma_x^2[m] \nearrow$ et $\mu_\nu[m] \searrow$

$$\hat{\psi}_m(\nu) = \sqrt{s} \hat{\Psi}(s\nu) \times e^{-j2\pi\nu x_0}$$

Quand le facteur d'échelle $s \nearrow \Rightarrow \sigma_\nu^2[m] \searrow$ et $\mu_\nu[m] \searrow$

$$\Psi(x) = \frac{2}{\sqrt{3}\sigma\pi^{\frac{1}{4}}} \left(1 - \frac{x^2}{\sigma^2}\right) e^{\frac{-x^2}{2\sigma^2}}$$

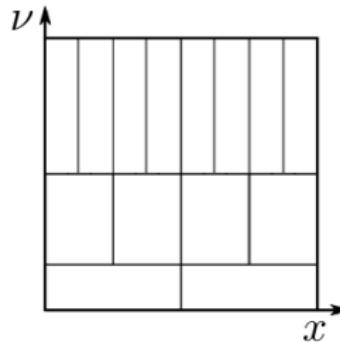


Choix de la base

Transformée en ondelette discrète

En pratique on utilise des transformées en ondelette dyadique, c'est-à-dire avec des facteurs d'échelle en 2^j

$$\psi_m[n] = 2^{-j/2} \Psi(2^{-j}n - n_0) \text{ ici } m = \{j, n_0\}$$

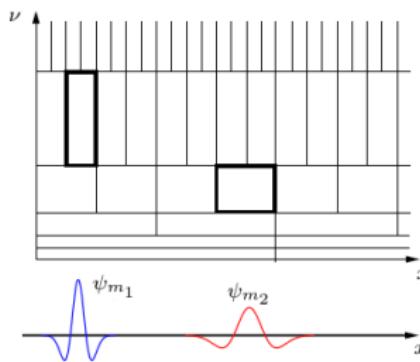


Choix de la base

Transformée en ondelette discrète

En pratique on utilise des transformées en ondelette dyadique, c'est-à-dire avec des facteurs d'échelle en 2^j

$$\psi_m[n] = 2^{-j/2} \Psi(2^{-j}n - n_0) \text{ ici } m = \{j, n_0\}$$

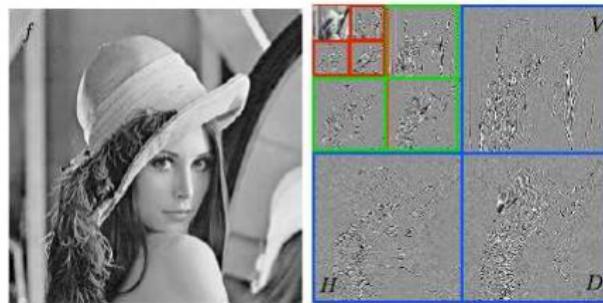
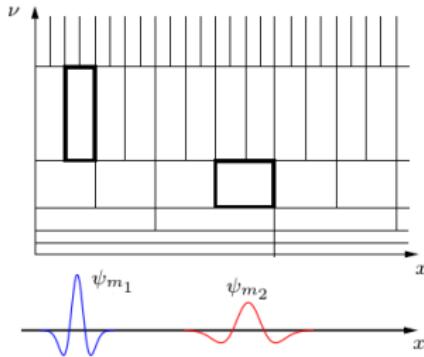


Choix de la base

Transformée en ondelette discrète

En pratique on utilise des transformées en ondelette dyadique, c'est-à-dire avec des facteurs d'échelle en 2^j

$$\psi_m[n] = 2^{-j/2}\Psi(2^{-j}n - n_0) \text{ ici } m = \{j, n_0\}$$



Choix de la base

Transformée en ondelette discrète : *-lette (pronunciation "starlette")

- La contrainte orthogonalité laisse peu de choix dans la base d'ondelettes (Daubechies) \Rightarrow transformation biorthogonale ;
- En image, l'info directionnelle est importante : d'autres bases existent comme les bandlettes, les curvelettes par exemple ;
- Ondelettes pour des variétés non-euclidiennes (sphères par ex.).

Choix de la base

Sélection des coefficients

original



1% of Fourier coeffs



10% of Fourier coeffs



Choix de la base

Sélection des coefficients

original



1% of wavelet coeffs



10% of wavelet coeffs



Choix de la base

Sélection des coefficients

Comment "choisir" les bons coefficients ? \Rightarrow **Quantification**

original



1% of curvelet coeffs



10% of curvelet coeffs



2 Théorie de l'information et Codage

3 Compression par transformée

- Principe d'un codage par transformée
- Choix de la base
- Quantification
- Exemples - Formats standard

4 Détection et correction d'erreurs

5 Cryptographie

Quantification

Quantification scalaire

Soit une source d'information à valeurs continues $X \in \Omega \subset \mathbb{R}$.
Comment coder cette information ? \Rightarrow **Quantification**.

Quantification scalaire

$X \in [a, b]$ un processus aléatoire discret stationnaire de densité de probabilité p_X .

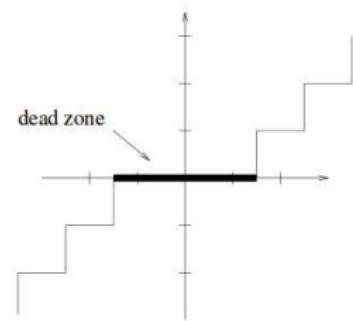
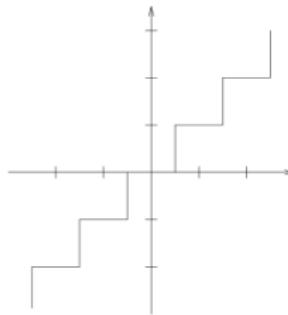
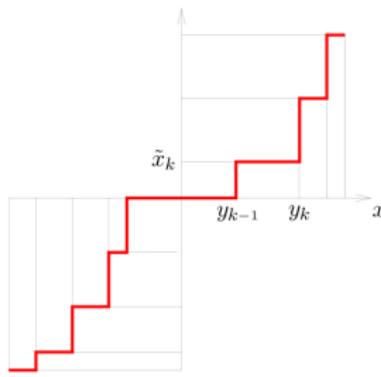
La quantification scalaire associe à la réalisation $x(u)$ une valeur quantifiée $\tilde{x}(u)$ (représentant). Pour cela on décompose l'intervalle $[a, b]$ en K intervalles $\{]y_{k-1}; y_k]\}_{1 \leq k \leq K}$ (partition).

Finalement, les valeurs quantifiées sont codées en binaire :

$$\tilde{x}_k \rightarrow m_k$$

Quantification

Quantification scalaire : Exemples



Quantification

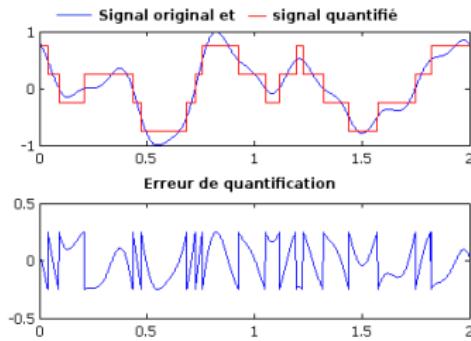
Quantification scalaire : Taux de distorsion

Taux de distorsion

On définit une mesure de distorsion (erreur quadratique moyenne) :

$$D = \mathbb{E} [|X(u) - \tilde{x}(u)|^2] = \sigma_Q^2$$

C'est également la puissance de l'erreur (ou bruit) de quantification.



Quantification

Quantification scalaire optimale

Quantification scalaire optimale

La quantification scalaire optimale consiste à trouver la partition et les représentants pour minimiser la distorsion :

$$\arg \min_{y_k, \tilde{x}_k} D$$

Il n'y a pas de solution simple.

Cependant il y a deux conditions nécessaires d'optimalités (optimisation séparée) : Algorithme de Lloyd-Max

Quantification

Quantification scalaire optimale : Algorithme de Lloyd-Max

Algorithm 2 Algorithme de Lloyd-Max

- 1: On dispose de la densité de probabilité de la source p_X
- 2: **while** Itérer jusqu'à convergence **do**
- 3: Optimiser D en calculant la partition :
 Règle du plus proche voisin : $y_k = \frac{\tilde{x}_k + \tilde{x}_{k+1}}{2}$
- 4: Optimiser D en calculant les représentants :
 Condition du centroïde : $\tilde{x}_k = \frac{\int_{y_{k-1}}^{y_k} u p_X(u) du}{\int_{y_{k-1}}^{y_k} p_X(u) du}$
- 5: **end while**

- Algorithme qui converge ;
- Minimum local sauf si $\log(p_X(u))$ est concave (source gaussienne, uniforme, ...)

Quantification

Quantification scalaire optimale : Quantificateur Haute Résolution (QHR)

Quantificateur Haute Résolution (*High-rate quantization*)

Un quantificateur est dit Haute Résolution si la densité de probabilité peut être assimilée à une constante dans chaque intervalle $]y_{k-1}, y_k]$ de taille Δ_k (pas de quantification).

$\Leftrightarrow \Delta_k$ est petit devant les variations de p_X .

Proposition

Pour un QHR, la distorsion est minimale lorsque $\tilde{x}_k = \frac{y_{k-1} + y_k}{2}$

$$D_{QHR} \simeq \frac{1}{12} \sum_{k=1}^K p_X(\tilde{x}_k) \Delta_k^3 = \frac{1}{12} \sum_{k=1}^K \mathbb{P}[X \in]y_{k-1}, y_k]] \Delta_k^2$$

Quantification

Quantification scalaire optimale : Quantificateur Haute Résolution (QHR)

Distorsion : Quantification Haute Résolution uniforme

Dans le cas où $\Delta_k = \Delta$ on obtient la formule classique : $D = \frac{\Delta^2}{12}$
Elle est indépendante de la densité de probabilité de la source.

Formule de Bennett

La formule de Bennett donne les performances d'une quantification scalaire (Lloyd-Max) où $K = 2^b$ et caractérisé par sa densité de probabilité p_X :

$$D \simeq \frac{1}{12} \left(\int_{-\infty}^{+\infty} [p_X(u)]^{\frac{1}{3}} du \right)^3 \times 2^{-2b}$$

Quantification

Quantification scalaire optimale : Quantificateur Haute Résolution (QHR)

- $X(u)$ suit une loi uniforme $\Rightarrow D = \sigma_x^2 \times 2^{-2b}$
- $X(u)$ suit une loi de Laplace $\Rightarrow D = \sigma_x^2 \times \frac{9}{2} \times 2^{-2b}$
- $X(u)$ suit une loi gaussienne $\Rightarrow D = \sigma_x^2 \times \pi \frac{\sqrt{3}}{2} \times 2^{-2b}$

Limite de Shannon

On définit $D_{min} = \sigma_x^2 \times 2^{-2b}$ la limite de Shannon (cas le plus favorable \Leftrightarrow loi uniforme)

Bruit de quantification : Rapport signal sur bruit

$$SNR_{\text{dB}} = 10 \log \frac{\sigma_X^2}{\sigma_Q^2} = 10 \log \frac{\sigma_X^2}{D}$$

Quantification

Quantification scalaire optimale : Quantificateur Haute Résolution (QHR)

- $X(u)$ suit une loi uniforme $\Rightarrow SNR_{\text{dB}} = 6,05 \times b$
- $X(u)$ suit une loi gaussienne $\Rightarrow SNR_{\text{dB}} = 6,05 \times b - 4,35$

Le cas gaussien (le plus défavorable) possède une différence de 4,35 dB avec l'optimal.

On parle de la "règle des 6dB par bit"

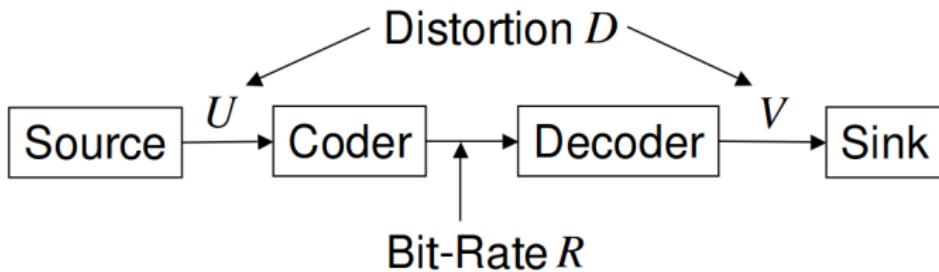
Exemple : CD audio, SNR_{dB} acceptable = 80 dB (conséquence perceptuelle). Donc $b = 14$ bits ($SNR_{\text{dB}} = 80,35$ dB) est suffisant, mais en pratique on utilise $b = 16$ bits ($SNR_{\text{dB}} = 92,45$ dB) car cela correspond à 2 octets.

Quantification

Quantification scalaire entropique

Quantification scalaire entropique (*Entropy-Constrained Quantization*)

La quantification scalaire entropique consiste à faire suivre la quantification par un codage entropique. On souhaite minimiser le nombre de bits nécessaire au codage des \tilde{x}_k (\Leftrightarrow minimiser $S(\tilde{X})$) pour une distorsion D donnée.



Quantification

Quantification scalaire entropique

Théorème de Gish et Pierce (1968)

Si on a un quantificateur haute résolution par rapport à p_X , alors

$$S(\tilde{X}) \geq S_d(X) - \frac{1}{2} \log_2 (12D)$$

où $S_d(X)$ est l'entropie différentielle de la source continue
 (quantité d'information moyenne)

$$S_d(X) = - \int_{-\infty}^{+\infty} p_X(u) \log_2 (p_X(u)) du$$

Remarque : il y a égalité ssi le quantificateur est uniforme. Dans ce cas $D = \frac{\Delta^2}{12}$, donc $S(\tilde{X}) = S_d(X) - \frac{1}{2} \log_2 (\Delta)$

Quantification

Quantification scalaire entropique

Meilleur QHR de la source $X(n)$ est un quantificateur uniforme suivi d'un codage entropique.

