Théorie de l'information

Thomas Grenier

Hugues Benoit-Cattin

Plan

1. Introduction

1948 : Shannon → Théorie de l'information

Réflexion sur les techniques de communication (XIX°)

- Mécanique, accoustique
- Ondes radio-électrique
- Télégraphe (code morse)
- Téléphone, ….

Système de communication = Σ fonctions physiques réalisables

↳ Mauvaise compréhension des perturbations, des débits …

Vue d'ensemble d'un système de communication indépendante des moyens techniques & physiques

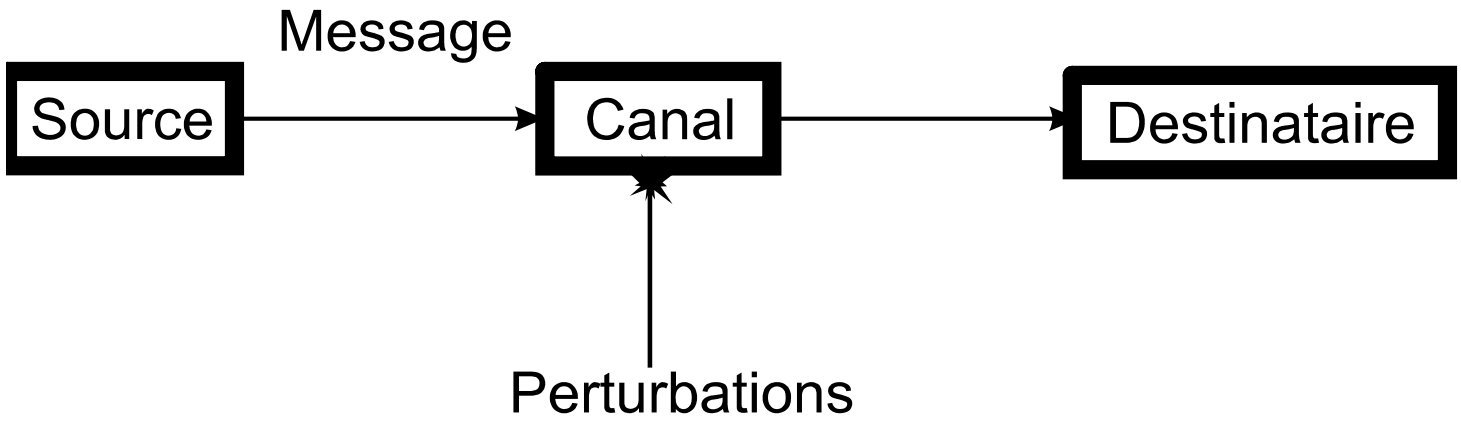
☹ Ca ne sert à rien !

☺ 1960 / conquête spatiale → codage de source

Aujourd'hui 

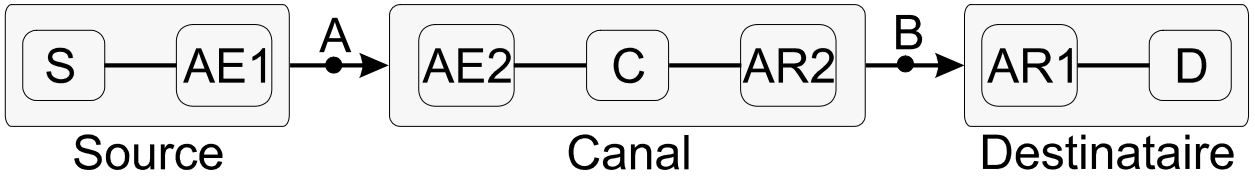
- ✓GSM ⇨ codage de source & canal
- ✓TV Num ⇨ codage de source & canal
- ✓Réseaux ⇨ codage de canal (erreurs)
- ✓@business ⇨ cryptage

- Paradigme de Shannon = modèle sys. com.



Source	=	je parle
Canal	=	l'air ambiant
Perturbations	=	bruit sonore
Destinataire	=	tu écoutes

- Modèle détaillé



✓Th. Signaux ⇨ décrit messages et perturbations

✓Modulation ⇨ modifie les signaux pour les propager

✓Electronique ⇨ réalise les fonctions

✓Th. Information ⇨ propose une mesure quantitative de l'**information** et étudie sa représentation, sa transmission, sa dégradation

✓**Source** : siège d'évènements aléatoires qui constituent le message émis ⇨ **Entropie**

✓**Canal** : transmet et dégrade le message ⇨ **Capacité**

Des messages différents portent la même information, le **codage** cherche le message avec les meilleures propriétés.

✓Codage de source ⇨ supprime la redondance, réduit le coût

✓Codage de canal ⇨ protège contre les perturbations

✓Cryptage/Authentification ⇨ protège des curieux

Deux théorèmes fondamentaux :

- Codage de source
- Codage de canal

2. Sources discrètes & Entropie

Sources débitant des messages sous forme discrète !

✓**Source discrète d'information** : suite de variables aléatoires discrètes X_1, X_2, … X_n

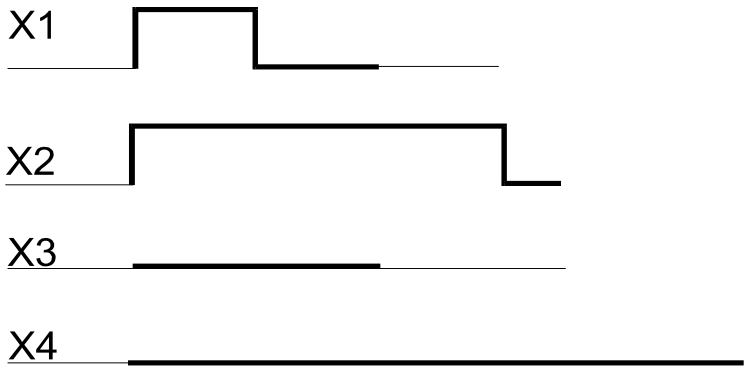
✓**Symbole** ou **lettre** : élément fondamental irréductible contenant une information, cad réalisation particulière de la source d'information.

✓**Mot** : succession finie de symboles

✓**Alphabet** : totalité des D lettres

$[X] = [x_1, x_2, \ldots, x_D]$

Ex : Code morse, 4 symboles



✓**Source discrète sans mémoire** : source pour laquelle la probabilité d'apparition d'un symbole ne dépend pas des symboles précédents

$$p(x_{i_n} / x_{i_{n-1}}, x_{i_{n-2}}, ...) = p(x_{i_n})$$

✓**Source discrète à mémoire** : source pour laquelle la probabilité d'apparition d'un symbole dépend du ou des symboles précédents

✓**Source sationnaire** : source pour laquelle les probabilités d'apparition des différents symboles ne dépendent pas de l'origine des temps

$$p(x_{i_n}) = p(x_{i_{n+k}}) \quad \forall k$$

✓**Source à débit contrôlable** : source pouvant générer des messages comme suite à une commande externe (Télégraphe, .)

✓**Source à débit non contrôlable** : source générant des messages avec un débit fixé, propriété de la source (CD audio)

✓**Source discrète à contraintes fixes** : source pour laquelle certains symboles ne peuvent être utilisés qu'en des conditions déterminées (Morse, …)

✓**Source discrète à contraintes probabilistes** : source à mémoire. Dans un état, la source peut générer n'importe lequel des symboles avec une probabilité qui dépend des symboles précédents (texte …)

✓**Source de Markov** : source pour laquelle la probabilité de générer un symbole ne dépend que du symbole à l'instant n-1

$$p(x_{i_n} / x_{i_{n-1}}, x_{i_{n-2}}, ...) = p(x_{i_n} / x_{i_{n-1}})$$

Quantité d'information & Entropie

- **Quantité d'information propre**

Propriété de l'information = **imprévisibilité**

Quantité d'information propre : $h(x) = f(1/p(x))$

Avec f croissante & f(1)=0

2 evt. indépendants apportent la somme de leur quantité d'info

$$h(x,y) = f(1/p(x,y)) = f(1/p(x).p(y)) = f(\frac{1}{p(x)}) + f(\frac{1}{p(y)}) = h(x) + h(y)$$

f → fonction **logarithme** (Base 2 >> bit)

$$h(x) = \log(1/p(x)) = -\log(p(x))$$

$$h(x,y) = \log\left(1/p(x,y)\right)$$

$$h(x/y) = \log\left(1/p(x/y)\right)$$

Règle de Bayes : $p(x,y) = p(x/y).p(y) = p(y/x).p(x) = p(y,x)$

$$h(x,y) = h(x/y) + h(y) = h(y/x) + h(x) = h(y,x)$$

$h(x/y) = h(x)$ si x et y indépendants

Ex → cartes

- **Entropie**

Hyp : source discrète finie stationnaire sans mémoire

Emission = variable aléatoire X

$$p_i = p(X = x_i) \qquad \text{pour} \quad i = 1,2,...,n$$

$$\sum_{i=1}^{n} p_i = 1$$

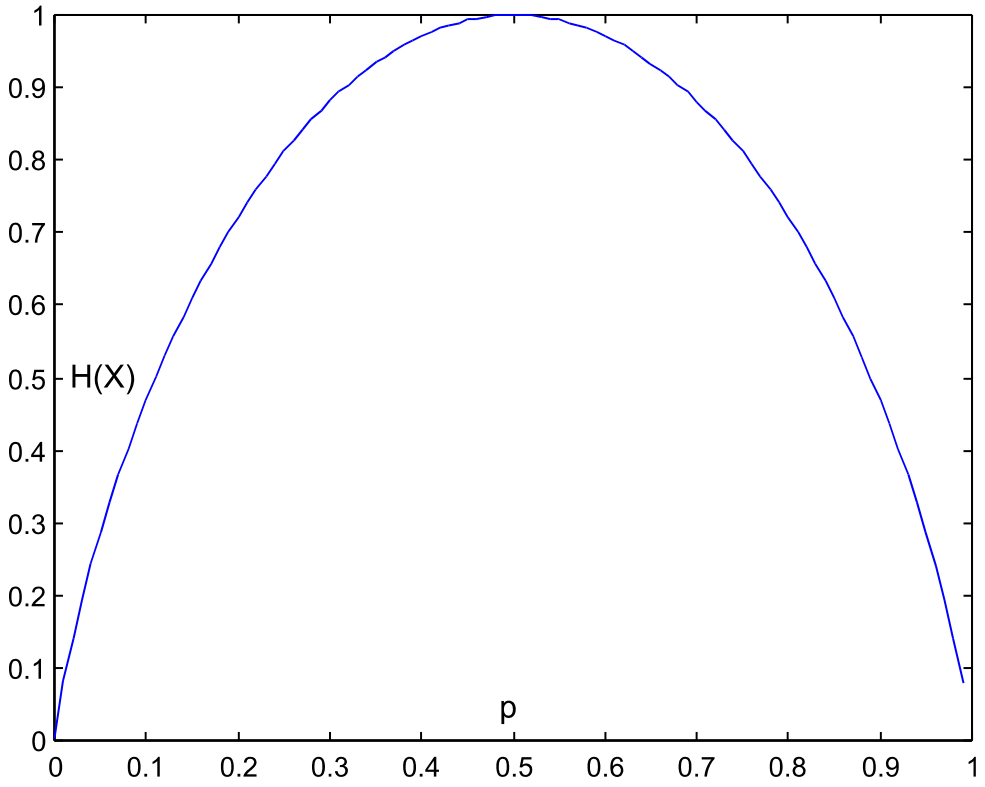
Quantité d'information moyenne associée à chaque symbole de la source = **entropie**

$$H(X) = E(h(X)) = \sum_{i=1}^{n} p_i . \log(1/p_i) = -\sum_{i=1}^{n} p_i . \log(p_i)$$

- Ex : Source binaire

$$p(1) = p$$
$$p(0) = 1 - p$$

$$H(X) = \begin{cases} -p.\log(p) - (1-p).\log(1-p) & \text{pour } 0 < p < 1 \\ 0 \quad \text{si} \quad p = 0 \text{ ou } 1 \end{cases}$$



- **Propriétés de l ’entropie**

✓**Continuité** : l'entropie est une fonction continue de chaque variable p_i.

✓**Additivité** : de part la définition de l'information propre.

✓**Positive** : $H(X) = H(p_1, p_2, ..., p_n) \geq 0$

✓**Bornée** : $$H(X) \leq H(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}) = \log(n)$$

- **Redondance**

$$R = H_{\max}(X) - H(X)$$

$$\rho = 1 - \frac{H(X)}{H_{\max}(X)}$$

- **Entropie & Débit d ’information**

  ✓Le débit d'information d'une source est donné par le produit de l'entropie de la source (valeur moyenne de l'info /symbole) par le nombre moyen de symboles par seconde soit :

$$D_X = \frac{H(X)}{\tau} \quad (bits.s^{-1}) \quad \text{avec } \tau \text{ durée moyenne d'un symbole}$$

- **Source Qaire**

  ✓**Source Q^{aire}** : source S dont l'alphabet possède Q éléments

  ✓**k^{ième} extension** : source S^k dont l'alphabet Q^{kaire} est obtenu en groupant par bloc de k celui de la source S

3. Canaux discrets & Capacité

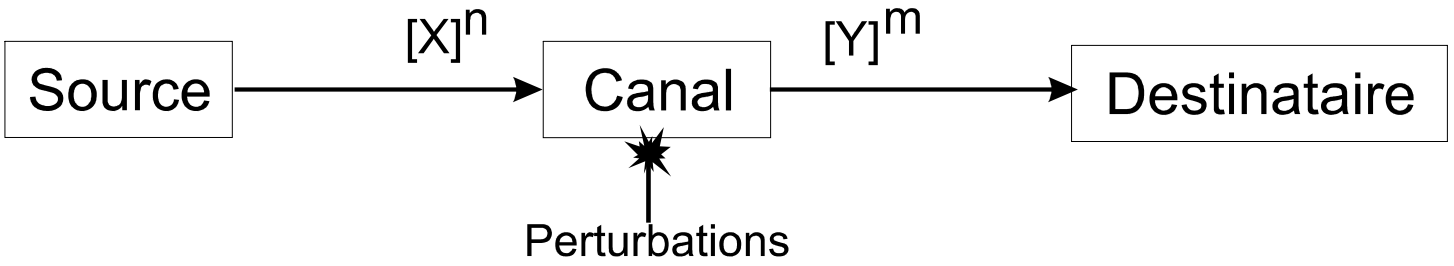
✓**Canal** : milieu de transmission de l'information situé entre la source et la destination. Le canal opère une transformation entre l'espace des symboles à l'entrée et celui de la sortie.

✓**Canal discret** : les espaces d'entrée et de sortie sont discrets

✓**Canal continu** : les espaces d'entrée et de sortie sont continus

✓**Canal sans mémoire** : si la transformation d'un symbole x à l'entrée en un symbole y en sortie ne dépend pas des transformations antérieures

✓**Canal stationnaire** : si les transformations ne dépendent pas de l'origine des temps



$$[X.Y]=\begin{bmatrix} x_1y_1 & x_1y_2 & \dots & x_1y_m \\ x_2y_1 & x_2y_2 & & x_2y_m \\ \dots & & & \dots \\ x_ny_1 & x_ny_2 & \dots & x_ny_m \end{bmatrix}$$

$$[P(X,Y)]=\begin{bmatrix} p(x_1,y_1) & p(x_1,y_2) & \dots & p(x_1,y_m) \\ p(x_2,y_1) & p(x_2,y_2) & & p(x_2,y_m) \\ \dots & & & \dots \\ p(x_n,y_1) & p(x_n,y_2) & \dots & p(x_n,y_m) \end{bmatrix}$$

- Probabilités marginales

$$p(x_i)=\sum_{j=1}^{m} p(x_i,y_j)$$

$$p(y_j)=\sum_{i=1}^{n} p(x_i,y_j)$$

$$H(X)=-\sum_{i=1}^{n} p(x_i).\log(p(x_i))$$

$$H(Y)=-\sum_{j=1}^{m} p(y_j).\log(p(y_j))$$

- Entropie réunie ou conjointe

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j).\log(\, p(x_i, y_j))$$

- Entropie conditionnelle ou équivoque

$$H(X/Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j).\log(\, p(x_i / y_j))$$

- Canaux non perturbés

$$H(X/Y) = H(Y/X) = 0$$

$$H(X,Y) = H(X) = H(Y)$$

- Canaux très (très) perturbés

$$H(X/Y) = H(X) \quad \text{et} \quad H(Y/X) = H(Y)$$

$$H(X,Y) = H(X) + H(Y)$$

Transinformation & capacité

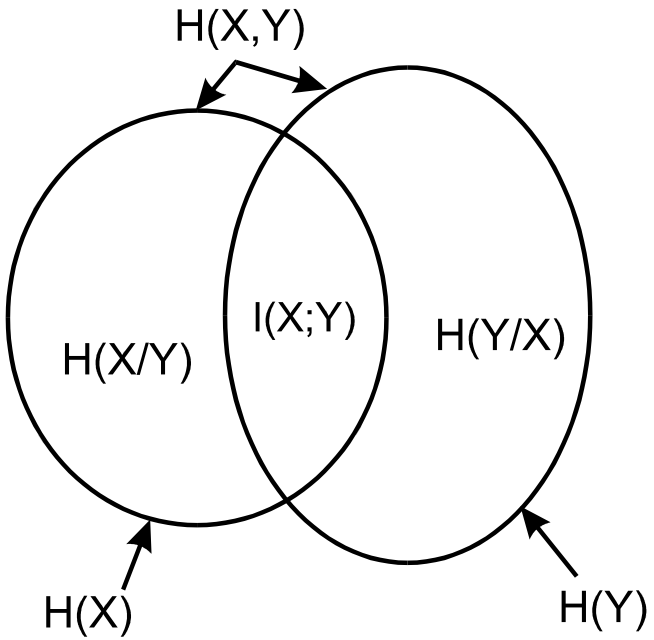
- Information mutuelle

$$i(x;y) = \log(\, p(x/y)/\, p(x)) \qquad \rightarrow i(x;y) = i(y;x)$$

- **Transinformation**

$$I(X;Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j).\log(\frac{p(x_i, y_j)}{p(x_i).p(y_j)})$$

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$
$$I(X;Y) = H(X) - H(X/Y) = H(Y) - H(Y/X)$$



- **Capacité d'un canal**

$$C = \underset{p(x)}{Max}(I(X;Y))$$

- **Redondance d'un canal**

$$Rc = C - I(X;Y) \qquad \rho_c = 1 - \frac{I(X;Y)}{C}$$

- **Efficacité d'un canal**

$$\eta_c = \frac{I(X;Y)}{C}$$

Ex → canal binaire

Plan

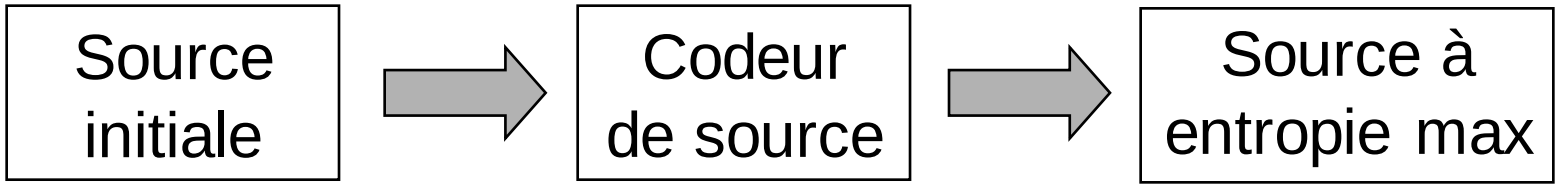
- 1. Introduction
- 2. Vue d’ensemble
- 3. Sources discrètes & Entropie
- 4. Canaux discrets & Capacité
- 5. Codage de source
- 6. Codage de canal
- 7. Cryptographie
- 8. Conclusion

5. Codage de source

✓**Adapter la source au canal** ⇨ l'alphabet
⇨ le débit

✓Utiliser la capacité du canal ⇨ maximiser I(X,Y)

- Hyp : Source stationnaire, canaux **sans perturbation**



Codeur de source ⇨ supprimer la **redondance**

- **Propriétés d'un codeur de source**

  ✓**Régularité** : messages ≠ ⇨ codes ≠

  ✓**Déchiffrabilité** : séparation des mots non ambiguë

- **Mot-code**

  $[S]=[s_1,s_2, \ldots, s_N]$ $[X]=[x_1,x_2, \ldots, x_D]$

  ↳ $[C]=[c_1,c_2, \ldots, c_N]$

- **Exemple**

Symbole	Code A	Code B	Code C	Code D
s_1	00	0	0	0
s_2	01	10	01	10
s_3	10	110	011	110
s_4	11	1110	0111	111

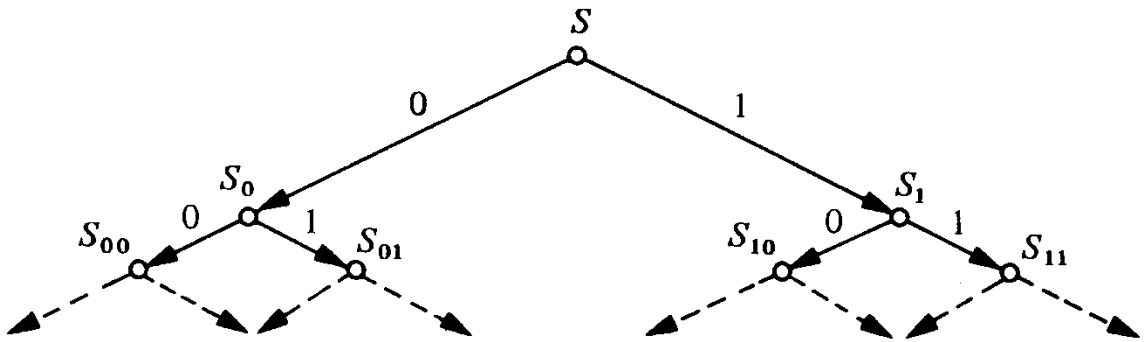
✓**Code à longueur variable / fixe**

✓**Code à décodage unique** : mot-code ⇔ symbole unique

✓**Code séparable** : pas de signe de démarcation entre les mots

✓**Code instantané** ou **irréductible** : on détermine les mots-codes à mesure que l'on reçoit les lettres de l'alphabet du code.
CNS : Aucun mot-code n'est le préfixe d'un autre !

• **Arbre & codes binaires instantanés**



- **Longueur moyenne d'un mot-code**

$$\bar{l} = \sum_{i=1}^{N} p(s_i).l_i$$

- **Limite de la longueur moyenne**

$$H(S) = H(C) = \bar{l}.H(X) \quad \Rightarrow \quad \bar{l} \geq \frac{H(S)}{\log D} = l_{\min}$$

- **Capacité - Efficacité - Redondance**

$$C = Max(H(X)) = \log D \qquad \eta = \frac{H(X)}{\log D} \qquad \rho = \frac{\log D - H(X)}{\log D}$$

$$\eta = \frac{H(S)}{\bar{l}.\log D}$$

Ex → code opt.

- **Codes optimaux absolus**

  Codes dont l'efficacité est maximale : $\eta = 1$

  ↳ $\bar{l} = l_{\min} = \dfrac{H(S)}{\log D}$

  ↳ $\boxed{\sum_{i=1}^{N} D^{-l_i} = 1}$ ⇨ Condition nécessaire pour les codes optimaux absolus

- **Théorème des canaux sans bruit (codage de source)**

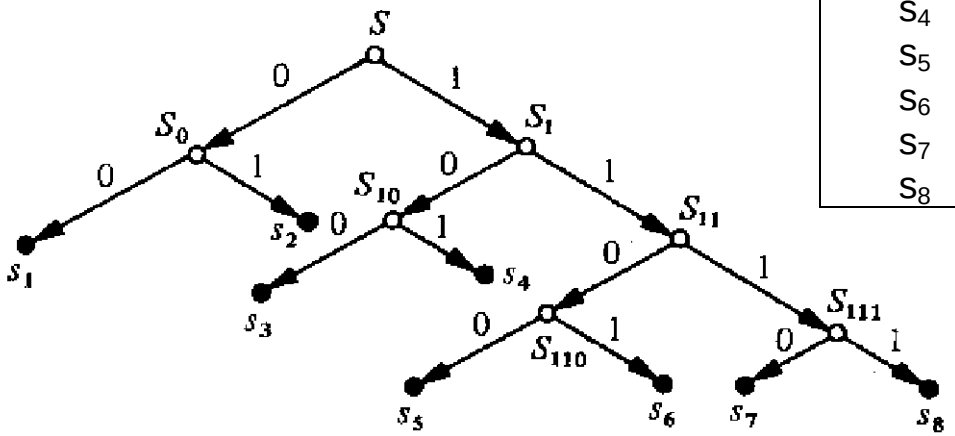
" Par un codage approprié (codage par groupe de n symboles de la source), l'information moyenne par lettre de l'alphabet du code peut être amenée aussi proche que l'on veut de la capacité du code, c'est-à-dire qu'**il existe toujours un codage optimal absolu** ."

Rq1 : à n fixé, le code qui donne $\eta_{max}<1$ est dit 'optimal'

Rq2 : en pratique, on travaillait à n=1

- **Codage de Shannon-Fano**

Algorithme de génération d'un **codage optimal absolu**, pour des sources divisibles récursivement (jusqu'à un symbole par ensemble) en deux sous-ensembles équiprobables.



Symboles s_k	Proba $p(s_k)$					Mots-codes c_k	Longueur l_k
s_1	0.25	0	0			00	2
s_2	0.25		1			01	2
s_3	0.125	1	0	0		100	3
s_4	0.125			1		101	3
s_5	0.0625		1	0	0	1100	4
s_6	0.0625				1	1101	4
s_7	0.0625			1	0	1110	4
s_8	0.0625				1	1111	4

- **Codage binaire de Huffman** (1952)

  - Algorithme de génération d'un **codage optimal** symbole par symbole.

  - Code à longueur variable ⇨ codes longs pour probas faibles

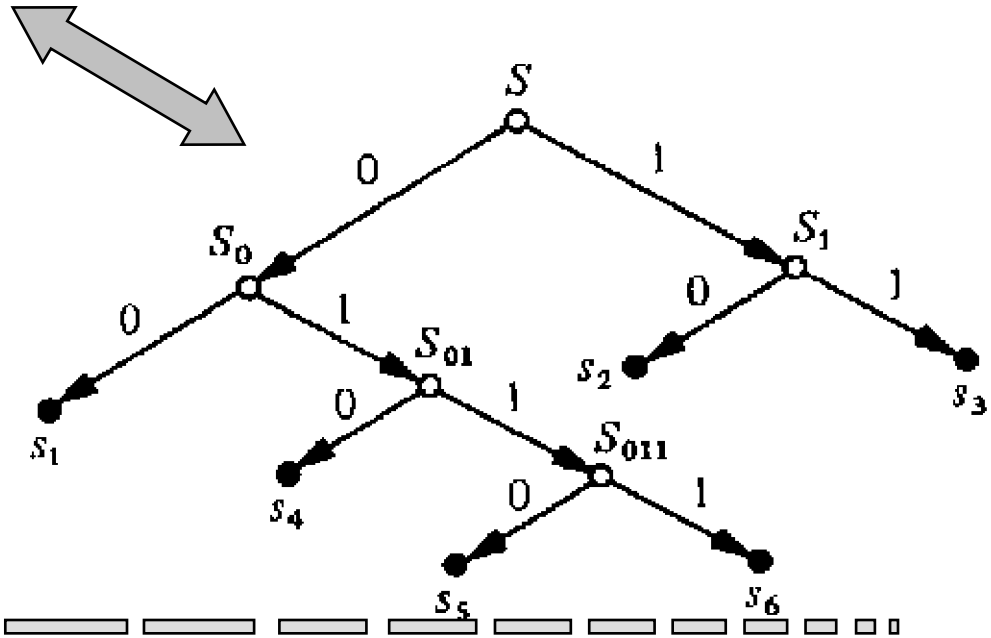
- Algorithme

  ① Extraction des probabilités
  ② Création de l'arbre
  ③ Création de la table d'Huffman
  ④ Codage

⇨On transmet la table + les codes en binaire ⇨

  ❶ Lecture de la table d'Huffman
  ❷ Création de l'arbre de décodage
  ❸ Lecture séquentielle et décodage

Messages s_k $[R_0]$	Probabilités $p(s_k)$	Mots-code c_k	Sources restreintes $[R_1]$		$[R_2]$		$[R_3]$		$[R_4]$	
							0,40	(1)	0,60	(0)
									0,40	(1)
s_1	0,30	00	0,30	(00)	0,30	(00)	0,30	(00)		
					0,30	(01)	0,30	(01)		
s_2	0,25	10	0,25	(10)	0,25	(10)				
s_3	0,15	11	0,15	(11)	0,15	(11)				
s_4	0,15	010	0,15	(010)						
			0,15	(011)						
s_5	0,10	0110								
s_6	0,05	0111								



Rq : code d'échappement
= Huffman + fixe

- **Codage Arithmétique** (1976)
  - Huffman ⇨ 1 symbole = 1 mot-code
  - Arithmétique ⇨ 1 flot de symboles = nbre en virgule flottante

❒ **Codeur**

m=0 ; M=1 ;
Tant que !(fin de fichier)
{
i = symbole suivant;
soit $[a_i ; b_i]$ associé à i ;
s = M-m ;
$M = m + s.b_i$;
$m = m + s.a_i$;
}
Renvoyer m, le compacté du fichier

❒ **Decodeur**

N = nombre codé ;
Faire
{
trouver i / $N \in [a_i ; b_i[$;
sortir i ;
$s = b_i - a_i$;
$N = (N - a_i) / s$;
}
Tant qu'il reste un symbole à lire

- Exemple

s_i	p_i	$[a_i ; b_i[$	$Huff_i$
␣	0.1	[0.0 ; 0.1[	111
A	0.1	[0.1 ; 0.2[	110
E	0.1	[0.2 ; 0.3[	101
I	0.1	[0.3 ; 0.4[	100
B	0.1	[0.4; 0.5[	0111
G	0.1	[0.5 ; 0.6[	0110
L	0.2	[0.6 ; 0.8[	00
S	0.1	[0.8; 0.9[	0100
T	0.1	[0.9 ; 1.0[	0101

0.4372207712 = ?

10111010 10100100 11011001 01
01111000 00011101 10110010 11010100

Arithmétique	≥	Huffman

☹ + de calcul

Proba très élevée →1 bit
Peu de symboles (↓)
⇩
Run Length

Codeurs statistiques

- Dépendants de la qualité de la statistique
- Statistique connue par le décodeur

- **Codage par longueur de plage** (Run length coding)

  ⇨Coder le nombre de symboles identiques

00000111110000000000000000	⇨	5w5b17w
00000000000111110000000000	⇨	11w5b11w
A B C C C C C C A B C A B C	⇨	A B !6C A B C A B C

- CCITT, Fax groupe III

  ↳ Huffman sur les plages de 0 précédant les 1

- JPEG

  ↳ Huffman sur les plages de 0 précédant les coeff. DCT

- Table d'Huffman FAX III

0	00110101
1	000111
2	0111
3	1000
4	1011
5	1100
6	1110
7	1111
8	10011
9	10100
10	00111
11	01000
12	001000
13	000011
14	110100
15	110101
16	101010
17	101011
18	0100111
19	0001100
20	0001000
21	0010111
22	0000011
23	0000100
24	0101000
25	0101011
26	0010011
27	0100100
28	0011000
29	00000010
30	00000011
31	00011010

32	00011011
33	00010010
34	00010011
35	00010100
36	00010101
37	00010110
38	00010111
39	00101000
40	00101001
41	00101010
42	00101011
43	00101100
44	00101101
45	00000100
46	00000101
47	00001010
48	00001011
49	01010010
50	01010011
51	01010100
52	01010101
53	00100100
54	00100101
55	01011000
56	01011001
57	01011010
58	01011011
59	01001010
60	01001011
61	00110010
62	00110011
63	00110100

64	11011
128	10010
192	010111
256	0110111
320	00110110
384	00110111
448	01100100
512	01100101
576	01101000
640	01100111
704	011001100
768	011001101
832	011010010
896	011010011
960	011010100
1024	011010101
1088	011010110
1152	011010111
1216	011011000
1280	011011001
1344	011011010
1408	011011011
1472	010011000
1536	010011001
1600	010011010
1664	011000
1728	010011011
EOL	000000000001

Table 6.5 - Mots de code spécifiant la longueur des séquences correspondant au blanc lors de la transmission de documents par télécopie.

- **Codage de type dictionnaire** (1977)

⇨ Coder une extension de la source de longueur variable

1977 : LZ (Lempel & Ziv) ⇨ 1984 : LZW (Welch)

✓**Dictionnaire** de symboles incrémenté dynamiquement
↳ **apprentissage**

✓ Fichier codé = suite des adresses des mots du dico

! Gérer l'incrément des bits d'adresse

PKZIP, ARJ ⇦ LZW + Huffman

❒ Codeur LZW

$\mathbb{D} = \{C_i, W_i\}$, $P = \varnothing$

Tant que (symboles à coder)
    C = symbole suivant
    Si $P \oplus C \in \mathbb{D}$
        $P = P \oplus C$
    Sinon
        sortir W_P
        $P \oplus C \rightarrow \mathbb{D}$
        P=C
    Fin si
Fin tant que

sortir W_P

A B B A B A B A C ….

