Information, Calcul et Communication

Correction d'erreurs : principes de base

Olivier Lévêque

École polytechnique fédérale de Lausanne

Comment transmettre des données ?

Il existe deux moyens de transmission :

- A travers le **temps** : enregistrement sur un support
  (et les données peuvent lues plus tard)
- A travers **l'espace** :
  - par câble électrique ou fibre optique (téléphonie, internet)
  - dans l'air (téléphonie sans fil, wifi)

Problème commun : des erreurs surviennent régulièrement !

- lors de l'écriture ou de la lecture de données
- lors de la transmission par câble électrique, fibre optique ou onde électromagnétique

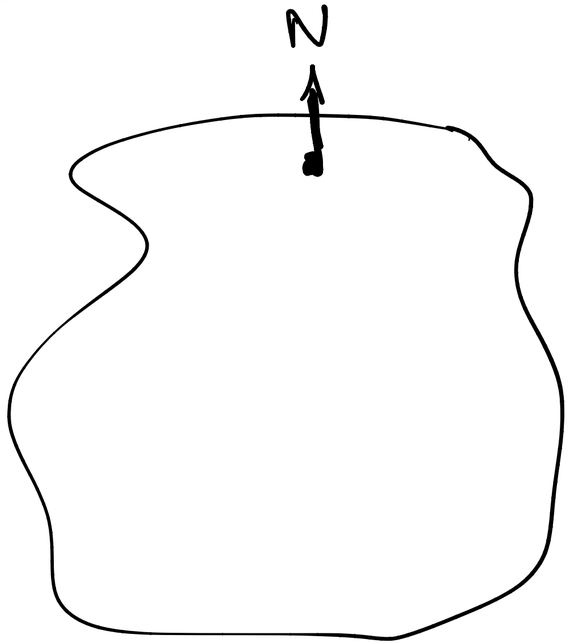
Exemple

- Vous désirez communiquer la direction à prendre (N,S,E,W) à un ami perdu.

Code binaire utilisé :

N	S	E	W
11	10	01	00

- Vous indiquez à votre ami d'aller au nord en envoyant 11.



Exemple

- Vous désirez communiquer la direction à prendre (N,S,E,W) à un ami perdu.

Code binaire utilisé :

N	S	E	W
11	10	01	00

- Vous indiquez à votre ami d'aller au nord en envoyant 11.
- Si votre ami reçoit :

11 Tout se passe bien : votre ami se dirige vers le nord.

1? Un **effacement** survient : votre ami ne sait pas s'il doit se diriger vers le nord ou vers le sud.

10 Une **erreur** survient : votre ami se dirige alors vers le sud !

Correction d'erreurs

modèles d'erreurs :

1. effacements : 0111010 ? 01

2. "erreurs" : 0101101010101

(1→0) position ?

(3. "destruction" : 011101001 ⚠)

Exemples de redondance dans la vie courante

- Un père à ses enfants :

« Mettez vos chaussures… mettez vos chaussures, j'ai dit ! »

- Vous à votre ami :

« Pour venir chez moi, c'est simple :
tu tournes dans la 2e rue à droite après le feu rouge, juste après la station-service ;
j'habite au numéro 3, dans l'immeuble bleu avec les grands balcons »

Les informations en rouge sont redondantes (donc inutiles en théorie…).

Corriger un effacement

- Codage par répétition

N	S	E	W
1111	1100	0011	0000

  - 11?1 → N
  - Simple mais coûteux : il faut envoyer 4 bits au lieu de 2 !

Ex : ?11? → 1111 (N)

11?? → N ou S ?

xor au bit de parité

N : 110

S : 101

E : 011

W : 000

Ex : 10? → 101 (S)

0?1 → 011 (E)

Corriger un effacement

1010?101

1⊕1⊕0⊕1 = 1 ⇒ ? = 0

- Codage par répétition

N	S	E	W
1111	1100	0011	0000

  - 11?1 → N
  - Simple mais coûteux : il faut envoyer 4 bits au lieu de 2 !