On peut également lire l'inégalité précédente sous la forme :

$$D \geq \frac{1}{12} 2^{-2[S(\tilde{X}) - S_d(X)]}$$

où la puissance de l'erreur de quantification est minorée par :

$$D_{min} = \frac{1}{12} 2^{-2[\bar{l}_X - S_d(X)]} \text{ où } \bar{l}_X \geq S(\tilde{X})$$

- $X(u)$ suit une loi gaussienne $\Rightarrow SNR_{dB} = 6,05 \times b - 1,53$

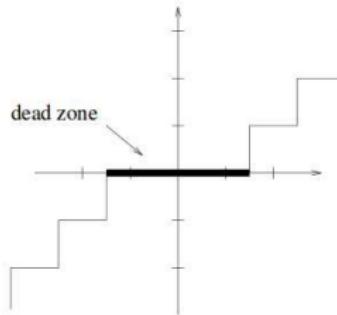
On obtient un gain de 2,81 dB par rapport au quantificateur de Lloyd-Max

Quantification

Quantification pour le codage par transformée

Généralement, en compression $\bar{l}_X \sim 0,5$ bits donc on est loin du quantificateur haute résolution : **quantificateur basse résolution¹**.

La solution du quantificateur à zone morte donne de bons résultats :



1. Stéphane Mallat, Frédéric Falzon, "Analysis of low bit rate image transform coding", *IEEE Transactions on Signal Processing*, 46 :1027-1042, 1998

2 Théorie de l'information et Codage

3 Compression par transformée

- Principe d'un codage par transformée
- Choix de la base
- Quantification
- Exemples - Formats standard

4 Détection et correction d'erreurs

5 Cryptographie

Exemples - Formats standard

Son

- MUSICAM (*Masking pattern Universal Subband Integrated Coding and Multiplexing*) ;
- MP3 (MPEG1-couche 3) ;
- MPEG2-AAC (*Advanced Audio Coding*).

Exemples - Formats standard

Image et Vidéo

- JPEG ;
- JPEG-2000 ;
- MPEG2.

2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

- Introduction - Notion de base
- Codes linéaires
- Codes cycliques
- Codes convolutionnels

5 Cryptographie

Introduction - Notion de base

Introduction

Théorème de Shannon de codage en présence de bruit \Rightarrow limites théoriques au codage de messages. Malheureusement, aucun conseil sur la façon de construire les codes en pratique
 \Rightarrow **théorie des codes correcteurs d'erreurs.**

Un mot de code z_i est transmis sur un canal bruité $\Rightarrow \hat{z}$ est finalement reçu.

Erreur de transmission correspond à la différence entre \hat{z} et z_i :
 $e = \hat{z} - z_i$.

L'idée derrière le codage est d'ajouter une structure algébrique à l'ensemble des mots de code de sorte que l'erreur de transmission puisse aisément être exprimée en termes des opérations définissant cette structure algébrique.

Introduction - Notion de base

Introduction

ligne	taux d'erreur
Disquette	10^{-9} : à 5 Mo/s, 3 bits erronés par minute
CD-ROM optique	10^{-5} : 7ko erronés sur un CD de 700 Mo
DAT audio	10^{-5} : à 48 kHz, deux erreurs par seconde
Mémoires à semi-conducteurs	$< 10^{-9}$
Liaison téléphonique	entre 10^{-4} et 10^{-7}
Télécommande infrarouge	10^{-12}
Communication par fibre optique	10^{-9}
Satellite	10^{-6} (Voyager), 10^{-11} (TDMA)
ADSL	10^{-3} à 10^{-9}
Réseau informatique	10^{-12}

Introduction - Notion de base

Introduction

Exemple : Code binaire (Règles de calculs dans GF (2) ou $\mathbb{Z}/2\mathbb{Z}$)

La différence « naturelle » sur les mots binaires est la différence bit par bit. Dans le corps GF (2), il y a deux loi internes.

-	0	1	\times	0	1
0	0	1	0	0	0
1	1	0	1	0	1

L'opération – correspond en fait à l'arithmétique « modulo 2 », c.-à-d. que nous considérons le corps de Galois GF (2) et que les mots de code sont des éléments de l'espace vectoriel $GF(2)^n$ (où n est la longueur des mots de code). Cette base s'étend à n'importe quel code \mathcal{D} -aire utilisant une arithmétique « modulo \mathcal{D} ».

Introduction - Notion de base

Introduction

Code par bloc

Un code par bloc \mathcal{D} -aire de longueur n est un sous-ensemble non vide de l'espace vectoriel des n -uplets $\text{GF}(\mathcal{D})^n$ (c.-à-d. des mots \mathcal{D} -aires de même longueur n , considérés comme des « vecteurs ligne »).

Exemple :

- $\{1101, 0110, 1110\}$: exemple de code par bloc binaire ;
- Un autre exemple de code par bloc, considérant les codes ternaires utilisant les symboles 0 1 et 2 est donné par l'ensemble $\{120, 201, 222, 010\}$;
- $\{011, 110, 10\}$ n'est pas un code par bloc, ces mots n'étant pas tous de même longueur.

Introduction - Notion de base

Distance de Hamming et poids d'un mot de code : Définitions

Distance de Hamming

La distance de Hamming $d(z, z')$ entre deux mots de même longueur z et z' (c.-à-d. deux n -uplets dans le cas le plus général) est le nombre de symboles (c.-à-d. de positions) pour lesquels z et z' diffèrent.

Poids d'un mot de code

Le poids d'un mot est le nombre de symboles non nuls qu'il contient.

Mot de code nul

Le mot de code nul est le mot de code constitué uniquement de zéros. Il sera noté $\mathbf{0}_n$.



Introduction - Notion de base

Distance de Hamming et poids d'un mot de code : Propriétés

- La distance de Hamming entre deux mots de code est le poids de leur différence : $d(z_i, z_j) = w(z_i - z_j)$ en désignant la distance de Hamming par $d(\bullet)$ et le poids par $w(\bullet)$;
- Le poids d'un mot de code est toujours positif ou nul ;
- Le poids d'un mot de code est 0 si et seulement si ce mot de code est le mot de code nul 0_n ;
- Le poids est symétrique : pour chaque mot de code z_i , $w(z_i) = w(-z_i)$ (où $-z_i$ est le mot dans lequel chaque symbole est l'opposé du symbole correspondant dans z_i) ;
- Pour tous mots de code z_i et z_j , on a :
$$w(z_i) + w(z_j) \geq w(z_i + z_j).$$

Introduction - Notion de base

Décodage à distance minimale et à vraisemblance maximale

Comment un décodeur de code peut-il décoder un mot reçu \hat{z} ? Une réponse intuitive est de supposer le plus petit nombre d'erreurs \Leftrightarrow prendre le mot de code z_i le plus proche (c.-à-d. $d(\hat{z}, z_i)$ minimale).

Décodage à Distance Minimale

On dit qu'un code \mathcal{C} utilise un décodage à distance minimale chaque fois que la décision de décodage \mathcal{D} consiste, pour un mot \hat{z} reçu, à choisir le (un des) mot(s) de code le(s) plus proche(s) :

$$\mathcal{D}(\hat{z}) = \arg \min_{z \in \mathcal{C}} d(z, \hat{z})$$

Dans le cas du canal symétrique binaire, le décodage à distance minimale est équivalente à un décodage à vraisemblance maximale.

Introduction - Notion de base

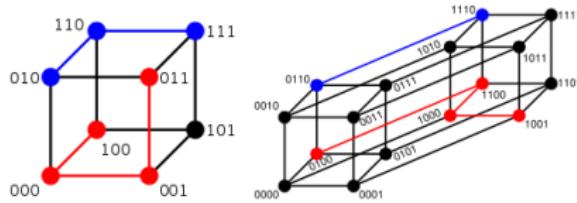
Détection et correction d'erreurs

Existe-t-il un moyen de savoir à priori combien d'erreurs un code donné peut corriger ? \Rightarrow Distance minimale

Distance Minimale d'un Code

La Distance Minimale $d_{\min}(\mathcal{C})$ d'un code $\mathcal{C} = \{z_1, \dots, z_i, \dots, z_M\}$ est la distance de Hamming minimale (non nulle) entre toute paire de ses mots de code :

$$d_{\min}(\mathcal{C}) = \min_{i \neq j} d(z_i, z_j)$$

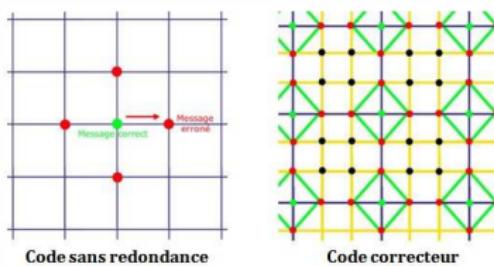


Introduction - Notion de base

Détection et correction d'erreurs : Théorème

Théorème : Correction d'Erreur et Capacité de Détection

Un code par bloc de longueur n utilisant le décodage à distance minimale peut, pour toute paire d'entiers t et s tels que $0 \leq t \leq n$ et $0 \leq s \leq n - t$, corriger tous les schémas à t erreurs ou moins et détecter tous les schémas à $t + 1, \dots, t + s$ erreurs si et seulement si sa distance minimale est strictement supérieure à $2t + s \Leftrightarrow d_{\min}(\mathcal{C}) > 2t + s$.



Introduction - Notion de base

Détection et correction d'erreurs : Propriétés

- **Capacité maximale de détection d'erreurs** : Un code par bloc \mathcal{C} utilisant le décodage à distance minimale peut être utilisé pour détecter tous les schémas d'erreur de $d_{\min}(\mathcal{C}) - 1$ erreurs, ou moins ;
- **Capacité maximale de correction d'erreurs** : Un code par bloc \mathcal{C} utilisant le décodage à distance minimale peut être utilisé pour corriger tous les schémas d'erreur de $\frac{d_{\min}(\mathcal{C})-1}{2}$ (division Euclidienne) ou moins d'erreurs, mais ne peut pas être utilisé pour corriger tous les schémas d'erreur de $1 + \frac{d_{\min}(\mathcal{C})-1}{2}$ erreurs :

$$\Leftrightarrow t_{\max} = \left\lfloor \frac{d_{\min}(\mathcal{C}) - 1}{2} \right\rfloor.$$

2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

- Introduction - Notion de base
- Codes linéaires
- Codes cycliques
- Codes convolutionnels

5 Cryptographie

Codes linéaires

Définitions

Codes linéaires = codes par bloc + structure algébrique \Rightarrow aider au décodage : la structure d'espace vectoriel.

Code linéaire

Un (n, m) code linéaire ($1 \leq m \leq n$) \mathcal{D} -aire est un sous-espace m -dimensionnel de l'espace vectoriel $\text{GF}(\mathcal{D})^n$ de n -uplets sur $\text{GF}(\mathcal{D})$.

- Le code $\{1101, 0110, 1110\}$ n'est pas un code linéaire car 0 n'en fait pas partie.
- Le code $\{0000, 1101, 0110, 1011\}$ est un code linéaire comme toute combinaison linéaire de mots de code est aussi un mot de code. C'est un code linéaire binaire (4, 2).

Codes linéaires

Quelques propriétés des codes linéaires

- Tout code linéaire contient le mot de code nul 0 ;
- Un code linéaire (n, m) \mathcal{D} -aire contient \mathcal{D}^n mots de code différents (le mot de code nul inclus) ;
- Le taux de transmission d'un code linéaire (n, m) est $R = m/n$.

Équivalence du Poids Minimal et de la Distance Minimale

Pour tout code linéaire \mathcal{C} : $d_{\min}(\mathcal{C}) = w_{\min}(\mathcal{C}) = \min_{\substack{z \neq 0 \\ z \in \mathcal{C}}} w(z)$

Ce résultat est important, car $w_{\min}(\mathcal{C})$ est beaucoup plus facile à calculer que $d_{\min}(\mathcal{C})$.

Codes linéaires

Codage avec des codes linéaires

Matrice Génératrice

Une matrice $m \times n$ G est dite matrice génératrice d'un code linéaire (n, m) \mathcal{C} si et seulement si ses m vecteurs lignes sont une base de l'espace vectoriel \mathcal{C} .

Le codage d'un message u (de taille m) est alors effectué par $z = u \cdot G$.

L'utilisation d'une matrice génératrice rend le codage très facile à implémenter car l'utilisation de quelques portes OU-exclusif (XOR) peuvent le réaliser.

Exemple : Soit le code linéaire $(4, 2)$: $\{0000, 1101, 0110, 1011\}$.

Proposez une matrice génératrice.

Codes linéaires

Forme systématique d'un code linéaire

Forme Systématique

Une matrice génératrice G d'un code linéaire (n, m) est dite sous forme systématique si elle est de la forme :

$$G = [I_m \ P] = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{1,1} & \cdots & p_{1,n-m} \\ 0 & 1 & \cdots & 0 & p_{2,1} & \cdots & p_{2,n-m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & p_{m,1} & \cdots & p_{m,n-m} \end{bmatrix}$$

où I_m est la matrice identité de taille m et P une matrice de taille $m \times (n - m)$, appelée matrice de parité.

Code Linéaire Systématique

Un code linéaire utilisant une matrice génératrice sous forme systématique est appelé code (linéaire) systématique.



Codes linéaires

Décodage : Matrice de vérification \Rightarrow Comment décoder ?

Matrice de Vérification

Une matrice $(n - m) \times n$ V est une matrice de vérification pour un code linéaire \mathcal{D} -aire \mathcal{C} si et seulement si :

$$\forall z \in \text{GF}(\mathcal{D})^n \quad z \cdot V^T = 0 \Leftrightarrow z \in \mathcal{C}$$

Le noyau de cette matrice est \mathcal{C} .

Théorème

Pour un code linéaire \mathcal{C} systématique dont la matrice génératrice est G , alors la matrice suivante est une matrice de vérification :

$$V = [-P^T \ I_{n-m}]$$

Codes linéaires

Syndromes

Définition

Le syndrome d'un mot \hat{z} relatif à une matrice de vérification V est le produit $\hat{z} \cdot V^T$.

- $\hat{z} \cdot V^T = (z + e) \cdot V^T = e \cdot V^T$;
- Le syndrome $s = \hat{z} \cdot V^T$ d'un mot reçu \hat{z} relatif à la matrice de vérification V d'un code \mathcal{C} dépend uniquement de l'erreur de transmission e et non du mot de code z ($\in \mathcal{C}$) transmis.

La correction peut alors être effectuée simplement en faisant correspondre les colonnes de V aux correcteurs et en additionnant ceux qui correspondent aux positions non nulles du syndrome (exemple).

Codes linéaires

Distance minimale et matrice de vérification

Matrice de Vérification et Distance Minimale

Si V est une matrice de vérification pour un code linéaire \mathcal{D} -aire \mathcal{C} (avec $1 \leq m < n$), alors la distance minimale $d_{\min}(\mathcal{C})$ de ce code est égale au plus petit nombre de colonnes linéairement dépendantes de V (rang de la matrice).