❒ Décodeur LZW

$\mathbb{D} = \{C_i, W_i\}$
cW = 1^{er} code ; sortir s(cW)

Tant que (codes à lire)
    pW = cW
    cW = code suivant
    Si (s(cW) $\in \mathbb{D}$)
        sortir s(cW)
        P = s(pW)
        C = 1^{er} symbole de s(cW)
        $P \oplus C \rightarrow \mathbb{D}$
    Sinon
        P = s(pW)
        C = 1^{er} symbole de s(pW)
        sortir $s(P \oplus C)$
        $P \oplus C \rightarrow \mathbb{D}$
    Fin si
Fin tant que

- **Conclusion sur le codage de source**

Utilisé en **compression audio & vidéo** (JPEG, MPEG ...) mais en étant associé à des algorithmes non réversibles (avec pertes)

Supprime la redondance

⇨ Sensibilité au bruit

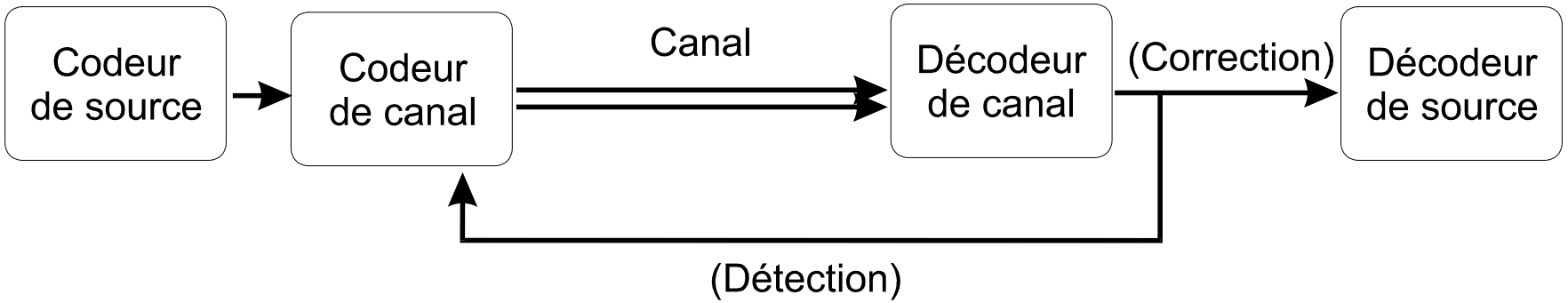
⇨ Codage de canal

Plan

- 1. Introduction
- 2. Sources discrètes & Entropie
- 3. Canaux discrets & Capacité
- 4. Codage de source
- 5. Codage de canal
- 6. Cryptographie
- 7. Conclusion

5. Codage de canal

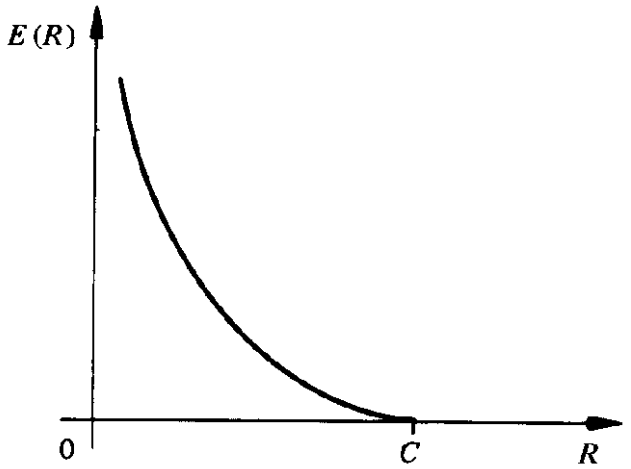
✓**Détecter** et/ou **corriger** les erreurs de transmission



Codeur de canal ⇨ **introduire** une **redondance** utilisable

- **Théorème des canaux à perturbation (codage de canal)**

" Pour une source à débit d'information de R bit/s et un canal de capacité C bit/s, si R<C, il existe un code ayant des mots de longueur n, de sorte que la probabilité d'erreur de décodage p_E vérifie : $p_E \leq 2^{-n.E(R)}$ "



Rq1 : un résultat inattendu !

Rq2 : existance ss méthode ...

Rq3 : à p_E constant, n augmente si R tend vers C.

Rq4 : en pratique, si R<0.5 C, des codes existent avec p_E faible.

- **Taux d'erreur**

$$T_e = \frac{Nombre\ de\ bits\ erronés}{Nombre\ de\ bits\ transmis}$$

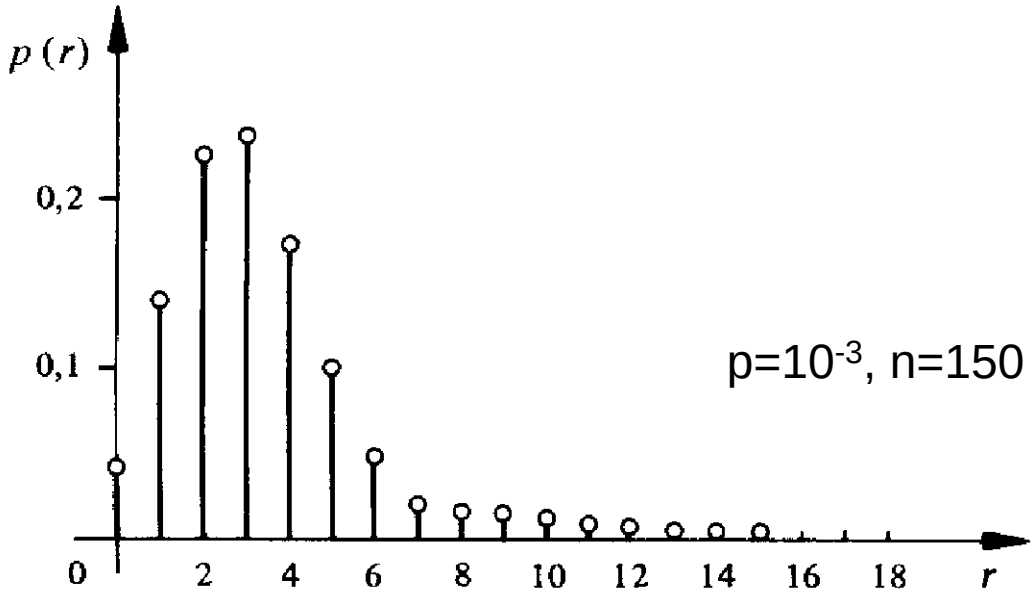
01100100100110010010100 1010 ⇨ 01100110110010110100 0010

↳ $T_e = 3/24 = 0.125$

- **Probabilité d'erreur**

$$P_{n\,bits\,corrects} = (1-p)^n$$

$$P_{r\,erreurs/n} = C_n^r . p^r . (1-p)^{n-r}$$



p=10^{-3}, n=150

- **Taux de codage**

$$R = \frac{k}{n}$$

- k taille du mot d ’information (avant codage)
- n taille du mot-code (après codage)

- **Détection et correction d'erreurs**

✓**Détection par écho**

✓**Détection par répétition**

✓**Détection par bit de parité**

✓**Détection par code**

✓**Détection et correction par code**

• Détection d'erreurs par bit de parité (caractère)

✓**VRC (**Vertical Redundancy Check)

⇨ Asynchrone

Caractère	O	S	I
Bit 0	1	1	1
Bit 1	0	0	0
Bit 2	0	1	0
Bit 3	1	0	1
Bit 4	1	0	0
Bit 5	1	1	0
Bit 6	1	1	1
Bit de parité	1	0	1
Bit d'imparité	0	1	0

✓**LRC (**Longitudinal Redundancy Check)

⇨ Synchrone

Caractère à envoyer	Bit de VRC	Caractère à envoyer	Bit de VRC	. . .	Caractère LRC	Bit de VRC

	H	E	L	L	O	LRC →
bit 1	0	1	0	0	1	0
bit 2	0	0	0	0	1	1
bit 3	0	1	1	1	1	0
bit 4	1	0	1	1	1	0
bit 5	0	0	0	0	0	0
bit 6	0	0	0	0	0	0
bit 7	1	1	1	1	1	1
VRC ↓	0	1	1	1	1	0

0001001	0	1010001	1	0011001	1	0011001	1	1111100	1	0100001	0
H		E		L		L		O		LRC	

- **Codes détecteur et/ou correcteur**

  ✓ **Codes linéaires**
  - **Codes groupes**

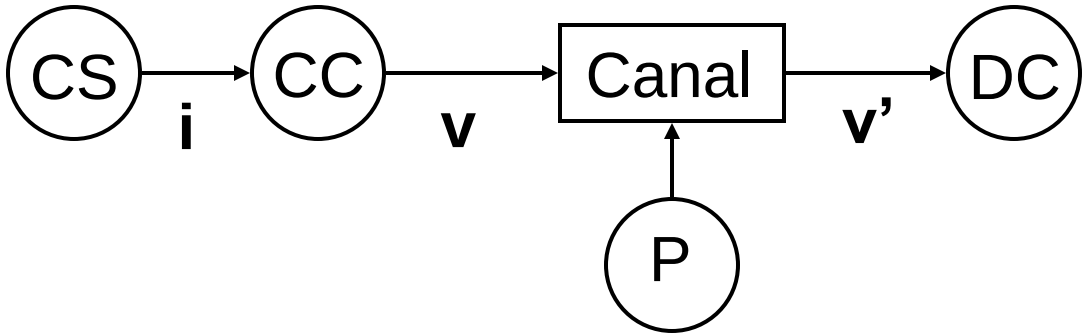
    Parité, Code de Hamming
  - **Codes cycliques**

    CRC/FCS, code BCH, Golay

  ✓ **Codes convolutifs**

    Algorithme de Viterbi

✓ Codes linéaires



- **Notations**
  - Mot-code : v

$$v = [a_1\ a_2\ ....\ a_m\ a_{m+1}\ a_{m+2}\ ....\ a_n] = [c \quad i]$$

[c] : *m* symboles de contrôle
[i] : *k* =*n*-*m* symboles d'information

  - Mot-erreur : ε

$$\varepsilon = [\varepsilon_1 \quad \varepsilon_2\ .....\ \varepsilon_n] \qquad v_i = v_i' + \varepsilon \quad \Leftrightarrow \quad v_i' = v_i + \varepsilon$$

$$\varepsilon_i = \begin{bmatrix} 1 \text{ si erreur à la ième position} \\ 0 \text{ sinon} \end{bmatrix}$$

- **Propriétés des codes linéaires**

  Les symboles de contrôle sont obtenus par une combinaison linéaire des symboles d ’information.

  → un code linéaire contient v=[0 0 …0]

- **Code systématique**

  Les symboles d ’information et de contrôle sont séparés.

- **Distance de Hamming**

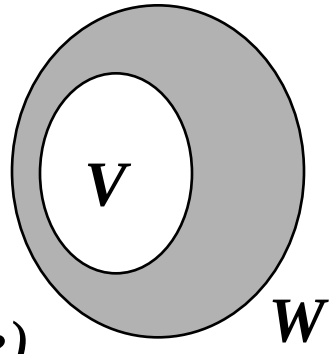
$$D(v_i, v_j) = (a_{i1} \oplus a_{j1}) + (a_{i2} \oplus a_{j2}) + ... + (a_{in} \oplus a_{jn})$$

  ↳ Le nombre de coordonnées par lesquelles les 2 mots diffèrent

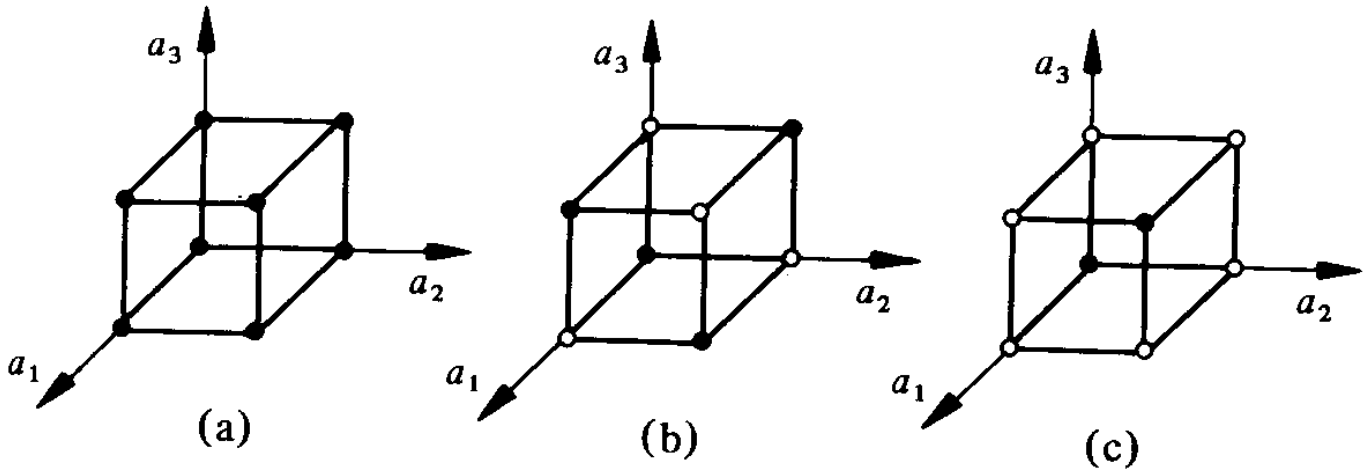
- **Illustration spatiale :** modèle code groupe

  - Un mot = un vecteur dans un espace à n dimensions !

    $w=[a_1\ a_2\ \ldots\ a_n]$



  - *W = ensemble des $N = 2^n$ mots*
  - *V = ensemble des $S = 2^k$ mots ayant un sens (mot-code)*



- **Capacité de détection et région de décision**

$v_i \rightarrow$ Région W_i Région $W_0 \rightarrow$ équidistant

→ Détection et correction ⇨ si W_i grand

Théorème de Hamming

✓ Détecter *d* erreurs → $D_{min} = d+1$

✓Corriger *e* erreurs → $D_{min} = 2e+1$

✓Corriger *e* & détecter *d* erreurs → $D_{min} = 2e + d + 1$

Ex → Hamming(S^4)

- **Principe de détection et correction**

Deux opérateurs : H D

$$\mathrm{H}(v_i) = 0 \text{ pour tout } i = 1 \text{ à } S = 2^k$$

✓ Si $\mathrm{H}(v_i') = 0$ alors $v_i' = v_i \rightarrow$ pas d'erreur

✓ Si $\mathrm{H}(v_i') = z \neq 0 \rightarrow$ détection d'erreur

Si z est connu $\rightarrow D(z) = \varepsilon$

$v_i' + \varepsilon = v_i \rightarrow$ correction d'erreur

- **Décodage et matrice de contrôle**

$$v = [a_1 \quad a_2 \quad \dots \quad a_n]$$

Soit $H_{(m,n)}$ la matrice de contrôle, $[H] = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & & h_{2n} \\ \dots & & & \dots \\ h_{m1} & h_{m2} & \dots & h_{mn} \end{bmatrix}$

Soit z le syndrome (ou correcteur), $z = H.v'^{T} = \begin{bmatrix} z_1 \\ : \\ z_m \end{bmatrix}$

Si z=[0] pas d ’erreur, sinon erreur et +- correction

- **Codage et matrice génératrice**

$$i = [i_1 \quad i_2 \quad \dots \quad i_k]$$

Soit $G_{(k,n)}$ la matrice génératrice, $[G] = \begin{bmatrix} g_{11} & g_{12} & \dots & g_{1n} \\ g_{21} & g_{22} & & g_{2n} \\ \dots & & & \dots \\ g_{k1} & g_{k2} & \dots & g_{kn} \end{bmatrix}$

$$\boxed{v = i.G}$$

Les matrices H et G sont liées par : $G.H^t = 0$

et peuvent se mettre sous la forme systématique

$$G = \left[\begin{array}{c:c:c} & & \\ I_k & : & A_{k,m} \\ & : & \end{array}\right] \qquad H = \left[\begin{array}{c:c:c} & & \\ A^t{}_{k,m} & : & I_m \\ & : & \end{array}\right]$$

- **Exemple** k=2, m=1, n=3

$$[G_1]=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix} \qquad [H]=\begin{bmatrix}1 & 1 & 1\end{bmatrix} \qquad [G_2]=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}$$

$$\begin{bmatrix}0 & 0\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\end{bmatrix} \qquad \begin{bmatrix}0 & 0\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}0 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\end{bmatrix} \qquad \begin{bmatrix}0 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\end{bmatrix} \qquad \begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\end{bmatrix} \qquad \begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\end{bmatrix}$$

• Code de Hamming groupe

⇨ Correction d'une erreur

↳ $\boxed{2^m \geq n+1 \Leftrightarrow 2^m \geq k+m+1}$

✓ $[H] = [h_1 \quad h_2 \quad \ldots \quad h_n] = \begin{bmatrix} 0 & 0 & \ldots & \\ : & : & : & \ldots \\ 0 & 1 & 1 & \ldots \\ 1 & 0 & 1 & \ldots \end{bmatrix}$ avec $h_i = bin(i)$

✓ Mot-erreur : $\varepsilon = [\ldots. \ \alpha_i \ \ldots.]$

$$v'_j = v_j + \varepsilon \Leftrightarrow z = H.v'_j = H.\varepsilon^T \Leftrightarrow z = h_i$$

↳ L'erreur est à la position $dec(h_i)$

Ex → Hamming

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

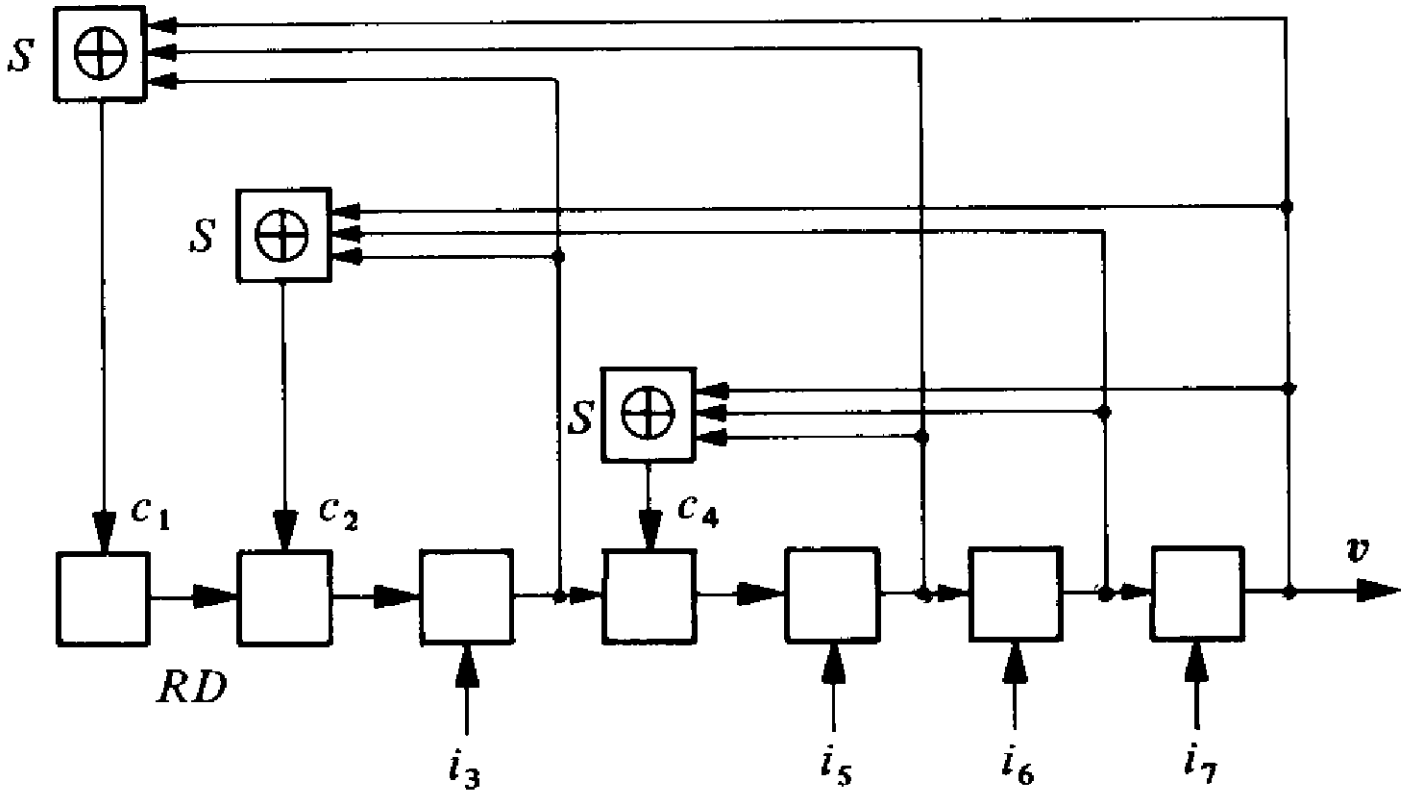
$$v = \begin{bmatrix} c_1 & c_2 & i_3 & c_4 & i_5 & i_6 & i_7 \end{bmatrix}$$

$$H.v^T = 0$$

$$\Downarrow$$

$$\begin{cases} c_1 = i_3 + i_5 + i_7 \\ c_2 = i_3 + i_6 + i_7 \\ c_4 = i_5 + i_6 + i_7 \end{cases}$$

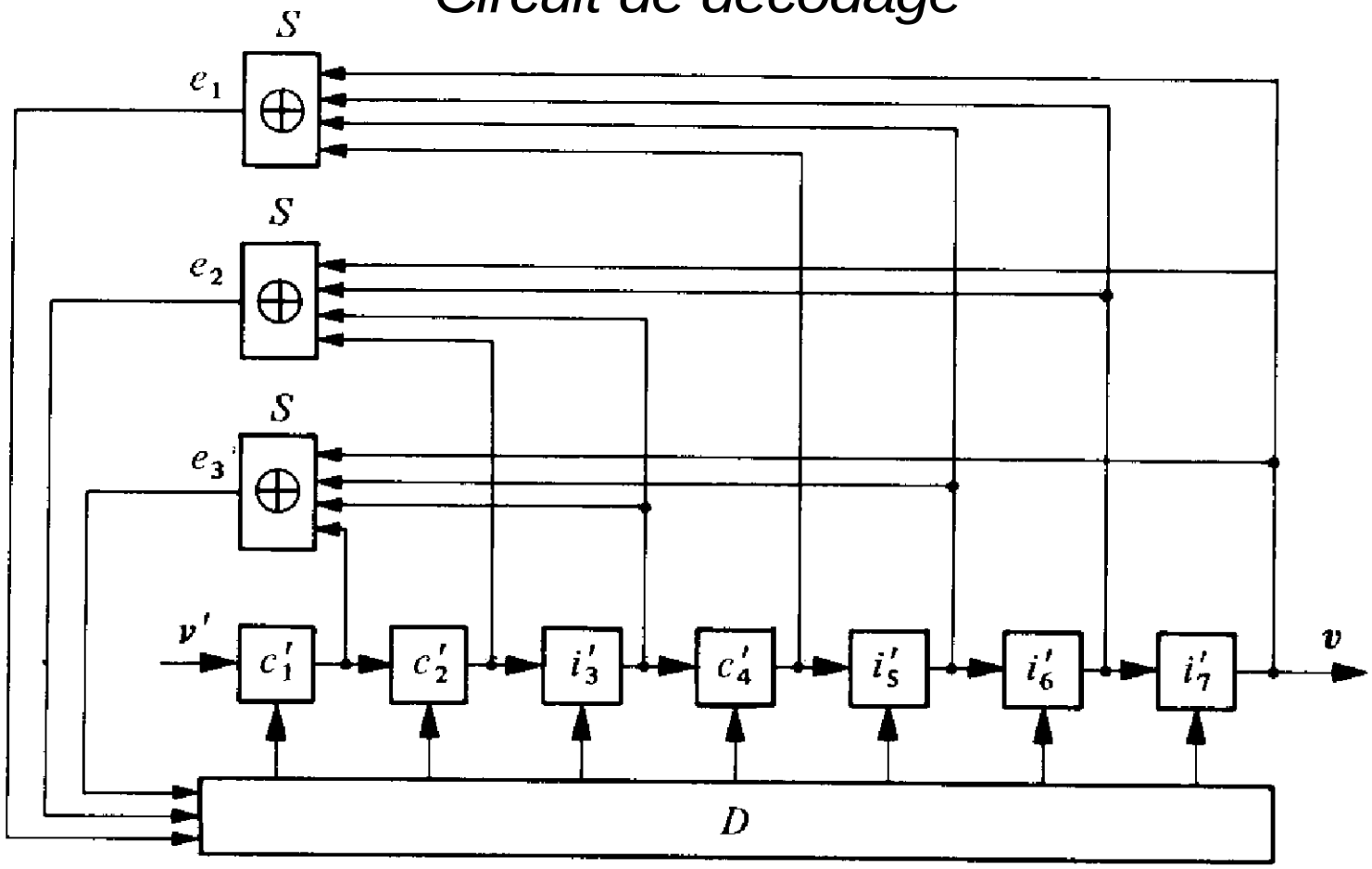
Circuit de codage



$$\begin{cases} e_3 = c'_1 + i'_3 + i'_5 + i'_7 \\ e_2 = c'_2 + i'_3 + i'_6 + i'_7 \\ e_1 = c'_4 + i'_5 + i'_6 + i'_7 \end{cases}$$

$$\varepsilon_i = 1 \quad \text{pour} \quad i = e_3.2^0 + e_2.2^1 + e_1.2^2$$

Circuit de décodage



✓ **Codes cycliques** (Cyclic Redundancy Check / Frame Check Sequence)

- Code cyclique = code linéaire + propriété de permutation
- Bloc de n symboles → **polynôme** de degré n-1 ! :

- Mot-code : $v = [a_0\ a_1\ ....\ a_{n-1}]$ $\quad v(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}$
- Information : $i = [i_0\ i_1\ ....i_{k-1}]$ $\quad i(x) = i_0 + i_1 x + i_2 x^2 + ... + i_{k-1} x^{k-1}$

$$[1 \quad 0 \quad 1 \quad 1] \leftrightarrow 1 + x^2 + x^3$$

- **Polynôme générateur : *g(x)***

  - g(x) définit le codeur (n,k)

  - g(x) est de degré m=n-k

  - Il vérifie : $1+x^n = g(x) \times p(x)$

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k} x^{n-k}$$

avec $g_{n-k} = g_m = 1$

et souvent $g_0 = 1$

Exemple : code cyclique (n=7, k=4)

$$1+x^7 = (1+x) \times (1+x^2+x^3) \times (1+x+x^3)$$

g(x) est de degré 3 soit :

$$g(x) = (1+x^2+x^3) \quad \text{ou} \quad g(x) = (1+x+x^3)$$

- **Matrice génératrice et polynôme générateur**

$$G_{(k,n)} = \begin{bmatrix} g(x) \\ x.g(x) \\ \dots \\ x^{k-1}.g(x) \end{bmatrix}$$

Exemple : $g(x)=(1+x^2+x^3)$

$$G_{(4,7)} = \begin{bmatrix} 1 & 0 & 1 & 1 & . & . & . \\ . & 1 & 0 & 1 & 1 & . & . \\ . & . & 1 & 0 & 1 & 1 & . \\ . & . & . & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$G^s_{(4,7)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$H^s_{(3,7)} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- **Codage par multiplication** $\boxed{v(x) = i(x) \times g(x)}$

$$g(x) = 1 + x + x^3 \quad \text{et} \quad i(x) = x + x^2 + x^3 \quad \rightarrow \quad v(x) = x + x^5 + x^6$$

$$[0 \quad 1 \quad 1 \quad 1] \times [1 \quad 1 \quad 0 \quad 1] = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1]$$

\# convolution discrète !

- **Codage par division**

$$v(x) = c(x) + x^m . i(x)$$

Systématique !

$$\boxed{c(x) = Reste\left(\frac{x^m . i(x)}{g(x)}\right)}$$

- **Décodage par division**

$$\boxed{z(x) = Reste\left(\frac{v'(x)}{g(x)}\right)}$$

Si *z(x)=0* → Transmission OK
Sinon → Détection ou correction

Ex →

- **Exemple de polynômes générateurs**

✓ATM

- $x^8 + x^2 + x + 1$ → Cellule ATM
- $x^{10} + x^9 + x^5 + x^4 + x + 1$ → Couche AAL type 3/4

✓CCITT N°41 → X25 (HDLC)

- $x^{16} + x^{12} + x^5 + 1$

✓IEEE 802 → Réseaux locaux

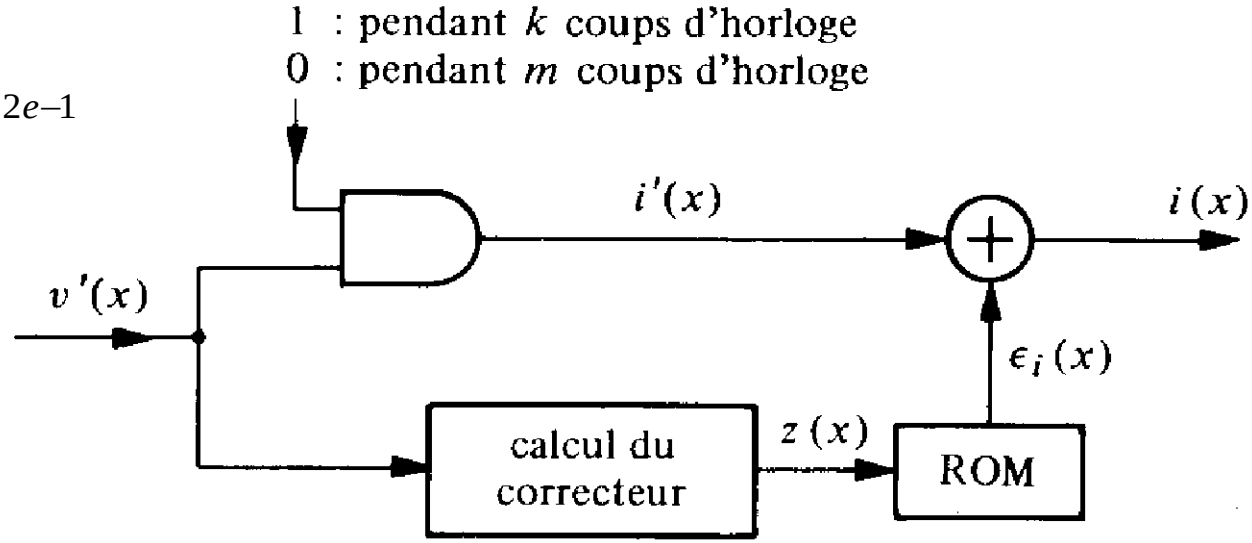
- $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$

- **Code BCH** (Bose-Chaudhuri - Hocquenghem)

  ⇨ Correction de e erreurs

$$g(x) = \prod_{i=1}^{r} m_i(x)$$

$$\beta_1 = \alpha, \beta_2 = \alpha^3, ..., \beta_r = \alpha^{2e-1}$$



- Exemple

  n=15 et e=3

  ↳ $g(x) = (1+x+x^4)(1+x+x^2+x^3+x^4)(1+x+x^2)$

  ↳ m=10

- **Code Golay**

⇨ Correction de *e* erreurs parfait

⇨ Nb correcteurs = Nb mots-erreur

e → Nombre d'erreurs à corriger = $2^m - 1$

⇩

$g(x)$ polynôme minimal de degré m

- Exemple

n=23 et e=3

↳ m=11 , k=12

↳ $g(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$

✓Codes convolutifs

- **Généralités**

  ⇨ Les symboles d'information sont traités en flux continu

- Rque :Blocs de n_0 symboles, mais dont les m_0 contrôleurs ne dépendent pas que des k_0 symboles d'information !