- Bit de parité

N	S	E	W
110	101	011	000

  - 1?0 → N
  - Le dernier bit correspond à la somme modulo 2 des deux premiers.

Pour envoyer un message de n bits et corriger un effacement, il faut ajouter :

$$n\ bits \quad | \quad 1\ bit\ de\ parité$$

La taille finale du message est :

$$2n\ bits \quad | \quad n+1\ bits$$

Corriger une erreur

- Doubler chaque bit
  - 1101 → N ou S ??

N	S	E	W
1111	1100	0011	0000

détection d'une erreur possible

mais correction impossible

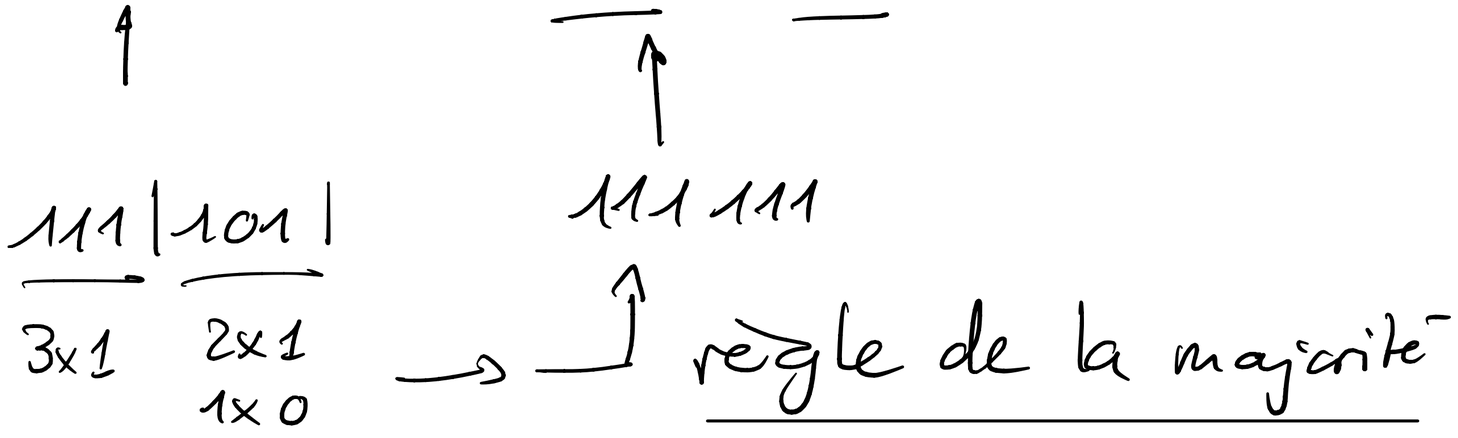
Corriger une erreur

- Doubler chaque bit
  - 1101 → N ou S ??

N	S	E	W
1111	1100	0011	0000

- Tripler chaque bit
  - 111101 → N

N	S	E	W
111111	111000	000111	000000



Corriger une erreur

- Doubler chaque bit
  - 1101 → N ou S ??

N	S	E	W
1111	1100	0011	0000

- Tripler chaque bit
  - 111101 → N

N	S	E	W
111111	111000	000111	000000

On applique ici la **règle de la majorité** : 111101 → N 111100 → S

Mais à nouveau, cette solution est très coûteuse :

Chaque bit est triplé, on aura donc besoin de $3n$ bits pour envoyer un message de n bits à l'origine !

Corriger une erreur

Un meilleure méthode : ajouter *plusieurs* bits de parité

(ajouter 1 bit de parité ne fonctionne pas :

1101**0** → on détecte qu'il y a une erreur

mais on n'est pas capable de la corriger)

Corriger une erreur

Un meilleure méthode : ajouter *plusieurs* bits de parité

Voici un essai :

- Lorsque $n = 4$: on ajoute 2 bits de parité
  - le premier indique la parité de la somme des bits 1 et 2
  - le deuxième indique la parité de la somme des bits 1 et 3

Exemple : Pour envoyer 1101, on envoie 110101.