Pour un code linéaire binaire \mathcal{C} avec une matrice de vérification V , ce résultat implique que si V n'a pas de colonne nulle,

$$V = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

a une distance minimale de 3.

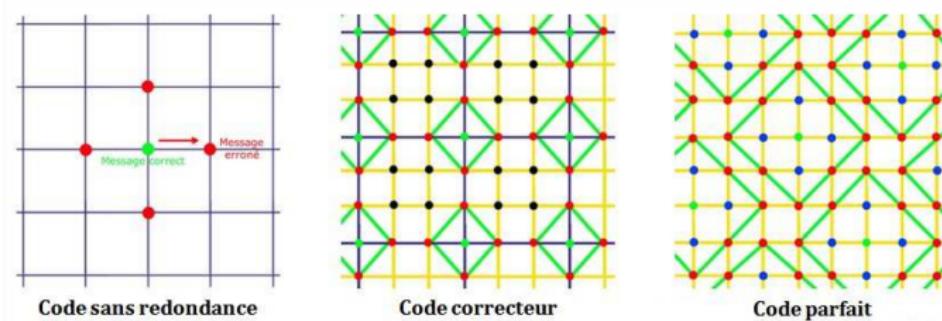
En effet, V n'a pas de colonne nulle, ni deux fois la même colonne, donc $d_{\min} \leq n - m + 1$

Codes linéaires

Codes parfait

Définition : Code parfait

Un code linéaire est dit parfait si et seulement s'il existe une valeur t tel que les boules fermées de centre les mots de code et de rayon t forment une partition de l'espace.



Codes linéaires

Codes de Hamming binaires

Définition : Code de Hamming

Un code de Hamming est un code linéaire binaire

$(2^r - 1, 2^r - r - 1)$ ($r \geq 2$), dont la matrice de vérification est :

$$V_r = [b_r(1)^T, \dots, b_r(n)^T] = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \end{bmatrix}$$

où $b_r(i)$ est la représentation binaire de i en n bits.

Tout code de Hamming binaire peut corriger tous les schémas à une erreur.

Codes linéaires

Codes de Hamming binaires

Un code de Hamming est parfait : pour une longueur de code donnée il n'existe pas d'autre code plus compact ayant la même capacité de correction. En ce sens son rendement est maximal.

Exemple : Code de Hamming (7, 4) :

$$V_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- Trouver une matrice génératrice ;
- Comment décoder ?

2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

- Introduction - Notion de base
- Codes linéaires
- Codes cycliques
- Codes convolutionnels

5 Cryptographie

Codes cycliques

Introduction

Les codes cycliques = catégorie des codes correcteurs d'erreurs dont l'utilisation est la plus répandue. Ils sont une sous-catégorie importante des codes linéaires car ils possèdent beaucoup de propriétés algébriques qui simplifient les implémentations du codage et du décodage.

Définition : Codes Cycliques

Un code cyclique est un code linéaire tel que, pour tout mot de code z_i à n symboles $z_i = z_{i,1}, \dots, z_{i,n}$, le mot $z_{i,2}, \dots, z_{i,n}, z_{i,1}$ résultant d'une permutation cyclique (vers la gauche) (aussi appelée « décalage ») des symboles de z_i soit aussi un mot de code.

Codes cycliques

Codes cycliques et polynômes

On représente un mot de code z_i à n symboles $z_i = z_{i,1} \cdots z_{i,n}$ par un polynôme

$$z_i(X) = z_{i,1} \cdot X^{n-1} + z_{i,2} \cdot X^{n-2} + \cdots + z_{i,n-1} \cdot X + z_{i,n}.$$

L'intérêt que que $X \cdot z_i(X) \bmod (X^n - 1)$ correspond à la permutation cyclique vers la gauche de z_i .

Propriété

Si $z(X)$ est le polynôme correspondant à un mot de code z d'un code cyclique de taille n , alors, pour tout polynôme $p(X)$, $p(X) \cdot z(X) \bmod (X^n - 1)$ est aussi un polynôme correspondant à un mot de code de ce code (décalages vers la gauche et combinaisons linéaires).

Codes cycliques

Codes cycliques et polynômes

Théorème

Pour tout code cyclique (n, m) \mathcal{C} , il existe un polynôme $g_c(X)$ de degré $n - m$ tel que

$$\mathcal{C} = \{g_c(X) \cdot p : p \in \text{GF}(\mathcal{D})[X], \deg(p) < m\}$$

c'est-à-dire que tout polynôme d'un mot de code est un multiple de $g_c(X)$, et réciproquement. En d'autres termes, le code \mathcal{C} est généré par $g_c(X)$.

$g_c(X)$ est en fait appelé le **générateur** de \mathcal{C} .

Le codage par un code cyclique (n, k) consiste à réaliser l'opération :

$$z(X) = [g_c(X) \cdot u(X)] \bmod (X^n - 1).$$

Codes cycliques

Décodage

Processus de décodage (similaire aux codes linéaires) :

- ① calculer un syndrome à partir du mot reçu ;
= reste de la division de $\hat{z}(X)$ par $g_c(X)$
- ② en déduire le correcteur ;
=
 - ① pour une erreur seule X^i de degré i inférieur à $n - m$ (le degré de $g_c(X)$), le syndrome est simplement X^i ;
 - ② pour l'erreur seule $X^{(n-m)}$, le syndrome est $X^{(n-m)} - g_c(X)$.
- ③ appliquer le correcteur au mot de code reçu.

Codes cycliques

Codes BCH et Reed-Solomon

Le taux de correction d'un code cyclique est difficile à calculer.

Théorème distance minimale

Soit \mathcal{C} un code cyclique (n, k) de polynôme générateur $g_c(X) \in \text{GF}(\mathcal{D})[X]$ avec n premier avec \mathcal{D} et soit β une racine primitive $n^{\text{ième}}$ de l'unité. S'il existe deux entiers l et s tels que $g_c(\beta^l) = g_c(\beta^{l+1}) = \dots = g_c(\beta^{l+s+1}) = 0$ alors :

$$d_{\min}(\mathcal{C}) \geq s + 1$$

Le code \mathcal{C} est donc au moins s -déTECTeur et $\lfloor s/2 \rfloor$ -correctEUR.

En application du théorème, Bose, Chaudhuri et Hocquenghem ont proposé une construction des polynômes générateurs BCH. Les codes BCH optimaux sont appelés codes de Reed-Solomon.

Codes cycliques

Exemples : QRCode



© 2012 Walter Tuck

OR Code – Structure

Model 2005 – ISO/IEC 18004:2006

- | | |
|---|---|
| <ul style="list-style-type: none"> <input checked="" type="checkbox"/> $0_2 \leftrightarrow \text{"Light"}$ <input checked="" type="checkbox"/> $1_2 \leftrightarrow \text{"Dark"}$ | Bits \leftrightarrow Modules (nominal color/reflectance) |
| | Version ($1 \leq V \leq 40$): $\text{Size} = N \times N$; $N = 4 + V + 17 \Rightarrow N^2$ modules |
| |  Quiet Zone: $4N$; ≥ 4 -mod light margin surrounding cell |
| <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Separators: $6N$; $1 \times 8 \Rightarrow 45$ mods <input checked="" type="checkbox"/> ID/Finder/Positioning/Orientation: $3N$; $7 \times 7 \Rightarrow 147$ mods <input checked="" type="checkbox"/> Alignment: $K N$; $K \in \{0, 1, 6, 13, 22, 33, 46\}$; $5 \times 5 = 25 \cdot K$ mods <input checked="" type="checkbox"/> Timing: $2N$; $1 \times (N-16) \Rightarrow 2 \cdot (N-16)$ mods | |
| <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Format Info: $5N$; $1 \times 8, 1 \times 6, 3 \Rightarrow 31$ mods <input checked="" type="checkbox"/> Version Info: $2N$; $V \geq 7$ only; $3 \times 6 \Rightarrow 0 \leq 36$ mods | |
|  | Content: Cell \ Artifacts = Data(&Pads) + EDC(&Remainder)
E.g.: $V = 3 \Rightarrow 29^2 - (45 + 147 + 25 + 26 + 31 + 0) = 567$ content mods |

Codes cycliques

Exemples : QRCode

4

QR Code — Levels & Masks

Model 2005 — ISO/IEC 18004:2006

Levels: $0_2=1 \triangleq \mathbf{L}[\mathbf{low}]$; $0_2=0 \triangleq \mathbf{M}[\mathbf{medium}]$; $1_2=3 \triangleq \mathbf{Q}[\mathbf{quality}]$; $1_2=2 \triangleq \mathbf{H}[\mathbf{high}]$

Detailed version/level definitions at ISO/IEC spec, §6.4.10, Table 7

E.g.: Version=3 \Rightarrow 567 content mods/bits \Rightarrow 70 codewords (+ 7 remainder bits)

EDC Level: Determines count/distribution of codewords in D-space & E-space

Mask: Raw (not metadata) D/E XOR-masked ("cooked") with **mask template**

Goal: Robustness for scanning/reading devices; balance light/dark modules; minimize occurrence of *pro forma* patterns in D/E spaces (not "crypto")

Robustness evaluation rubric defined at ISO/IEC spec, §6.8.1

D/E mask templates ($/$ \triangleq integer division; $\%$ \triangleq integer remainder):

- Mask $000_2=0 \triangleq (row+col)\%2 = 0$
- Mask $001_2=1 \triangleq row\%2 = 0$
- Mask $010_2=2 \triangleq col\%3 = 0$
- Mask $011_2=3 \triangleq (row+col)\%3 = 0$
- Mask $100_2=4 \triangleq (row/2 + col/3)\%2 = 0$
- Mask $101_2=5 \triangleq (row\%2 + (row+col)\%3)\%2 = 0$
- Mask $110_2=6 \triangleq ((row\%2 + (row+col)\%3)\%2 + (row\%2 + (row+col)\%3)\%2)\%2 = 0$
- Mask $111_2=7 \triangleq ((row\%2 + (row+col)\%3)\%2 + (row\%2 + (row+col)\%3)\%2)\%2 = 0$

Raw format info (level, mask) also masked ("cooked") — not by the D/E mask (below), but by this special 15-bit **Format Mask**:

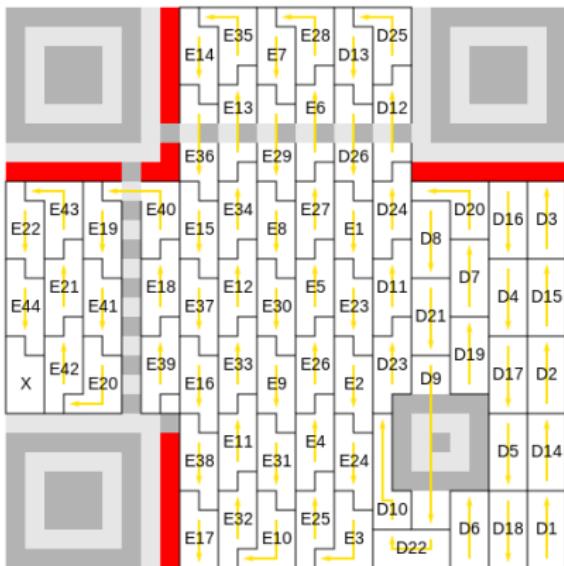
1	0	1	0	1	0	0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

Mask templates (Version 3, 29x29):

© 2012 Walter Tuyvel

Codes cycliques

Exemples : QRCode



Fixed Patterns Format Info

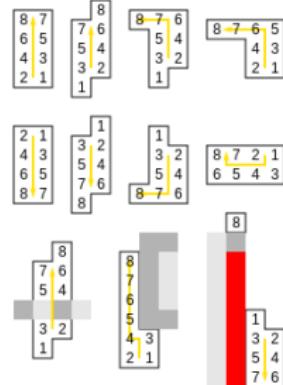
D: Data, E: Error Correction, X: Unused
Error Correction Level H is shown

Block 1 Codewords: D1–D13, E1–E22

Block 2 Codewords: D14–D26, E23

Message Data: D1–D13, D14–D26

Bit order (1 is the most significant bit):



2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

- Introduction - Notion de base
- Codes linéaires
- Codes cycliques
- Codes convolutionnels

5 Cryptographie

Codes convolutionnels

Introduction

Les **codes convolutifs** = classe extrêmement souple et efficace de codes correcteurs d'erreur (les + utilisés dans les communications fixes et mobiles).

Ils ont les mêmes caractéristiques que les codes en bloc sauf qu'ils s'appliquent à des séquences infinies de symboles d'information et génèrent des séquences infinies de symboles de code.

Toutefois, il y a une différence significative dans la conception de ces techniques de codage/décodage.

Codes convolutionnels

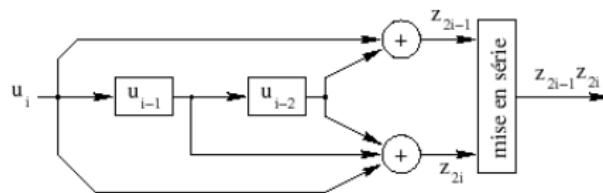
Introduction

Les **codes convolutifs** = classe extrêmement souple et efficace de codes correcteurs d'erreur (les + utilisés dans les communications fixes et mobiles).

Ils ont les mêmes caractéristiques que les codes en bloc sauf qu'ils s'appliquent à des séquences infinies de symboles d'information et génèrent des séquences infinies de symboles de code.

Toutefois, il y a une différence significative dans la conception de ces techniques de codage/décodage.

Exemple :

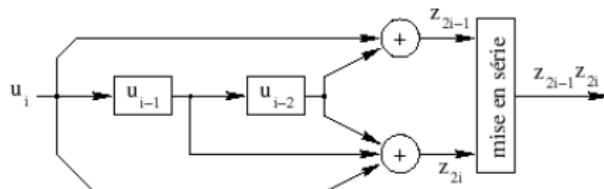


Codes convolutionnels

Codage

Le codeur qui engendre un code convolutif comporte un effet de mémoire : le mot code ne dépend pas que du bloc de k symboles entrant, mais aussi des m mots de code qui l'ont précédé, stockés dans un registre.