- Contrainte : m = nb de blocs contrôlés par un bloc donné
- Longueur de contrainte : $n=m.n_0$

- Taux d'émission : $R = \frac{k_0}{n_0}$

- **Codes convolutifs systématiques**

  - Mot-code : $V = \left[X_1 Y_1 X_2 Y_2 ..... X_j Y_j ....._1 \right]$

    avec $X_j = \left[X_j^1 ..... X_j^{k_0} \right]$ Information

    $Y_j = \left[Y_j^1 ..... Y_j^{m_0} \right]$ Contrôle

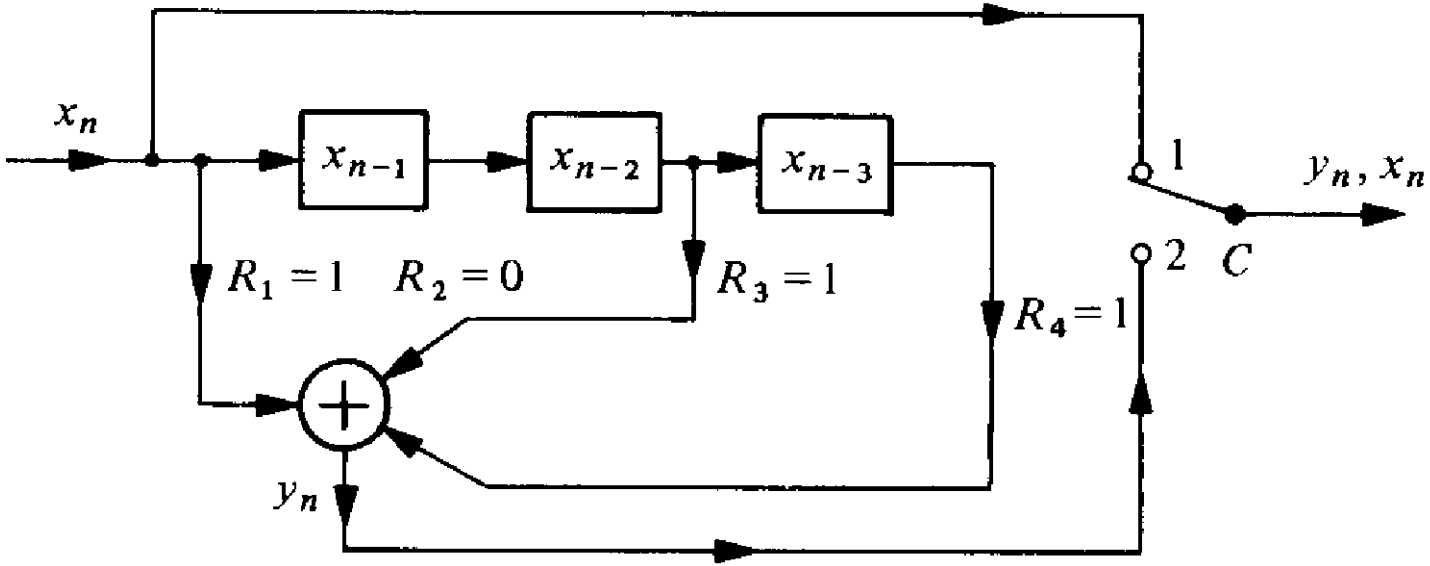
- **Codes convolutifs non systématiques**

  ⇨ Contrôle et information sont mélangés

  - Mot-code : $V = \left[U_1 U_2 ..... U_j ..... \right]$

- Exemple : m=4, k_0=1, m_0=1, n_0=2

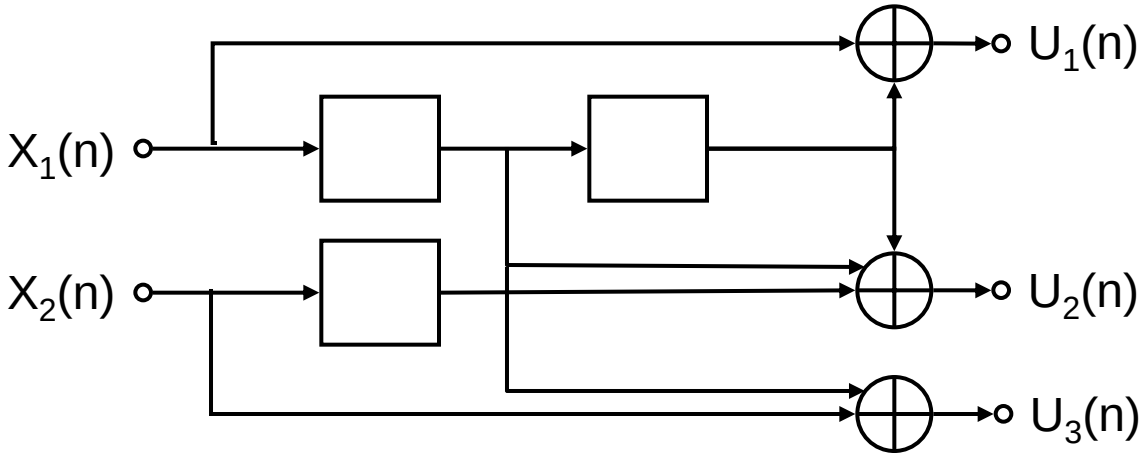
$$y_n = R_4.x_{n-3} + R_3.x_{n-2} + R_2.x_{n-1} + R_1.x_n$$



⇨ *R*=[1011]

- **Représentation des codes convolutifs**

- Par le codeur



- Par une matrice de transfert

$$G_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$G = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

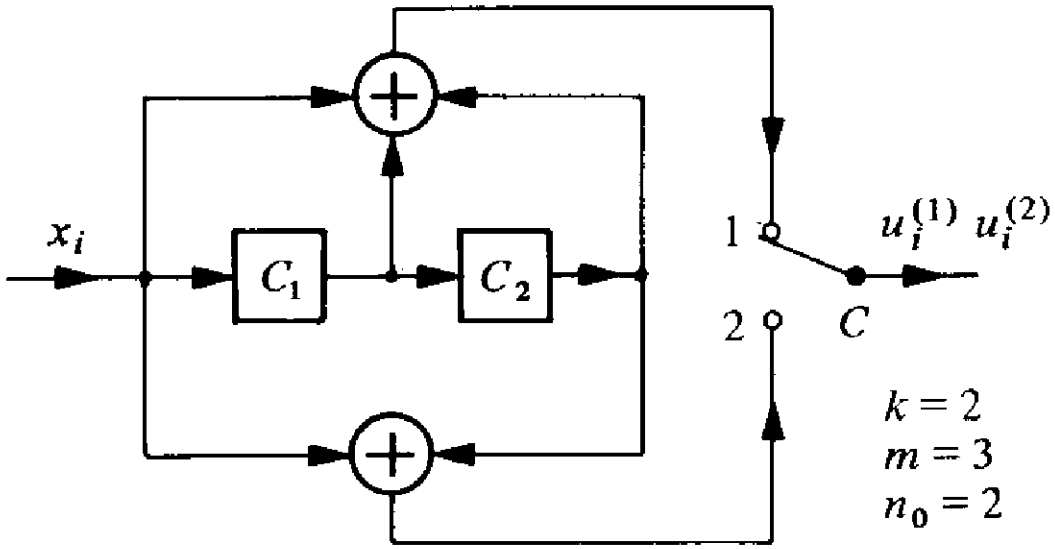
- Un diagramme d'état

- Un treillis → chemin → décodage par chemin le + probable

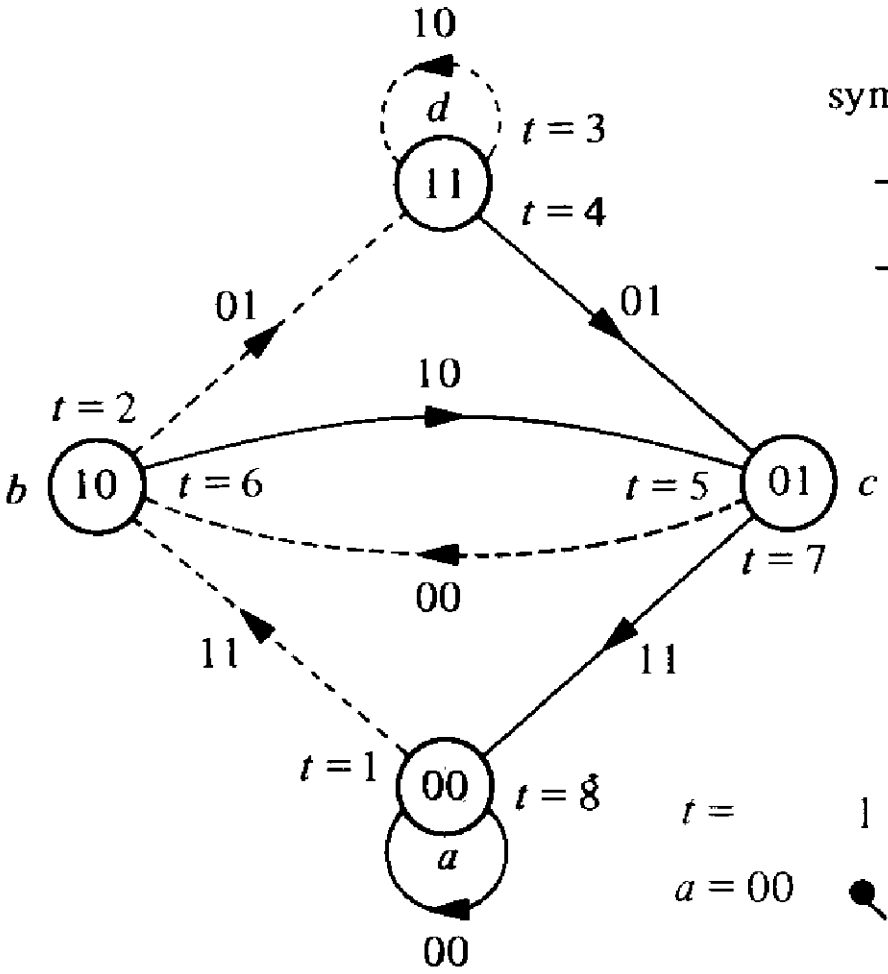
- Exemple : n_0=2, R=0.5 , m=3

$$U_n^{(1)} = x_n + x_{n-1} + x_{n-2}$$

$$U_n^{(2)} = x_n + x_{n-2}$$



t_i	1	2	3	4	5	6	7	8
x_i	1	1	1	0	1	0	0	0
$C_1 C_2$	00	10	11	11	01	10	01	00
$u_i^{(1)} u_i^{(2)}$	11	01	10	01	00	10	11	00



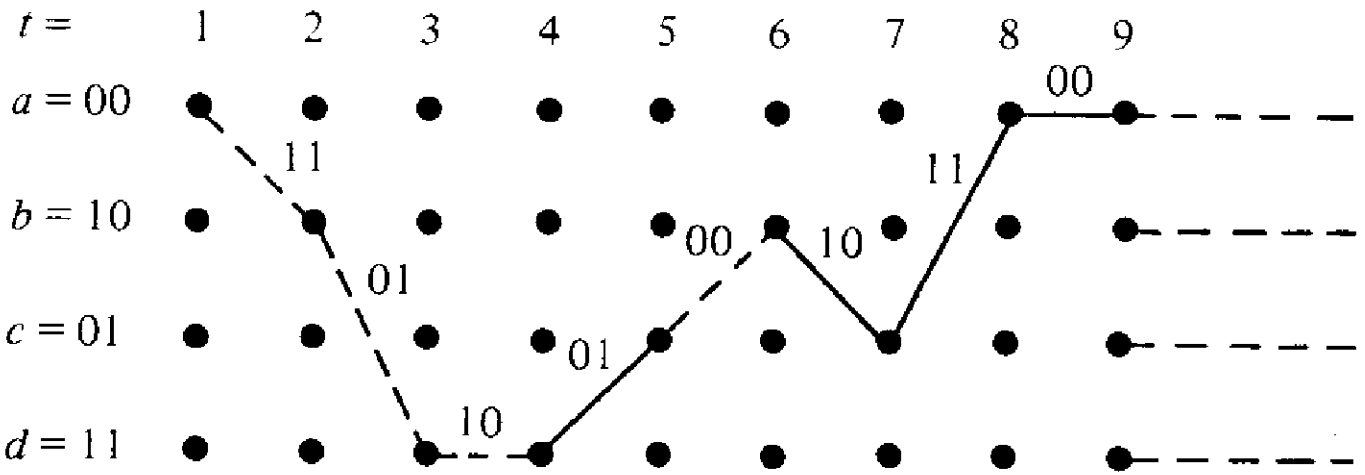
symbole entrant :

——— 0 :

- - - - 1 :

✓Recherche d'erreur à la fréquence N

→ Dmin = 2e+1



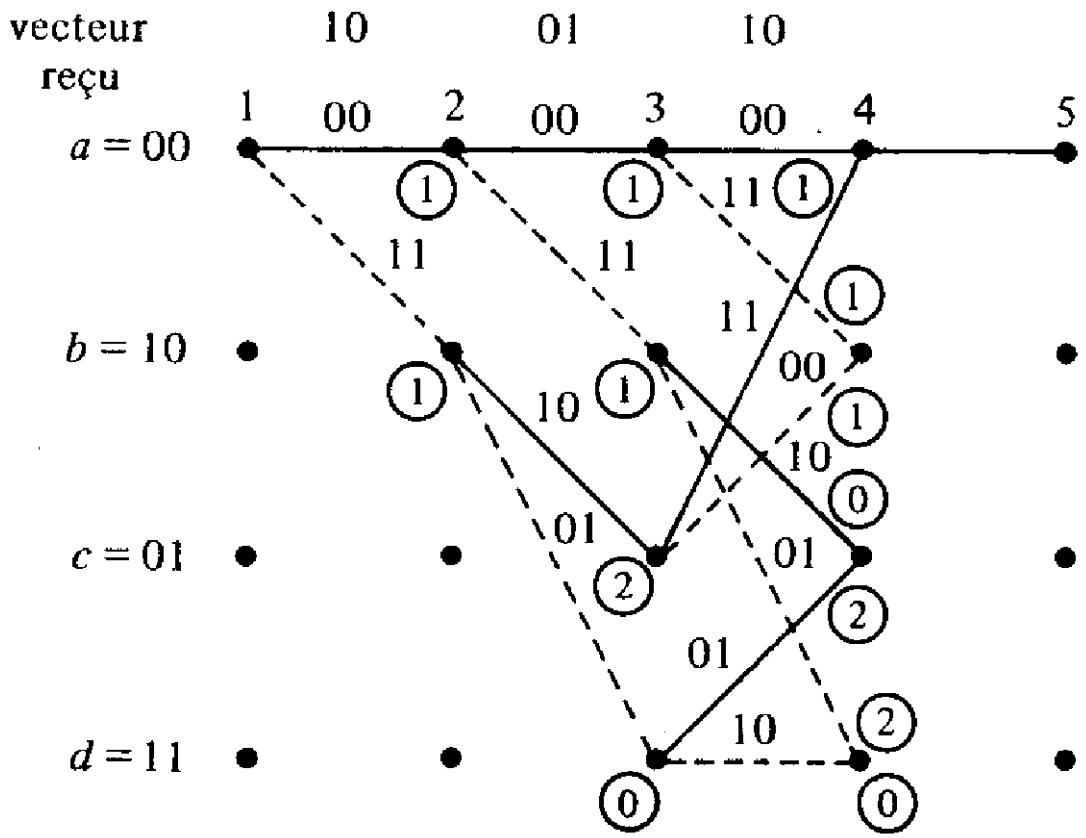
- **Décodage : algorithme de Viterbi**

⇨ Stratégie de recherche de D_{min}

✓Exemple pour N=3

10 01 10 → $Min\left(\sum_{i=1}^{3} d_i\right) = ?$

→ 1<u>1</u> 01 10



• Conclusion sur le codage de canal

✓ Indispensable

✓ Théories mathématiques complexes → des solutions concrètes

- Reed-Solomon (1984) : BCH Qaire → DVB(204,188,8)
- Turbo-Codes (1993) : Code convolutif + brassage

✓ Recherche de codeurs conjoint source / canal

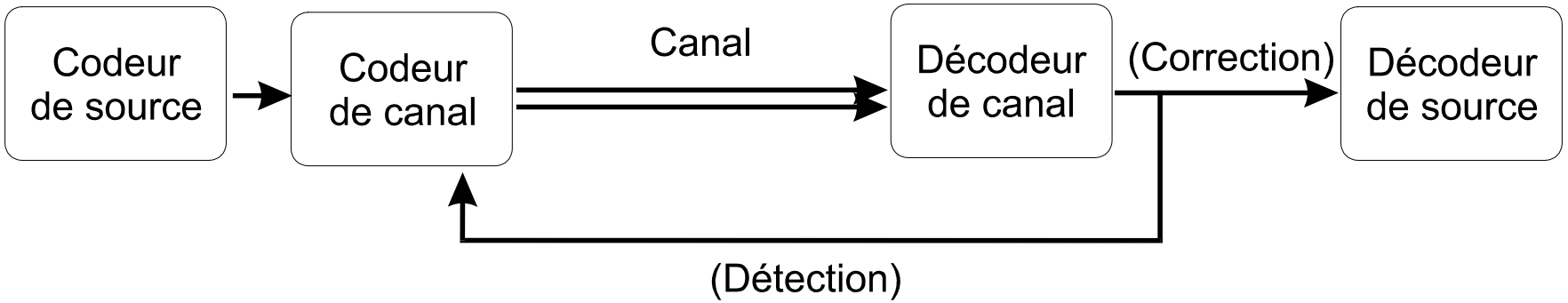
- complexité --
- robustesse ++
- flexibilité ++

Plan

- 1. Introduction
- 2. Sources discrètes & Entropie
- 3. Canaux discrets & Capacité
- 4. Codage de source
- 5. Codage de canal
- 6. Cryptographie
- 7. Conclusion

5. Codage de canal

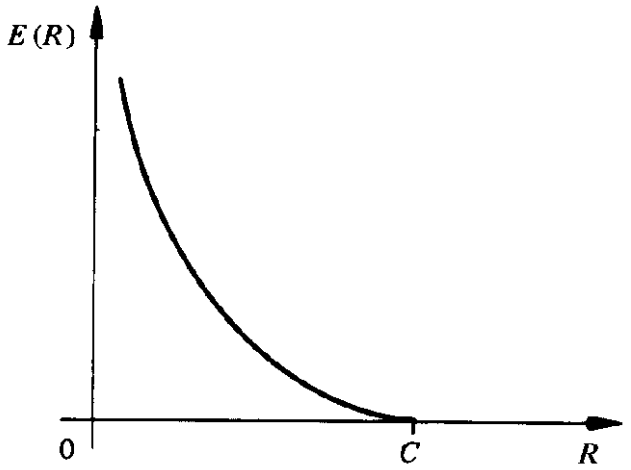
✓**Détecter** et/ou **corriger** les erreurs de transmission



Codeur de canal ⇨ **introduire** une **redondance** utilisable

- **Théorème des canaux à perturbation (codage de canal)**

" Pour une source à débit d'information de R bit/s et un canal de capacité C bit/s, si R<C, il existe un code ayant des mots de longueur n, de sorte que la probabilité d'erreur de décodage p_E vérifie : $p_E \leq 2^{-n.E(R)}$ "



Rq1 : un résultat inattendu !

Rq2 : existance ss méthode ...

Rq3 : à p_E constant, n augmente si R tend vers C.

Rq4 : en pratique, si R<0.5 C, des codes existent avec p_E faible.

- **Taux d'erreur**

$$T_e = \frac{Nombre\ de\ bits\ erronés}{Nombre\ de\ bits\ transmis}$$

0110010010011001001010010 ⇨ 0110011011001011010000010

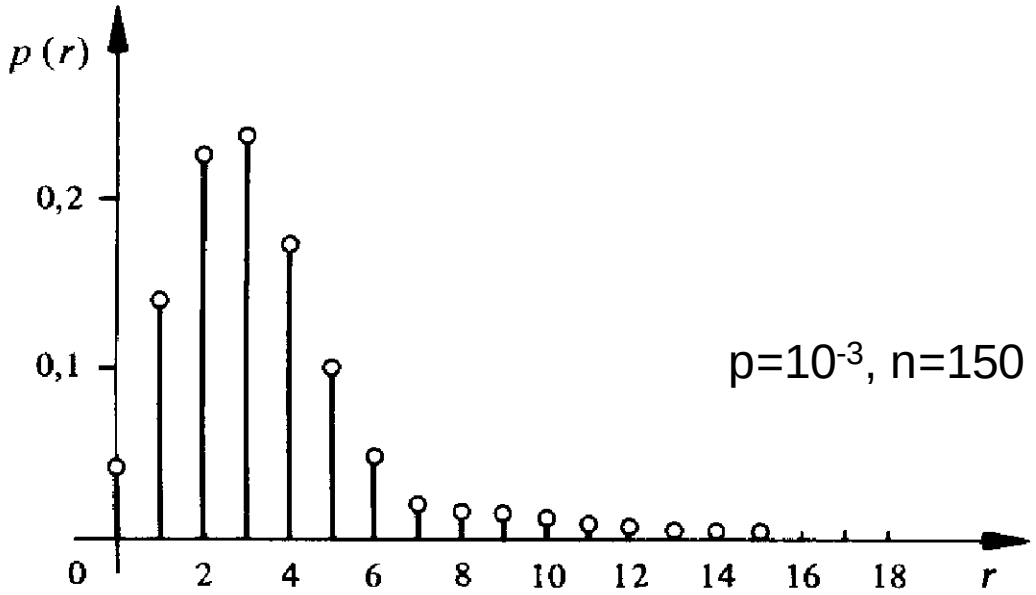
↳ $T_e = 3/24 = 0.125$

- **Probabilité d'erreur**

$$P_{n\,bits\,corrects} = (1-p)^n$$

$$P_{r\,erreurs/n} = C_n^r . p^r . (1-p)^{n-r}$$

$p(r)$



p=10^{-3}, n=150

- **Taux de codage**

$$R = \frac{k}{n}$$

- k taille du mot d ’information (avant codage)
- n taille du mot-code (après codage)

- **Détection et correction d'erreurs**

✓**Détection par écho**

✓**Détection par répétition**

✓**Détection par bit de parité**

✓**Détection par code**

✓**Détection et correction par code**

• Détection d'erreurs par bit de parité (caractère)

✓**VRC (**Vertical Redundancy Check)

⇨ Asynchrone

Caractère	O	S	I
Bit 0	1	1	1
Bit 1	0	0	0
Bit 2	0	1	0
Bit 3	1	0	1
Bit 4	1	0	0
Bit 5	1	1	0
Bit 6	1	1	1
Bit de parité	1	0	1
Bit d'imparité	0	1	0

✓**LRC (**Longitudinal Redundancy Check)

⇨ Synchrone

Caractère à envoyer	Bit de VRC	Caractère à envoyer	Bit de VRC	. . .	Caractère LRC	Bit de VRC

	H	E	L	L	O	LRC →
bit 1	0	1	0	0	1	0
bit 2	0	0	0	0	1	1
bit 3	0	1	1	1	1	0
bit 4	1	0	1	1	1	0
bit 5	0	0	0	0	0	0
bit 6	0	0	0	0	0	0
bit 7	1	1	1	1	1	1
VRC ↓	0	1	1	1	1	0

0001001	0	1010001	1	0011001	1	0011001	1	1111100	1	0100001	0
H		E		L		L		O		LRC	

- **Codes détecteur et/ou correcteur**

  ✓ **Codes linéaires**
  - **Codes groupes**

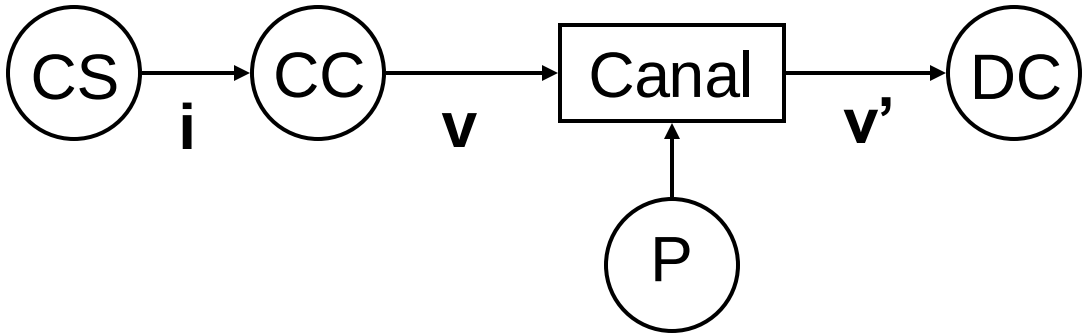
    Parité, Code de Hamming
  - **Codes cycliques**

    CRC/FCS, code BCH, Golay

  ✓ **Codes convolutifs**

    Algorithme de Viterbi

✓ Codes linéaires



- **Notations**
  - Mot-code : v

$$v = [a_1\ a_2\ ....\ a_m\ a_{m+1}\ a_{m+2}\ ....\ a_n] = [c \quad i]$$

[c] : *m* symboles de contrôle
[i] : *k* =*n*-*m* symboles d'information

  - Mot-erreur : ε

$$\varepsilon = [\varepsilon_1\ \ \varepsilon_2\ .....\ \varepsilon_n] \qquad v_i = v_i' + \varepsilon \quad \Leftrightarrow \quad v_i' = v_i + \varepsilon$$

$$\varepsilon_i = \begin{bmatrix} 1 \text{ si erreur à la ième position} \\ 0 \text{ sinon} \end{bmatrix}$$

- **Propriétés des codes linéaires**

  Les symboles de contrôle sont obtenus par une combinaison linéaire des symboles d 'information.

  → un code linéaire contient v=[0 0 …0]

- **Code systématique**

  Les symboles d 'information et de contrôle sont séparés.

- **Distance de Hamming**

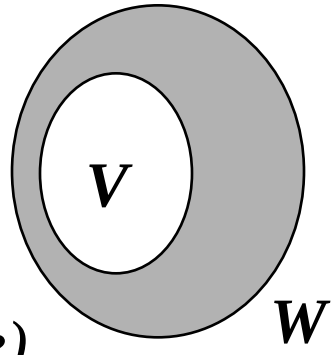
$$D(v_i, v_j) = (a_{i1} \oplus a_{j1}) + (a_{i2} \oplus a_{j2}) + ... + (a_{in} \oplus a_{jn})$$

  ↳ Le nombre de coordonnées par lesquelles les 2 mots diffèrent

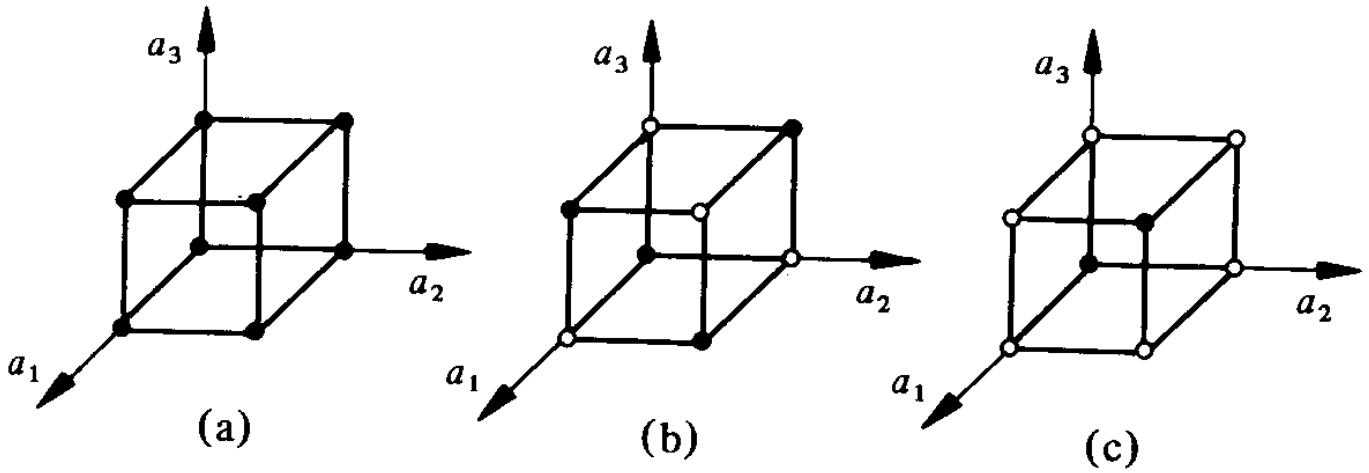
- **Illustration spatiale :** modèle code groupe

  - Un mot = un vecteur dans un espace à n dimensions !

    $w=[a_1\ a_2\ \ldots\ a_n]$



  - *W = ensemble des $N = 2^n$ mots*
  - *V = ensemble des $S = 2^k$ mots ayant un sens (mot-code)*



- **Capacité de détection et région de décision**

$v_i \rightarrow$ Région W_i Région $W_0 \rightarrow$ équidistant

→ Détection et correction ⇨ si W_i grand

Théorème de Hamming

✓ Détecter *d* erreurs → $D_{min} = d+1$

✓Corriger *e* erreurs → $D_{min} = 2e+1$

✓Corriger *e* & détecter *d* erreurs → $D_{min} = 2e + d + 1$

Ex → Hamming(S^4)

- **Principe de détection et correction**

Deux opérateurs : H D

$$\mathrm{H}(v_i) = 0 \text{ pour tout } i = 1 \text{ à } S = 2^k$$

✓ Si $\mathrm{H}(v_i') = 0$ alors $v_i' = v_i \rightarrow$ pas d'erreur

✓ Si $\mathrm{H}(v_i') = z \neq 0 \rightarrow$ détection d'erreur

Si z est connu $\rightarrow D(z) = \varepsilon$

$v_i' + \varepsilon = v_i \rightarrow$ correction d'erreur

- **Décodage et matrice de contrôle**

$$v = [a_1 \quad a_2 \quad \dots \quad a_n]$$

Soit $H_{(m,n)}$ la matrice de contrôle, $[H] = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & & h_{2n} \\ \dots & & & \dots \\ h_{m1} & h_{m2} & \dots & h_{mn} \end{bmatrix}$

Soit z le syndrome (ou correcteur), $z = H.v'^{T} = \begin{bmatrix} z_1 \\ : \\ z_m \end{bmatrix}$

Si z=[0] pas d ’erreur, sinon erreur et +- correction

- **Codage et matrice génératrice**

$$i = [i_1 \quad i_2 \quad \dots \quad i_k]$$

Soit $G_{(k,n)}$ la matrice génératrice, $[G] = \begin{bmatrix} g_{11} & g_{12} & \dots & g_{1n} \\ g_{21} & g_{22} & & g_{2n} \\ \dots & & & \dots \\ g_{k1} & g_{k2} & \dots & g_{kn} \end{bmatrix}$

$$\boxed{v = i.G}$$

Les matrices H et G sont liées par : $G.H^t = 0$

et peuvent se mettre sous la forme systématique

$$G = \begin{bmatrix} & : & \\ I_k & : & A_{k,m} \\ & : & \end{bmatrix} \qquad H = \begin{bmatrix} & : & \\ A^t_{k,m} & : & I_m \\ & : & \end{bmatrix}$$

- **Exemple** k=2, m=1, n=3

$$[G_1]=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix} \qquad [H]=\begin{bmatrix}1 & 1 & 1\end{bmatrix} \qquad [G_2]=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}$$

$$\begin{bmatrix}0 & 0\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\end{bmatrix} \qquad \begin{bmatrix}0 & 0\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}0 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\end{bmatrix} \qquad \begin{bmatrix}0 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\end{bmatrix} \qquad \begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\end{bmatrix} \qquad \begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\end{bmatrix}$$

• Code de Hamming groupe

⇨ Correction d'une erreur

↳ $\boxed{2^m \geq n+1 \Leftrightarrow 2^m \geq k+m+1}$

✓ $[H] = [h_1 \quad h_2 \quad \ldots \quad h_n] = \begin{bmatrix} 0 & 0 & \ldots & \\ : & : & : & \ldots \\ 0 & 1 & 1 & \ldots \\ 1 & 0 & 1 & \ldots \end{bmatrix}$ avec $h_i = bin(i)$

✓ Mot-erreur : $\varepsilon = [\ldots. \ \alpha_i \ \ldots.]$

$$v'_j = v_j + \varepsilon \Leftrightarrow z = H.v'_j = H.\varepsilon^T \Leftrightarrow z = h_i$$

↳ L'erreur est à la position $dec(h_i)$

Ex → Hamming

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

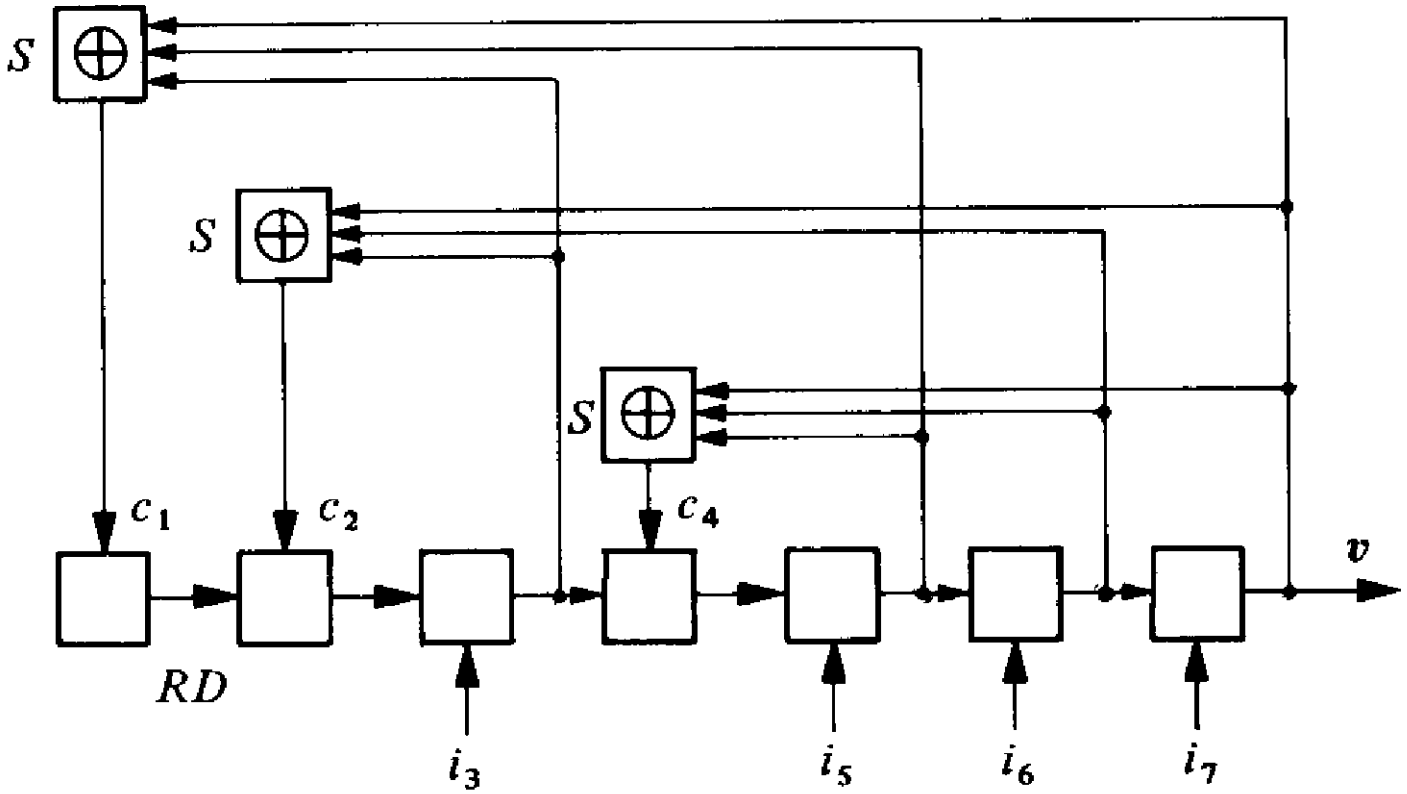
$$v = \begin{bmatrix} c_1 & c_2 & i_3 & c_4 & i_5 & i_6 & i_7 \end{bmatrix}$$

$$H.v^T = 0$$

$$\Downarrow$$

$$\begin{cases} c_1 = i_3 + i_5 + i_7 \\ c_2 = i_3 + i_6 + i_7 \\ c_4 = i_5 + i_6 + i_7 \end{cases}$$