$x_6 = x_1 \oplus x_3 = 1$

$x_5 = x_1 \oplus x_2 = 0$

- Si on reçoit 100101 :
  - bit 1 et 2 : $1 \oplus 0 = 1 \neq 0$
  - bit 1 et 3 : $1 \oplus 0 = 1$

} On déduit que le bit 2 est faux :
Le message d'origine est donc 110101

2 Pbs : • si le bit 4 est faux → ?

• si le bit 5 (ou 6) est faux, on fait une correction fausse

Corriger une erreur (suite)

Un meilleure méthode : ajouter *plusieurs* bits de parité

Voici un essai :

- Lorsque $n = 4$: on ajoute 2 bits de parité
  - le premier indique la parité de la somme des bits 1 et 2
  - le deuxième indique la parité de la somme des bits 1 et 3

Problème : Pour envoyer 1101, on envoie toujours 110101.

- Si l'erreur survient sur le bit 4, on reçoit alors 110001, mais les deux bits de parité sont corrects → erreur indétectée !

- Pour réparer ce problème, *3 bits de parité* sont en fait nécessaires → code de Hamming

Corriger plusieurs effacements ou erreurs ?

Tout l’art de la théorie du codage (70 ans d’histoire…) consiste à choisir parcimonieusement les bits de parité pour corriger un nombre maximum d’erreurs…

Code correcteur d'erreurs

$C = \{c^{(1)}, \dots, c^{(M)}\}$ **code binaire**

avec $c^{(j)}$ = mot de code de longueur n $\quad (1 \leq j \leq M)$

Distance de Hamming entre deux mots de code:

$$d(c, c') = \#\{1 \leq i \leq n : c_i \neq c_i'\}$$

(ex: $d(001, 010) = 2$)

distance minimale de C : $d = \min_{j \neq k} d(c^{(j)}, c^{(k)})$

Trois paramètres importants :

1°) M = nombre de mots de code
(↔ "quantité d'information" à envoyer)

2°) n = longueur des mots de code
(↔ quantité de redondance)

3°) d = distance minimale entre les mots de code
(↔ capacité de correction)

nb déffacements = ... , nb d'erreurs = ...

Ex 1 : $C = \{ \underset{N}{11}, \underset{S}{10}, \underset{E}{01}, \underset{W}{00} \}$

$n = 4$, $n = 2$, $d = 1$ (= minimum)

ce code ne corrige ni 1 effacement ni 1 erreur

Ex 2 : $C = \{ \underset{N}{111}, \underset{S}{1100}, \underset{E}{0011}, \underset{W}{0000} \}$

$n = 4$, $n = 4$, $d = 2$ (= minimum)

ce code corrige 1 effacement, détecte 1 erreur mais ne permet de la corriger

Ex 3 : $C = \{ \underset{N}{110}, \underset{S}{101}, \underset{E}{011}, \underset{W}{000} \}$

$M = 4, \; n = 3, \; d(c^{(j)}, c^{(k)}) = 2 \quad \forall j \neq k$

$\Rightarrow d = 2$

corrige 1 effacement, détecte 1 erreur
mais ne la corrige

Et en général, combien d'effacements
et combien d'erreurs peuvent être corrigés ?

→ exercices ! (indice : dépend uniquement de d)