Supposons vouloir envoyer le message $u = 101$:



i	u_i	Etat ($u_{i-1}u_{i-2}$)	$z_{2i-1}z_{2i}$
1	1	00	11
2	0	10	01
3	1	01	00
4	(0)	10	01
5	(0)	01	11

Codes convolutionnels

Définition générale

Code convolutionnel

Un **code convolutionnel** (n, k, r) est un code linéaire non borné formé de $r + 1$ matrices binaires $k \times n$ et le codage d'une séquence binaire infinie u est égal à $z = u \cdot G$, où G est une matrice binaire infinie définie par :

$$G = \begin{bmatrix} G_0 & G_1 & \cdots & G_r \\ & G_0 & G_1 & \cdots & G_r \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ & G_0 & G_1 & \cdots & G_r \\ & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

On peut écrire un codeur convolutionnel sous plusieurs forme :

$$z_{j+r} = \sum_{i=0}^r u_{j+i} G_{r-i}.$$

Codes convolutionnels

Représentation sous forme de treillis - Diagramme d'état

Pour faciliter l'algorithme de décodage, la représentation la plus courante du codage est la représentation en treillis.

Diagramme d'État

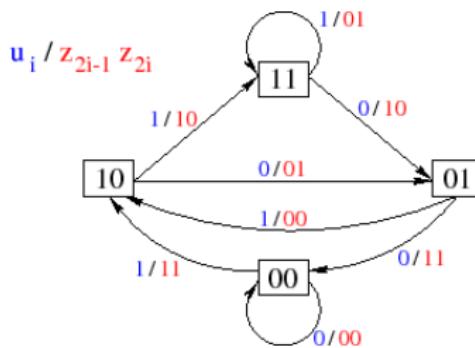
Le diagramme d'état d'un codeur est un graphe dont les nœuds sont tous les états internes possibles du codeur. Un arc entre un nœud S_i et un nœud S_j dans ce graphe représente le fait qu'il existe une entrée qui, lorsqu'elle est reçue dans l'état S_i met le codeur dans l'état S_j .

Ces arcs sont habituellement étiquetés avec le(s) symbole(s) d'entrée et les symboles de sortie correspondants.

Codes convolutionnels

Représentation sous forme de treillis - Diagramme d'état

Exemple : Par exemple, pour le codeur présenté précédemment :



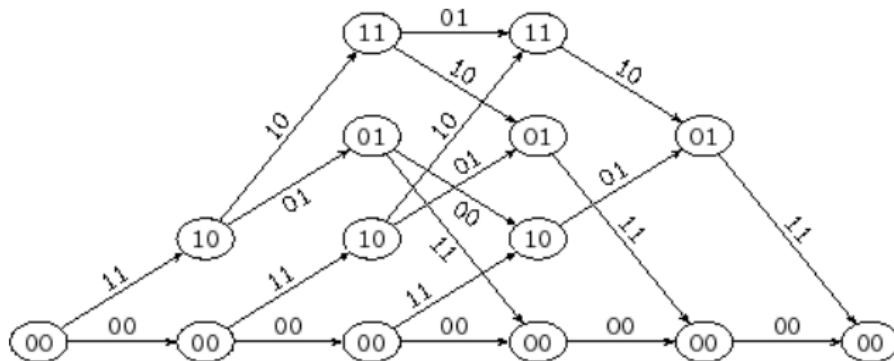
Codes convolutionnels

Représentation sous forme de treillis

Treillis de codage

Le **treillis de codage** de taille m est obtenu en étendant le diagramme d'état dans le temps du code convolutionnel (n, k, r) .

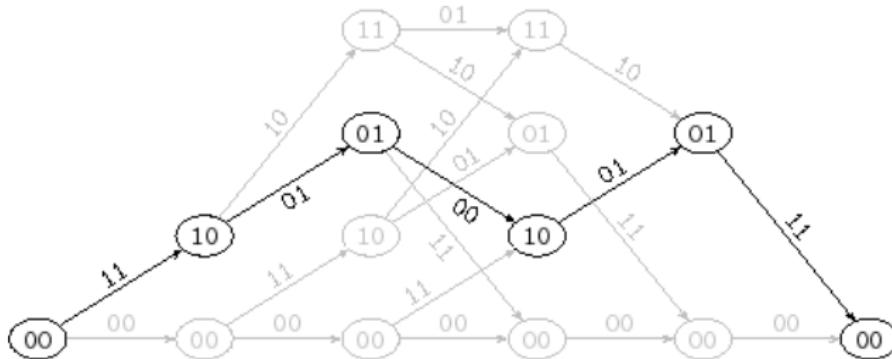
Exemple : Pour le code $(2, 1, 2)$, le treillis de longueur $m = 3$ est :



Codes convolutionnels

Codage

Codage dans le Treillis : Par exemple, le codage du message $u = 101(0)(0)$ correspond au chemin suivant :



c'est-à-dire au mot de code $z = 1101000111$

Codes convolutionnels

Décodage

Un mot de code correspond à un chemin du nœud de départ au nœud de fin du treillis.

Le décodage consiste alors à trouver le chemin le plus approprié correspondant au message reçu à décoder.

Dans le cadre du décodage à distance minimale (le mot de code avec un nombre minimal d'erreurs), "le chemin le plus approprié" signifie le chemin avec la distance de Hamming minimale par rapport au message à décoder \Rightarrow programmation dynamique par l'algorithme de Viterbi.

Codes convolutionnels

Décodage : Algorithme de Viterbi

L'**algorithme de Viterbi** consiste à retrouver le chemin qui correspond au mot de code le plus proche du mot reçu.

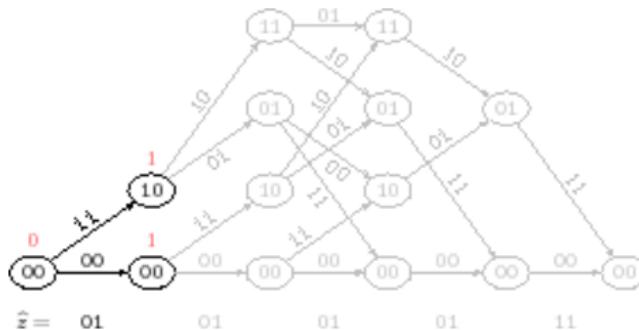
Algorithm 3 Algorithme de Viterbi

- 1: $\gamma_0(00) = 0$
 - 2: **for** i de 0 à $|\hat{z}|$ **do**
 - 3: $\gamma_i(s) = \min_{st \rightarrow s} (d(z_{2i-1}z_{2i}, \hat{z}_{2i-1}\hat{z}_{2i}) + \gamma_{i-1}(st))$
 - 4: Marquer le/un arc de st à s qui réalise le minimum
 - 5: **end for**
 - 6: Reconstruire le chemin optimal en sens inverse, de l'état final à l'état nul initial.
-

Codes convolutionnels

Décodage : Algorithme de Viterbi

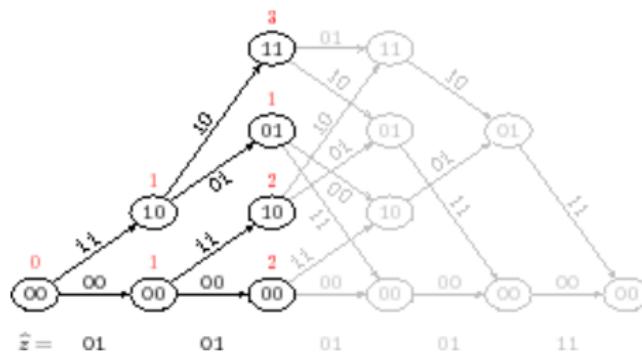
Exemple : Supposons que le mot de code $z = 1101000111$ soit envoyé à travers un canal bruité et que $\hat{z} = 0101010111$ soit reçu.



Codes convolutionnels

Décodage : Algorithme de Viterbi

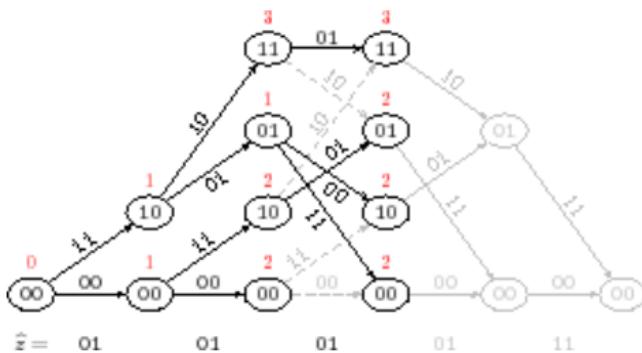
Exemple : Supposons que le mot de code $z = 1101000111$ soit envoyé à travers un canal bruité et que $\hat{z} = 0101010111$ soit reçu.



Codes convolutionnels

Décodage : Algorithme de Viterbi

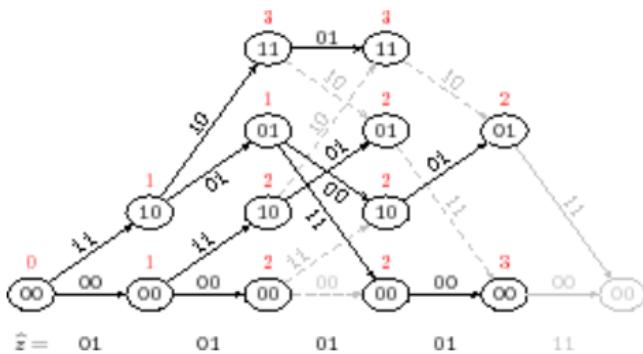
Exemple : Supposons que le mot de code $z = 1101000111$ soit envoyé à travers un canal bruité et que $\hat{z} = 0101010111$ soit reçu.



Codes convolutionnels

Décodage : Algorithme de Viterbi

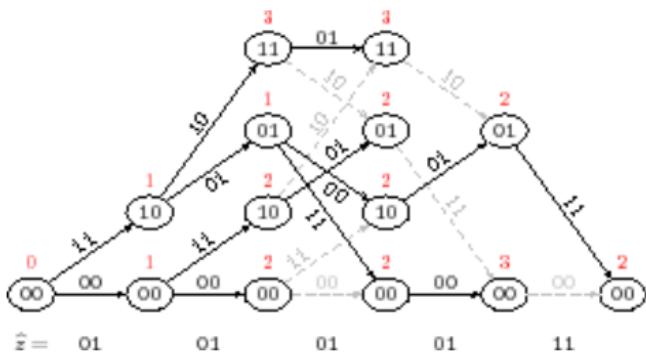
Exemple : Supposons que le mot de code $z = 1101000111$ soit envoyé à travers un canal bruité et que $\hat{z} = 0101010111$ soit reçu.



Codes convolutionnels

Décodage : Algorithme de Viterbi

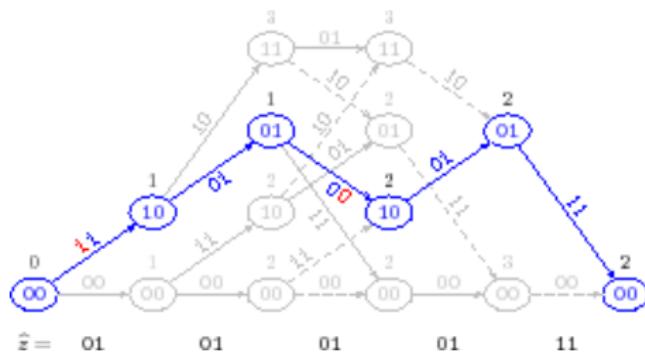
Exemple : Supposons que le mot de code $z = 1101000111$ soit envoyé à travers un canal bruité et que $\hat{z} = 0101010111$ soit reçu.



Codes convolutionnels

Décodage : Algorithme de Viterbi

Exemple : Supposons que le mot de code $z = 1101000111$ soit envoyé à travers un canal bruité et que $\hat{z} = 0101010111$ soit reçu.



Codes convolutionnels

Décodage : Algorithme de Viterbi

En pratique, dans un code convolutif, il est plus intéressant de décoder à la volée, le code pouvant être infini. L'idée est qu'une erreur n'a un impact que sur un nombre limité de mots de code : ce nombre est appelé **longueur de contrainte du code** et est égal à kr . Ainsi, pour décoder à la volée, il faut appliquer l'algorithme de Viterbi sur $T = kr$ mots de code reçus. Il est alors possible de décider que le premier mot de source ne dépendra plus du dernier sommet exploré. On reprend ensuite l'algorithme de Viterbi avec le mot de code reçu suivant qui permet de s'assurer du deuxième mot de source, etc.

Codes convolutionnels

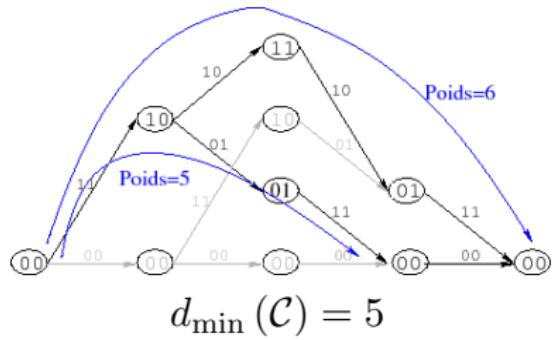
Distance minimale

Comment évaluer la capacité de correction d'un code convolutif ?

La notion de distance entre deux mots de code n'est pas adaptée (mots de code potentiellement infinis). On utilise la **distance libre du code**. Comme le code est linéaire, la distance libre est le poids minimal du code.

Poids minimal d'un code convolutionnel

Pour un code convolutionnel, le **poids minimal** est le nombre minimal de symboles non nuls sur un chemin menant de l'état nul à l'état nul.



2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

5 Cryptographie

- Les bases de la cryptographie
- Chiffrement à clef secrète
- Chiffrement à clef publique
- Clefs et Cryptanalyse

Terminologie

Cryptologie ("Science du Secret") = Cryptographie + Cryptanalyse

- *Cryptographie* : Science qui utilise les mathématiques pour chiffrer et déchiffrer des données ;
- *Cryptanalyse* : Science de l'analyse des données chiffrées afin d'en retrouver la version "claire" ;
- ≠ *Stéganographie* : Méthode pour cacher l'existence d'un message ;
- *Système cryptographique (ou cryptosystème)* : Algorithme de cryptographie + gestion des clefs + protocoles.

Référence : Request for Comments RFC 2828. "Internet Security Glossary". Mai 2000. <http://rfc.net/rfc2828.html>

Principe fondamental

Principes de Kerckhoffs (1883)

La sécurité d'un cryptosystème ne doit reposer que sur une donnée de petite taille, la clef. (Auguste Kerckhoffs, "La cryptographie militaire", Journal des sciences militaires, 1883.)