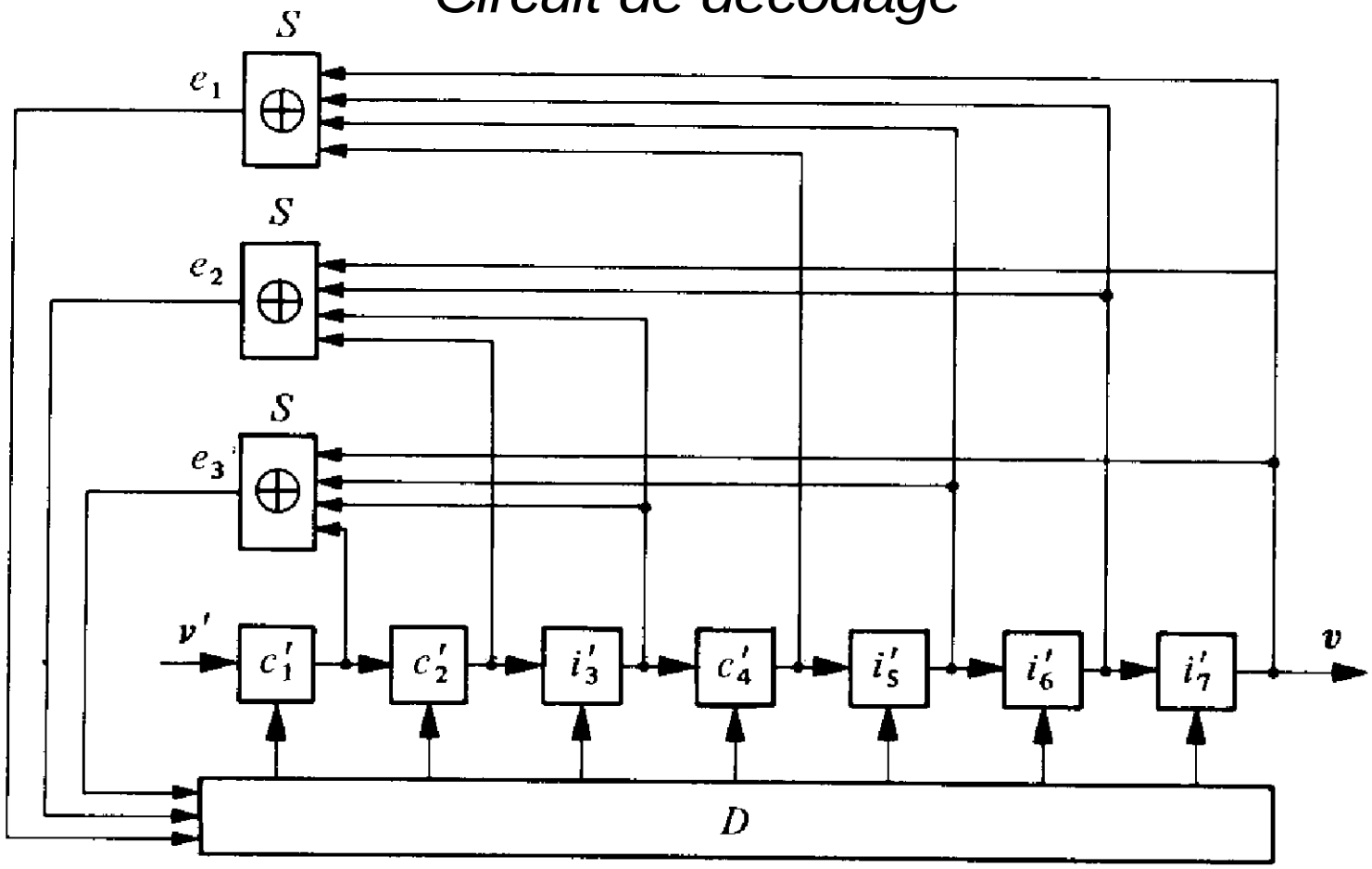
Circuit de codage



$$\begin{cases} e_3 = c'_1 + i'_3 + i'_5 + i'_7 \\ e_2 = c'_2 + i'_3 + i'_6 + i'_7 \\ e_1 = c'_4 + i'_5 + i'_6 + i'_7 \end{cases}$$

$$\varepsilon_i = 1 \quad \text{pour} \quad i = e_3.2^0 + e_2.2^1 + e_1.2^2$$

Circuit de décodage



✓ **Codes cycliques** (Cyclic Redundancy Check / Frame Check Sequence)

- Code cyclique = code linéaire + propriété de permutation
- Bloc de n symboles → **polynôme** de degré n-1 ! :

- Mot-code : $v = [a_0\ a_1\ ....\ a_{n-1}]$ $\quad v(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}$
- Information : $i = [i_0\ i_1\ ....i_{k-1}]$ $\quad i(x) = i_0 + i_1 x + i_2 x^2 + ... + i_{k-1} x^{k-1}$

$$[1 \quad 0 \quad 1 \quad 1] \leftrightarrow 1 + x^2 + x^3$$

- **Polynôme générateur : *g(x)***

  - g(x) définit le codeur (n,k)

  - g(x) est de degré m=n-k

  - Il vérifie : $1+x^n = g(x) \times p(x)$

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k} x^{n-k}$$

avec $g_{n-k} = g_m = 1$

et souvent $g_0 = 1$

Exemple : code cyclique (n=7, k=4)

$$1+x^7 = (1+x) \times (1+x^2+x^3) \times (1+x+x^3)$$

g(x) est de degré 3 soit :

$$g(x) = (1+x^2+x^3) \quad \text{ou} \quad g(x) = (1+x+x^3)$$

- **Matrice génératrice et polynôme générateur**

$$G_{(k,n)} = \begin{bmatrix} g(x) \\ x.g(x) \\ \dots \\ x^{k-1}.g(x) \end{bmatrix}$$

Exemple : $g(x)=(1+x^2+x^3)$

$$G_{(4,7)} = \begin{bmatrix} 1 & 0 & 1 & 1 & . & . & . \\ . & 1 & 0 & 1 & 1 & . & . \\ . & . & 1 & 0 & 1 & 1 & . \\ . & . & . & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$G^s_{(4,7)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$H^s_{(3,7)} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- **Codage par multiplication** $\boxed{v(x) = i(x) \times g(x)}$

$$g(x) = 1 + x + x^3 \quad \text{et} \quad i(x) = x + x^2 + x^3 \quad \rightarrow \quad v(x) = x + x^5 + x^6$$

$$[0 \quad 1 \quad 1 \quad 1] \times [1 \quad 1 \quad 0 \quad 1] = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1]$$

\# convolution discrète !

- **Codage par division**

$$v(x) = c(x) + x^m . i(x)$$

Systématique !

$$\boxed{c(x) = Reste\left(\frac{x^m . i(x)}{g(x)}\right)}$$

- **Décodage par division**

$$\boxed{z(x) = Reste\left(\frac{v'(x)}{g(x)}\right)}$$

Si *z(x)=0* → Transmission OK

Sinon → Détection ou correction

Ex →

- **Exemple de polynômes générateurs**

✓ATM

- $x^8 + x^2 + x + 1$ → Cellule ATM
- $x^{10} + x^9 + x^5 + x^4 + x + 1$ → Couche AAL type 3/4

✓CCITT N°41 → X25 (HDLC)

- $x^{16} + x^{12} + x^5 + 1$

✓IEEE 802 → Réseaux locaux

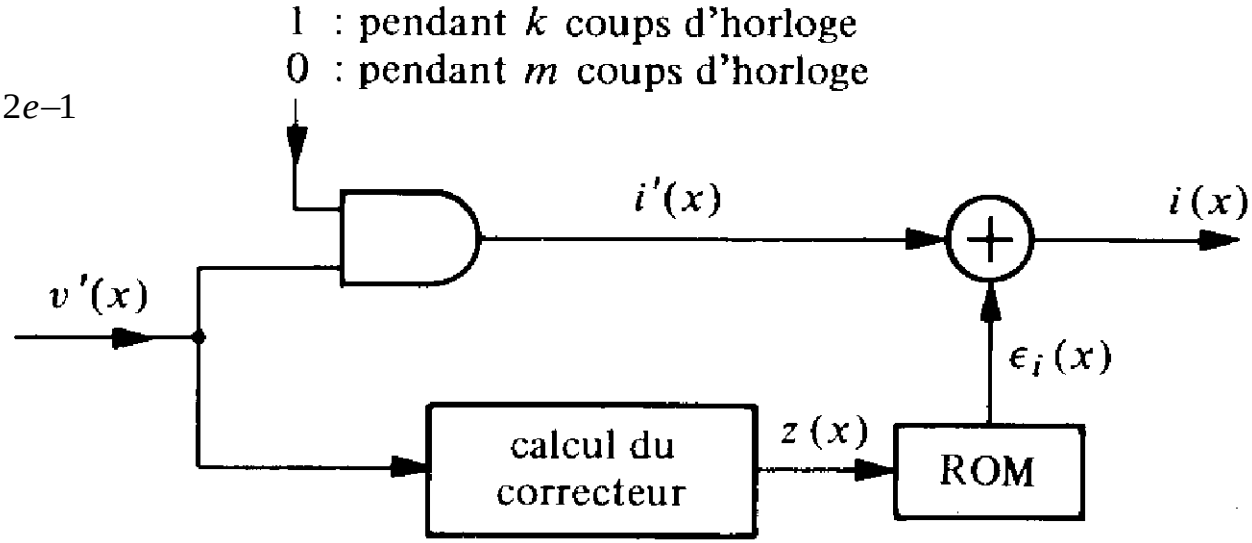
- $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$

- **Code BCH** (Bose-Chaudhuri - Hocquenghem)

  ⇨ Correction de e erreurs

$$g(x) = \prod_{i=1}^{r} m_i(x)$$

$$\beta_1 = \alpha, \beta_2 = \alpha^3, ..., \beta_r = \alpha^{2e-1}$$



- Exemple

  n=15 et e=3

  ↳ $g(x) = (1+x+x^4)(1+x+x^2+x^3+x^4)(1+x+x^2)$

  ↳ m=10

- **Code Golay**

⇨ Correction de *e* erreurs parfait

⇨ Nb correcteurs = Nb mots-erreur

e → Nombre d'erreurs à corriger = $2^m - 1$

⇩

$g(x)$ polynôme minimal de degré m

- Exemple

n=23 et e=3

↳ m=11 , k=12

↳ $g(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$

✓Codes convolutifs

- **Généralités**

  ⇨ Les symboles d'information sont traités en flux continu

- Rque :Blocs de n_0 symboles, mais dont les m_0 contrôleurs ne dépendent pas que des k_0 symboles d'information !

- Contrainte : m = nb de blocs contrôlés par un bloc donné
- Longueur de contrainte : $n=m.n_0$

- Taux d'émission : $R = \frac{k_0}{n_0}$

- **Codes convolutifs systématiques**

  - Mot-code : $V = \left[X_1 Y_1 X_2 Y_2 ..... X_j Y_j ..... _1 \right]$

    avec $X_j = \left[X_j^1 ..... X_j^{k_0} \right]$ Information

    $Y_j = \left[Y_j^1 ..... Y_j^{m_0} \right]$ Contrôle

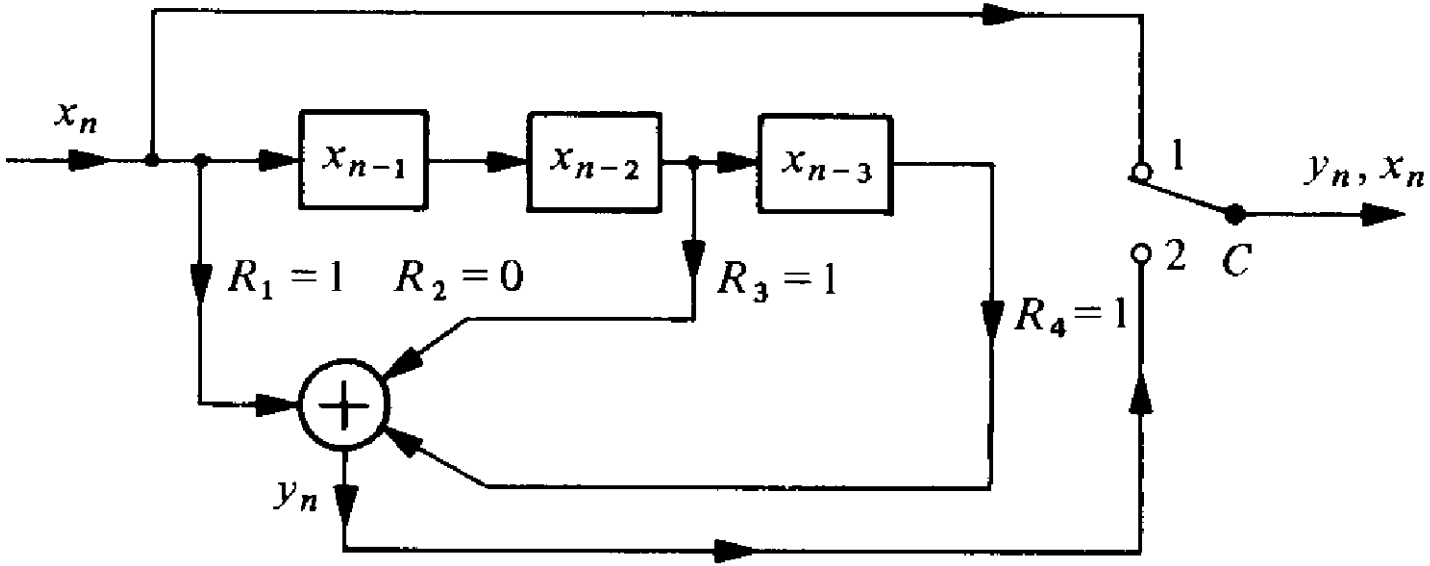
- **Codes convolutifs non systématiques**

  ⇨ Contrôle et information sont mélangés

  - Mot-code : $V = \left[U_1 U_2 ..... U_j ..... \right]$

- Exemple : m=4, k_0=1, m_0=1, n_0=2

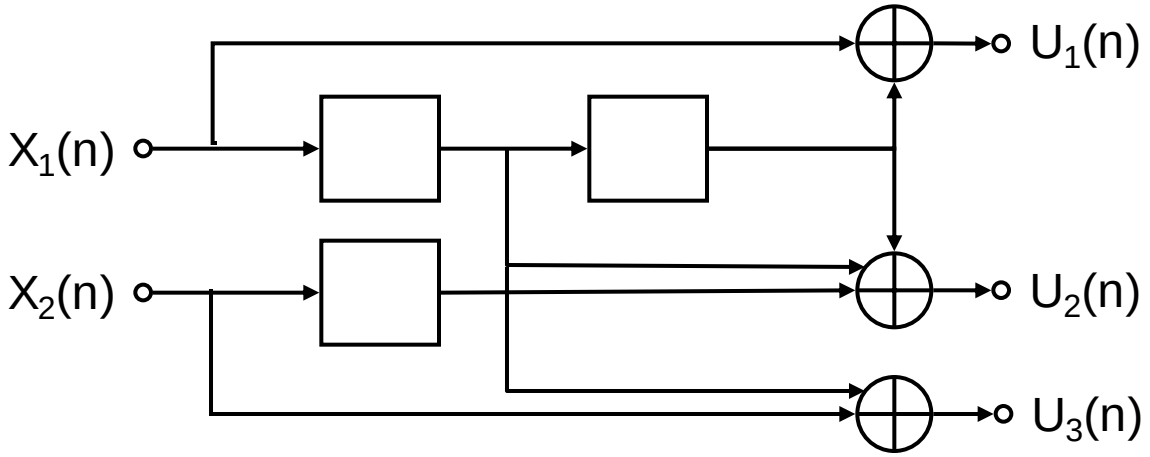
$$y_n = R_4.x_{n-3} + R_3.x_{n-2} + R_2.x_{n-1} + R_1.x_n$$



⇨ R=[1011]

- **Représentation des codes convolutifs**

- Par le codeur



- Par une matrice de transfert

$$G_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$G = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

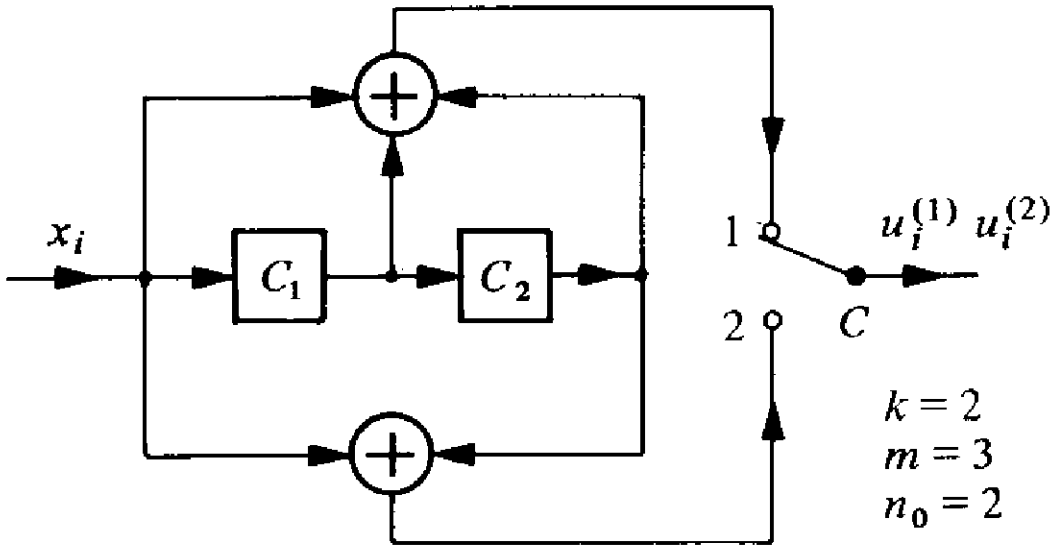
- Un diagramme d'état

- Un treillis → chemin → décodage par chemin le + probable

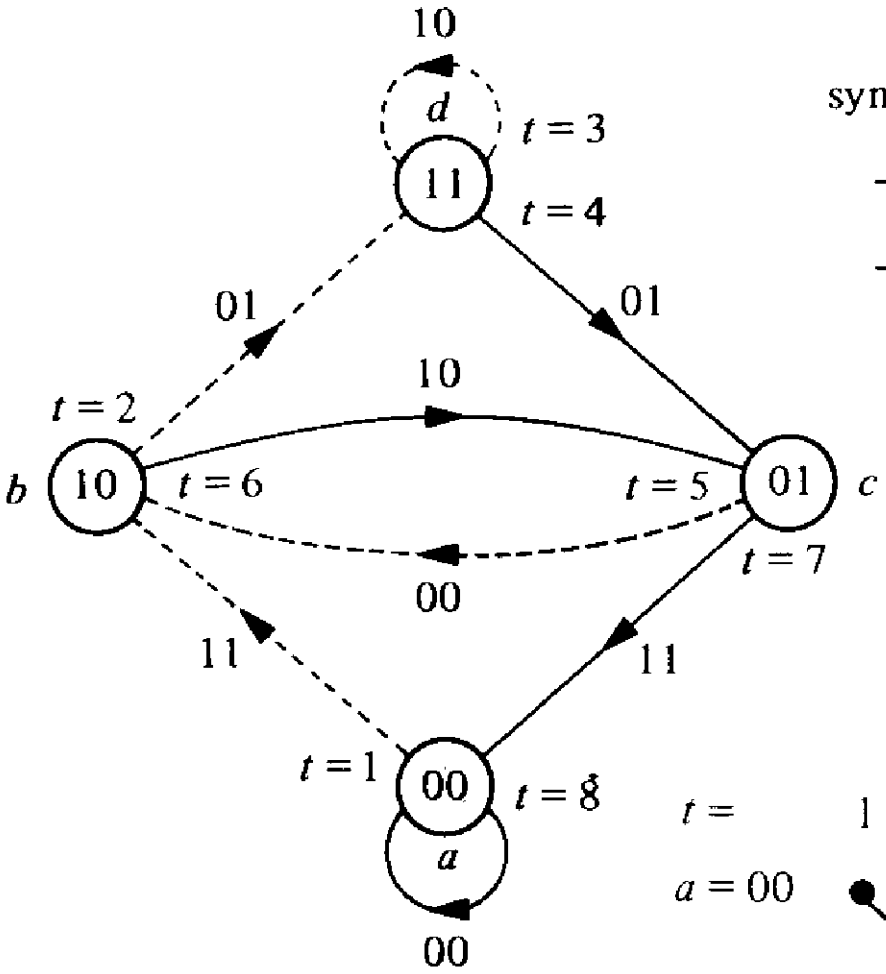
- Exemple : n_0=2, R=0.5 , m=3

$$U_n^{(1)} = x_n + x_{n-1} + x_{n-2}$$

$$U_n^{(2)} = x_n + x_{n-2}$$



t_i	1	2	3	4	5	6	7	8
x_i	1	1	1	0	1	0	0	0
$C_1 C_2$	00	10	11	11	01	10	01	00
$u_i^{(1)} u_i^{(2)}$	11	01	10	01	00	10	11	00



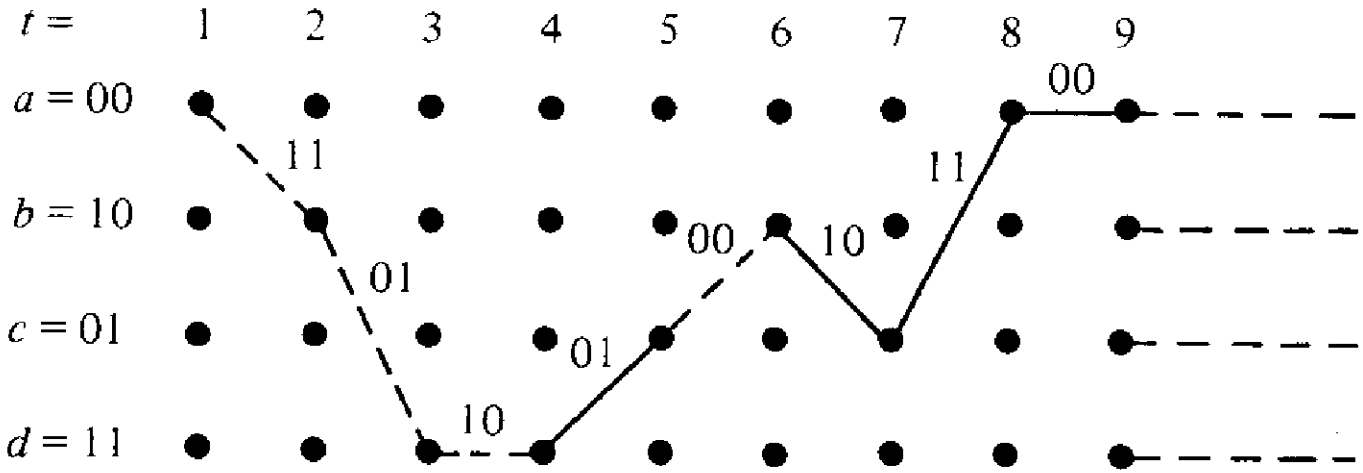
symbole entrant :

——— 0 :

– – – – 1 :

✓Recherche d'erreur à la fréquence N

→ Dmin = 2e+1



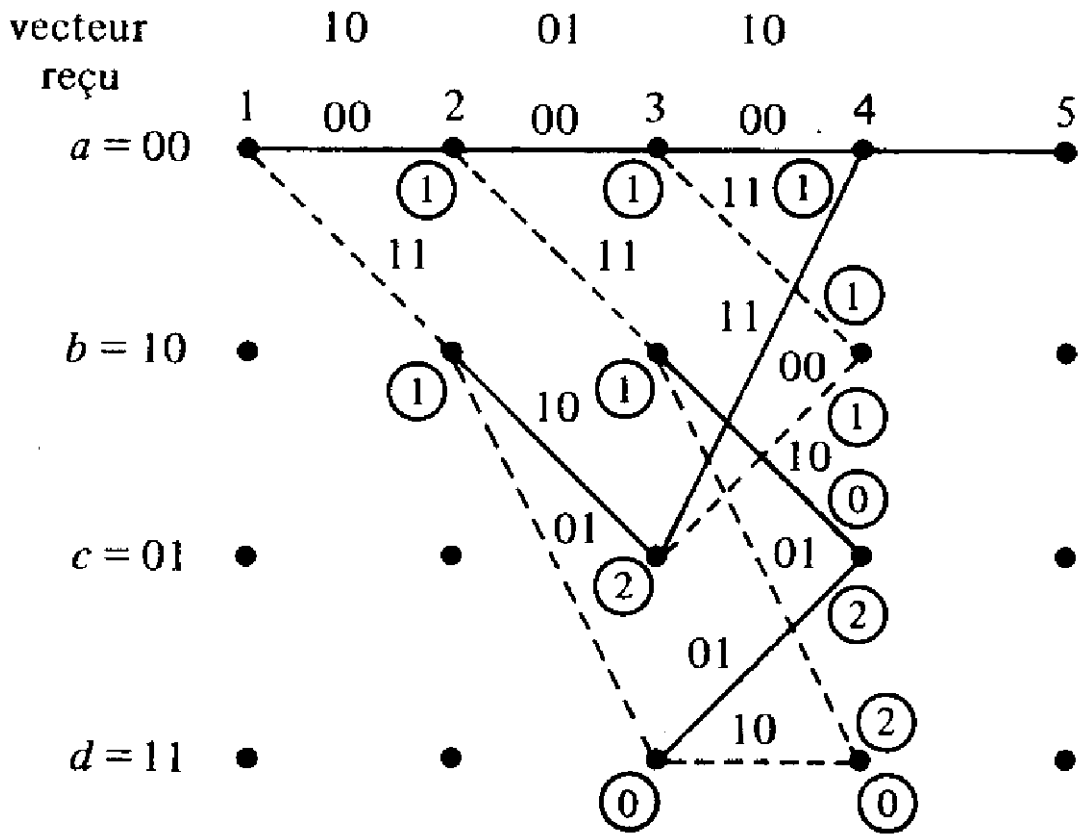
- **Décodage : algorithme de Viterbi**

⇨ Stratégie de recherche de D_{min}

✓Exemple pour N=3

10 01 10 → $Min\left(\sum_{i=1}^{3} d_i\right) = ?$

→ 1<u>1</u> 01 10



• Conclusion sur le codage de canal

✓ Indispensable

✓ Théories mathématiques complexes → des solutions concrètes

- Reed-Salomon (1984) : BCH Qaire → DVB(204,188,8)
- Turbo-Codes (1993) : Code convolutif + brassage

✓ Recherche de codeurs conjoint source / canal

- complexité --
- robustesse ++
- flexibilité ++

Introduction aux Turbo Codes

Suite du cours Codage de Canal

Thomas Grenier

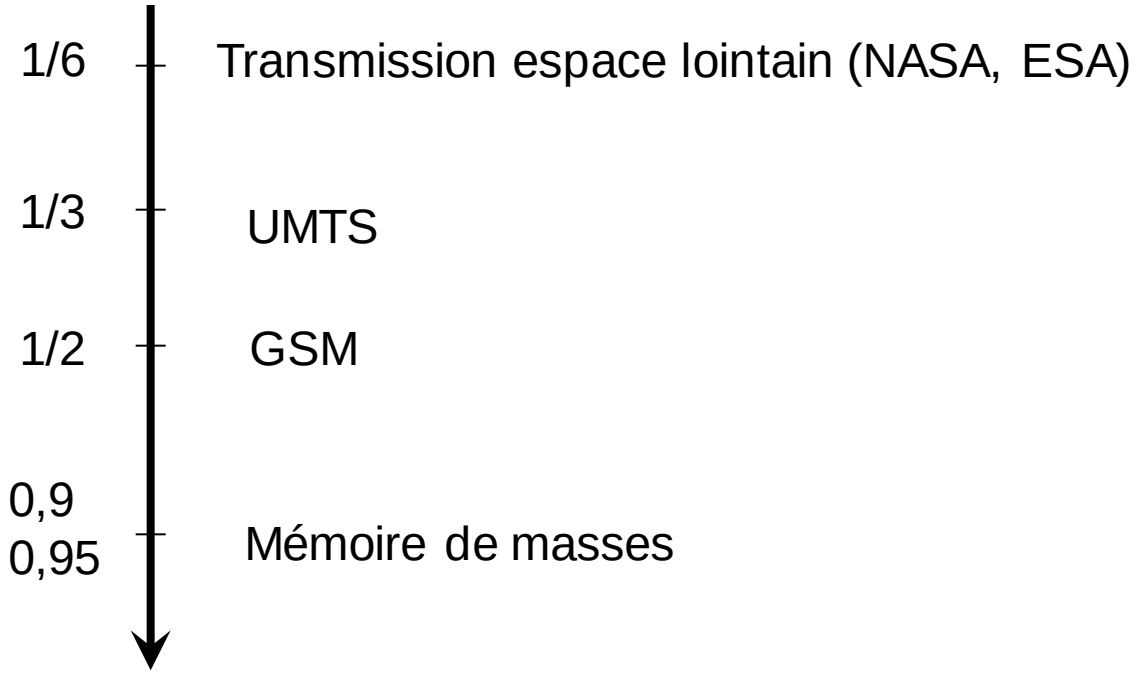
Codage de canal: but & enjeu

➔ Bénéficier de la diversité de l'information

- Notations:
  - k : nb de bits de chaque bloc d'information.
  - n : nb de bits du bloc après codage.
- R=k/n défini le rendement de codage
- k et R fixé par les normes. Ex:
  - ATM: k=53 octets,
  - GSM: k=192 bits, R=1/2

Codage de canal: enjeu

- Ordre de grandeur des rendements et applications:



Code correcteur

- Le plus simple : codage par répétition
- Le code de Hamming

  Code parfait pour k=4 et n=7 (Dmin=3)
- Le code le plus puissant : code Aléatoire

  Utilisé par Shannon pour établir la loi du théorème de codage de canal

$$\overline{D\min} \approx \frac{n-k}{2}$$

Borne réaliste :

$$\overline{D\min} \approx \frac{n-k}{4}$$

Ex: sur du MPEG (k=188octets, R=1/2) → Dmin=375

→ correction de 190 erreurs

Code correcteur

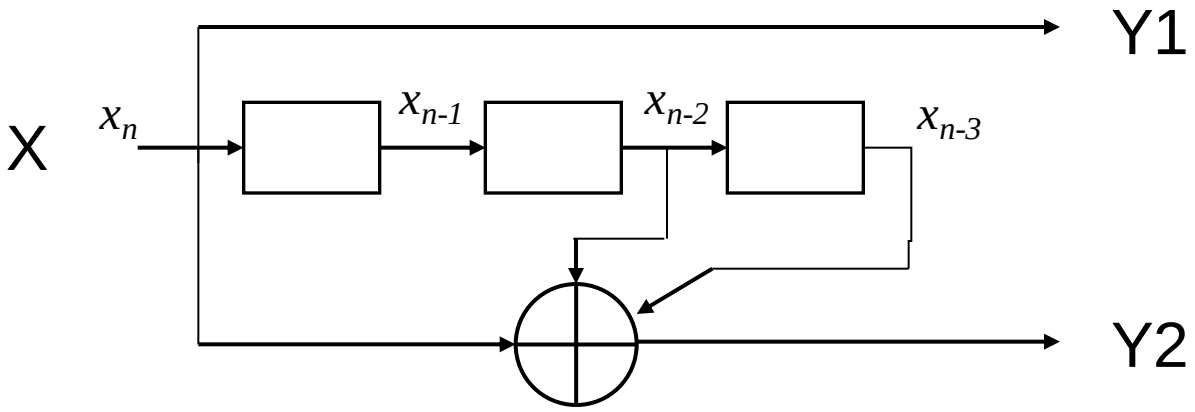
- Le code aléatoire existe sans possibilité de décodage:

  Pour chaque bit à décoder : considérer 2^k cas!