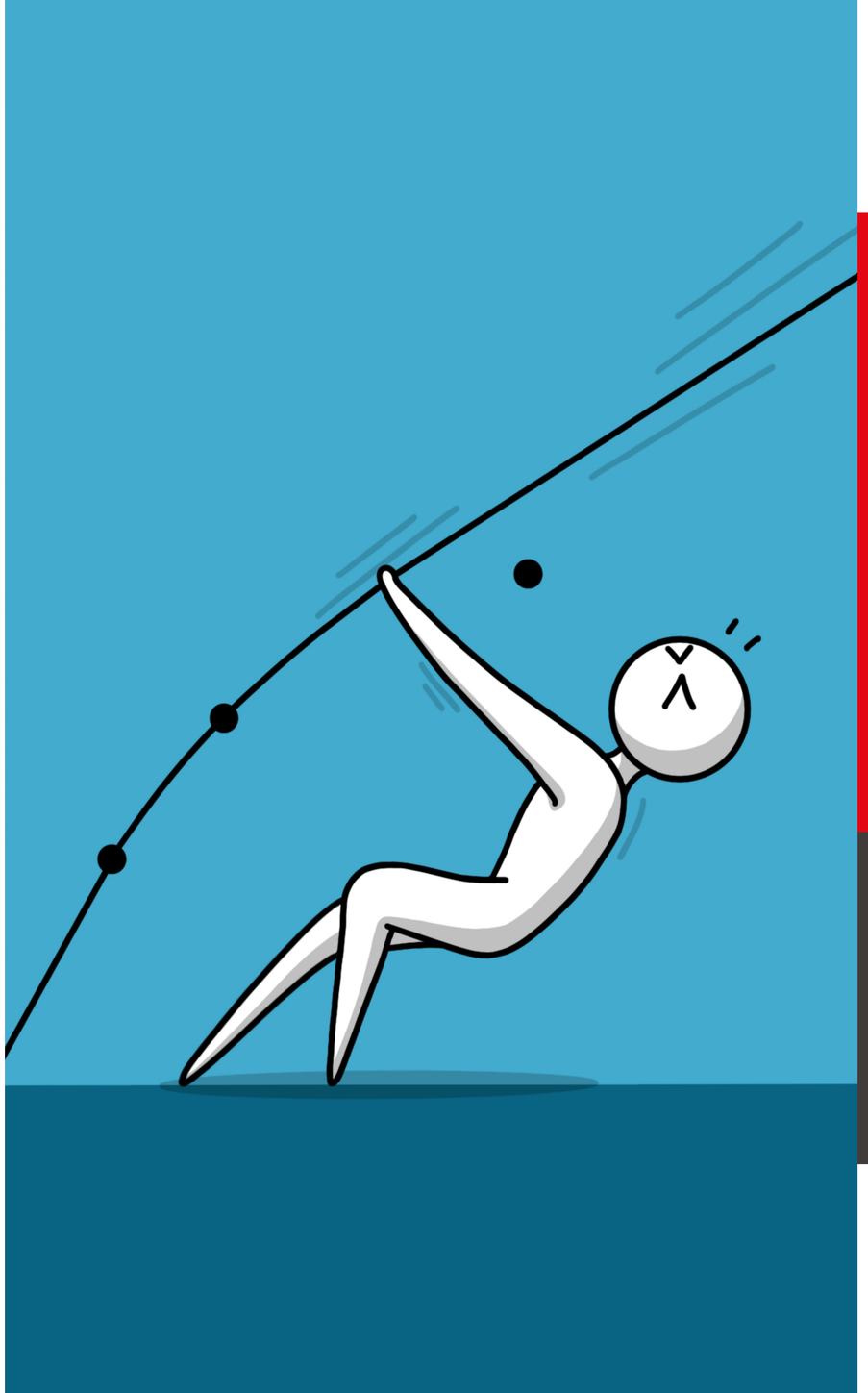
Idée : trouver des codes avec :

- M et d aussi grands que possible
- n aussi petit que possible

⚠ Il y a des limitations :

1°) $n \geq \log_2 M$ ou $M \leq 2^n$

2°) $M \leq 2^{n-d+1}$ (borne de Singleton)



Information, Calcul et Communication

Les codes de Reed-Solomon

Olivier Lévêque

École polytechnique fédérale de Lausanne

Introduction

- Deux personnes (disons Alice et Bob) cherchent à communiquer, mais une proportion non-négligeable des informations transmises par Alice sont effacées/bruitées avant d'être reçues par Bob:

- Les codes de Reed-Solomon sont un bon moyen de gérer une telle situation.
- Pour simplifier leur description, nous allons supposer qu'Alice et Bob travaillent avec des moyens de communication capables de manipuler des *nombres réels* (et non des bits).