Ce principe a été reformulé par Claude Shannon (1949) :

« l'adversaire connaît le système » = *maxime de Shannon*.

Interprétations : B. Schneier ("Ce qui est gardé secret doit être ce qui est le moins coûteux à changer" - 2002), E. Raymond ("ne jamais faire confiance à un logiciel à code source fermé" - 2004).

Il est considéré aujourd'hui comme un principe fondamental par les cryptologues, et s'oppose à la sécurité par l'obscurité.

Objectifs

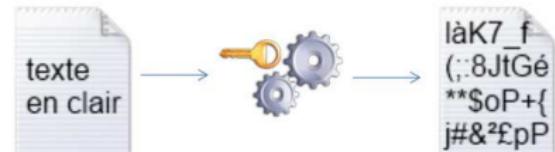
Cryptographie → services suivants :

- **Confidentialité** : ne permettre l'accès aux données qu'aux seules personnes autorisées (seul le destinataire est capable de déchiffrer) ;
- **Intégrité** : les données n'ont pas été modifiées sans autorisation (le destinataire doit être capable de déterminer si le message a été modifié) ;
- **Preuve** : \Leftrightarrow authentification et non-répudiation. Fournir un moyen de preuve garantissant la véritable identité des entités ainsi que l'imputation de leurs actions (le destinataire doit être capable d'identifier l'expéditeur qui ne peut nier être l'auteur du message).

Vocabulaire (1/3)

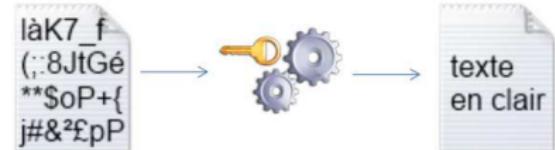
Chiffrer

Transformer une donnée de telle façon qu'elle devienne incompréhensible. Seules les entités autorisées pourront la comprendre.



Déchiffrer

Transformer une donnée précédemment chiffrée pour reconstituer la donnée d'origine. Seules les entités autorisées ont la capacité de procéder à cette action.



Recours à un algorithme et à une clé cryptographique.

Vocabulaire (2/3)

Signer

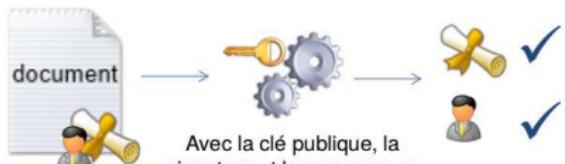
Créer une signature électronique unique à la donnée et à son auteur.
La signature lie donc la donnée d'origine et son auteur.



Avec une clé privée et un message en entrée, on obtient une signature en sortie

Vérifier la signature

S'assurer que la donnée d'origine n'a pas été modifiée et que son auteur est authentifié. Sinon il ne faut pas faire confiance au document.

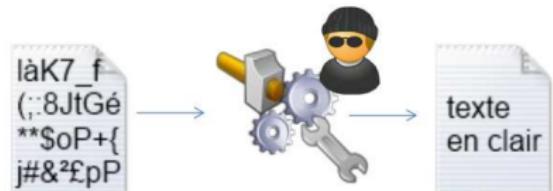


Avec la clé publique, la signature et le message en entrée, on obtient un verdict OK/NOK en sortie

Vocabulaire (3/3)

Décrypter

Reconstituer la donnée d'origine en tentant de "casser" la donnée chiffrée ou l'algorithme cryptographique.



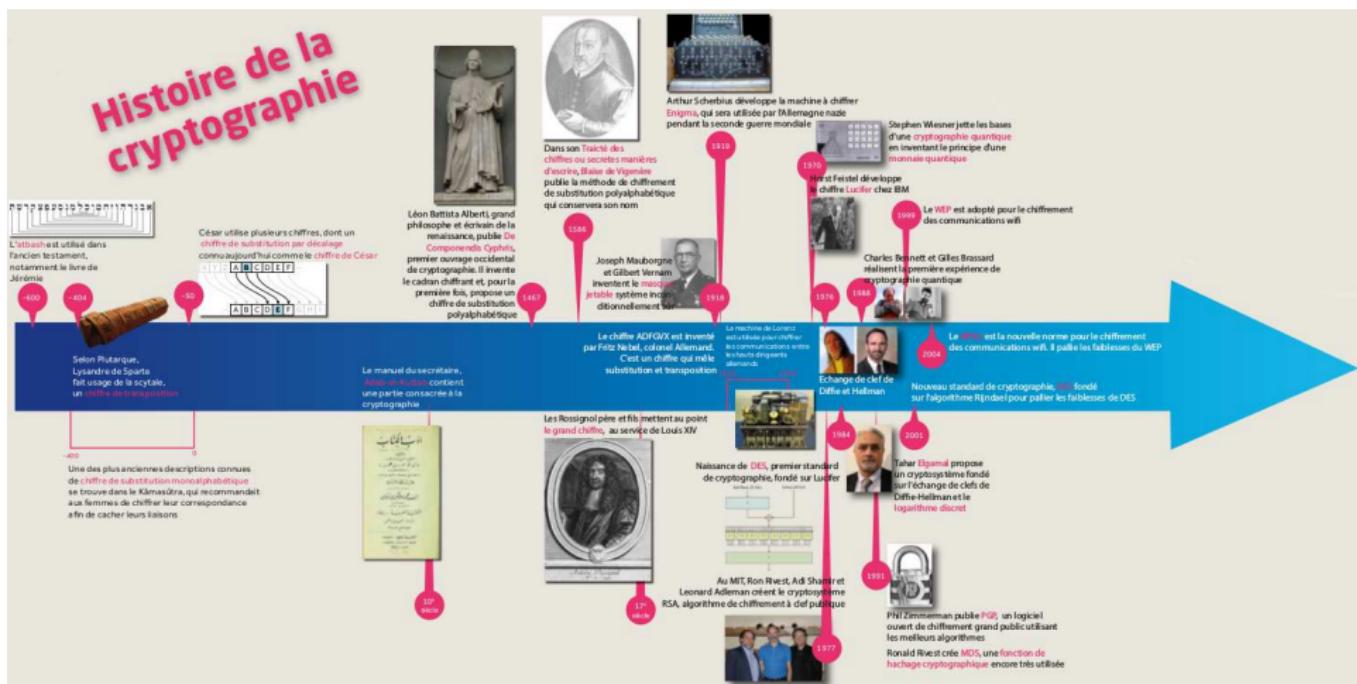
Crypter

La notion de crypter n'existe pas. Il s'agit d'un abus de langage.

<http://www.ssi.gouv.fr/administration/formations/cyberedu/>

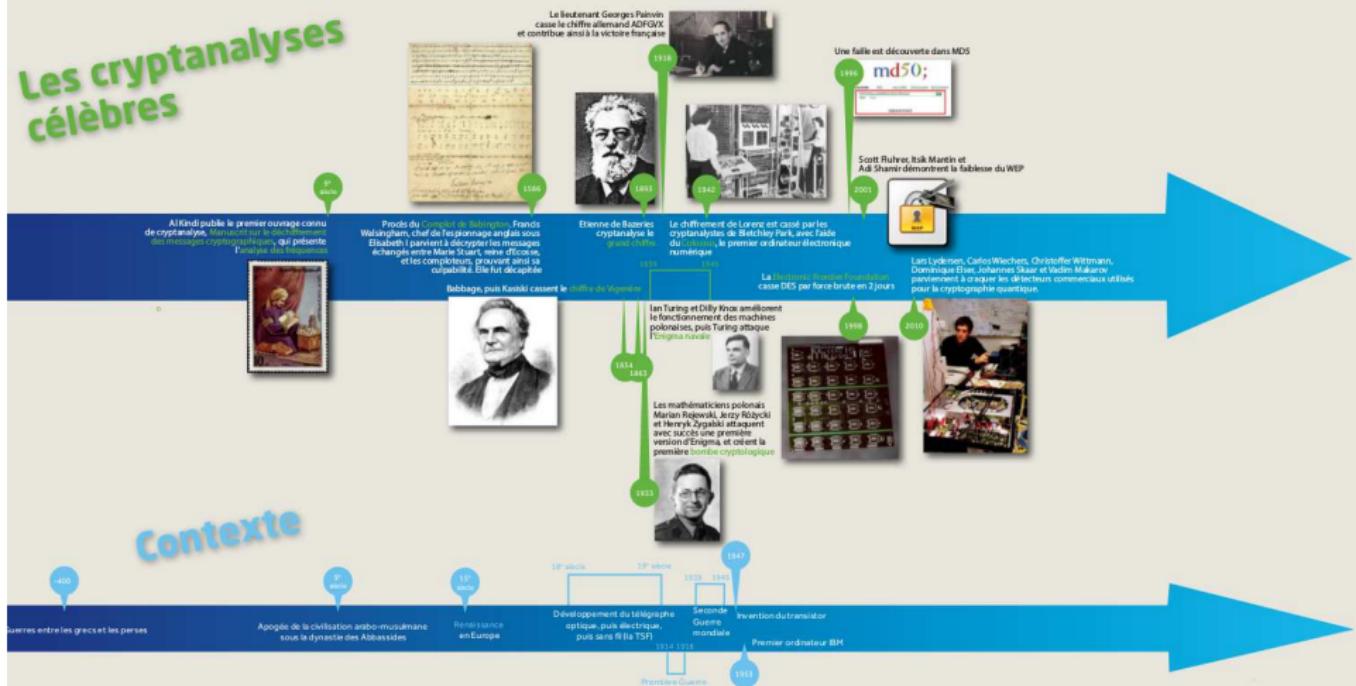
Histoire de la cryptographie

https://www-fourier.ujf-grenoble.fr/~rossigno/Vulgarisation/vulg_files/poster_fds2012.pdf



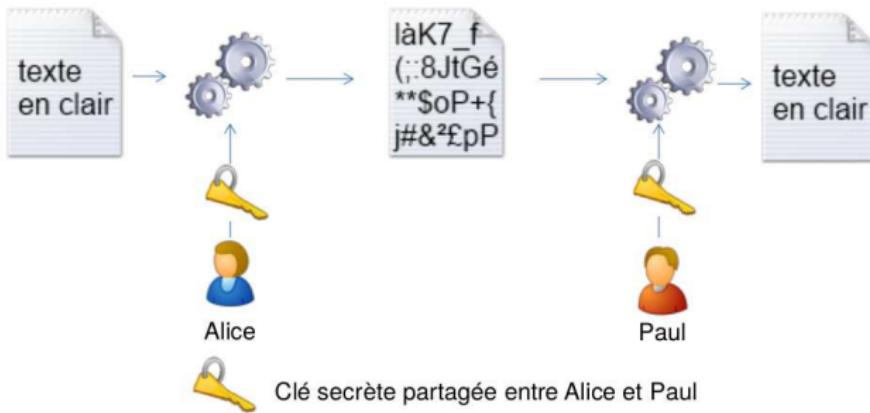
Histoire des cryptanalyses

https://www-fourier.ujf-grenoble.fr/~rossigno/Vulgarisation/vulg_files/poster_fds2012.pdf



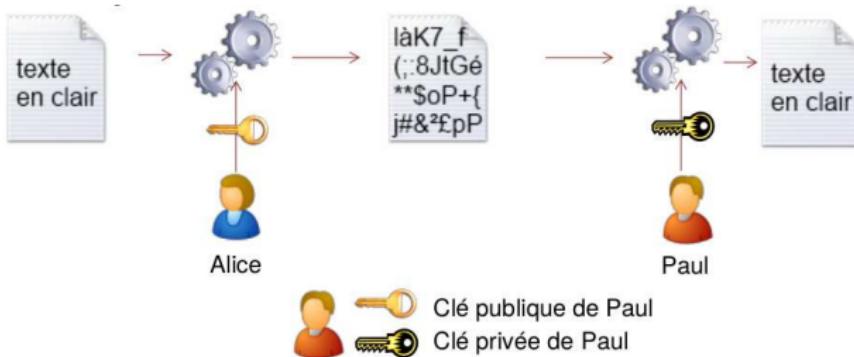
Chiffrement symétrique ou à clef secrète

- La clé utilisée pour le chiffrement est la même que celle utilisée pour le déchiffrement ;
- Cette clé doit être secrète : seules les personnes habilitées doivent posséder cette clé, sinon la confidentialité du message n'est plus assurée !



Chiffrement asymétrique ou à clef publique

- Utilisation de 2 clefs : chiffrement & déchiffrement
 - Clé publique : elle peut être donnée à tout le monde ;
 - Clé privée : elle est connue de son seul propriétaire.
- Ces deux clés sont mathématiquement liées.
 - La clé publique ne permet pas de trouver la clé privée ;
 - Chaque personne doit donc posséder les 2 clés.



Chiffrement symétrique vs asymétrique

Chiffrement symétrique

Rapidité des opérations (adapté à du trafic en temps réel) ;
Clés courtes (256 bits suffisent actuellement) ;

Chiffrement asymétrique

Avantages

Facilité d'échange des clefs : les seules clés qui ont besoin d'être échangées sont des clés publiques ;

Inconvénients

Difficulté d'échange sécurisé des clefs secrètes : comment le faire en protégeant ce secret ?

Lenteur des opérations (peu adapté à du trafic en temps réel) ;
Grande taille des clefs (2048 bits minimum actuellement) ;

Exemples d'algorithmes sûrs (janvier 2015)

AES

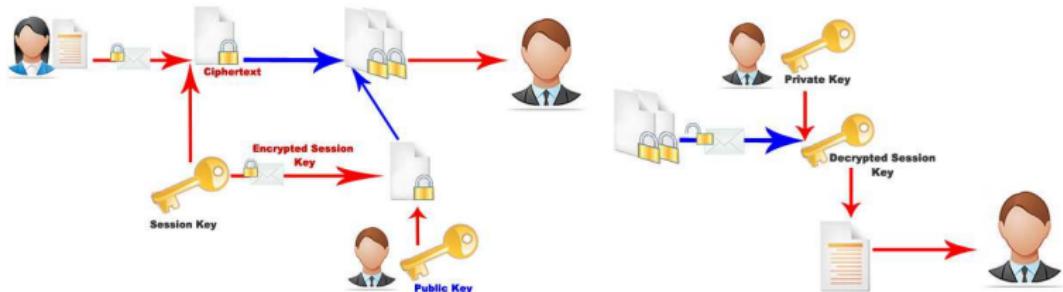
RSA

Chiffrement hybride

Cryptographie asymétrique = Lente - Échange de clefs simple

Cryptographie symétrique = Rapide - Difficulté d'échange des clefs

Cryptographie hybride combine les deux systèmes afin de bénéficier des avantages (rapidité de la cryptographie symétrique pour le contenu du message) et utilisation de la cryptographie "lente" uniquement pour la clé.