➔ Idée pour le codage de canal : construire un code très proche du code aléatoire, de manière à obtenir un Dmin très (très) grand, mais décodable ☺

Code convolutif (1954)

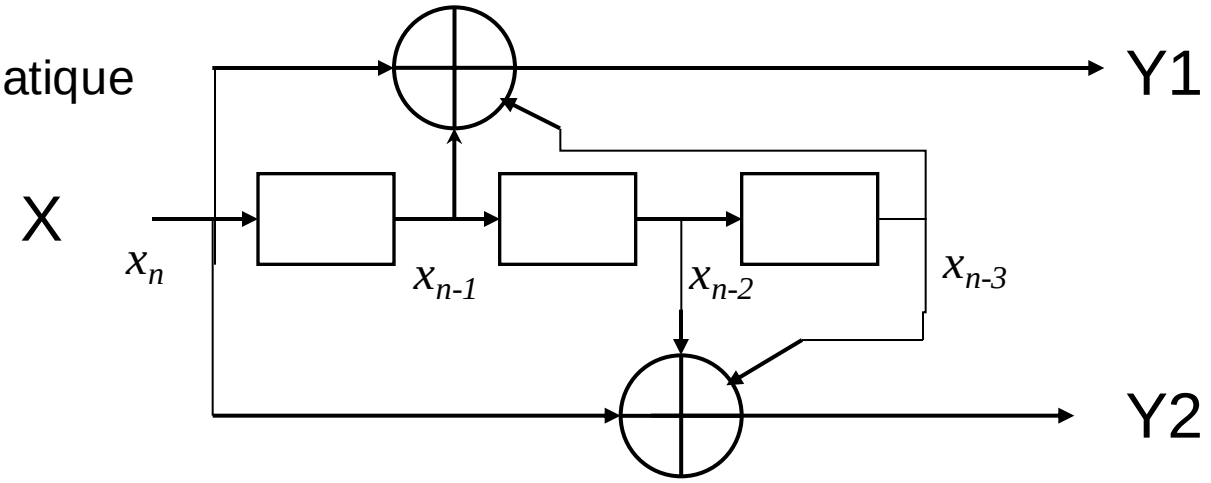
- L'information est considérée comme une suite de donnée (en non pas en « bloc »)
- Codage = Convolution discrète de l'information



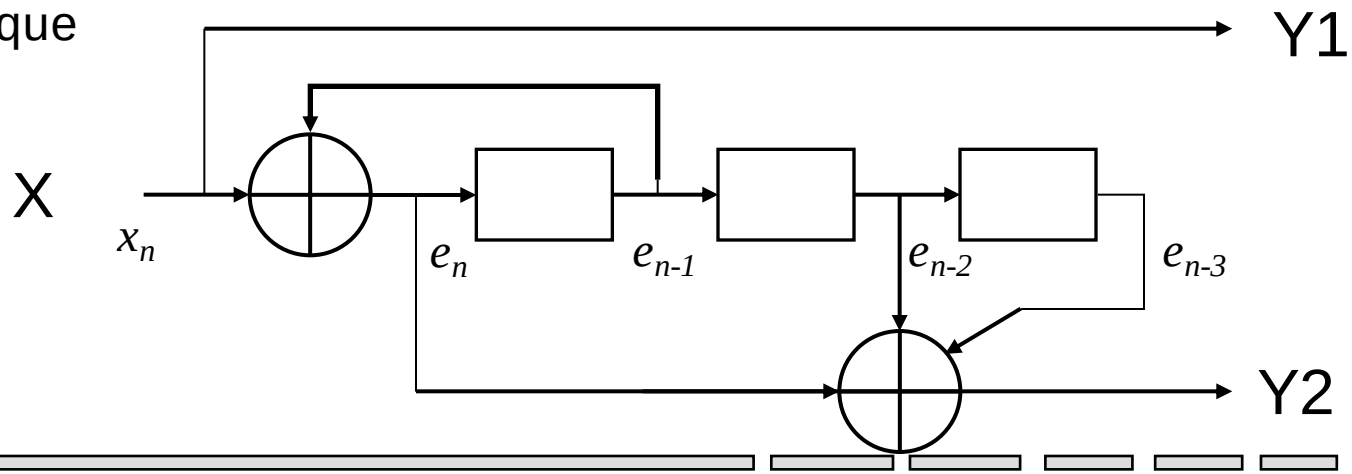
Code convolutif systématique

Codes convolutifs

Non systématique

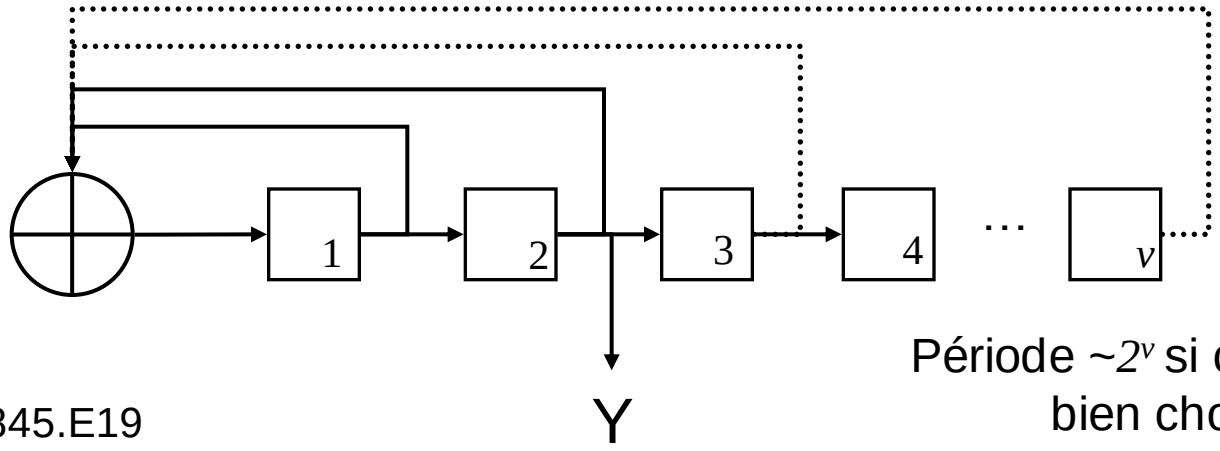


Systématique
Récursif
(1970)



Rappel qq structures

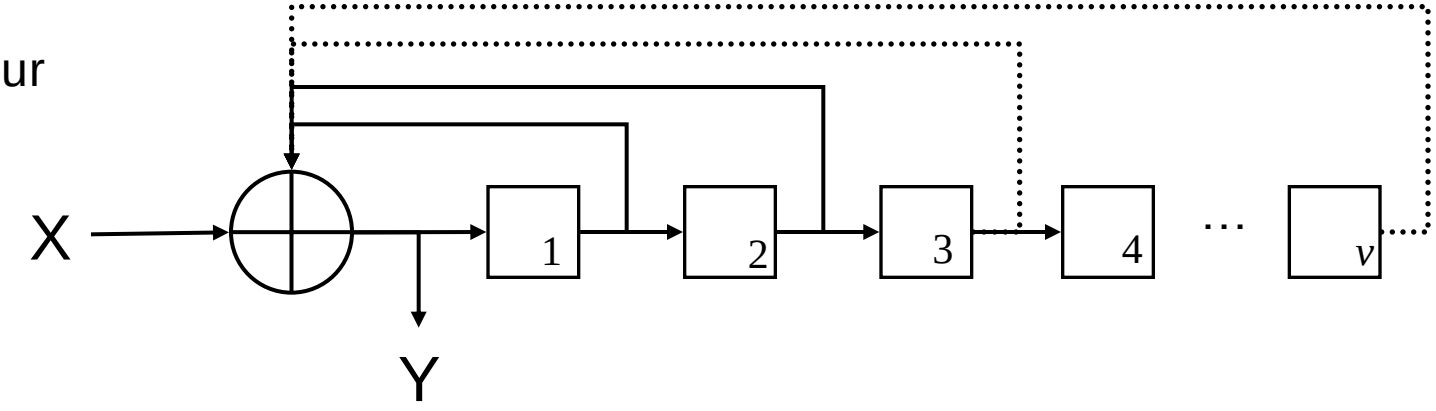
Générateur pseudo aléatoire



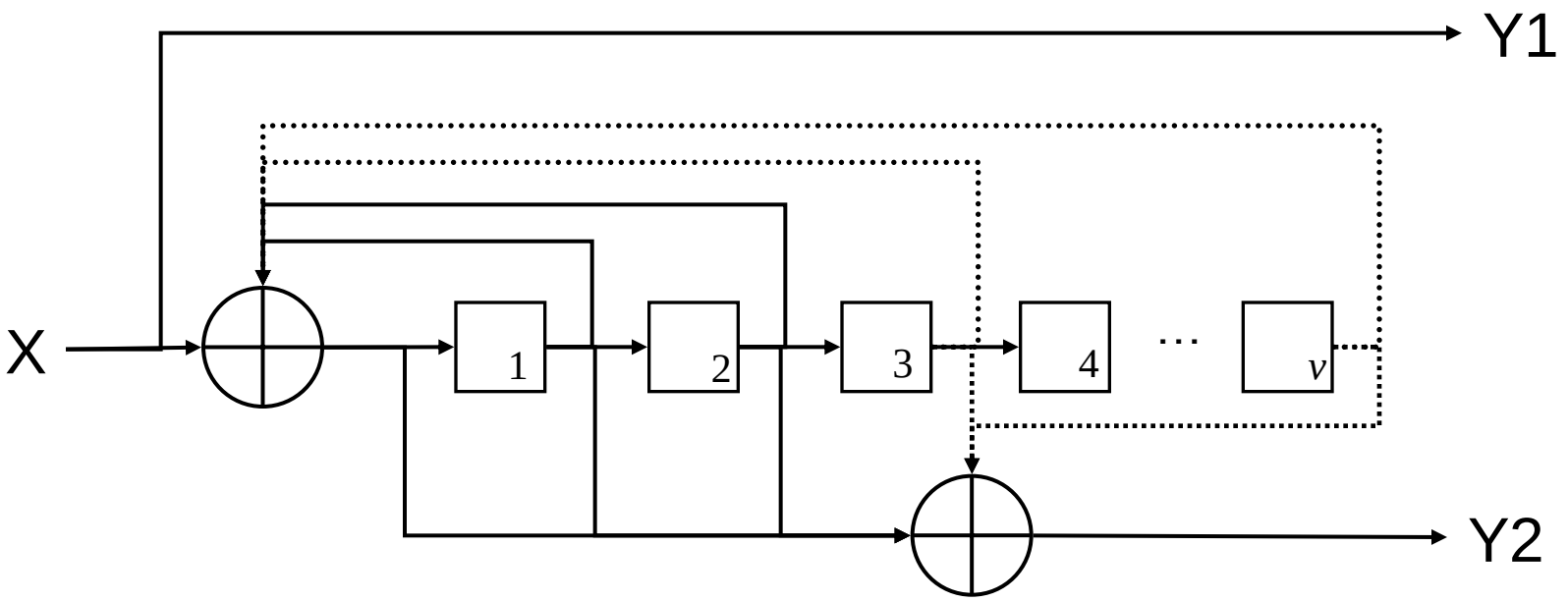
Période ~2^v si connexions bien choisies

rem: 2^{64}=1,845.E19

Brasseur



Code convolutif récursif



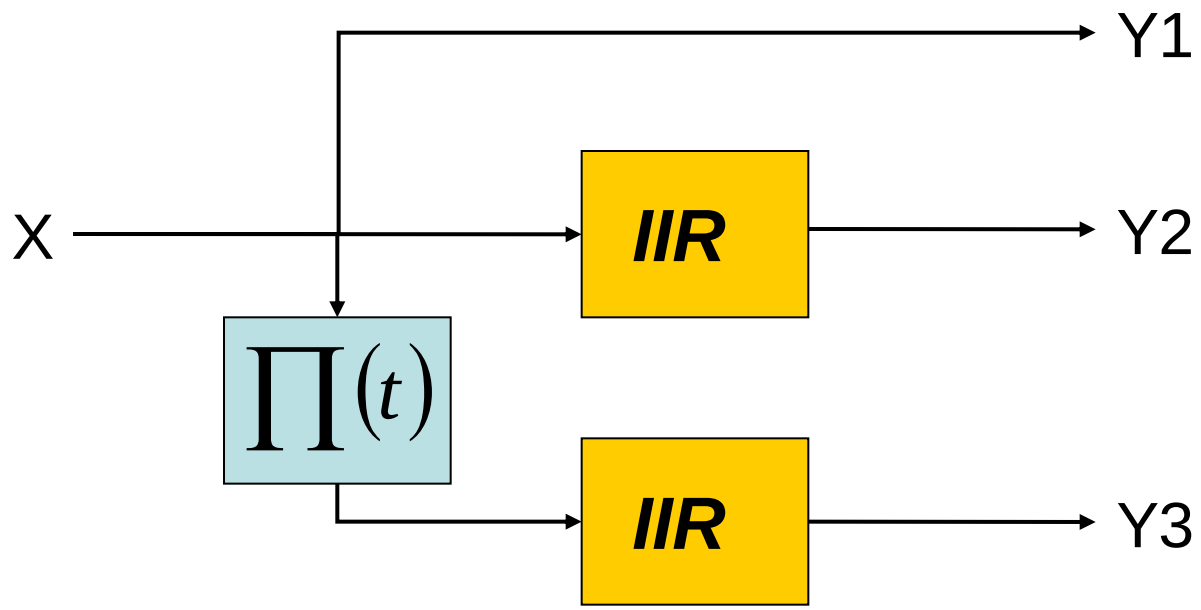
➔ Nature aléatoire des codes convolutifs récursifs

➔ Potentiellement optimaux pour le codage de canal !!!

➔ Décodage ? : Algorithme de Viterbi, mais décoder 2^v états ☹

Solution : Turbo Codage

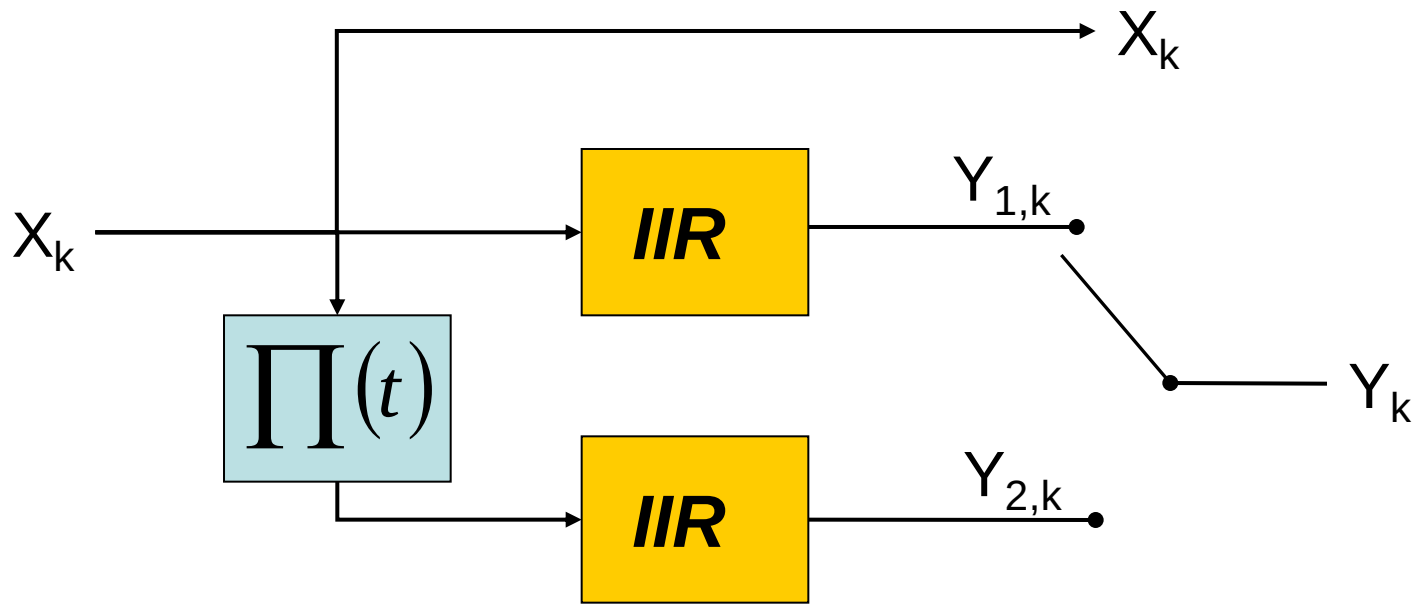
- Créer un codeur équivalent à un codeur convolutif recursif de ν très grand



$\prod(t)$ Permutation ou entrelaceur de bits

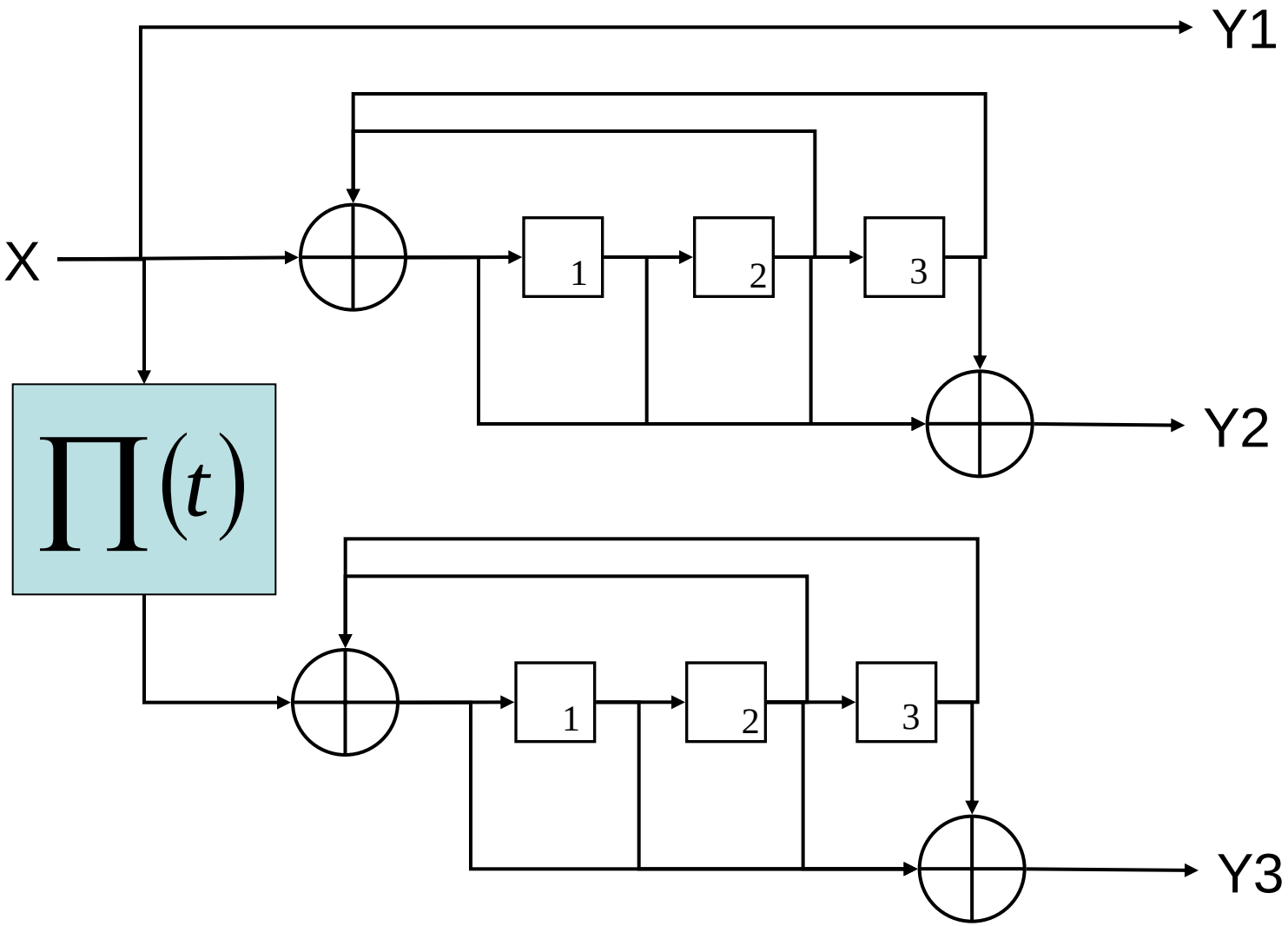
Solution : Turbo Codage

- Créer un codeur équivalent à un codeur convolutif recursif de ν très grand



$\Pi(t)$ Permutation ou entrelaceur de bits

Schéma turbo codage



Turbo code

- Valeur équivalente du nb d'états (2^v) ?

  Dépend de la matrice/fonction de permutation $\prod(t)$
- Mauvais $\prod(t)$
  - Matrice identité: codage ⇔ 2 codeurs en //

  Performance équivalente à un seul codeur
- Bon $\prod(t)$
  - Dans cas idéal $v = 3+3+k$ ➔ 2^{k+6} états

  (impossible)

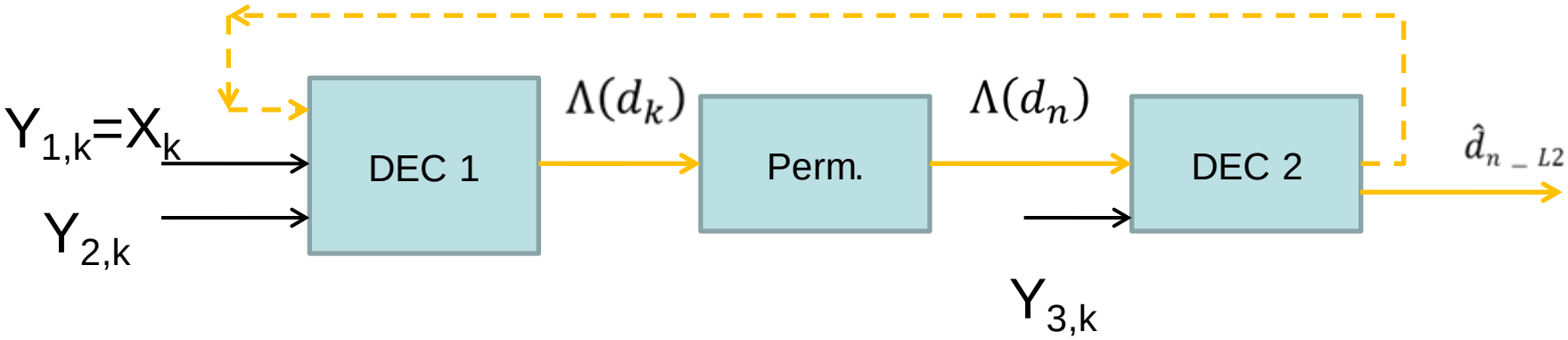
➔ $\prod(t)$ ne doit pas être régulière (…pas être inversible)

Intérêt turbo code

- Décodage Simple !
  - Décodeur bidimensionnel
  - 2 décodeurs de *3* bascules synchrones
  - Échange de probabiliste entre les 2 décodeurs (convergence)

D'où le nom de *turbo: réinjection de la sortie d'un décodeur dans l'entrée de l'autre*

Décodage Turbo



$$\Lambda(d_k) = log\frac{p(d_k = 1)}{p(d_k = 0)}$$ Logarithme du ration de la vraissemblance

$\hat{d}_{n\_L2}$ 'soft bit' *n* avec un retard L2

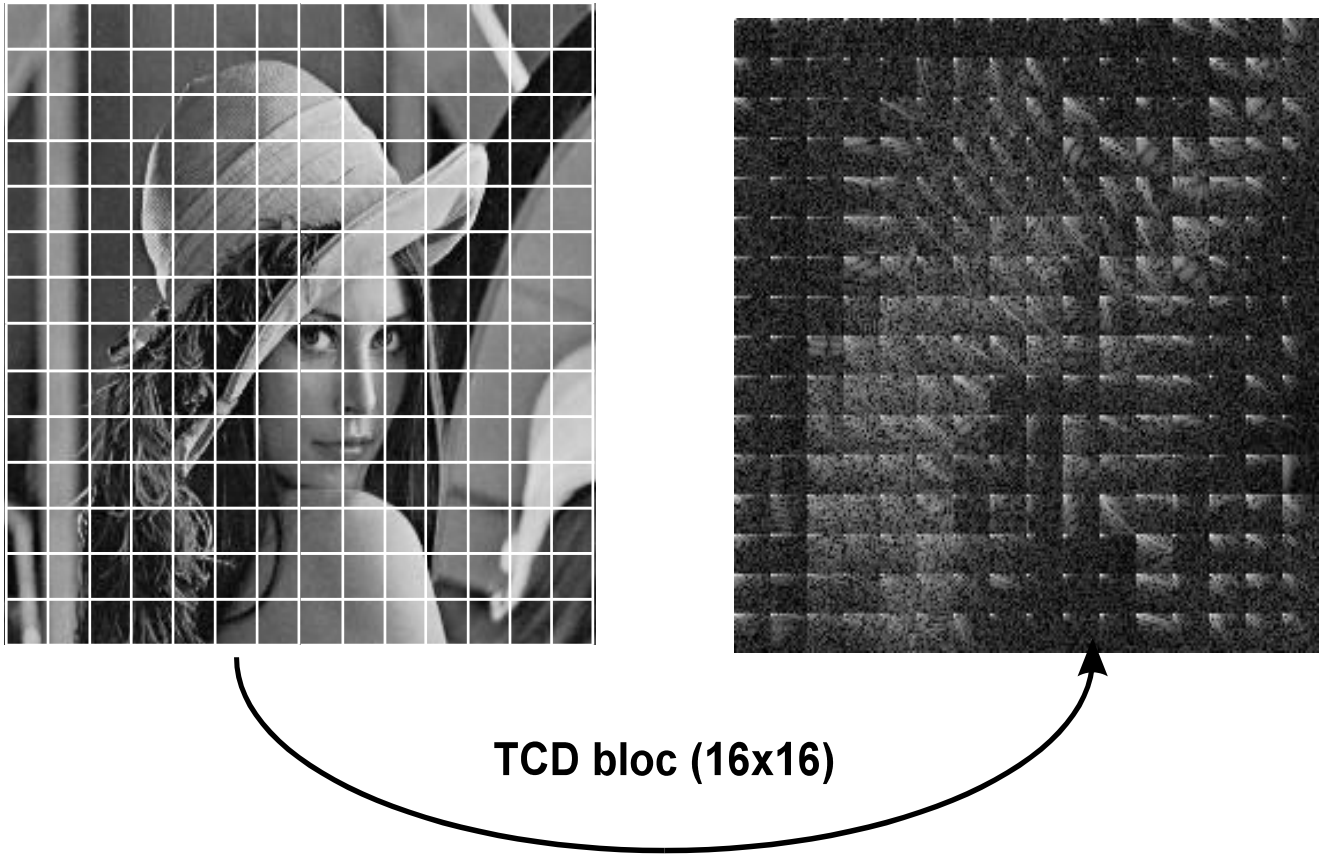
DEC1 : Algorithme BCJR (treilli, « viterbi bi-directionnel », contraintes)
délai : L1
DEC2 : Algorithme Virtebi
délai : L2

Applications turbo

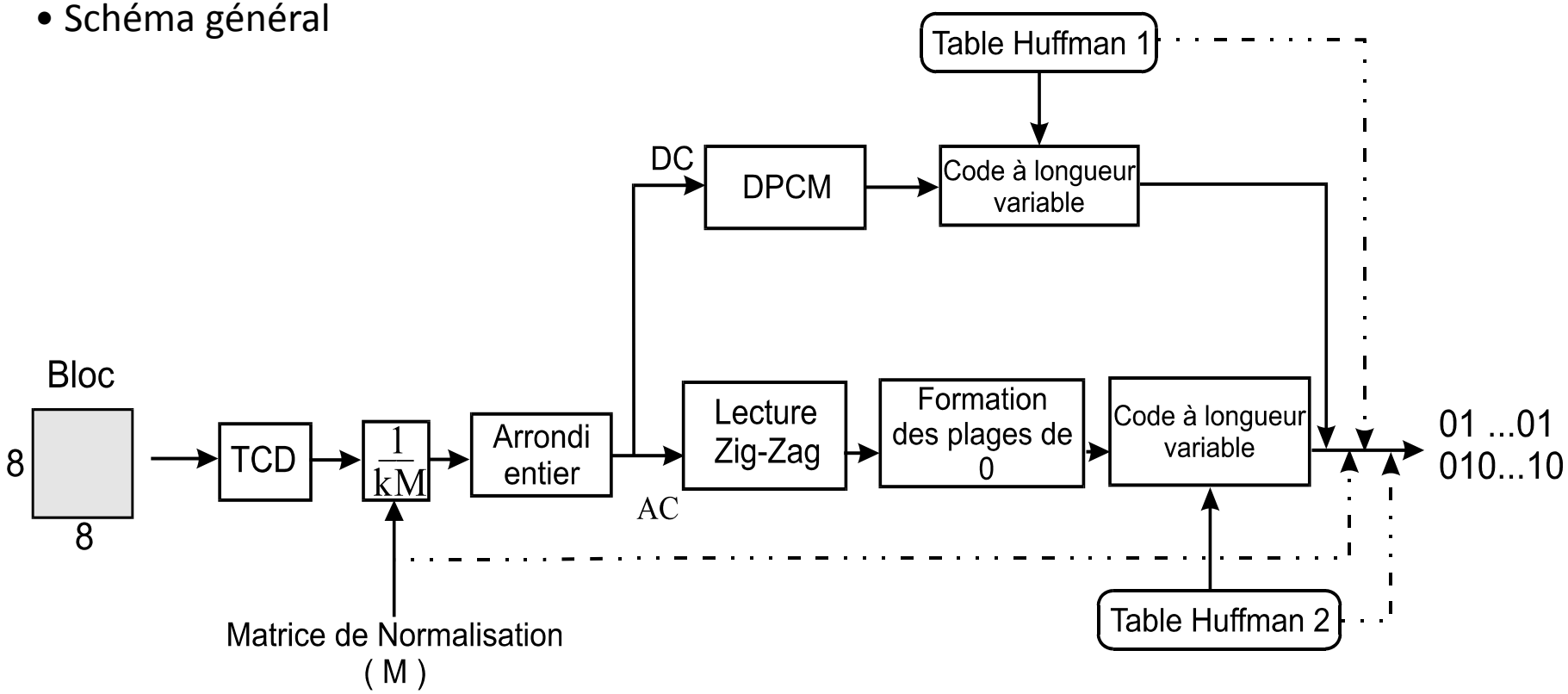
- Encore de nombreuses recherches
- Brevet France Telecom
- Gain des Turbo Code: 2,5dB à 4dB
- Normes, formats utilisant les turbo codes:
  - UMTS, ADSL2
  - IEEE 802.16 (WiMAX)
  - 3G et 4G (HSPA, EV-DO, LTE)
  - DVB-RTS (il doit y en avoir d'autres…)

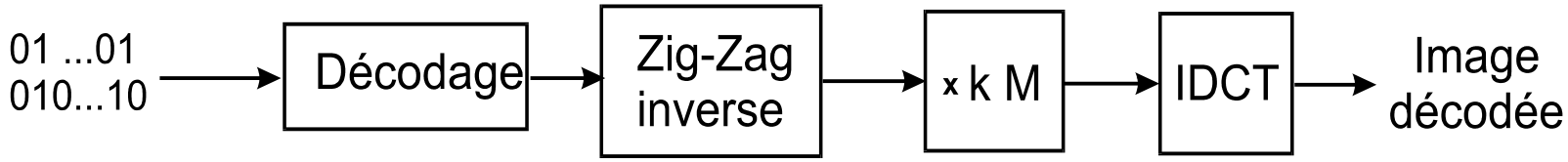
✓ Compression DCT bloc : JPEG (1989)

- DCT bloc 8x8
  - ⇨ homogénéité locale de l'image
  - ⇨ l'erreur de quantification est localisée au bloc



• Schéma général





- Matrice de normalisation
  - ⇨ allocation des bits aux coeffs avant quantification par arrondi

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

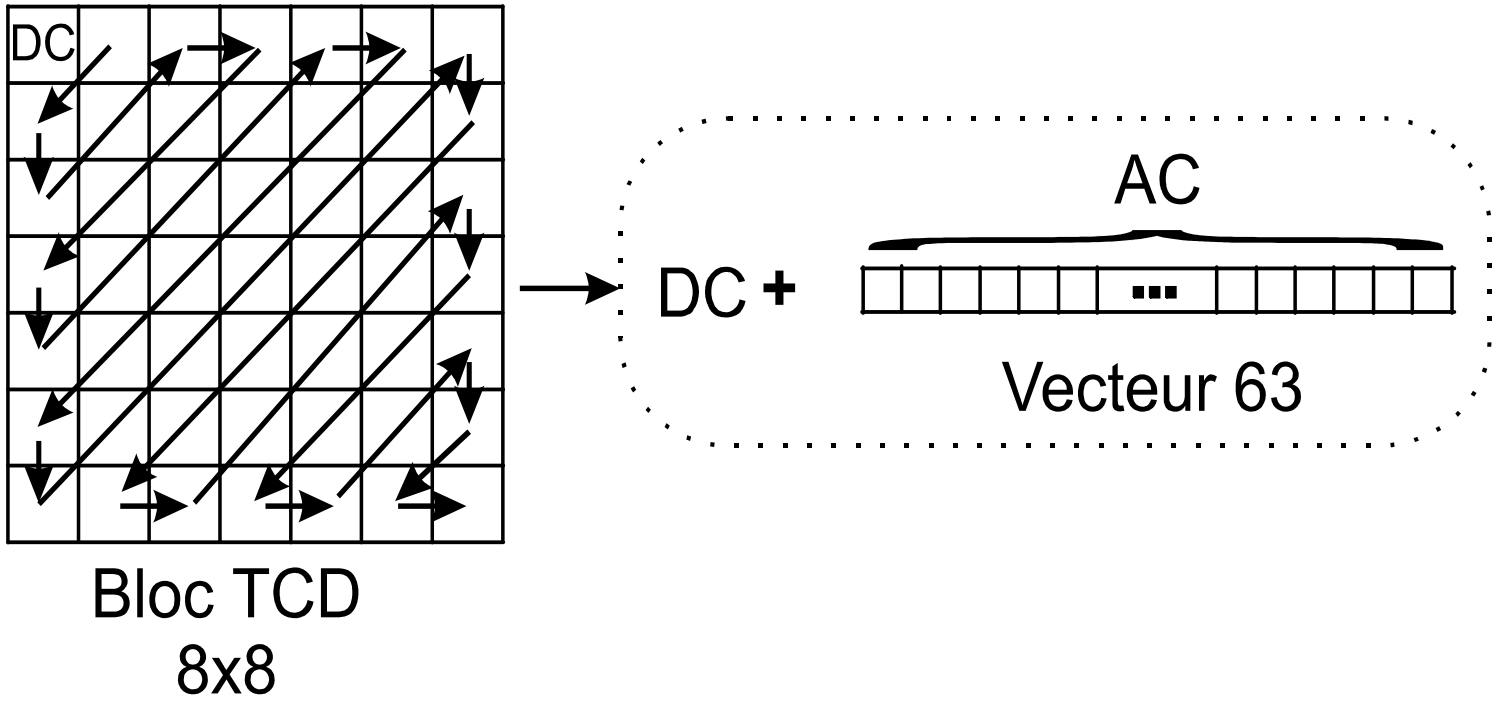
Matrice luminance

Matrice chrominance

$$\begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

- Lecture zig-zag

  ⇨ prise en compte de la répartition spatiale de l'énergie pour faire apparaître de longues plages de coeffs nuls



- Codage du coeff DC

  ⇨ DPCM d'ordre 1 + Huffman

- Codage des coeffs AC

  ⇨ Codage hybride : runlength + ... + Huffman

  - Huffman = Code (plage de 0 + catégorie)

    162 codes : $10_{cat} x 16_{lp} + 2_{(EOB+16)}$

Cat.	Intervalle des coefficients AC
1	-1 _ 1,
2	-3, .. ,-2 _ 2, .. ,3
3	-7, .. ,-4 _ 4, .. ,7
4	-15, .. ,-8 _ 8, .. ,15
5	-31, .. ,-16 _ 16, .. ,31
6	-63, .. ,-32 _ 32, .. ,63
7	-127, .. ,-64 _ 64, .. ,127
8	-255, .. ,-128 _ 128, .. ,255
9	-511, .. ,-256 _ 256, .. ,511
10	-1023, .. ,-512 _ 512, .. ,1023