Protocole de communication : Envoi

- *Avant de communiquer*, Alice et Bob se mettent d'accord sur un ensemble de n nombres réels *tous différents* :

$$\{t_1, t_2, t_3, \dots, t_n\}$$

- Alice désire envoyer un message x composé d'une suite de nombres réels :

$$x = \{x_1, x_2, x_3, \dots, x_k\} \quad \text{avec} \quad k < n$$

- Elle définit le polynôme suivant :

$$P(t) = \sum_{i=1}^{k} x_i \cdot t^{i-1} \qquad \deg(P) \leq k - 1$$

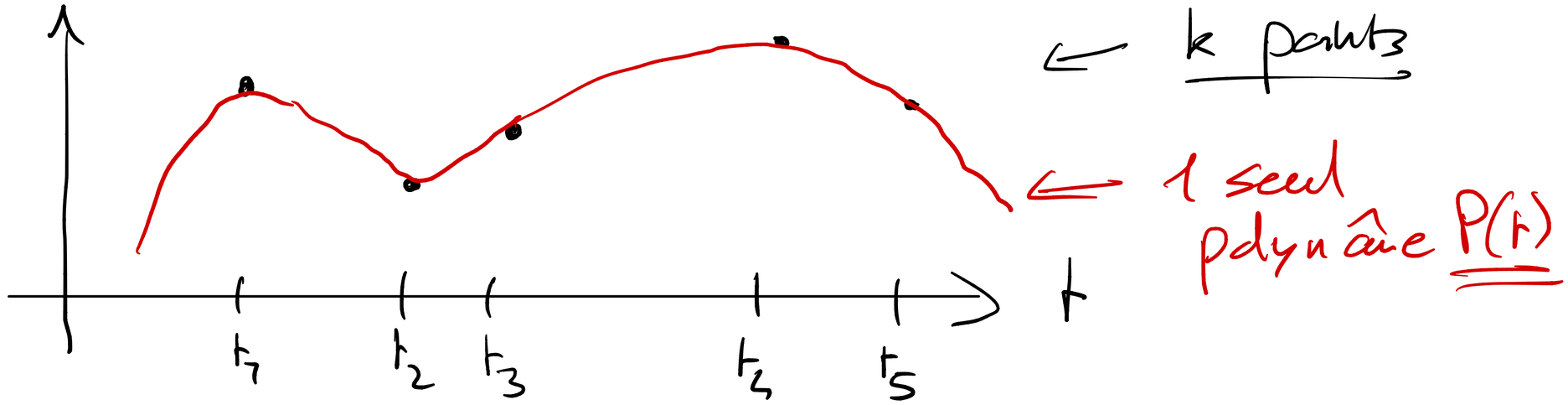
- et envoie finalement le message :

$$y = \{y_1 = P(t_1)\,,\ \ y_2 = P(t_2)\,,\ \ y_3 = P(t_3)\,, \dots,\ \ y_n = P(t_n)\}$$

Protocole de communication : Réception

- Bob reçoit le message y. On suppose que $n - k$ nombres du message sont effacés; il reçoit donc que k nombres de y. Pour simplifier, admettons de plus que Bob ne reçoive que les k premiers nombres $\{y_1, y_2, y_3, \ldots, y_k\}$.

$y_1 = P(t_1)$ ..... $y_k = P(t_k)$



Protocole de communication : Réception

- Bob reçoit le message y. On suppose que $n - k$ nombres du message sont effacés; il reçoit donc que k nombres de y. Pour simplifier, admettons de plus que Bob ne reçoive que les k premiers nombres $\{y_1, y_2, y_3, \dots, y_k\}$.
- Le but de Bob est de retrouver $\{x_1, x_2, x_3, \dots, x_k\}$ à partir de $\{y_1, y_2, y_3, \dots, y_k\}$