Signature électronique (1/3)

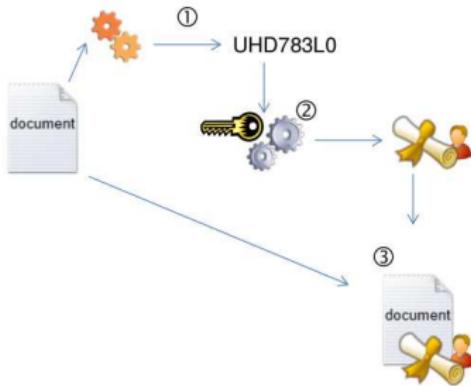
Rappel de l'objectif : s'assurer de la non-modification d'une donnée, et s'assurer de l'identité de son auteur. Si la signature n'est pas valide, cela indique que l'auteur « n'est pas le bon » ou que la donnée reçue n'est pas celle que son auteur avait signé.

Notes :

- La signature électronique n'assure pas la confidentialité des données, mais leur intégrité et la notion de preuve ;
- Lorsque l'on chiffre un message, il est fortement recommandé de le signer également afin d'assurer l'intégrité du message.

Signature électronique (2/3)

Principe



Etapes de la signature :

- ① Le signataire génère le condensat unique associé au message ;
- ② Le signataire utilise l'algorithme de signature, qui prend en entrée sa clé privée et le condensat précédent, pour produire une signature électronique ;
- ③ Le signataire envoie (ou stocke) le message et la signature électronique, permettant ainsi à un lecteur d'en prendre connaissance ;

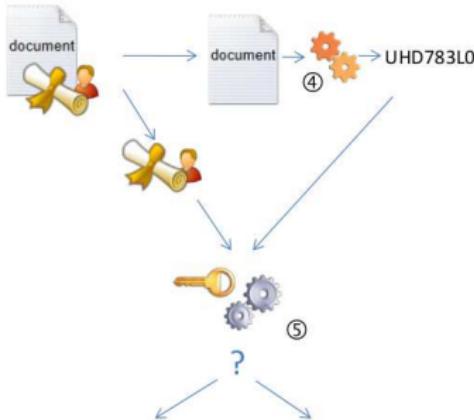
La vérification par le destinataire/lecteur est décrite sur la diapositive suivante.



Clé publique du signataire
Clé privée du signataire

Signature électronique (3/3)

Principe



✓ La signature est valide. Le message est intégrée.

✗ La signature est invalide. Le message n'est pas intégré.



Clé publique du signataire

Clé privée du signataire

Certificats électroniques

Les interlocuteurs de Paul ont besoin d'utiliser sa clé publique. Comment peuvent-ils être certains que la « clé publique de Paul » appartient effectivement à Paul et qu'elle n'a pas été générée frauduleusement en son nom ?



Clé publique de Paul

Clé privée de Paul

Autre exemple, comment les visiteurs d'un site web bancaire peuvent être certains que le site web est légitime et qu'il ne s'agit pas d'un site frauduleux imitant celui d'une banque ?

Solution :

Utilisation de certificats électroniques.

Certificats électroniques

Un certificat est un fichier électronique qui comprend notamment :

- La clef publique d'un individu ;
- Les détails de cet individu : nom, prénom, etc. ;
- La signature par un tiers de confiance, chargé de garantir que le propriétaire de la clé publique a été vérifié. Elle porte sur l'identité du détenteur et la clef publique (intégrité de l'ensemble) ;
- D'autres informations telles que l'usage de la clef, les dates de validité, des informations concernant la révocation, etc.

Le tiers de confiance, une autorité de certification, en charge de :

- Vérifier l'identité de la personne qui crée le certificat ;
- Créer le certificat après vérification, puis le signer (avec la clef privée de l'autorité de certification) ;
- Tenir à jour une liste des certificats qui ont été révoqués.

Certificats électroniques

Exemple sous un navigateur Firefox

General Details

This certificate has been verified for the following uses:

SSL Client Certificate
SSL Server Certificate

Issued To

Common Name (CN) www.france-universite-numerique-mooc.fr
Organization (O) <Not Part Of Certificate>
Organizational Unit (OU) Domain Control Validated
Serial Number 03:57:E9:2A:6D:41:A9:5F:5E:73:18:D1:D7:DE:C0:C0

Issued By

Common Name (CN) TERENA SSL CA 2
Organization (O) TERENA
Organizational Unit (OU) <Not Part Of Certificate>

Period of Validity

Begins On 09/02/2015
Expires On 09/02/2018

Fingerprints

SHA-256 Fingerprint C1:0B:DC:B6:95:F1:57:48:EE:5B:D5:BF:46:9E:72:D8
BC:7D:F6:C8:6F:DB:EA:8D:BB:E2:1E:6B:5F:F5:7B:D8
SHA1 Fingerprint 69:91:55:BB:4D:01:69:09:5C:B0:DD:BF:13:CE:44:C7:1A:6B:F7:53

Website Identity

Website: www.france-universite-numerique-mooc.fr
Owner: This website does not supply ownership information.
Verified by: TERENA

Privacy & History

Have I visited this website prior to today? Yes, 16 times
Is this website storing information (cookies) on my computer? Yes
Have I saved any passwords for this website? Yes

Technical Details

Connection Encrypted (TLS_DHE_RSA_WITH_AES_256_CBC_SHA, 256 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information travelling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Help Close

2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

5 Cryptographie

- Les bases de la cryptographie
- Chiffrement à clef secrète
- Chiffrement à clef publique
- Clefs et Cryptanalyse

Chiffrement parfait

Définition d'un chiffrement parfait

Un système de chiffrement est dit **parfait** si la connaissance d'un message chiffré n'apporte aucune information sur le message clair même à un adversaire ayant des ressources calculatoires illimitées.

Chiffrement de Vernam - Chiffrement à clef jetable (1918)

$$C = M \oplus K \text{ avec } |M| = |K| \text{ et } K \text{ totalement aléatoire}$$

Seul chiffrement connu à l'heure actuelle \Rightarrow mathématiquement prouvé inconditionnellement sûr (Shannon 1949).

Remarque : Définition $\Leftrightarrow \Pr [M = m | C = c] = \Pr [M = m]$ ou $S(M|C) = S(M)$.

Chiffrements pratiquement sûr

Chiffrements pratiquement sûr

On s'approche du chiffrement parfait (très dur car K doit être totalement aléatoire et utilisée une seule fois) par des *chiffrements pratiquement sûr*.

$\Leftrightarrow C$ ne permet pas de retrouver K ni M en un temps humainement raisonnable.

Temps humainement raisonnable ? :

- 128 bits \Leftrightarrow 39 décimaux, 512 bits \Leftrightarrow 155 chiffres ;
- PC à 1 GHz \Leftrightarrow 1 milliard (10^9) opérations par seconde ;
- Compter jusqu'à un nombre de 39 chiffres = énumérer 10^{39} ;
- Age de l'univers = 15 milliards d'années = $5 \cdot 10^{17}$ s ;
- Nombre d'électrons dans l'univers : 10^{84} .

Chiffrements pratiquement sûr

Chiffrements pratiquement sûr

On s'approche du chiffrement parfait (très dur car K doit être totalement aléatoire et utilisée une seule fois) par des *chiffrements pratiquement sûr*.

$\Leftrightarrow C$ ne permet pas de retrouver K ni M en un temps humainement raisonnable.

Temps humainement raisonnable ? :

- 128 bits \Leftrightarrow 39 décimaux, 512 bits \Leftrightarrow 155 chiffres ;
- PC à 1 GHz \Leftrightarrow 1 milliard (10^9) opérations par seconde ;
- Compter jusqu'à un nombre de 39 chiffres = énumérer 10^{39} ;
- Age de l'univers = 15 milliards d'années = $5 \cdot 10^{17}$ s ;
- Nombre d'électrons dans l'univers : 10^{84} .

Chiffrements pratiquement sûr

Chiffrements pratiquement sûr

On s'approche du chiffrement parfait (très dur car K doit être totalement aléatoire et utilisée une seule fois) par des *chiffrements pratiquement sûr*.

$\Leftrightarrow C$ ne permet pas de retrouver K ni M en un temps humainement raisonnable.

Temps humainement raisonnable ? :

- 128 bits \Leftrightarrow 39 décimaux, 512 bits \Leftrightarrow 155 chiffres ;
- PC à 1 GHz \Leftrightarrow 1 milliard (10^9) opérations par seconde ;
- Compter jusqu'à un nombre de 39 chiffres = énumérer 10^{39} ;
- Age de l'univers = 15 milliards d'années = $5 \cdot 10^{17}$ s ;
- Nombre d'électrons dans l'univers : 10^{84} .

Chiffrements pratiquement sûr

Chiffrements pratiquement sûr

On s'approche du chiffrement parfait (très dur car K doit être totalement aléatoire et utilisée une seule fois) par des *chiffrements pratiquement sûr*.

$\Leftrightarrow C$ ne permet pas de retrouver K ni M en un temps humainement raisonnable.

Temps humainement raisonnable ? :

- 128 bits \Leftrightarrow 39 décimaux, 512 bits \Leftrightarrow 155 chiffres ;
- PC à 1 GHz \Leftrightarrow 1 milliard (10^9) opérations par seconde ;
- Compter jusqu'à un nombre de 39 chiffres = énumérer 10^{39} ;
- Age de l'univers = 15 milliards d'années = $5 \cdot 10^{17}$ s ;
- Nombre d'électrons dans l'univers : 10^{84} .

Classes de chiffrement à clef secrète

On distingue deux grandes catégories de chiffrement symétrique :

- Chiffrement Symétrique par flot (*stream cipher*) ;
- Chiffrement Symétrique par bloc (*block cipher*).

Le principe du chiffrement par flot est de chiffrer une suite de caractères un à la fois, à l'aide d'une transformation qui varie au fur et à mesure du texte.

Le chiffrement par bloc utilise une transformation fixe, sur des blocs plus gros, typiquement 64 ou 128 bits.

L'avantage du chiffrement par flot est qu'il s'implante mieux en hardware, et qu'il ne nécessite pas de zone tampon (*buffer*).

Chiffrement par flot - *stream cipher*

Leur construction est basée sur le principe du chiffrement de Vernam, mais évite ses inconvénients : la clé est différente et aussi longue que le message mais elle est issue d'une autre clé, fixe et de petite taille.

Ils génèrent, à partir d'une clef de petite taille, un flot d'aléa vu comme une clef aléatoire plus longue servant à un système de Vernam. La clé aléatoire, appelée clef du flot (keystream), peut être générée bit par bit ou octet par octet suivant le système.

L'algorithme permettant la génération de la clé du flot est un générateur pseudo-aléatoire (GPA) :

$$K_s = GPA(K) \text{ puis } C = M \oplus K_s.$$

Chiffrement par flot

Exemple : A5/1

A5/1 est utilisé dans le cadre des communications GSM et produit une suite pseudo-aléatoire où on effectue un XOR avec les données.

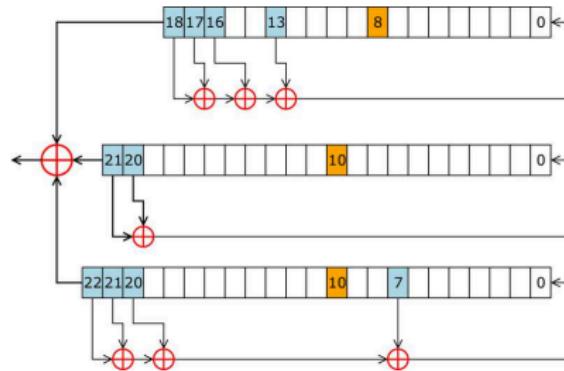
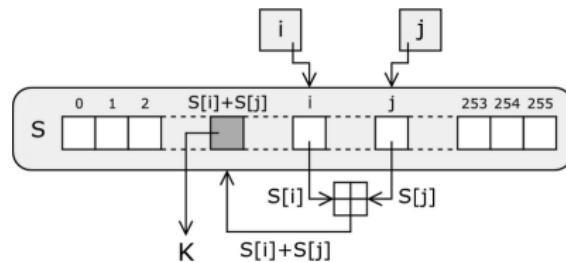


Schéma du A5/1 : un registre est décalé si le bit en orange correspond à la majorité des trois bits orange. On insère alors un bit correspondant à un XOR entre les bits en bleu.

Chiffrement par flot

Exemple : RC4 (Rivest Cipher 4)

Conçu en 1987 par Ronald Rivest (l'un des inventeurs du RSA), pour les Laboratoires RSA. Il est supporté par différentes normes, par exemple dans TLS (anciennement SSL) ou encore WEP.



Permutation S des 256 octets est construite à partir de la clef.

Chiffrement par flot

Exemple : RC4 (Rivest Cipher 4) - *PRGA (Pseudo Random Generator Algorithm)*

Algorithm 4 Génération de la suite chiffrante RC4

Require: S permutation, $n \in \mathbb{N}$, $N \in \mathbb{N}$, ($N = 256$ par exemple)

Ensure: $(z_1, \dots, z_n) \in \{0, \dots, N - 1\}^n$

- 1: $i = 0$
- 2: $j = 0$
- 3: **for** l de 1 à n **do**
- 4: $i \leftarrow (i + 1) \bmod N$
- 5: $j \leftarrow (j + S[i]) \bmod N$
- 6: $S[i] \leftrightarrow S[j]$
- 7: $z_l \leftarrow S[(S[i] + S[j]) \bmod N]$
- 8: **end for**
- 9: **return** z_1, \dots, z_n (keystream word)

Chiffrement par flot

Exemple : RC4 (Rivest Cipher 4)- *KSA (Key Scheduling Algorithm)*

Algorithm 5 Génération de la permutation initiale de RC4

Require: $K \in \{0, \dots, N - 1\}^l$, $N \in \mathbb{N}$, ($N = 256$ par exemple)

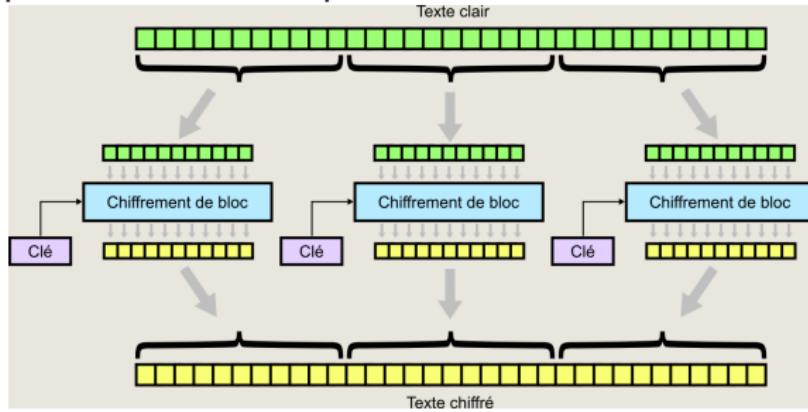
Ensure: S permutation initiale

- 1: **for** i de 0 à $N - 1$ **do**
- 2: $S[i] \leftarrow i$
- 3: **end for**
- 4: $j = 0$
- 5: **for** i de 0 à $N - 1$ **do**
- 6: $j \leftarrow (j + S[i] + K[i \text{ } (\text{mod } l)]) \text{ mod } N$
- 7: $S[i] \leftrightarrow S[j]$
- 8: **end for**
- 9: **return** S

Chiffrement par bloc (*block cipher*) - Modes opératoires

Dictionnaire de codes - *Electronic codebook (ECB)*

Le message à chiffrer est subdivisé en plusieurs blocs qui sont chiffrés séparément les uns après les autres.