AC ⇨ | huffman | signe | k-1 bits |

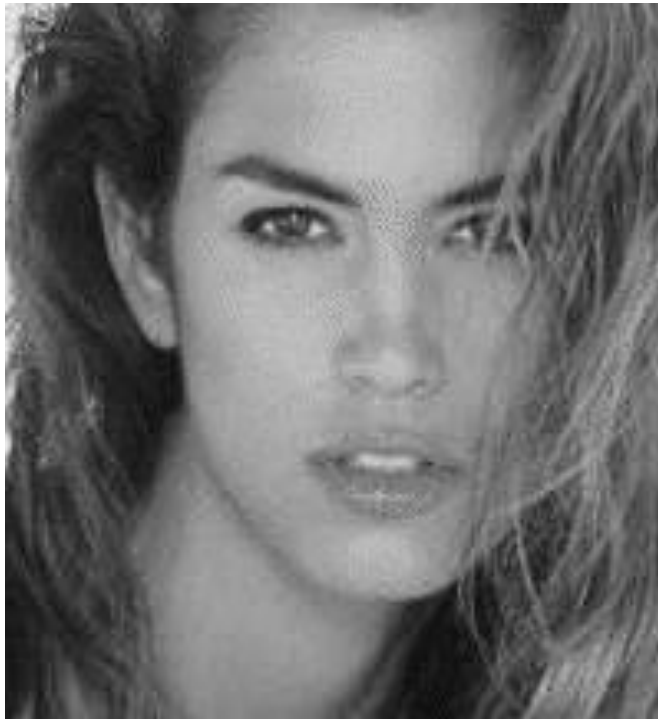
- Exemple

  0 -2 -1 0^2 -1 0^{46} ⇨ 111001 0 0 / 00 0 / 11011 0 / 1010

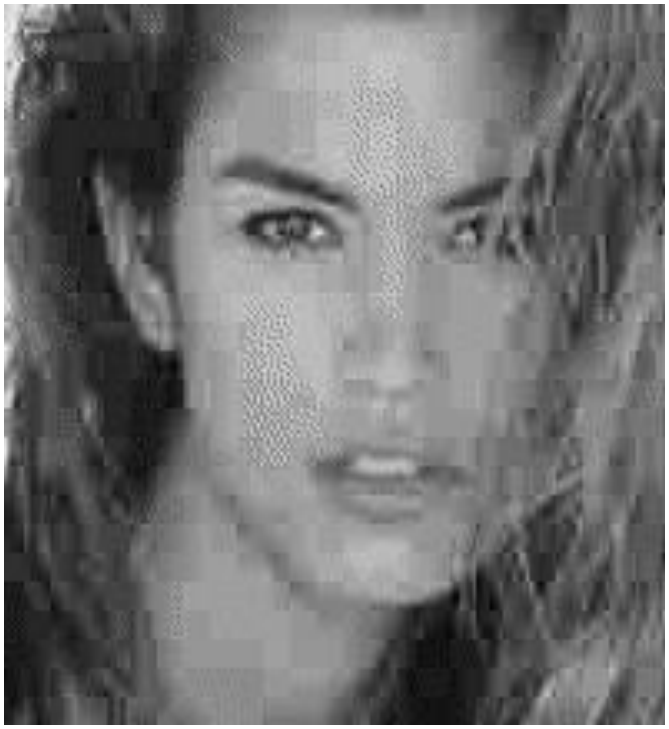
- Extrait de la table d'Huffman des AC

Plage de Zéros	Catégorie	Code
0	1	00
0	2	01
0	3	100
0	4	1011
.	.	.
1	1	1100
1	2	111001
1	3	1111001
1	4	111110110
.	.	.
2	1	11011
2	2	11111000
.	.	.
3	1	111010
.	.	.
16		11111010
EOB		1010

- Remarques
  - ⇨ JPEG = méthode générale ⌫ à adapter ...
  - ☺ Très performant à taux faibles (#10)
  - ☹ Effets de blocs à taux élevés

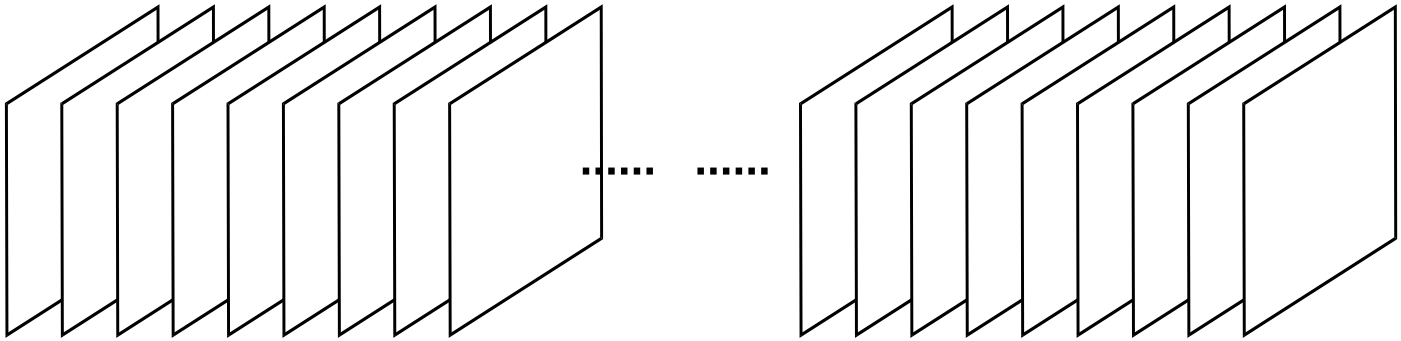


Tc = 10 / RSB = 30.1 dB



Tc = 20 / RSB = 28.7 dB

d) Compression de séquences d'images



- ✓ Supprimer la redondance spatiale ou intra-image

  >> Approches 2D

- ✓ Supprimer la redondance temporelle ou inter-image

  >> Utiliser le déjà vu et le mouvement

✓ Les normes MPEG

- **H261** (1988)
  - ↳La base de la compression de séquences d'images
    - Block matching
    - DCT bloc + Run length + DPCM

- **MPEG 1** (1988-92)
  - ↳Vidéo + Audio / 1.5 Mbs ⇨ **CDI**

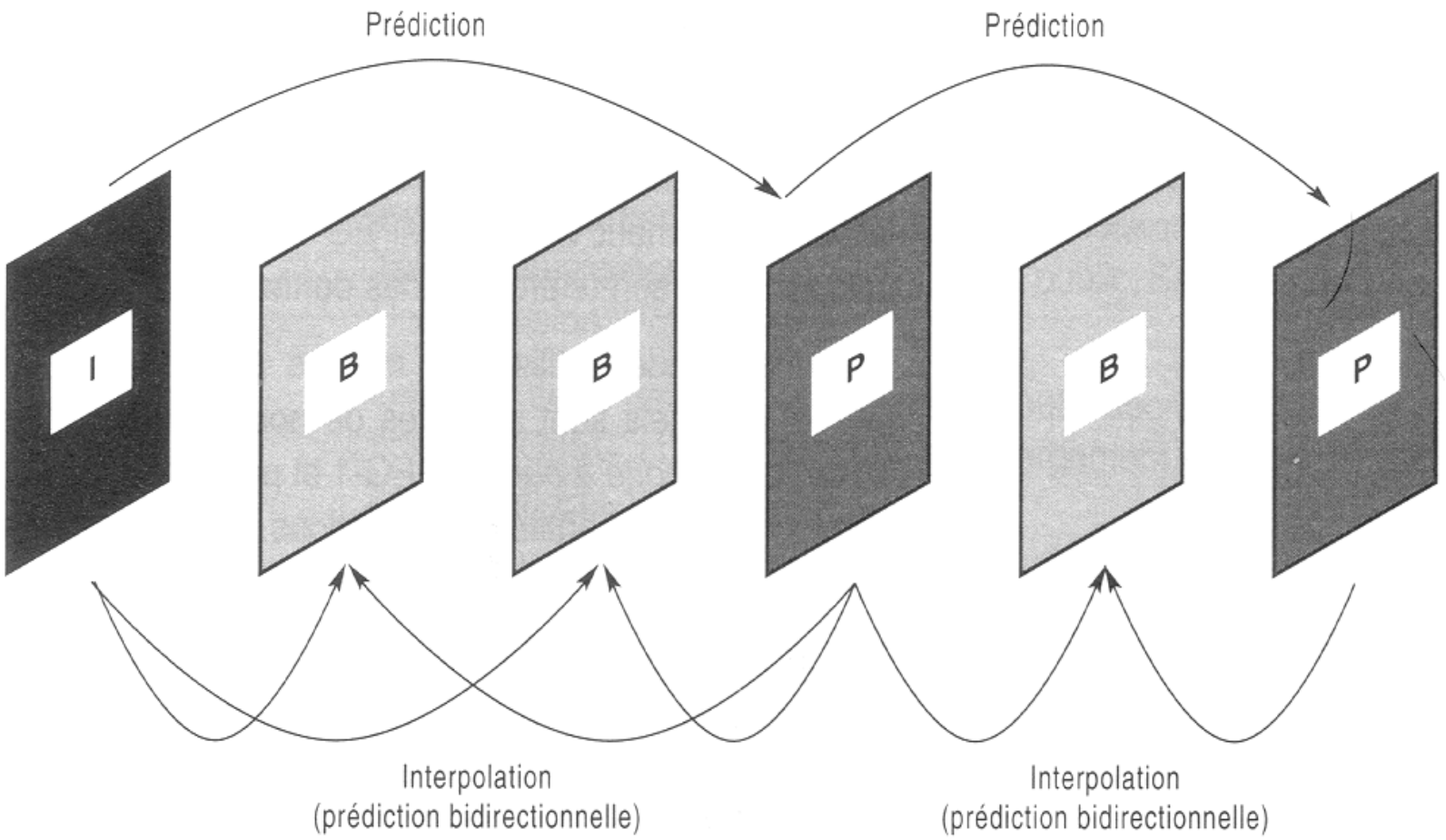
- **MPEG 2** (1990-94)
  - ↳4-30 Mbs ⇨ **TV numérique** (Digital Video Broadcasting)

- **MPEG 4** (1996-99)
  - ↳L'approche **multimédia** interactif

- **MPEG 7** (1997-01)
  - ↳ **Indexation** & recherche d'information

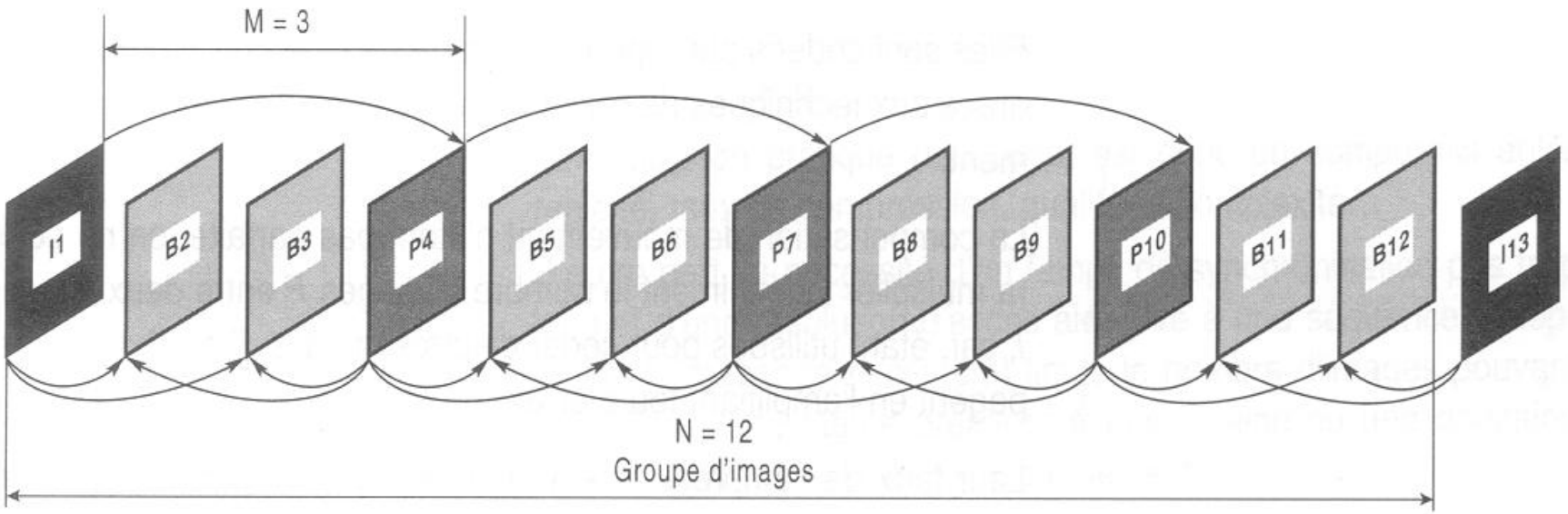
✓ Les bases de H261 à MPEG2

- 3 types d'images : 3 codages



- Images I (intra)
  - Codées JPEG'
  - Point d'accès séquence (0.5s)
  - Tc faible
- Images P (Prédites)
  - Prédites à partir de I ou P
  - Codage DPCM des vecteurs mvt
  - Codage JPEG* de l'erreur de prédiction
  - Tc élevé
  - Propagation de l'erreur
- Images B (Bidirectionnelles)
  - Interpolées à partir des I P
  - Tc le plus élevé

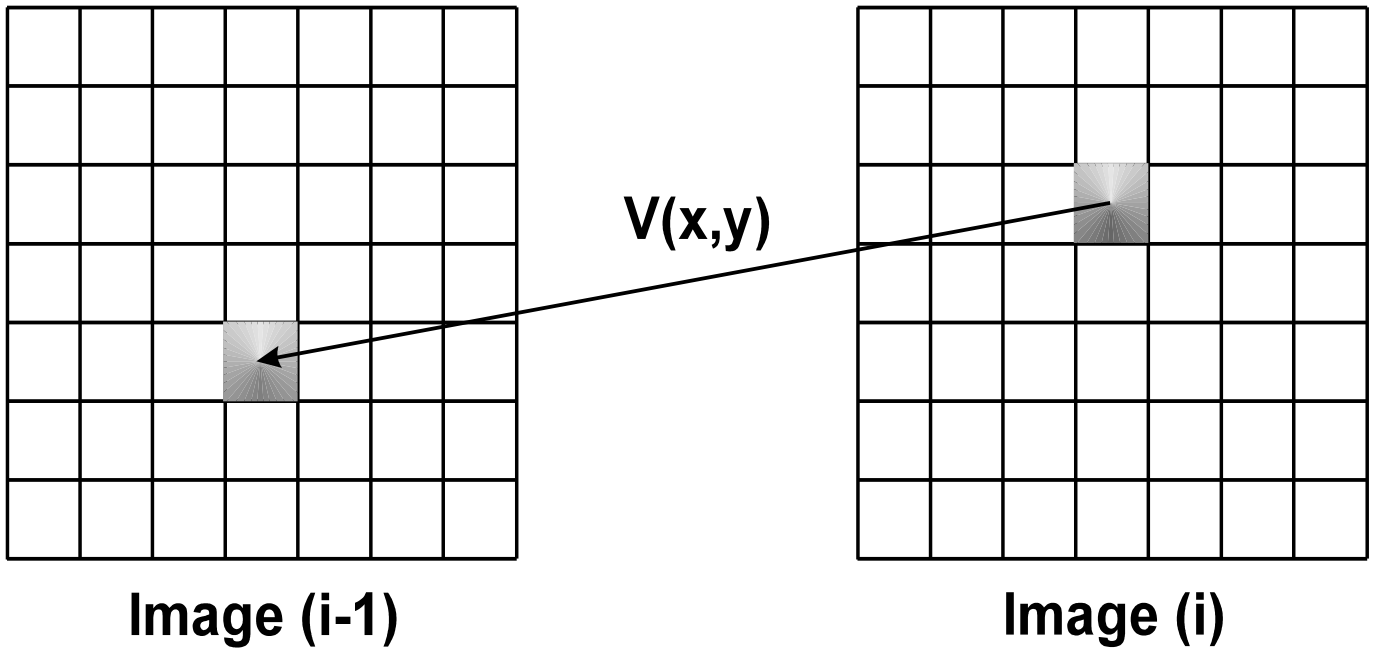
- 2 paramètres de réglage
  - N : distance inter-I (#12)
  - M : distance inter-P (#3)



GOP

✓ Estimation du mouvement par block matching

- Blocs 16x16
- Compromis simplicité / efficacité
- Rapide : algorithme logarithmique



✓ Le codage des images P

1- Calcul des V_j entre $I(n)$ et $\hat{I}(n-1)$

2- Synthèse de $I_p(n)$: $V_j(n) + \hat{I}(n-1) = I_p(n)$

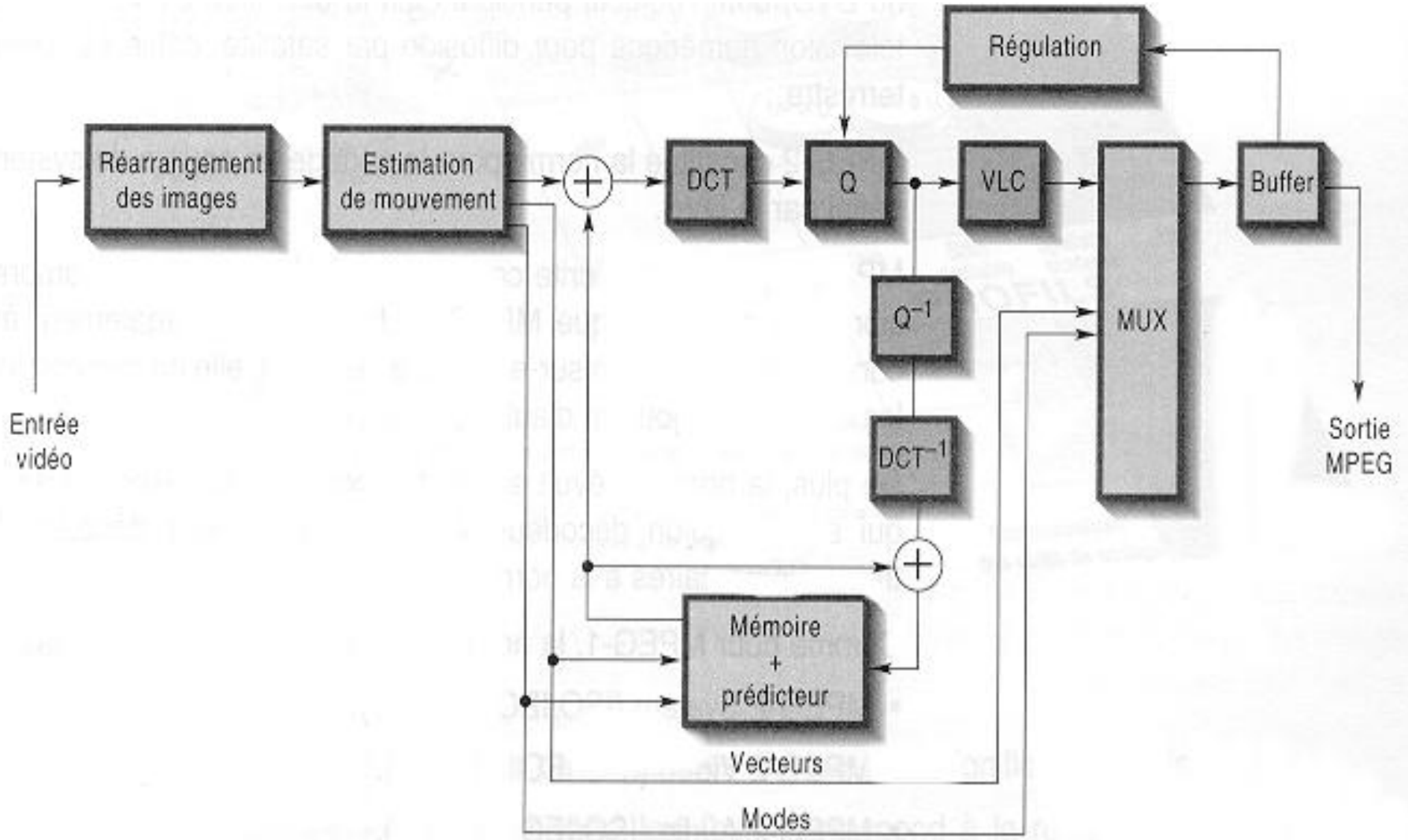
3- Calcul de l'erreur : $E(n) = I(n) - I_p(n)$

4- Codage JPEG* de $E(n)$

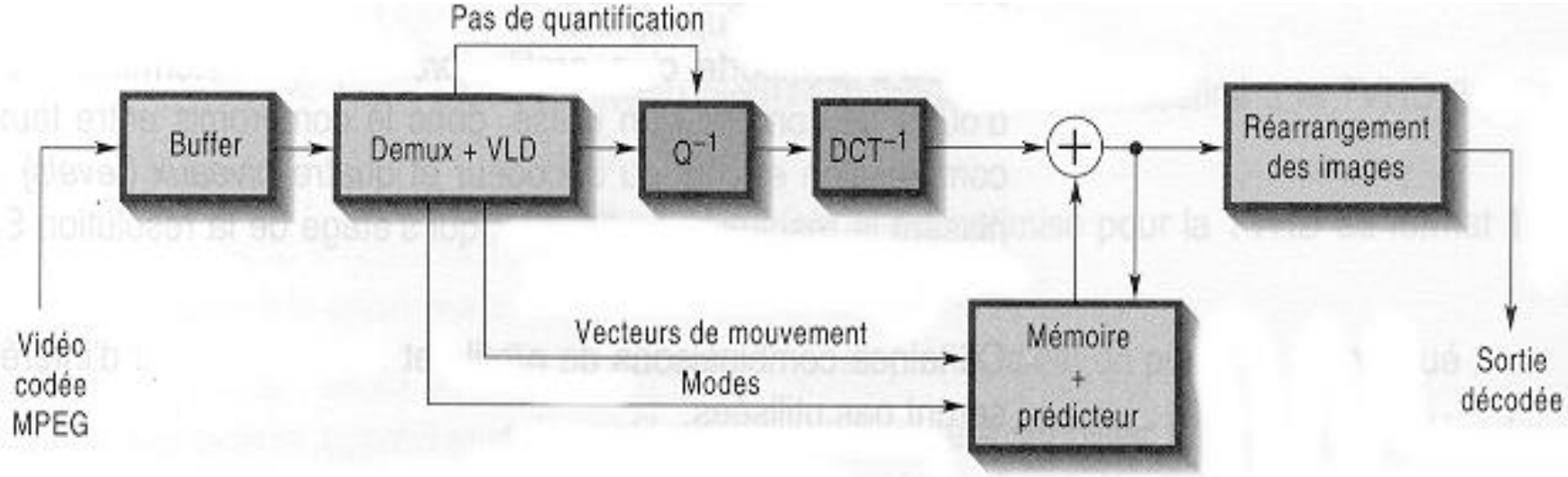
4_{bis}- *Mémorisation de* $\hat{I}(n)$

5- Codage DPCM des V_j

✓ Codeur MPEG2



✓ Décodeur MPEG2



✓ Codage et TVnum

- Numérisation brute : 200 Mb/s
- DVB # DVD = MPEG2 MP@ML
  - 720 x 480/576 (30/25 Hz) avec IPB
  - 4 Mb/s (PAL/SECAM) à 9 Mb/s (studio)
  - Tc de 40 à 18

Plan

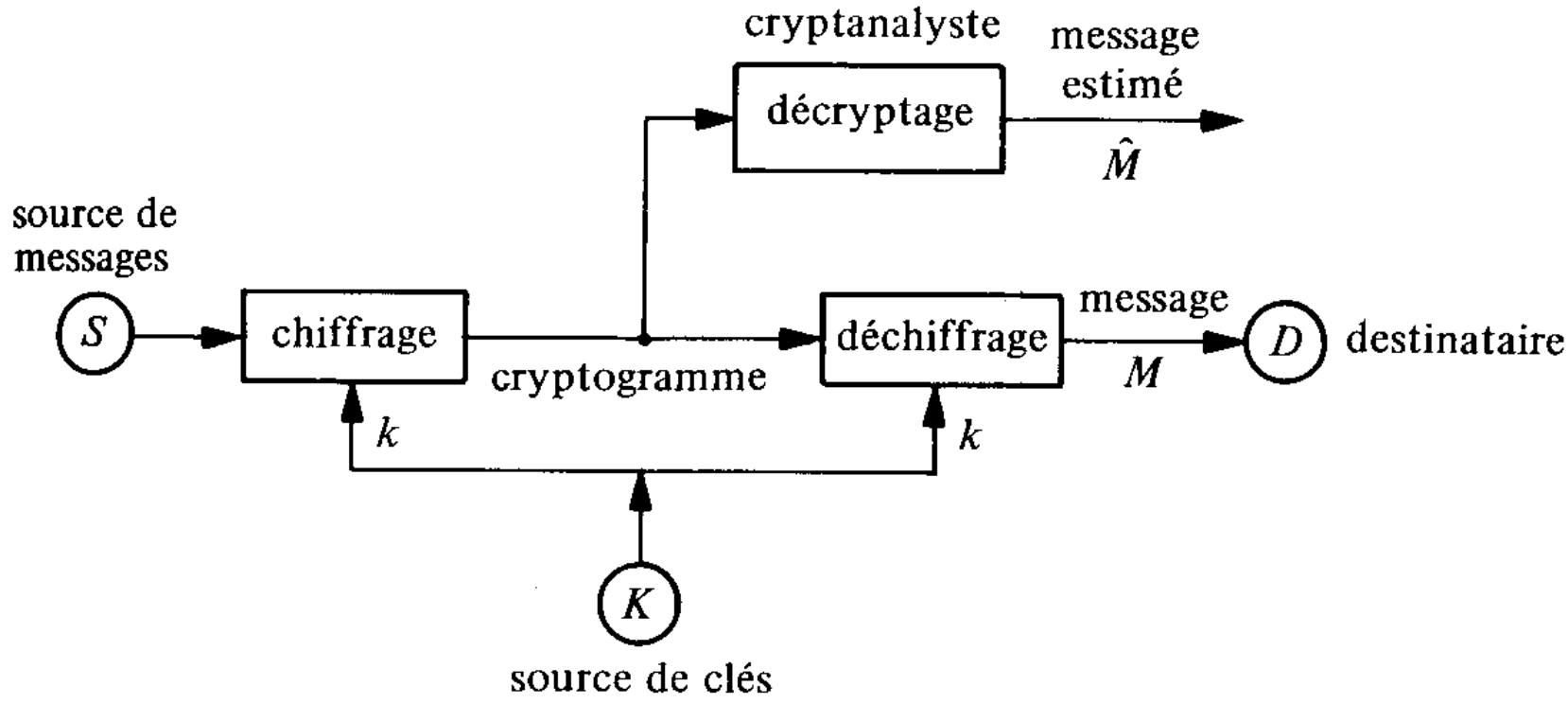
- 1. Introduction
- 2. Sources discrètes & Entropie
- 3. Canaux discrets & Capacité
- 4. Codage de source
- 5. Codage de canal
- 6. Cryptographie
- 7. Conclusion

6. Cryptographie

- **Objectifs**
  - ✓ Garantir la **confidentialité** des données
  - ✓ Garantir l'**intégrité** des données
  - ✓ Garantir l'**identité** des correspondants
    - ⇨ Non répudiation des transactions

- **Applications**
  - ✓Militaires
  - ✓Mots de passe
  - ✓Sécurité réseaux
  - ✓Téléphonie
  - ✓Commerce électronique
  - ✓@Business

- **Vocabulaire**

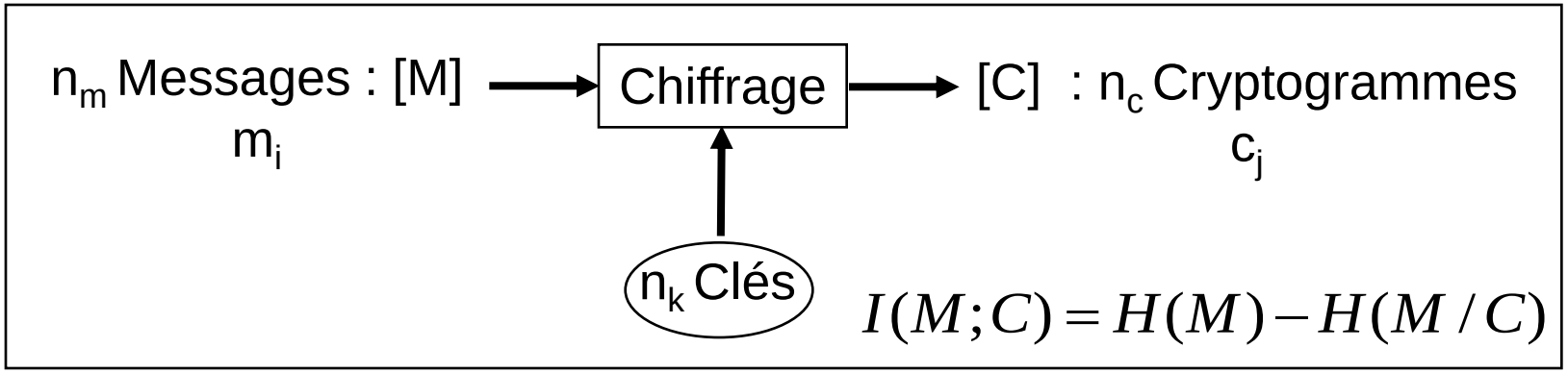


✓**Cryptographie** : techniques de chiffrage

✓**Cryptologie** : cryptographie & cryptanalyse

• Vue de la théorie de l'information

↳ Chiffrage = Canal très perturbé



✓Secret parfait ssi : $H(M/C) = H(M)$ soit $I(M;C) = 0$

- Clé unique permet m_i ⇨ c_j soit $n_m = n_c = n_k$
- Toutes les clés sont équiprobables

Chiffrage efficace

ssi

(Coût + temps) de décryptage >> Valeur de l'info

- **Les grandes approches**

  - ✓ **Approches classiques**

    - ✓**Chiffrage par substitution**
      Jules César, l'Abbé Trithème
    - ✓**Chiffrage par transposition**

  - ✓ **Approches modernes**

    - ✓**Chiffrage à clé privée (symétrique)**
      DES, IDEA,
    - ✓**Chiffrage à clé publique (asymétrique)**
      RSA, PGP

• Chiffrage par substitution

⇨ Chaque lettre (ou groupe de lettres) est remplacée par une lettre (ou un groupe de lettres)

• Abbé Trithème (1499)

Dans son royaume à perpétuité,
En Paradis à perpétuité,
Ainsi qu'en toute éternité;
Dans la gloire à perpétuité,
Mais dans son règne;
Sempiternel, toujours dans la félicité,
Tant dans la lumière que dans la béatitude,
Et toujours dans la gloire à perpétuité,
Mais dans son règne;
En une infinité encore à perpétuité,
Comme dans la gloire autant que dans les Cieux,
A tout jamais, oui ! à tout jamais à perpétuité;
Dans son royaume et dans la félicité,
Irrévocablement, dans son royaume,
Et sans cesse qu'il soit à perpétuité dans la lumière,
Et encore à perpétuité !

A = dans les cieux
B = à tout jamais
C = un monde sans fin
D = en une infinité
E = à perpétuité
F = sempiternel
G = durable
H = sans cesse
I-J = irrévocablement
K = éternellement
L = dans la gloire
M = dans la lumière
N = en paradis
O = toujours
P = dans la divinité
Q = dans la déité
R = dans la félicité
S = dans son règne
T = dans son royaume
U-V-W = dans la béatitude
X = dans la magnificience
Y = au trône
Z = en toute éternité

- **Chiffrage par transposition**

  ⇨Change l'ordre des lettres sans les substituer

- Exemple

B	R	I	Q	U	E	S
1	5	3	4	7	2	6
t	r	a	n	s	f	é
r	e	z	u	n	m	i
l	l	i	a	r	d	d
e	f	r	a	n	c	s
à	m	o	n	c	o	m
p	t	e	s	u	i	s
s	e	n	u	m	é	r
o	t	é	z	é	r	o
z	é	r	o	s	e	p
t	a	b	c	d	e	f

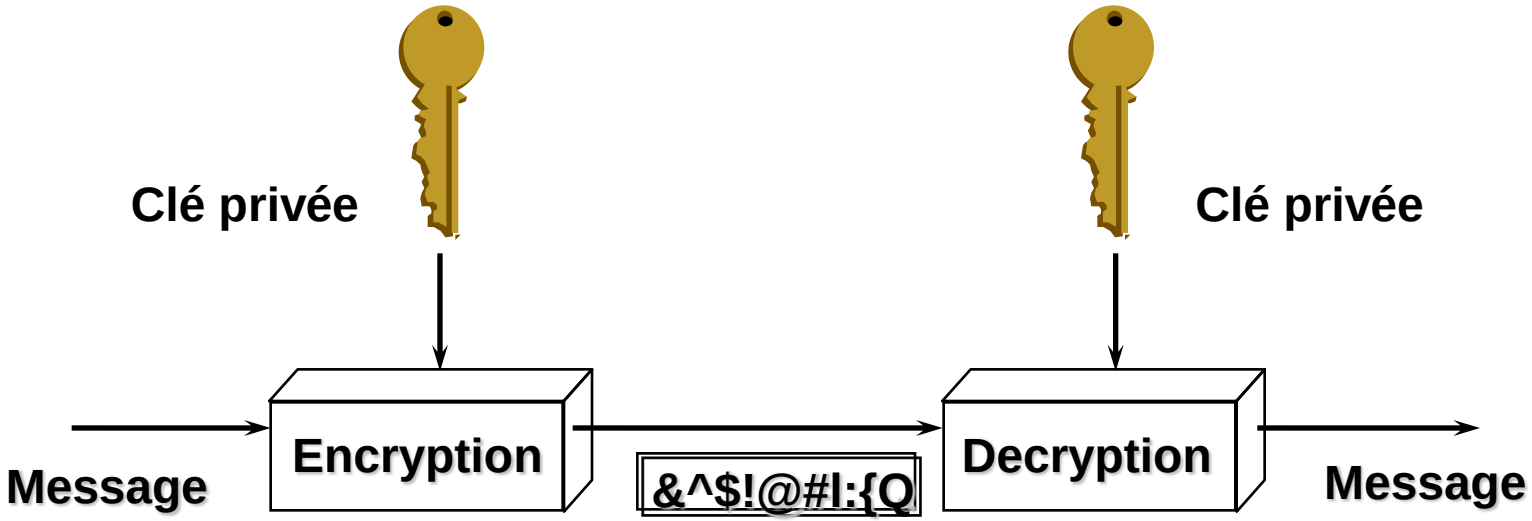
texte en clair

tranférezunmilliarddefrancsàmon
comptesuissenumérotézérozérosept

texte chiffré

TRLEAPSOZTFMDCOIEREEAZIROENERB
NUAANSUZOCRELFMTETEAEIDSMSROPF
SNRNCUMESD

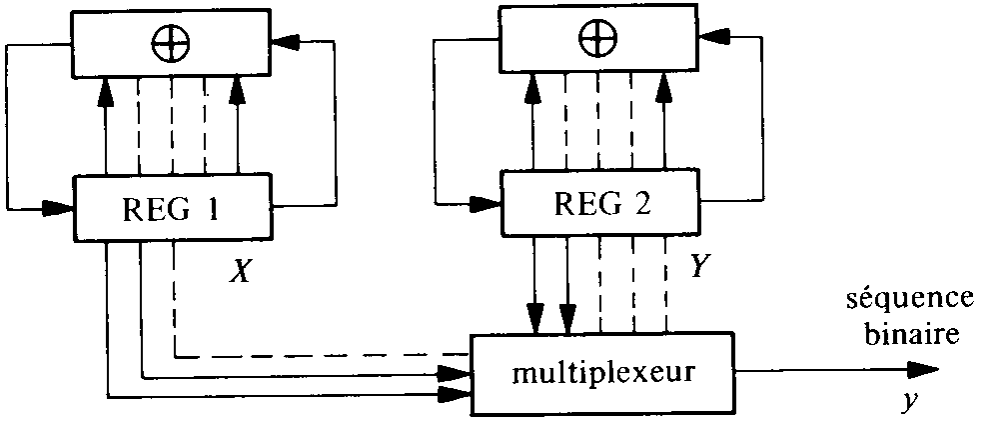
• Chiffrage à clé privée



- ✓ Encryption and decryption use same key
- ✓ Encryption and decryption use same mathematical function
- ✓ Fast
- ✓ Example: Data Encryption Standard (DES, IDEA ,RC2, ...)