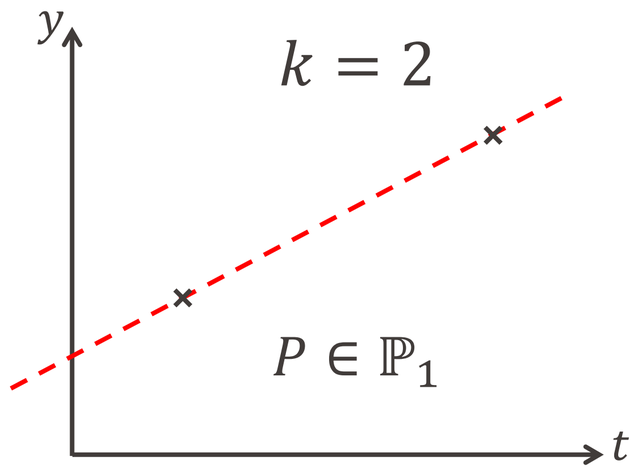
$$\text{Rappelons que } y_j = P(t_j) = \sum_{i=1}^{k} x_i \cdot {t_j}^{i-1} \quad \text{pour} \quad 1 \leq j \leq k$$

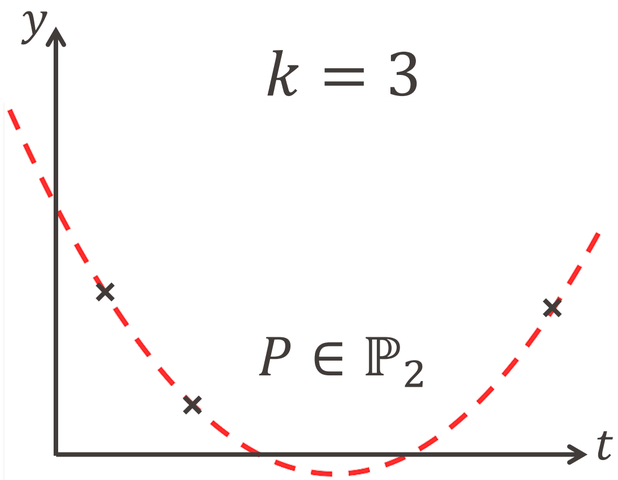
- Il s'agit donc de résoudre un système linéaire de k équations à k inconnues!

$$\text{Exemple pour } k = 3 \text{ et } t = \{1, 2, 3\} : \begin{cases} x_1 \cdot 1 + x_2 \cdot 1 + x_3 \cdot 1 = y_1 \\ x_1 \cdot 1 + x_2 \cdot 2 + x_3 \cdot 4 = y_2 \\ x_1 \cdot 1 + x_2 \cdot 3 + x_3 \cdot 9 = y_3 \end{cases}$$

Une autre façon de visualiser le problème

- En réalité, Bob reçoit les coordonnées des points (t_j, y_j) $\forall\ 1 \leq j \leq k$. Il s'agit donc de trouver l'unique polynôme P tel que $\deg(P) \leq k - 1$ passant par les k points différents.