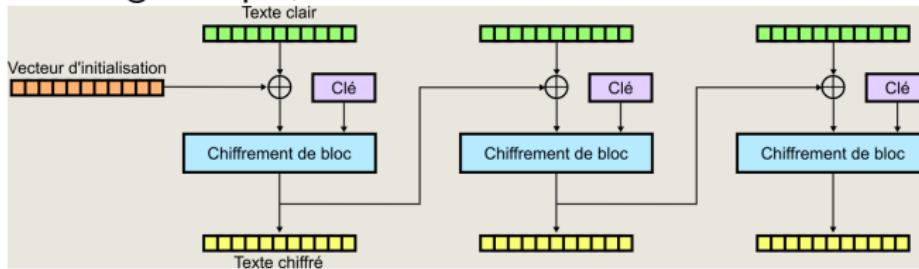
Défaut : deux blocs avec le même contenu seront chiffrés de la même manière (cryptanalyse).

Fortement déconseillé en cryptographie. Le seul avantage = accès rapide à une zone quelconque du texte chiffré.

Chiffrement par bloc - Modes opératoires

Enchainement des blocs - *Cipher Block Chaining (CBC)*

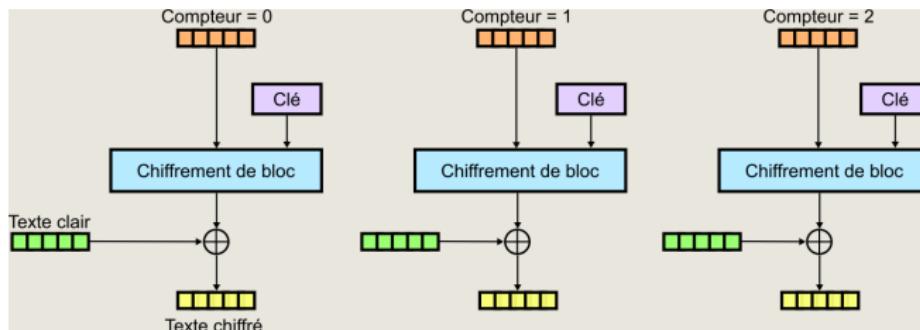
On applique sur chaque bloc un XOR avec le chiffrement du bloc précédent avant qu'il soit lui-même chiffré. De plus, afin de rendre chaque message unique, un vecteur d'initialisation est utilisé.



Chiffrement par bloc - Modes opératoires

Chiffrement basé sur un compteur - *CounTeR (CTR)*

Le flux de clef est obtenu en chiffrant les valeurs successives d'un compteur.



Nombreux avantages : précalculable, accès aléatoire aux données, parallélisable et n'utilise que la fonction de chiffrement.

Chiffrement par bloc - Architecture

Réseaux de substitutions-permutations

Le réseau comprend des boîtes de substitution et de permutation. Ces opérations sont conçues de manière à être efficace sur du matériel et font souvent appel à l'opération XOR.

S-Box - substitution box

Table de substitution qui contribue à la "confusion" en rendant l'information originale inintelligible. Elles permettent de casser la linéarité de la structure de chiffrement.

P-Box - permutation box

Table de permutation qui indique comment échanger les éléments d'une structure. Elle contribue à la "diffusion" en mélangeant les données et en améliorant l'effet avalanche.

Chiffrement par bloc - Architecture

Réseaux de substitutions-permutations

Une boîte *S-Box* ou *P-Box* seul \neq force cryptographique $\Rightarrow S\text{-}Box \Leftrightarrow$ chiffre de substitution et *P-Box* \Leftrightarrow chiffre de transposition.

Un réseau de substitutions permutations (bien conçu) répond aux propriétés de confusion et de diffusion de Shannon :

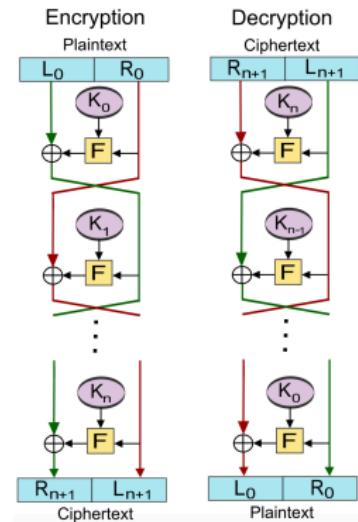
- **Diffusion** : la redondance statistique dans un texte en clair est dissipée dans les statistiques du texte chiffré. Un biais en entrée ne doit pas se retrouver en sortie : notion d'effet avalanche.
Bonne diffusion = l'inversion d'un seul bit en entrée doit changer chaque bit en sortie avec une probabilité de 0,5.
- **Confusion** : chaque chiffre binaire (bit) du texte chiffré devrait dépendre de plusieurs parties de la clef.

Chiffrement par bloc- Architecture

Réseau de Feistel

Un réseau de Feistel est subdivisé en plusieurs tours ou étages. À chaque tour, les deux blocs sont échangés puis un des blocs est combiné avec une version transformée de l'autre bloc. Puis au tour suivant, on inverse. Chaque tour utilise une clé intermédiaire (issue de la clé principale = *key schedule*).

Avantages : chiffrement et déchiffrement \Rightarrow architecture similaire. L'implémentation matérielle est facile.

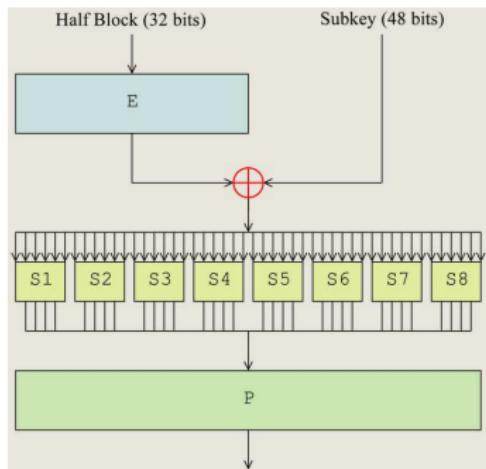


Chiffrement par bloc - Exemple

DES - Data Encryption Standard

DES utilise des clefs de 56 bits. Standard de chiffrement par le NIST en 1977. Son emploi n'est plus recommandé.

- Fractionnement du texte en blocs de 64 bits (8 octets) ;
- Permutation initiale des blocs ;
- Découpage des blocs en deux parties : gauche et droite, nommées G et D ;
- Étapes de permutation et de substitution répétées 16 fois (appelées rondes) ;
- Recollement des parties gauche et droite puis permutation initiale inverse.



Chiffrement par bloc - Exemple

DES - Data Encryption Standard

S₅		Middle 4 bits of input																	
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111		
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001		
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110		
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110		
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011		

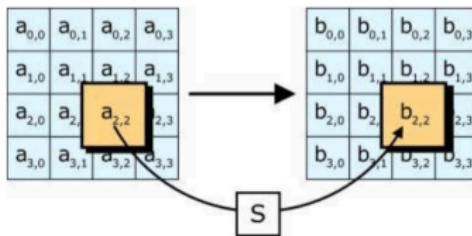
Chiffrement par bloc- Exemple

AES - Advanced Encryption Standard

- Besoin d'un chiffre en remplacement du DES ;
- Ouverture d'un concours (septembre 1997) par le NIST (National Institute of Standards and Technology) :
 - un algorithme public, non-classifié, sans droits d'utilisation ;
 - un chiffre symétrique sur des blocs d'au moins 128 bits et une taille de clef de 128, 192 et 256 bits ;
 - de sécurité au sens des méthodes de cryptanalyse connues ;
 - performant pour toutes les implantations (des petits processeurs, carte à puce, aux circuits spécialisés).
- 15 candidats ⇒ présélection mars 1999 ⇒ cinq finalistes : MARS, RC6, Rijndael (1996), Serpent, TwoFish ;
- Résultats définitifs (2000) : le NIST choisit Rijndael (issu des noms Vincent Rijmen & Joan Daemen)

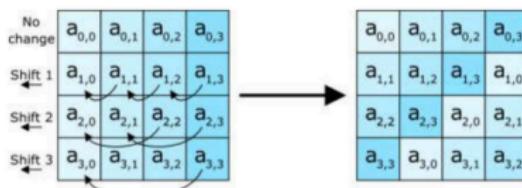
Chiffrement par bloc- Exemple

AES - Advanced Encryption Standard



Substitution

Chaque octet est remplacé par un autre en fonction d'une table

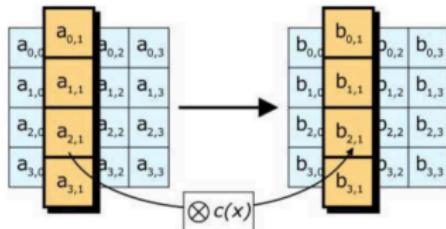


Transposition

Chaque ligne est décalé cycliquement en fonction du numéro de ligne

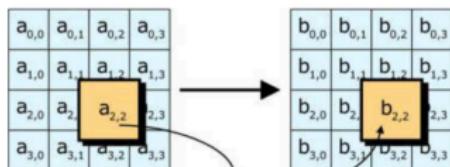
Chiffrement par bloc- Exemple

AES - Advanced Encryption Standard



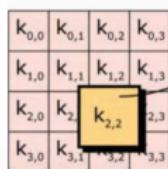
Mélange

On applique une transformation linéaire, fonction d'une matrice auxiliaire.



Cycle de clé

Chaque octet est combiné avec une sous-clé, elle-même dérivée de la clé principale (key schedule)



Chiffrement par bloc- Synthèse

Comparatif d'algorithmes de chiffrement symétriques

	3xDES	IDEA	BLOWFISH	CAST 5	TWOFISH	AES
Publication	1978	1991	1993	1996	1998	1998
Taille de bloc (bits)	64	64	64	64	128	128
Taille de clé	168 (112 effectif)	128	32-448 (par 8)	40-128 (par 8)	128, 192, 256	128, 192, 256
Structure	Feistel	substitution / permutation	Feistel	Feistel	Feistel,	substitution / permutation
Nbr. de Tours	3x16	8 ½	16	12 ou 16	16	10, 12 ou 14
Clé faible		OUI	OUI	NON		NON

LOGICIELS	PGP 9.0	OUI	OUI		OUI	OUI
	GnuPG	OUI	OUI	OUI	OUI	OUI
	OpenSSL	OUI	OUI	OUI		OUI

2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

5 Cryptographie

- Les bases de la cryptographie
- Chiffrement à clef secrète
- Chiffrement à clef publique
- Clefs et Cryptanalyse

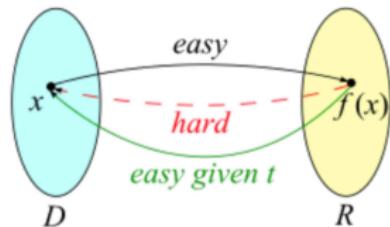
Introduction

L'idée de la cryptographie à clef publique est née dans les années 1970. Elle s'appuie sur une **fonction dite à sens unique** f telle que :

- il est facile d'évaluer $f(x)$ lorsqu'on connaît x ;
- connaissant un y dans l'espace d'arrivée, il est difficile de trouver un x tel que $f(x) = y$.

Fonction à sens unique à trappe

f à sens unique est à **trappe** si le calcul dans le sens inverse est efficace en disposant d'une information secrète t (la trappe - *trapdoor*) qui permet de construire g tel quel $g \circ f = Id$.



Le problème du logarithme discret

Éléments mathématiques

Logarithme discret

Pour tout entier m compris entre 1 et $p - 1$, il existe un entier k compris entre 0 et $p - 2$ et un seul tel que

$$m \equiv g^k \pmod{p}$$

m et g^k ont le même reste quand on les divise par p .

On dit que g est une racine primitive modulo p et que k est le logarithme discret de m dans la base g .

- Calculer m à partir de g et k facile par l'algorithme de l'exponentiation rapide ;
- Aucun moyen de calculer rapidement k à partir de m et g (si p est grand).

Le problème du logarithme discret

Algorithmique

Exponentiation rapide ou Exponentiation par carré

$$m \equiv g^k \pmod{p}$$

L'exposant k converti en notation binaire : $k = \sum_{i=0}^{n-1} k_i 2^i$.

Longueur de k est de n bits. k_i peut prendre la valeur 0 ou 1 pour n'importe quel i . Par définition, $k^{n-1} = 1$.

La valeur g^k s'écrit : $g^k = g^{\left(\sum_{i=0}^{n-1} k_i 2^i\right)} = \prod_{i=0}^{n-1} \left(g^{2^i}\right)^{k_i}$.

La solution m est, par conséquent :

$$m \equiv \prod_{i=0}^{n-1} \left(g^{2^i}\right)^{k_i} \pmod{p}$$

Le problème du logarithme discret

Algorithmique

Exemple : Exponentiation rapide avec $g = 4$, $k = 13$, et $p = 497$.