• Challenges with symmetric encryption

- ✓ Key length matters
- ✓ Keys must often be changed
- ✓ Shared keys must be generated and distributed securely
  - Ramdomized Key generator

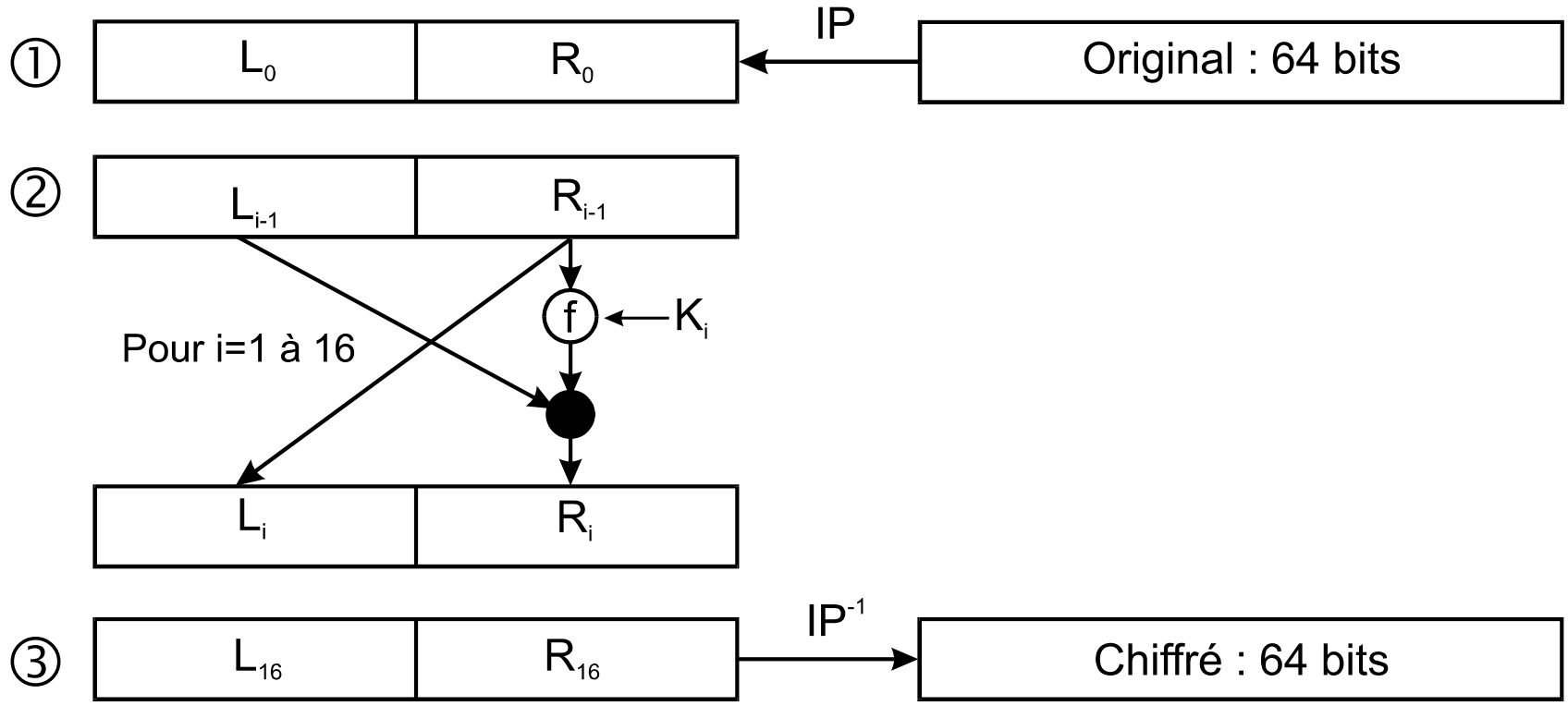


- **DES (Data Encryption Standard / IBM 1977)**

⇨ Un ensemble de permutations / substitutions

- Mot de 64 bits
- Clé de 56 bits
- 16 rondes

✓Principe



- **DES (Data Encryption Standard / IBM 1977)**

IP

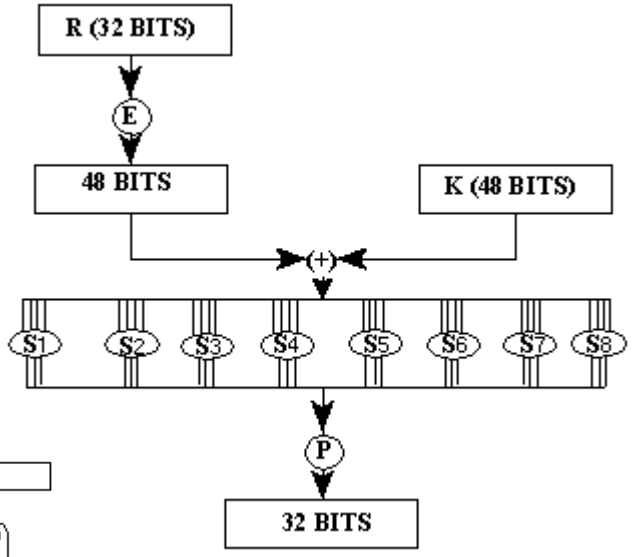
58 50 42 34 26 18 10 2
60 52 44 36 28 20 12 4
62 54 46 38 30 22 14 6
64 56 48 40 32 24 16 8
57 49 41 33 25 17 9 1
59 51 43 35 27 19 11 3
61 53 45 37 29 21 13 5
63 55 47 39 31 23 15 7

IP^{-1}

40 8 48 16 56 24 64 32
39 7 47 15 55 23 63 31
38 6 46 14 54 22 62 30
37 5 45 13 53 21 61 29
36 4 44 12 52 20 60 28
35 3 43 11 51 19 59 27
34 2 42 10 50 18 58 26
33 1 41 9 49 17 57 25

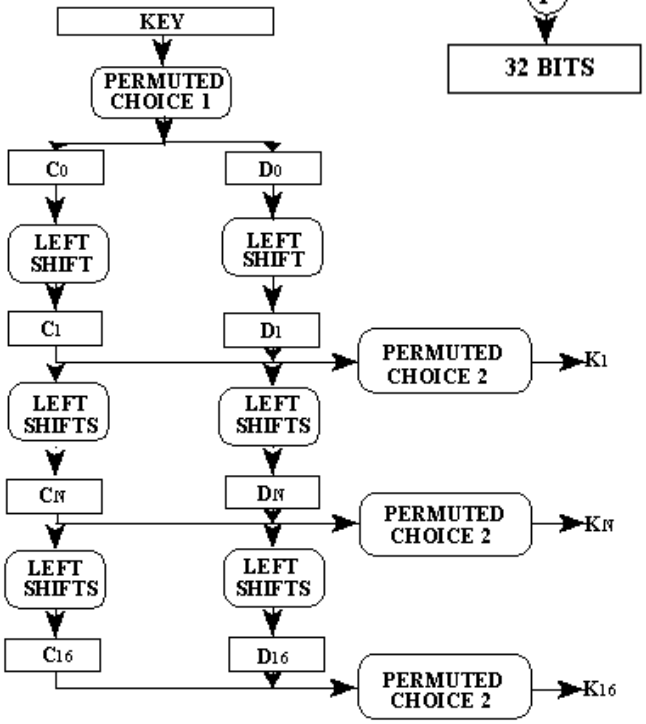
✓Fonction *f*

- Extension à 48 bits (E)
- Xor avec clé secondaire (K_i)
- Réduction (S_j)
- Permutation (PI)



✓Génération des clés secondaires

- Permutation (PC1)
- Décalage (LS)
- Réduction (PC2)



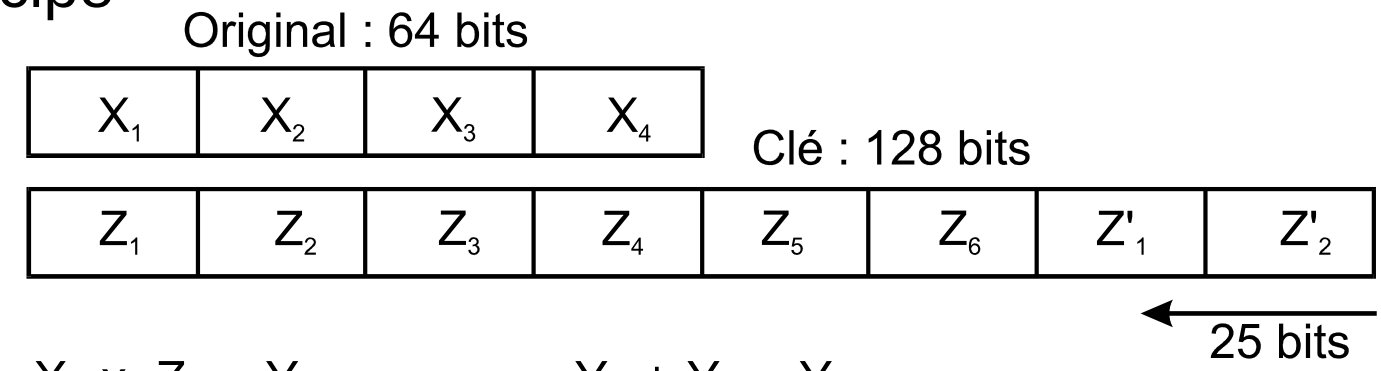
💣 La **clé** à échanger est à garder **secrète**

- **IDEA (International Data Encryption Algorithm / *Lai, Massey* 1991)**

⇨ Une succession d'addition (**+**) , multiplication (**x**), et Xor (⊕)

- Mot de 64 bits
- Clé de 128 bits
- 8 rondes

✓Principe



- $X_1 \times Z_1 = Y_1$
- $X_2 + Z_2 = Y_2$
- $X_3 + Z_3 = Y_3$
- $X_4 \times Z_4 = Y_4$
- $Y_1 \oplus Y_3 = Y_5$
- $Y_2 \oplus Y_4 = Y_6$
- $Y_2 \times Z_5 = Y_7$

⇨

- $Y_6 + Y_7 = Y_8$
- $Y_8 \times Z_6 = Y_9$
- $Y_7 + Y_9 = Y_{10}$
- $Y_1 \oplus Y_9 = X_1'$
- $Y_3 \oplus Y_9 = X_3'$
- $Y_2 \oplus Y_{10} = X_2'$
- $Y_4 \oplus Y_{10} = X_4'$

- $X_1 \times Z_1 = X_1'$
- $X_2 + Z_2 = X_2'$
- $X_3 + Z_3 = X_3'$
- $X_4 \times Z_4 = X_4'$

- **DES / IDEA**

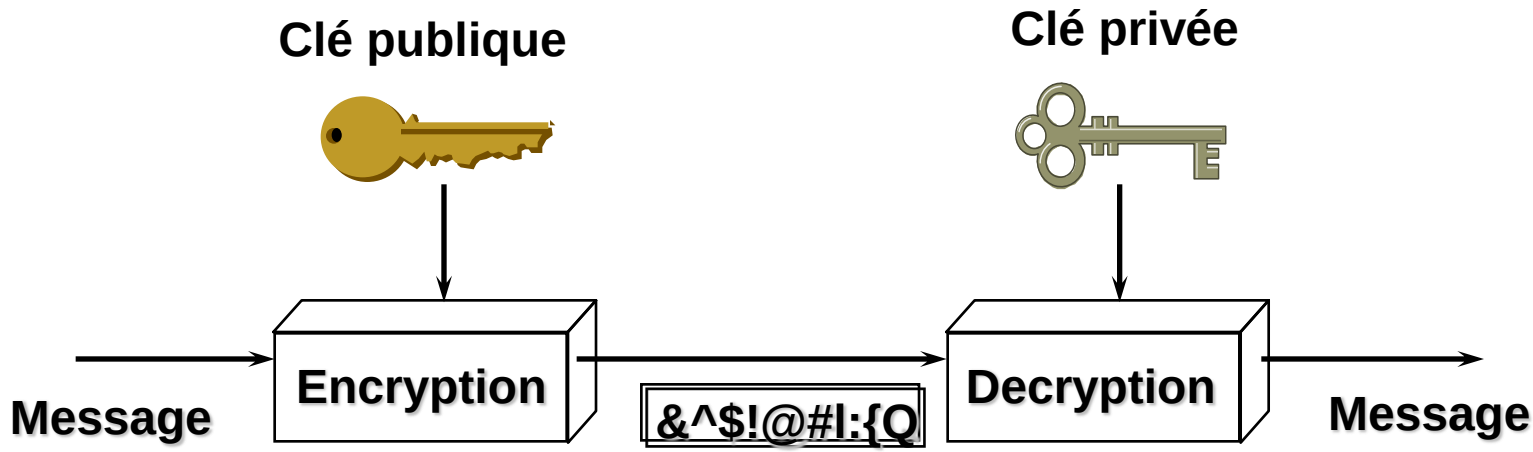
	Taille des blocs	Taille de clé	Nombre de rondes
DES	64	56	16
IDEA	64	128	8

✓ IDEA est deux fois plus rapide !

✓ Chip VLSI IDEA ⇨ 200 Mb/s

✓ IDEA le remplaçant de DES

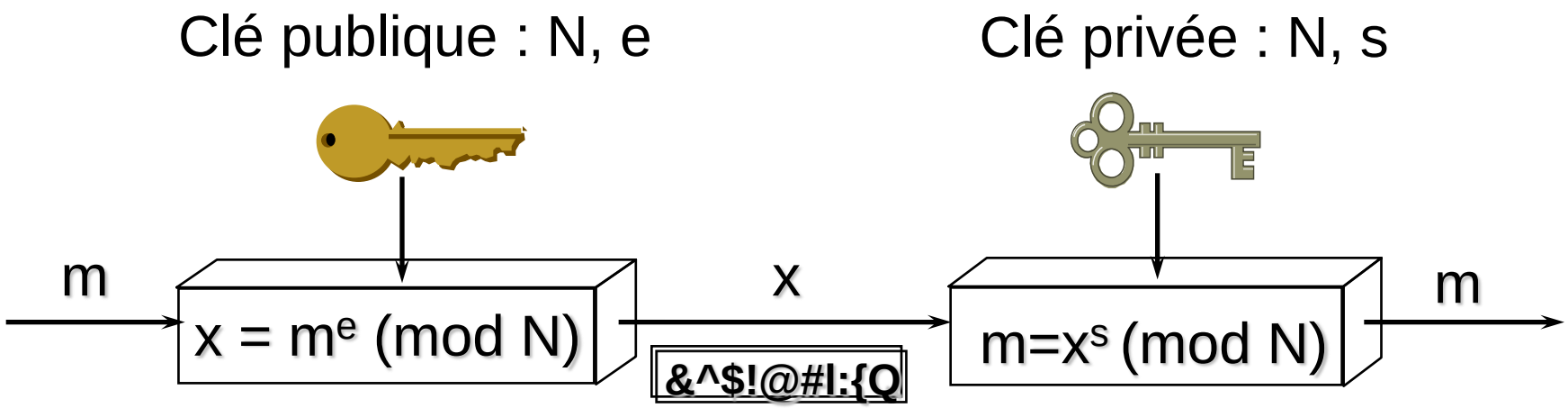
• Chiffrage à clé publique



- ✓ Encryptor and decryptor use different keys
- ✓ Encryptor and decryptor use different mathematical functions
- ✓ Slow
- ✓ Example: public key algorithms (RSA, Diffie-Hellman, ...)

- **RSA (Rivest Shamir Adleman / 1978)**

⇨ Basé sur des propriétés algébriques : - multiplication ☺
- factorisation ☹



- Choisir N = p . q avec p et q premiers (512 bits soit # 200 chiffres)
- Choisir s / s premier avec z=(p-1) .(q-1)
- e / e. s = 1 (mod z) e<<s

💣 Sécurité dépend des connaissances arithmétiques !

✓Exemple simple de RSA

- p=3 et q=11 ⇨ N = 33 ⇨ z = 20
- s = 7 ⇨ 7.e = 1 (mod 20) ⇨ e = 3
- $C = M^3 . (\text{mod } 33)$ et $M = C^7 . (\text{mod } 33)$

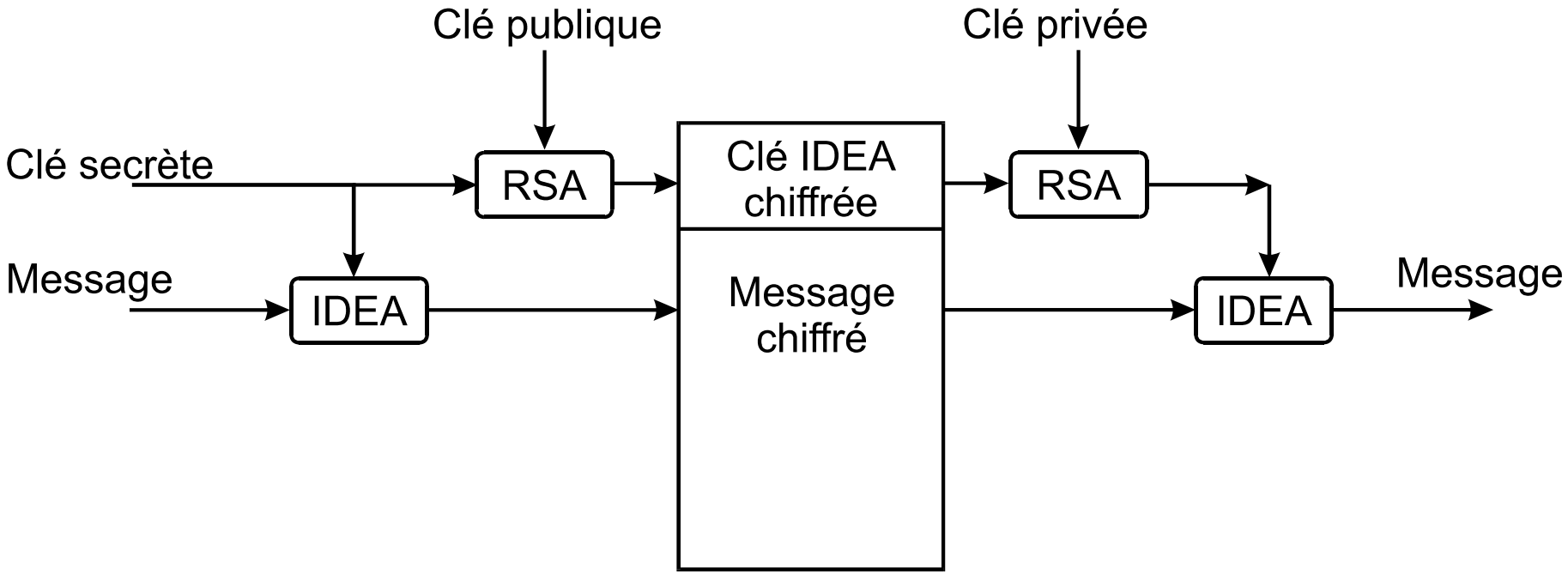
Texte en clair (M)		Texte chiffré (C)			Après déchiffrage	
Carac-tère	Valeur	P^3	$P^3 (\text{mod } 33)$	C^7	$C^7 (\text{mod } 33)$	Carac-tère
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	1	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	5	E

Calculs de l'émetteur

Calculs du récepteur

- **PGP (Pretty Good Privacy / 1991)**

⇨ Algorithme hybride : PGP = (RSA + IDEA)



💣 Interdit en France ! (jusqu'en 1999)

- **Comparaison**

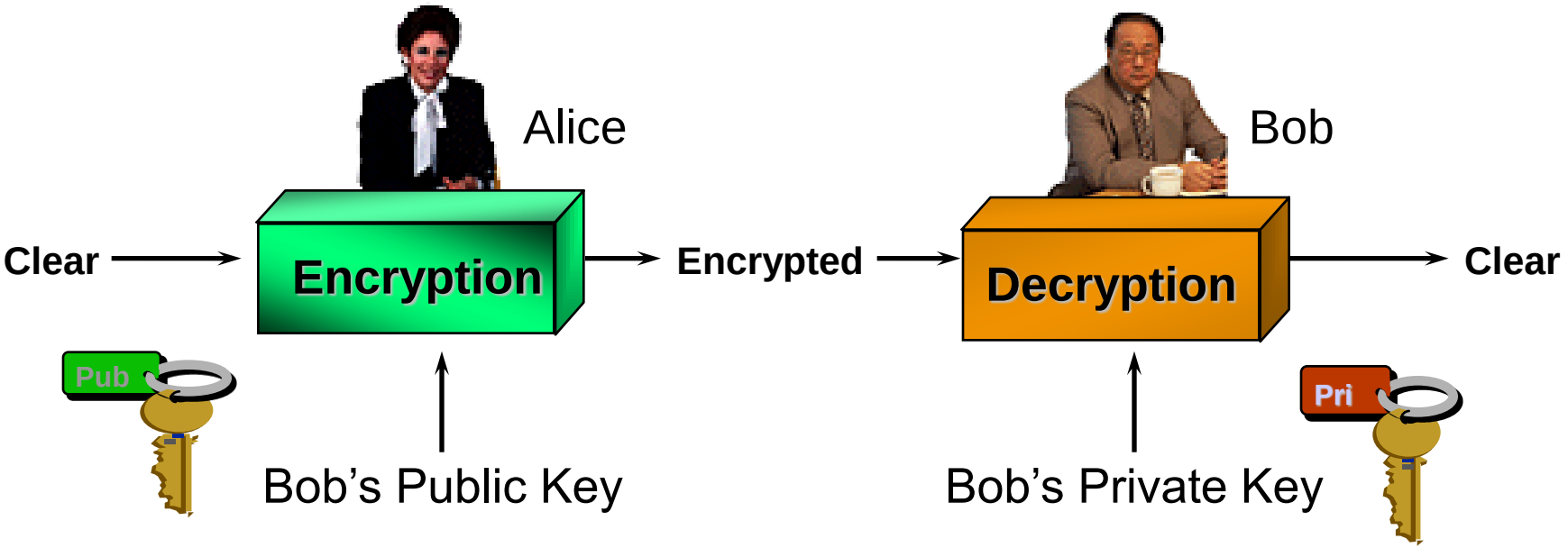
	Symmetric	Asymmetric
Number of keys	1	2
Usual key length	56 bits	512+ bits
Performance	fast	very slow
Dedicated hardware	yes	very rare
Code breaking	difficult	almost impossible

Nombre de personnes	Nombre de clés secrètes	Nombre total de clés privées ET publiques
2	1	4
3	3	6
4	6	8
5	10	10
6	15	12
7	21	14
8	28	16
9	36	18
10	45	20
15	105	30
20	190	40
50	1 225	100
100	4 950	200
500	124 750	1000
1 000	499 500	2000
10 000	49 995 000	20 000
n	n(n-1)/2	2n

✓ Usage des approches clé publique

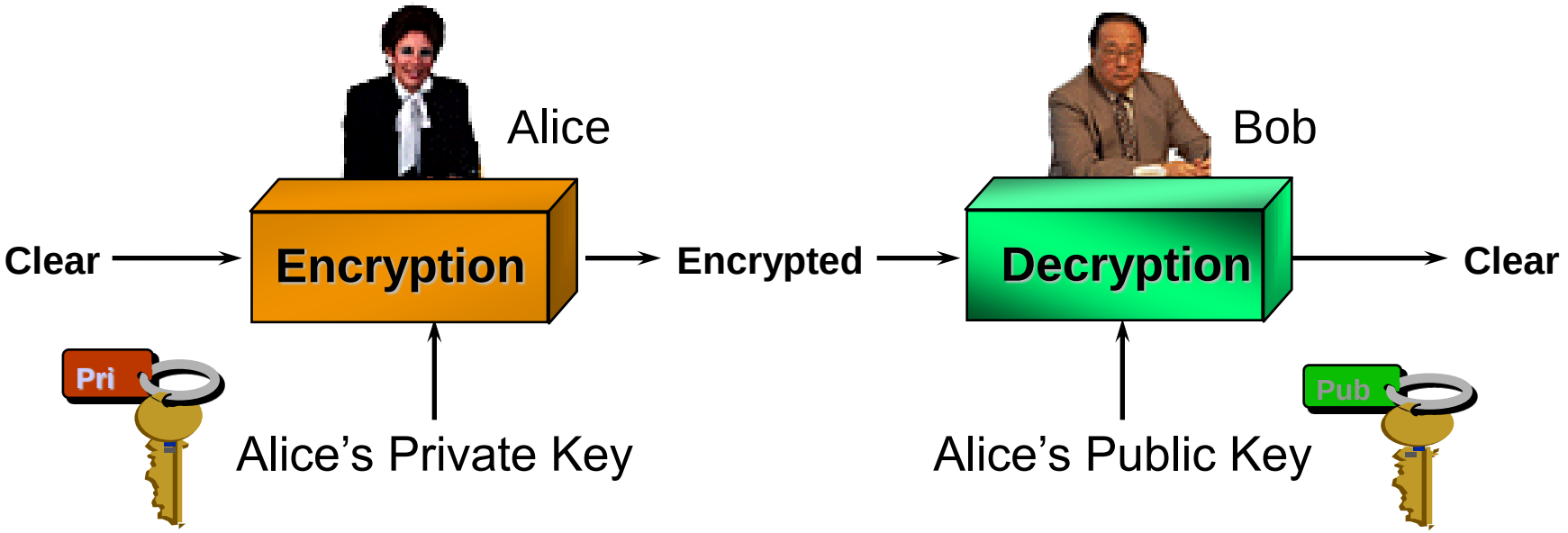
- **Confidentialité**
- **Authentification**
- **Confidentialité & authentification**
- **Signature**
- **Certificat**
- **Échanges sécurisés**

• Confidentialité



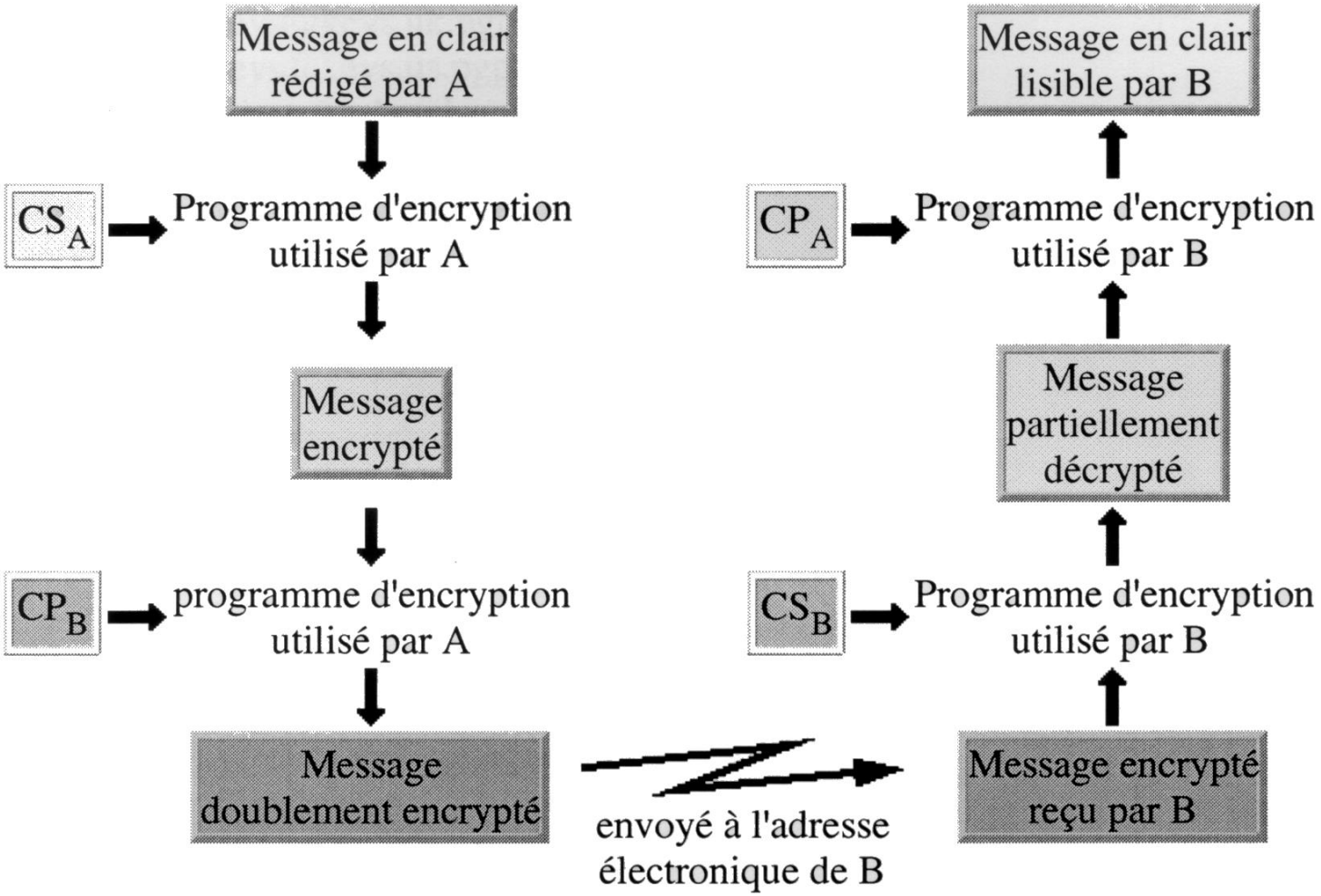
- Alice gets Bob's public key
- Alice encrypts message with Bob's public key
- Bob decrypts using his private key

• Authentification



- Alice encrypts message with her private key
- Bob gets Alice's public key
- Bob decrypts using Alice's public key

• Confidentialité & Authentification



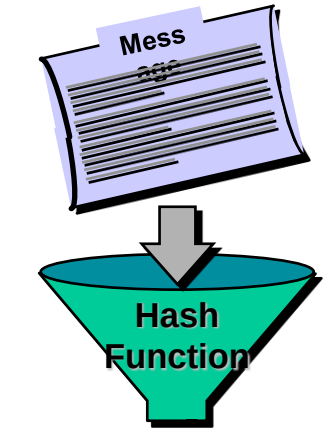
• Signature : Authentification & Intégrité

✓ DSS

- *Digital Signature Standard* from NIST
- Public and private keys (512+ bits)
- Applied on a *digest* of the message to be signed