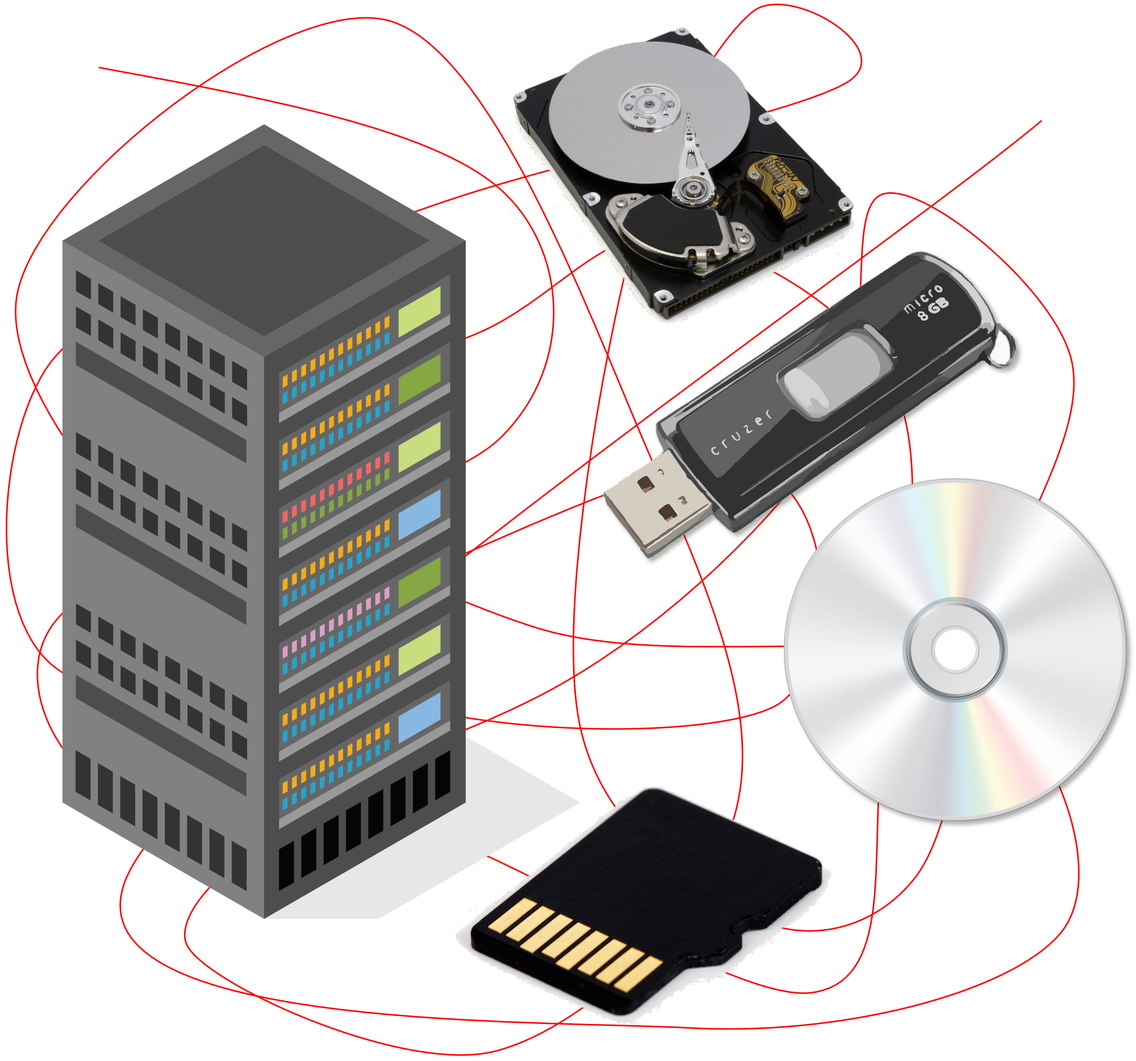
- Ainsi, Bob retrouve le message envoyé x (composé des coefficients du polynôme P).

En pratique

- On ne travaille pas avec des nombres réels, mais avec des nombres entiers modulo p (où p est un nombre premier) : $\{0,1,2,\ldots,p-1\}$, ou plus généralement des nombres appartenant à un groupe fini.
- Et les mêmes principes concernant les polynômes s'appliquent dans ce cadre.

Remarque : ces codes atteignent la borne de Singleton !

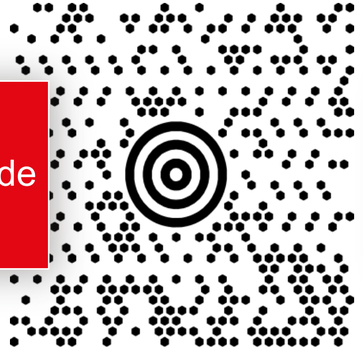
Application : Stockage de données et codes-barres 2D



QR code

Aztec Code

MaxiCode



Data Matrix

PDF-417