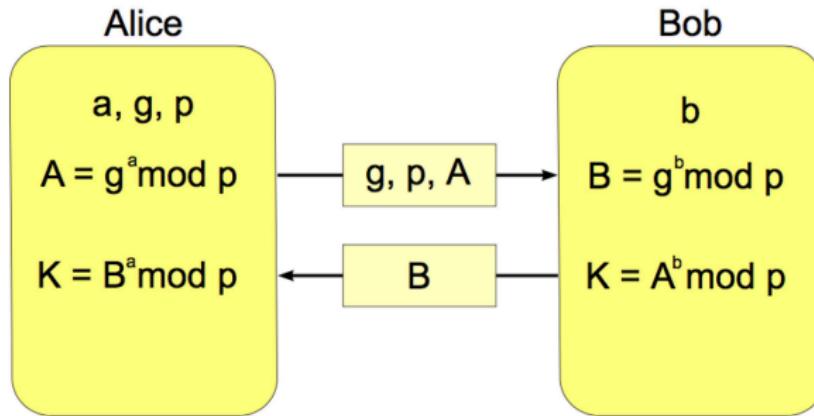
L'exposant k est égale à 1101 en base 2. Comme k est de longueur 4, l'algorithme va avoir 4 itérations :

- ① base = 4, exp = 1101 et res = 1. Comme exp =xxx1, res = $(1 \times 4) \text{ mod } 497 = 4$, puis exp = 110, puis base = $(4 \times 4) \text{ mod } 497 = 16$;
- ② Comme exp = xx0, res = res, exp = 11, base = $(16 \times 16) \text{ mod } 497 = 256$;
- ③ Comme exp =x1, res = $(4 \times 256) \text{ mod } 497 = 30$, puis exp =1, base = $(256 \times 256) \text{ mod } 497 = 429$;
- ④ Comme exp = 1, res = $(30 \times 429) \text{ mod } 497 = 445$.

Le problème du logarithme discret

Protocole Diffie-Hellman

Diffie et Hellman ont proposé en 1976 un protocole d'échange de clés totalement sécurisé.



Attention : ce protocole est vulnérable à "*Man in the middle attack*" \Rightarrow rajouter des signatures.

Le problème du logarithme discret

Protocole Diffie-Hellman : Exemple d'échange de clefs

Alice et Bob choisissent un nombre premier $p = 23$ et une base $g = 3$.

- ① Alice choisit un nombre entier secret $a = 6$;
- ② Elle envoie à Bob la valeur $A \equiv g^a \pmod{p} = 3^6 \pmod{23} = 16$;
- ③ Bob choisit à son tour un nombre secret $b = 15$;
- ④ Bob envoie à Alice la valeur $B \equiv g^b \pmod{p} = 3^{15} \pmod{23} = 12$;
- ⑤ Alice peut maintenant calculer la clé secrète : $B^a \pmod{p} = 12^6 \pmod{23} = 9$;
- ⑥ Bob fait de même et obtient la même clé qu'Alice : $A^b \pmod{p} = 16^{15} \pmod{23} = 9$.

Le problème du logarithme discret

Cryptosystème El Gamal (Inventé par Taher ElGamal en 1985)

Le destinataire, Alice, possède deux clés :

- une clé secrète : un entier k_s ;
- une clé publique (p, a, k_p) , qui consiste en un nombre premier p , un entier $a \in \llbracket 1, p - 1 \rrbracket$ et l'entier $k_p \equiv a^{k_s} \pmod{p}$.

Si Bob veut envoyer le message m à Alice ($m \in \llbracket 1, p - 1 \rrbracket$) :

- il tire au hasard un nombre k ;
- calcul $c_1 \equiv a^k \pmod{p}$ et $c_2 \equiv mk_p^k \pmod{p}$;
- Message chiffré = (c_1, c_2) qu'il transmet à Alice.

A la réception, celle-ci calcule

- $r \equiv c_1^{k_s} \pmod{p} = a^{k_s k} \pmod{p} = k_p^k \pmod{p}$;
- puis il retrouve $m \equiv c_2 r^{-1} \pmod{p} = m k_p^k k_p^{-k} \pmod{p}$.

Le problème du logarithme discret

Cryptosystème El Gamal

Robustesse du système :

Si Eve a en sa possession a, p, k_p, c_1 et c_2 :

Pour retrouver m , elle doit savoir calculer $k_p^k \bmod p$. Ceci impose de trouver k et donc savoir résoudre l'équation suivante (en k)
 $y \equiv a^k \bmod p$ pour n'importe quel entier y .

On appelle ce problème le **calcul du logarithme discret modulo p** . C'est un problème pour lequel on ne dispose pas d'algorithme rapide.

Le problème du logarithme discret

Cryptosystème El Gamal : Exemple

Considérons l'ensemble $\mathbb{Z}/p\mathbb{Z}$ où $p = 19$ et un élément $a = 10$.

- Alice calcul sa clef :
 - Choisie $k_s = 5$ et calcul $k_p \equiv a^{k_s} \pmod{p} = 10^5 \pmod{19} = 3$.
- Bob envoie un message $m = 17$ par $(c_1, c_2) = (11, 5)$ car
 - Choisi aléatoirement $k = 6$;
 - Calcul $r \equiv k_p^k \pmod{p} = 3^6 \pmod{19} = 7$;
 - Calcul $c_1 \equiv a^k \pmod{p} = 10^6 \pmod{19} = 11$;
 - Calcul $c_2 \equiv mr \pmod{p} = 7 \times 17 \pmod{19} = 5$.
- Alice déchiffre le message original :
 - Retrouve $r \equiv c_1^{k_s} \pmod{p} = 11^5 \pmod{19} = 7$;
 - Calcul l'inverse $r^{-1} = 7^{-1} = 11$;
 - Retrouve $m \equiv c_2 r^{-1} \pmod{p} = 5 \times 11 \pmod{19} = 17$.

Factorisation des entiers et primalité

Éléments mathématiques

Petit Théorème de Fermat (1636)

Si p est un nombre premier et si a est un entier non divisible par p :

$$a^{p-1} \equiv 1 \pmod{p}$$

Indicateur d'Euler

Si n est un entier > 2 , l'indicateur d'Euler de n , noté $\varphi(n)$ désigne le nombre d'entiers premiers avec n et compris entre 1 et n .

- Si n est premier : $\varphi(n) = n - 1$;
- Si n est produit de 2 entiers premiers, $n = p \times q$, alors $\varphi(n) = (p - 1)(q - 1)$;
- Si m et n sont premiers entre eux alors $\varphi(mn) = \varphi(m)\varphi(n)$.



Factorisation des entiers et primalité

Éléments mathématiques

Théorème d'Euler

Si n est un entier naturel et a est une premier avec n , alors :

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Théorème RSA

Soit $n = p \times q$ un produit de deux nombres premiers. Soit a un élément quelconque de $\mathbb{Z}/n\mathbb{Z}$. Quel que soit l'entier positif k :

$$a^{k\varphi(n)+1} \equiv a \pmod{n}$$

Factorisation des entiers et primalité

Chiffrement RSA

Le chiffrement RSA repose sur un utilisateur (Alice) qui crée son couple (clef publique / clef privée) :

- Choisir deux grands nombres premiers distincts p et q ;
- Calculer $n = p \times q$, appelé *module de chiffrement* ;
- Calculer $\phi(n) = (p - 1)(q - 1)$, qui est la valeur de l'*indicatrice d'Euler* en n ;
- Choisir un entier naturel e premier avec $\phi(n)$ et strictement inférieur à $\phi(n)$, appelé *exposant de chiffrement* ;
- Calculer d , l'inverse de e modulo $\phi(n)$ appelé *exposant de déchiffrement* ;
- Publier la paire $K_e = (e, n)$ comme *clef publique RSA* ;
- Garder secrète la paire $K_d = (d, n)$ qui est la *clef privée RSA*.

Factorisation des entiers et primalité

Chiffrement RSA

Chiffrement : Si M est un entier naturel strictement inférieur à n représentant un message, alors le message chiffré sera représenté par

$$E(M) = C \equiv M^e \pmod{n}$$

L'entier naturel C étant choisi strictement inférieur à n .

Déchiffrement : Pour déchiffrer C , on utilise d , l'inverse de e modulo $(p - 1)(q - 1)$, et on retrouve le message clair M par

$$D(C) = M \equiv C^d \pmod{n}.$$

D'après le théorème RSA on remarque bien que

$$D_{K_d}(E_{K_e}(M)) \equiv M^{ed} \pmod{n} = M^{1+\alpha\phi(n)} \pmod{n} = M.$$

Factorisation des entiers et primalité

Chiffrement RSA : Exemple

- Choisir deux entiers premiers : $p = 47$, $q = 71$;
- Calculer $n = p \times q = 3337$;
- Calculer : $\phi(n) = (p - 1)(q - 1) = 46 \times 70 = 3220$;
- Choisir e : exemple $e = 79$ (premier avec $\phi(n)$) ;
- Calculer $d = 1019$: inverse de e modulo $\phi(n)$;
- Chiffrer un message M : $M = 6882326879666683$;
- Décomposition en blocs de taille inférieure $< n = 3337$;
- Ici des blocs de 3 chiffres $M = 688\ 232\ 687\ 966\ 668\ 3$;
- Chiffrer 688 : $688^{79} \bmod 3337 = 1570$ puis les autres blocs ;
- $\Rightarrow E(M) = 1570\ 2756\ 2091\ 2276\ 2423\ 158\ 6$;
- Déchiffrer bloc par bloc en commençant par 1570 ;
- $1570^{1019} \bmod 3337 = 688$ etc ...

Factorisation des entiers et primalité

Algorithmique

Inverse modulaire

L'inverse modulaire d'un entier relatif a pour la multiplication modulo n est un entier u satisfaisant l'équation :

$$au \equiv 1 \pmod{n} \text{ noté également } u \equiv a^{-1} \pmod{n}.$$

L'inverse de a modulo n existe si et seulement si a et n sont premiers entre eux.

Calcul : algorithme d'Euclide étendu ou théorème d'Euler.

2 Théorie de l'information et Codage

3 Compression par transformée

4 Détection et correction d'erreurs

5 Cryptographie

- Les bases de la cryptographie
- Chiffrement à clef secrète
- Chiffrement à clef publique
- Clefs et Cryptanalyse

Taille des clefs nécessaires

Longueurs de clefs asymétriques recommandées (RSA)

1995 ⇒ 1280 bits, 2000 ⇒ 1280 bits, 2005 ⇒ 1536 bits, 2010 ⇒ 1536 bits, 2015 ⇒ 2048 bits.

Taille d'une clef publique

80 bits offre une sécurité équivalente à celle d'une clef publique (RSA) de 1024 bits, 128 bits ⇔ 3072 bits, 256 bits ⇔ 15360 bits.

Comparaison

Sym (bits)	80	112	128	192	256
RSA (bits)	1024	2048	3072	7680	15360
ECC (bits)	160	224	256	384	512

Comparaison RSA / ECC

ECC (*Elliptic Curve Cipher*) : Passeport électronique européen, carte d'identité australienne, carte d'identité allemande, nouvelle carte de santé allemande.

RSA (1978) libre et publique en 1993 / ECC (1985) normes fin 1990 ;

Brevets : RSA expirés en 2000, encore 130 brevets sur ECC (Certicom) ; RSA semble plus simple que ECC \Rightarrow problème d'interopérabilité supplémentaire (NIST : courbes d'étalonnage) ; Beaucoup d'implantations supportent que 2 d'entre eux (P-256 et P-384 : Suite B de la recommandation NSA). Crédit à la sécurité + rapide pour ECC que RSA.

Création des clefs - Logiciels

OpenSSL - *Cryptography and SSL/TLS Toolkit*

Boîte à outils cryptographiques implémentant les protocoles SSL/TLS :

- Bibliothèques de programmation en C : libcrypto fournit les algorithmes cryptographiques, libssl implémente le protocole Transport Layer Security (TLS) ainsi que son prédecesseur Secure Sockets Layer (SSL) ;
- Interface en ligne de commande openssl :
 - création de clés RSA, DSA (signature) ;
 - création de certificats X509
 - calcul d'empreintes (MD5, SHA, RIPEMD160, ...)
 - chiffrement et déchiffrement (DES, IDEA, RC2, RC4, ...)
 - réalisation de tests de clients et serveurs SSL/TLS
 - signature et le chiffrement de courriers (S/MIME)



Création des clefs - Logiciels

Format OpenPGP : PGP - *Pretty Good Privacy*, GnuPG

Logiciel de chiffrement et de déchiffrement cryptographique :

- Signature électronique et vérification d'intégrité de messages (fonction de hachage -MD5 et du système RSA) ;
- Chiffrement des fichiers locaux (IDEA) ;
- Génération de clefs publiques et privées (clefs privées IDEA + transfert de clefs par RSA) ;
- Gestion des clefs, Certification de clefs ;
- Révocation, désactivation, enregistrement de clefs.

Il figure parmi les outils généralement recommandés. Sa robustesse face aux attaques de la NSA a été confirmée dans un article publié le 28 décembre 2014 par *Der Spiegel*².

2. <http://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html>

Cryptanalyse - Techniques

Classification des techniques de cryptanalyse par les données qu'elles nécessitent :

- attaque sur texte chiffré seul (*ciphertext-only*) : on possède des exemplaires chiffrés des messages : faire des hypothèses sur les messages originaux ;
- attaque à texte clair connu (*known-plaintext attack*) : on possède des messages (ou des parties) en clair ainsi que les versions chiffrées. Ex : **Cryptanalyse linéaire** ;
- attaque à texte clair choisi (*chosen-plaintext attack*) : on possède des messages en clair et les versions chiffrées de ces messages avec l'algorithme. Ex : **Cryptanalyse différentielle** ;
- attaque à texte chiffré choisi (*chosen-ciphertext attack*) : on possède des messages chiffrés et on demande la version en clair de certains de ces messages pour mener l'attaque.

THE END !

Bibliographie :

Jean-Guillaume Dumas, Jean-Louis Roch, Eric Tannier, Sébastien Varrette, "Théorie des codes : Compression, cryptage, correction", Collection : Sciences Sup, Dunod 2013 - 2ème édition - 384 pages, EAN13 : 9782100599110.

Stephane Mallat, "A Wavelet Tour of Signal Processing : The Sparse Way", Editeur : Academic Press, 2008, ISBN-13 : 978-0123743701.

Jean-Luc Starck, Fionn Murtagh, Jalal M. Fadili, " Sparse Image and Signal Processing : Wavelets, Curvelets, Morphological Diversity, Editeur : Cambridge University Press, 2010, ISBN-13 : 978-0521119139.