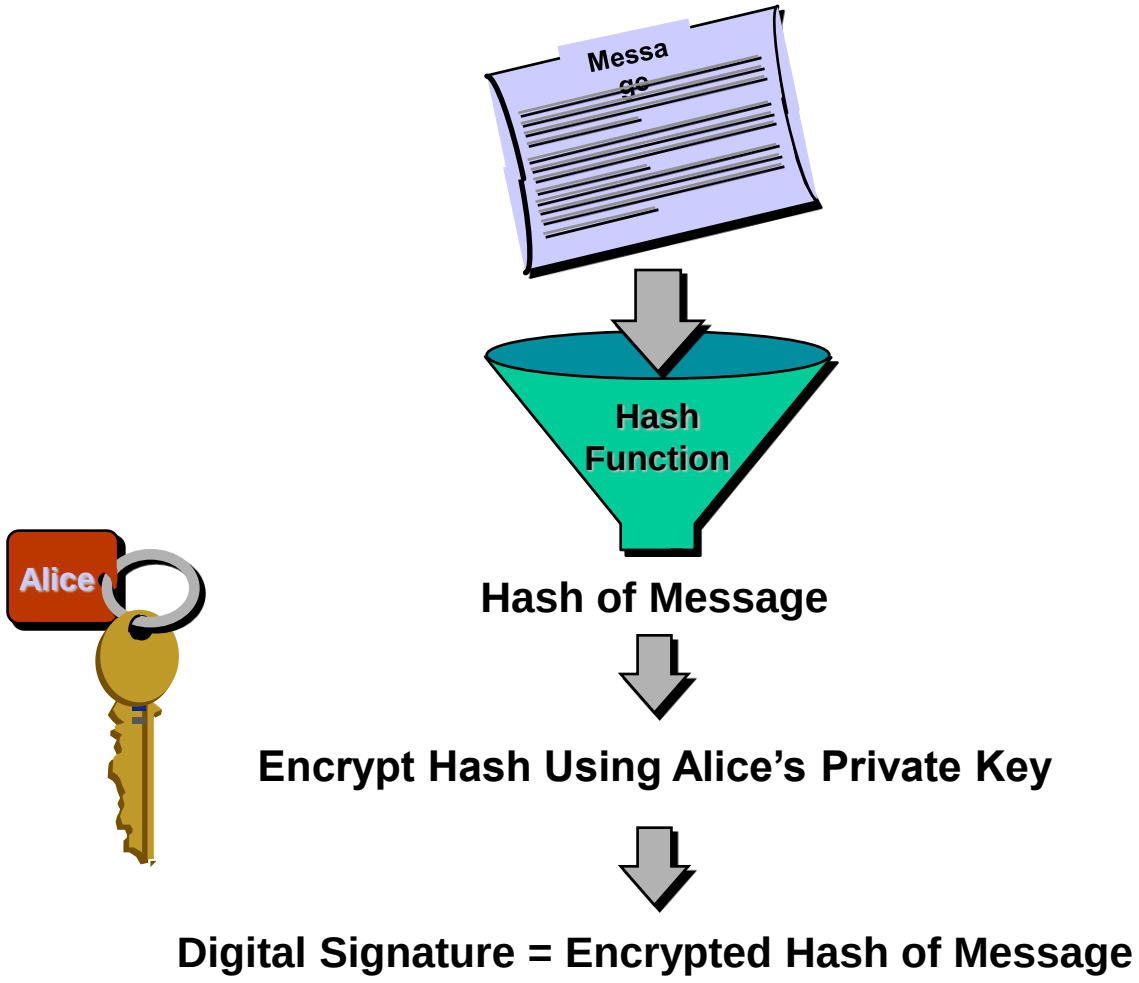
✓ Digest (Hash)

- one-way cryptographic function
- maps a large message into a short hash
- typical hash size 128 bits
- examples: MD5, SHA

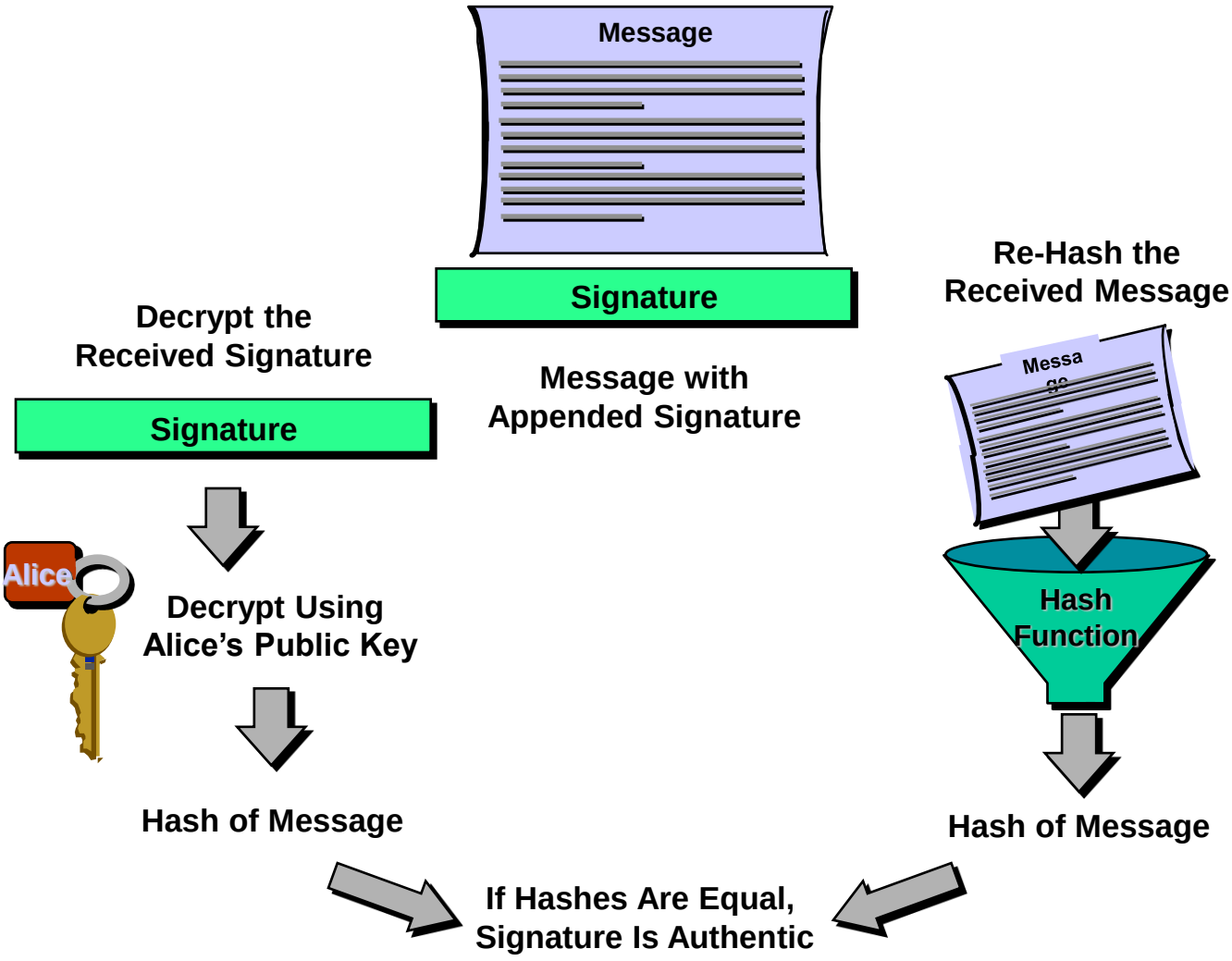


Hash of Message

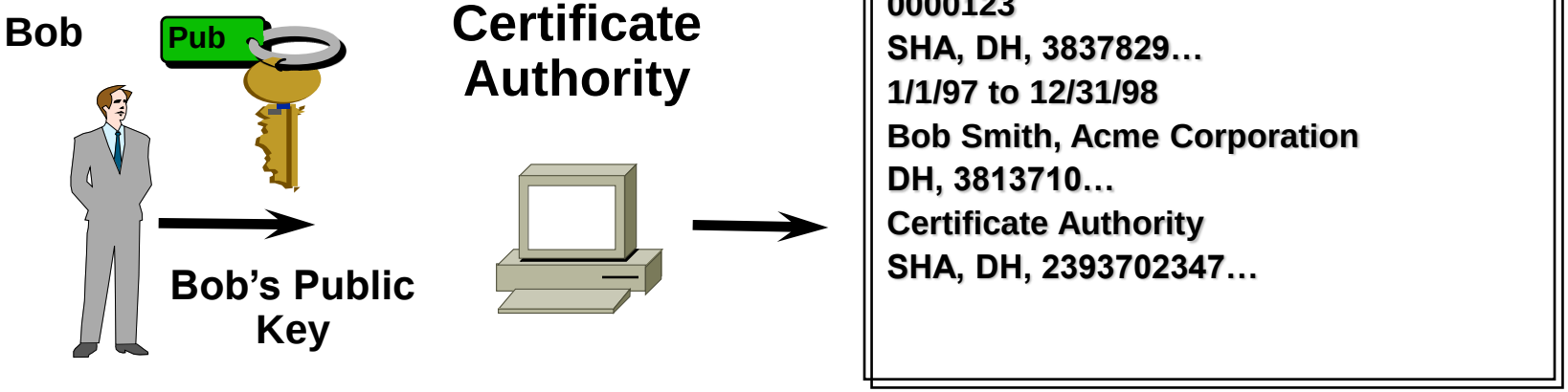
- **How does Alice sign her message?**



- **How does Bob verify Alice's signature?**



- **How can Bob be assured that the Alice's public key belongs to Alice?**



- **Digital certificate** is signed message that attests to authenticity of user's public key

• Certificat : l'identité électronique

- A digital certificate contains
  - Serial number of the certificate
  - Issuer algorithm information
  - Valid to/from date
  - User public key information
  - Signature of issuing authority

0000123
SHA,DH, 3837829....
1/1/93 to 12/31/98
Alice Smith, Acme Corp
DH, 3813710...
Acme Corporation, Security Dept.
SHA,DH, 2393702347 ...

- Tiers de confiance / sequestre
- Norme CCITT **X. 509**

- **Protocoles réseaux sécurisés**

  - ✓ **SSL (Secure Socket Layer)**
  - ✓ **SET (Secure Electronic Transaction)**
  - ✓ **Secure HTTP**
  - ✓ **Secure TCP/IP ⇨ IP v.6**
  - ✓ **...**

- **S S L**

  - ✓ **Communication sécurisée entre deux entités**

  - ✓ **Protocole de handshake**
    - Client vérifie le certificat du serveur
    - Client génère paire de clé
    - Demande la clé publique du serveur
    - Envoie de la clé publique du client chiffrée au serveur
    - Test émis par le serveur

  - ✓ **Échange de données sur liaison sécurisée**

⇨ Commerce électronique

• Commerce électronique

✓ Evolution exponentielle, initiée par les professionnels, tirée par les particuliers

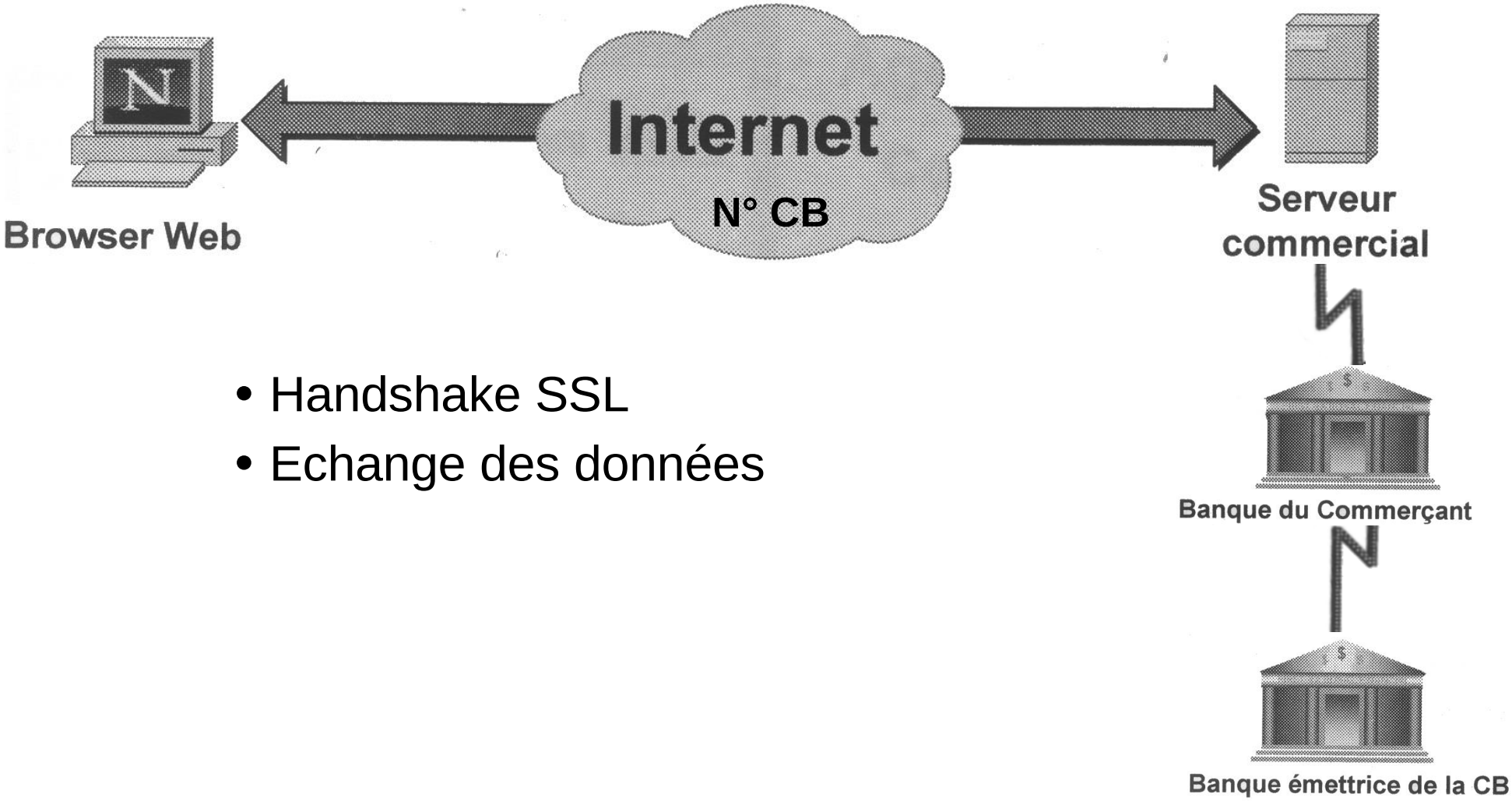
✓ Pose tous les problèmes traités par la cryptologie

- Authentification
- Confidentialité
- Intégrité
- Non répudiation

✓ 2 voies principales

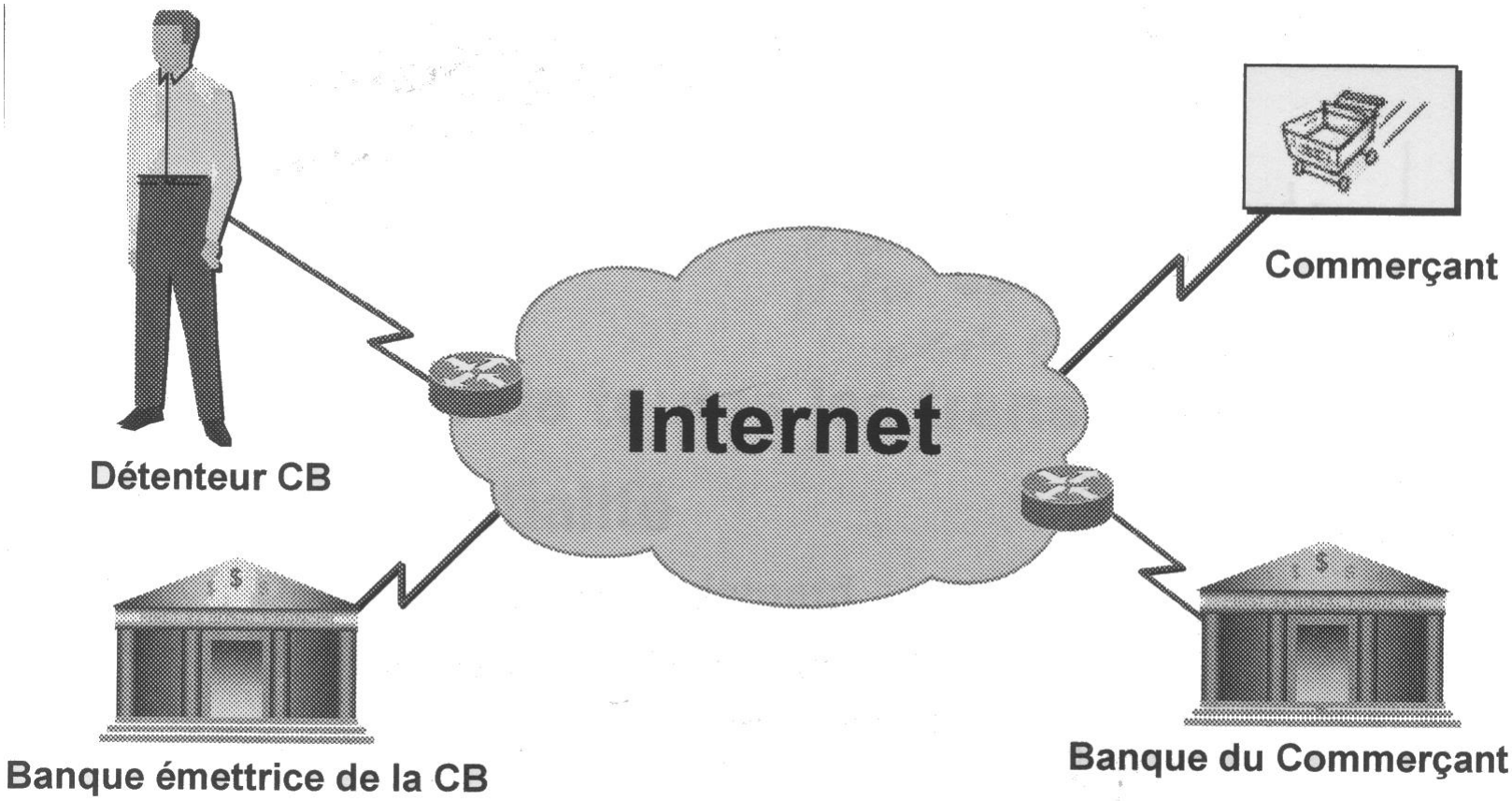
- Acheteur / Vendeur ⇨ SSL
- Acheteur / Vendeur + Banques ⇨ SET

✓ Commerce Acheteur / Vendeur via SSL



- Handshake SSL
- Echange des données

✓Secure Electronic Transaction (SET)



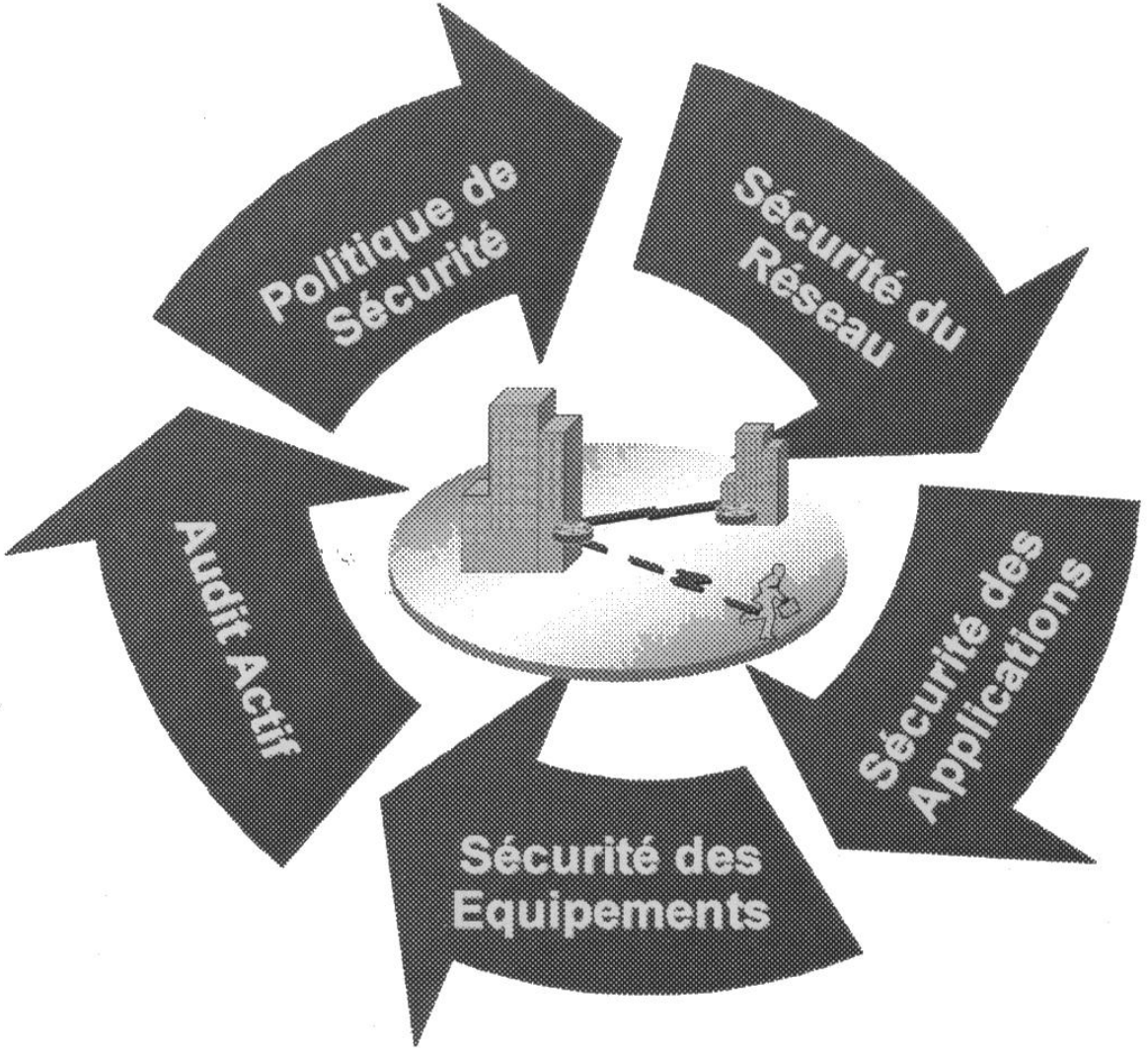
✓ SET prend en charge

- Authentification acheteur & Vendeur
- Intégrité des transmissions
- Confidentialité du paiement et de la commande

✓ Déroulement d'une vente SET

- Demande de l'acheteur
- Vendeur vérifie la commande
- Les banques vérifient Vendeur & Acheteur
- Acquittement de l'ordre

Cryptologie = élément essentiel du cycle de sécurité



- **Conclusion sur la cryptograhie**

✓Indispensable aux réseaux de communication

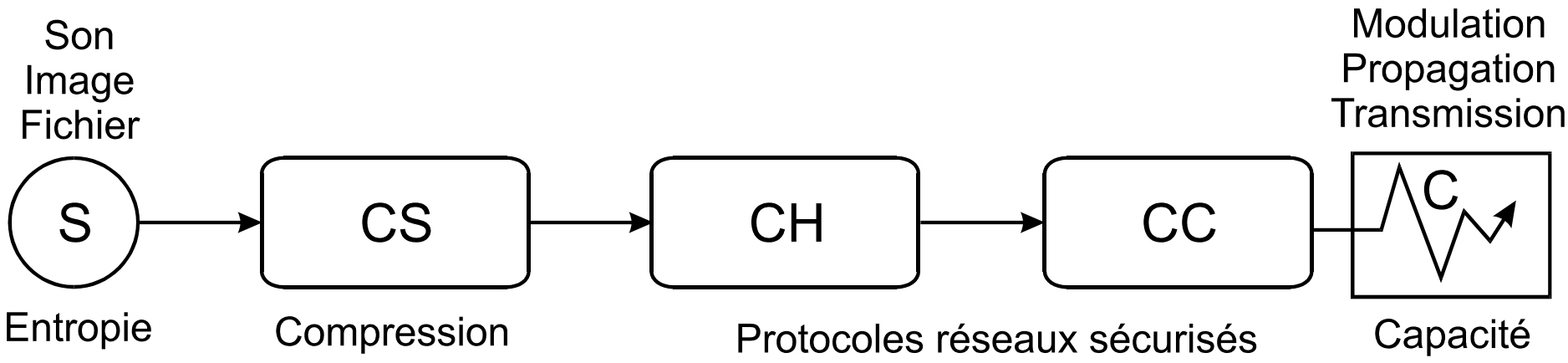
⇨ Sécurité Intranet / Extranet / Internet

✓Moteur de développement du @Business

✓Conséquences juridiques

7. Conclusion

Théorie de l'information ⇨ Domaine vaste (Continu, Modèle de réseaux, Théorie de la distorsion, …)



Théorie de l'information

Exercices

Exercices A

Exercice n° A.1

Dans un processus d'automatisation, une source génère de façon indépendante quatre niveaux de tension :
$x_{1=}$1 V, x_2=2 V, x_3=3 V, x_4=4 V.
La durée du niveau x_1 est $t_{1=}$1ms, celle du niveau x_2 est t_2=0,5 ms, celle du niveau x_3 est t_3=0,1 ms et enfin, celle du niveau x_4 est t_4=1 ms.
Les niveaux ci-dessus sont générés avec les probabilités suivantes :

$$p(x_1)=\frac{1}{8},\ p(x_2)=\frac{1}{4},\ p(x_3)=\frac{1}{2},\ p(x_4)=\frac{1}{8}$$

a) Quel est le débit d'information de cette source ?
b) Après une succession de 10 symboles, la source se met au repos (émet le niveau 0) pendant 15 ms. Quel est le débit d'information de cette source ?

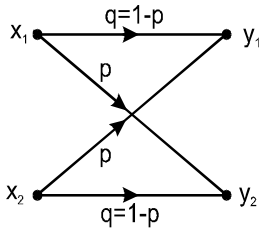
Exercice n° A.2

Un signal vidéo est transmis au travers un filtre idéal passe-bas ayant la fréquence de coupure de 5 MHz, après quoi il est échantillonné à la fréquence de Nyquist (2xFc=10 MHz). Les échantillons sont quantifiés uniformément par un dispositif de quantification uniforme possédant 256 niveaux et codés sur 8 bits.

➔ Calculez le débit de la source vidéo ainsi numérisée ?

Exercice n° A.3 : *Canal binaire symétrique*

Le schéma suivant représente la forme générale d'un canal binaire symétrique et permet d'obtenir la matrice de bruit [P(Y/X)].



a) Calculez la capacité C de ce canal. Que remarquez vous ?

b) Représentez la courbe C=f(p). Commentez les points d'intérêt.

c) Pour quelles valleurs de p(x) et de p(y) la transinformation est elle maximale ?

d) Application numérique

Avec la matrice de bruit suivante :

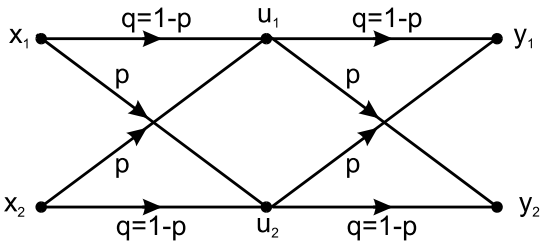
$$[P(Y/X)] = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} \quad \text{et} \quad [P(X)] = \begin{bmatrix} 1/4 \\ 3/4 \end{bmatrix}$$

Calculez :

- l'entropie du champ à l'entrée du canal
- l'entropie du champ à la sortie du canal
- l'entropie conjointe
- l'erreur moyenne et l'équivoque
- la transinformation
- la capacité du canal
- la redondance et l'efficacité de ce canal.
- Comment faut il modifier P[X] pour utiliser au mieux le canal.

Exercice n° A.4

Deux canaux binaires symétriques, ayant la même capacité C sont reliés en cascade.



a) Quel est la valeur de la capacité résultante ?

c) Calculez la capacité d'un seul canal et celle du canal résultant pour p=0,1.

Exercices B

Exercice n° B.1 : *Codage optimal*
Une source S génère 7 symboles s_i avec les probabilités suivantes : $p(s_1)=1/3$, $p(s_2)=1/3$, $p(s_3)=1/9$, $p(s_4)=1/9$, $p(s_5)=1/27$, $p(s_6)=1/27$, $p(s_7)=1/27$.

a) Construire un code binaire optimal et calculer son efficacité et sa redondance.

b) Construire un code optimal ayant l'alphabet $X=\{A,B,C\}$ et calculer son efficacité.

Exercice n° B.2 : *Codage de source Qaire*
Une source binaire S_1 génère les symboles s_1 et s_2 avec les probabilités $p(s_1)=0,9$ et $p(s_2)=0,1$. Les deux symboles sont indépendants.

a) Calculer l'entropie de la source S1 puis celle des sources S_2 et S_3 respectivement constituées de n=2 puis n=3 symboles de la source S_1.

b) Calculer les codes de Huffman associés à S_1, S_2, S_3 ainsi que leur efficacité et la longueur moyenne des mots-codes. Conclusion.

Exercice n° B.3 : *Efficacité*
Une source S génère de manière indépendante 6 symboles s_i avec les probabilités suivantes : $p(s_1)=0.40$, $p(s_2)=??$, $p(s_3)=0.25$, $p(s_4)=??$, $p(s_5)=0.15$, $p(s_6)=0.1$.

a) Quelles sont les valeurs des probabilités $p(s_2)$ et $p(s_4)$ qui permettront d'avoir le codeur binaire d'Huffman le plus efficace ?
b) Quelle est cette efficacité ?

Exercice n° B.4 : *Codage LZW d'une source binaire*
Une source binaire S génère de manière indépendante 2 symboles '0' et '1'. Voici un début de message émis par cette source et qui est codé par un codeur LZW.
0100010010010110101010010101010100100 …..

a) Donnez les 10 premiers mots du dictionnaire de ce codeur et proposez un codage des mots du dictionnaire.

b) Donnez alors le code de ce message.

Exercices C

Exercice C.1 : *Code de Hamming*

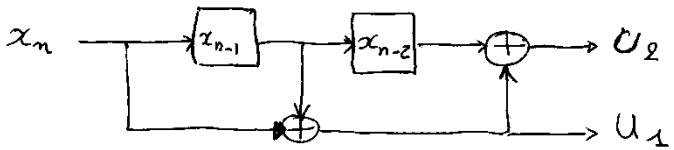
Soient la source S qui émet les cinq symboles suivants et leur code associé :

S_i	C_i
A	000
B	001
C	010
F	011
E	100

a) Détermine les codes associés à chaque lettre lorsque l'on met en œuvre un codeur de Hamming systématique.
b) Donnez l'expression des correcteurs.
c) Quelle est la distance minimale de ce code ?
d) Quel est son taux d'émission R = (Nbre de symboles d'info) / (longueur de mot-code) ?
e) On désire en plus de la correction d'une erreur, détecter toutes les erreurs affectant un nombre impair de bits. Proposez une solution et les mots-codes associés. Quel sont le nouveau taux d'émission et la distance minimale de ce nouveau code ?
f) Commentez les différents scénarios possibles au décodage.

Exercice C.2 : *Code convolutif*

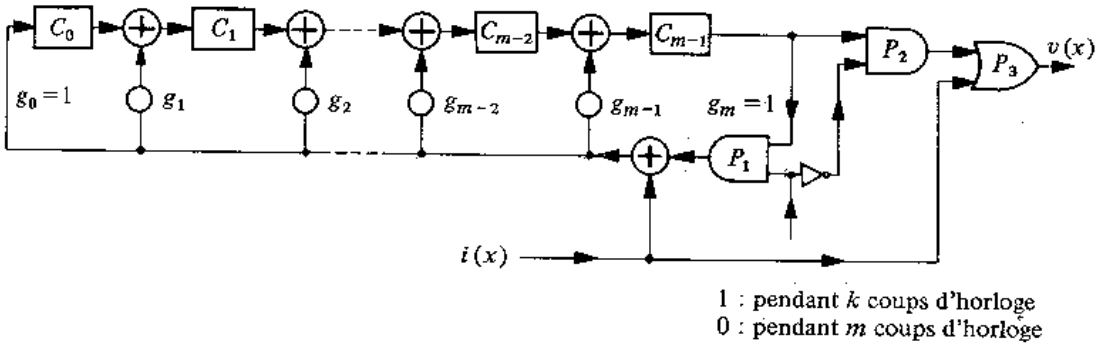
Soit le codeur convolutif suivant :



a) Quelles sont les relations entre u1, u2 et les symboles d'information ? En déduire la matrice de codage. Quel est le taux d'émission de ce codeur et la contrainte ?
b) Donnez le diagramme d'état de ce codeur.
c) En déduire si les bouts de séquence suivants ont pu être générés par ce codeur :
   1) 11100001
   2) 11011011
   3) 11001001
d) On applique l'algorithme de Vitterbi tous les 3 tops d'horloge. Le décodeur étant dans l'état 00, comment va t-il corriger 001010 ?
e) Ce résultat était t-il prévisible ?
f) Que peut on faire pour améliorer la capacité de décodage ? Quelle en est la conséquence ? Quelle est la limite de correction de ce codeur ?

Exercice C.3 : *Code cyclique*

Le '(n-k) stage shift register encoding circuit' est un circuit de codage par division utilisé dans les satellites pour son faible coût de mise en œuvre. Un schéma général de ce type de circuit est donné ci-dessous.



Durant l'introduction des k symboles d'information, la porte P_1 reste ouverte et le circuit calcule le reste de la division qui apparaît au coup d'horloge k+1, lorsque la porte P_2 est ouverte tandis que la porte P_1 est fermée. La porte P_2 reste ouverte durant m coups d'horloge pour permettre d'évacuer le registre, plus exactement les symboles de contrôle. Ces derniers sont placés, par le truchement de la porte P_3, à la suite des symboles d'information pour former un mot-code systématique.

Soit un codeur à circuit de division par le polynôme $g(x)=x^3+x^2+1$ avec $n=7$, conformément au schéma précédant :

a) Tracer le schéma particulier pour le cas étudié.

b) Trouver au moyen de ce schéma , les valeurs des symboles de contrôle a_0, a_1 et a_2 en fonction des symboles d'information a_3, a_4, a_5 et a_6. (a_6 étant le symbole d'information qui entre au premier top d'horloge)

c) Soit le mot d'information I=[1 0 0 1], donner le mot-code obtenu avec $g(x)=x^3+x^2+1$ par codage par division.

d) En utilisant un codage par division, donnant les expressions générales liants caractères de contrôle et d'information.

e) Même question que c) en effectuant un codage par multiplication. Remarques par rapport au mot obtenu en c) ?

f) Avec le même polynôme g(x), pour un codage par multiplication, donnez l'expression générale des élément du mot-code en fonction des symboles d'information.

g) Comment décode t-on les mot-codes après un codage par division ? par multiplication ?

Vérifiez que les mot-codes générés en c) et e) sont décodables.