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

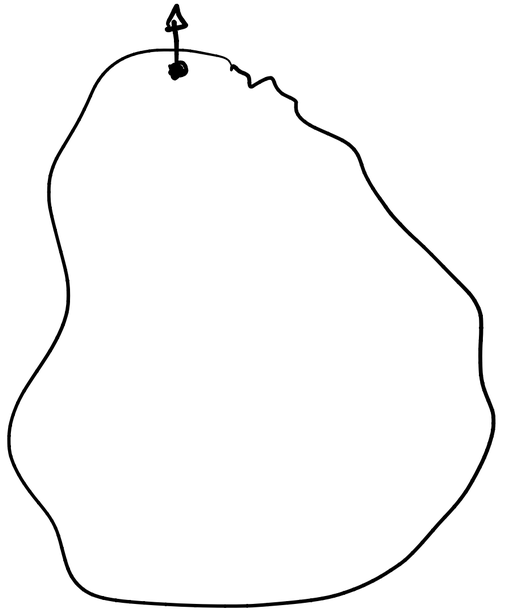
Exemple

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

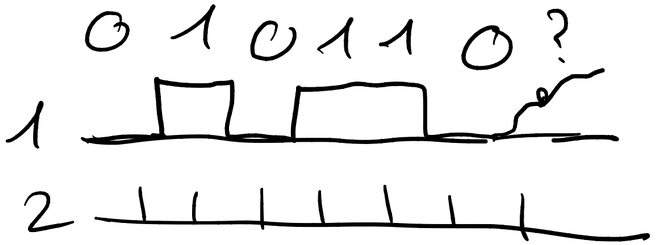
Code binaire utilisé :

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.

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 !

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 !

- 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.

$x_1 x_2 x_3$

N 11 0

S 10 1

E 01 1

W 00 0

éq: $x_3 = x_1 \oplus x_2$

↳ $x_3 = 1$ ssi $x_1 + x_2$ impair

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 !

- 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 :

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

1110101

↳ 11111100 11 ?? 11

La taille finale du message est :

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

1110101 1

Si on reçoit 11?0101 1

pour ?

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

règle de la majorité : $x_1 x_2 x_3 \mid x_4 x_5 x_6$

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 !

Règle de la majorité

Ex:

N	S	
11111	00000	5 bits

hyp:
≤ 2 erreurs, p.ex. si on reçoit 10101

$\Rightarrow$ on en déduit: N

Ce code corrige jusqu'à 2 erreurs.
(ou jusqu'à 4 effacements)

Rappel

↳ bouteilles (1, 2, 3, 4), dont une empoisonnée

2 rats pour tester → A, B

Solution : donner les bouteilles { 1 et 2 au rat A
                                 { 1 et 3 au rat B

1000 bouteilles et 10 rats

ex : bouteille 511 = 0 1 1 1 1 1 1 1 1 1

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

$1 \oplus 1 = 0$ 5e bit

$1 \oplus 0 = 1$ 6e bit

Exemple : Pour envoyer 1101, on envoie 110101.

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

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

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 !

• Si l'erreur survient sur 1 des deux bits de parité, on corrige alors qu'on ne devrait pas.

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

Code de Hamming (→ corrige 1 erreur)

$n = 4$ ($\rightarrow 2^4 = 16$ messages)

x_1, x_2, x_3, x_4, $x_5 = x_1 \oplus x_2 \oplus x_3$, $x_6 = x_1 \oplus x_2 \oplus x_4$

$x_7 = x_1 \oplus x_3 \oplus x_4$

4 bits d'information

3 bits de parité

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…

Déf:

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

  avec $c^{(j)}$ = mots de code, tous de longueur n

  $1 \leq j \leq M$

- Soient c et c' deux mots de code : la **distance de Hamming** entre c et c' est par définie par :

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

Ex: $d(0011, 1001) = 2$

- la distance minimale d'un code binaire C

$$= \min_{\substack{c \neq c' \\ c, c' \in C}} d(c, c')$$

N S E W

ex: $C = \{110, 101, 011, 000\}$

$d(c, c') = 2 \quad \forall c, c' \in C, \ c \neq c'$

$\Rightarrow$ distance minimale de $C = 2$

3 paramètres importants :

- M = nombre de messages possibles ($\leftrightarrow$ qté d'information)
- n = longueur des mots de code ($\leftrightarrow$ redondance)
- d = distance minimale ($\leftrightarrow$ capacité de correction)

But : créer des codes binaires avec $\begin{cases} M, d \text{ aussi grands que possible} \\ n \text{ aussi petit que possible} \end{cases}$

Ex 1 : $C = \{11, 10, 01, 00\}$

$M = 4$, $n = 2$, $\underline{d = 1}$

Capacité de correction nulle !

Ex 2 : $C = \{0000, 0011, 1100, 1111\}$

$M = 4$, $n = 4$, $d = 2$

Corrige $\begin{cases} \leq 1 \text{ effacement} \\ 0 \text{ erreur} \end{cases}$

Ex 3 : $C = \{000, 110, 101, 011\}$

$M = 4, \quad n = 3, \quad d = 2$

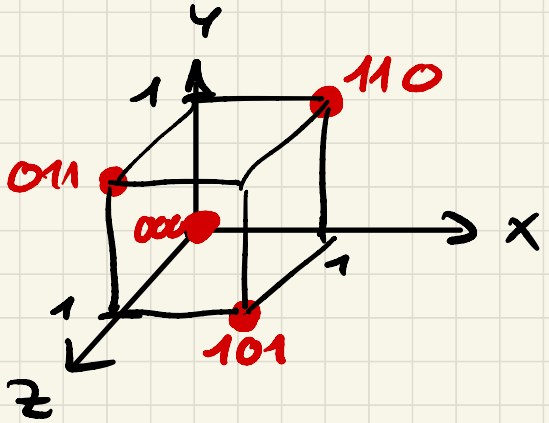
$\Rightarrow$ corrige $\begin{cases} \leq 1 \text{ effacement} \\ 0 \text{ erreur} \end{cases}$

Ex 4 : $C = \{000, 111\}$

$M = 2, \quad n = 3, \quad d = 3$

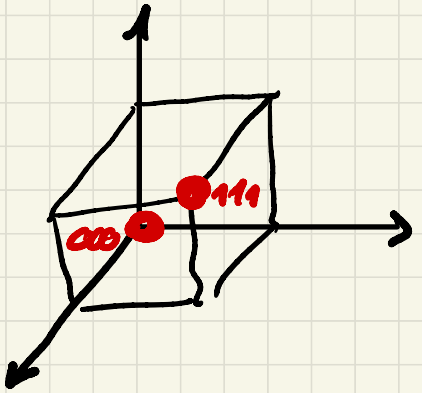
$\Rightarrow$ corrige $\begin{cases} \leq 2 \text{ effacements} \\ \leq 1 \text{ erreur} \end{cases}$

Ex 3:



$d = 2$

Ex 4:



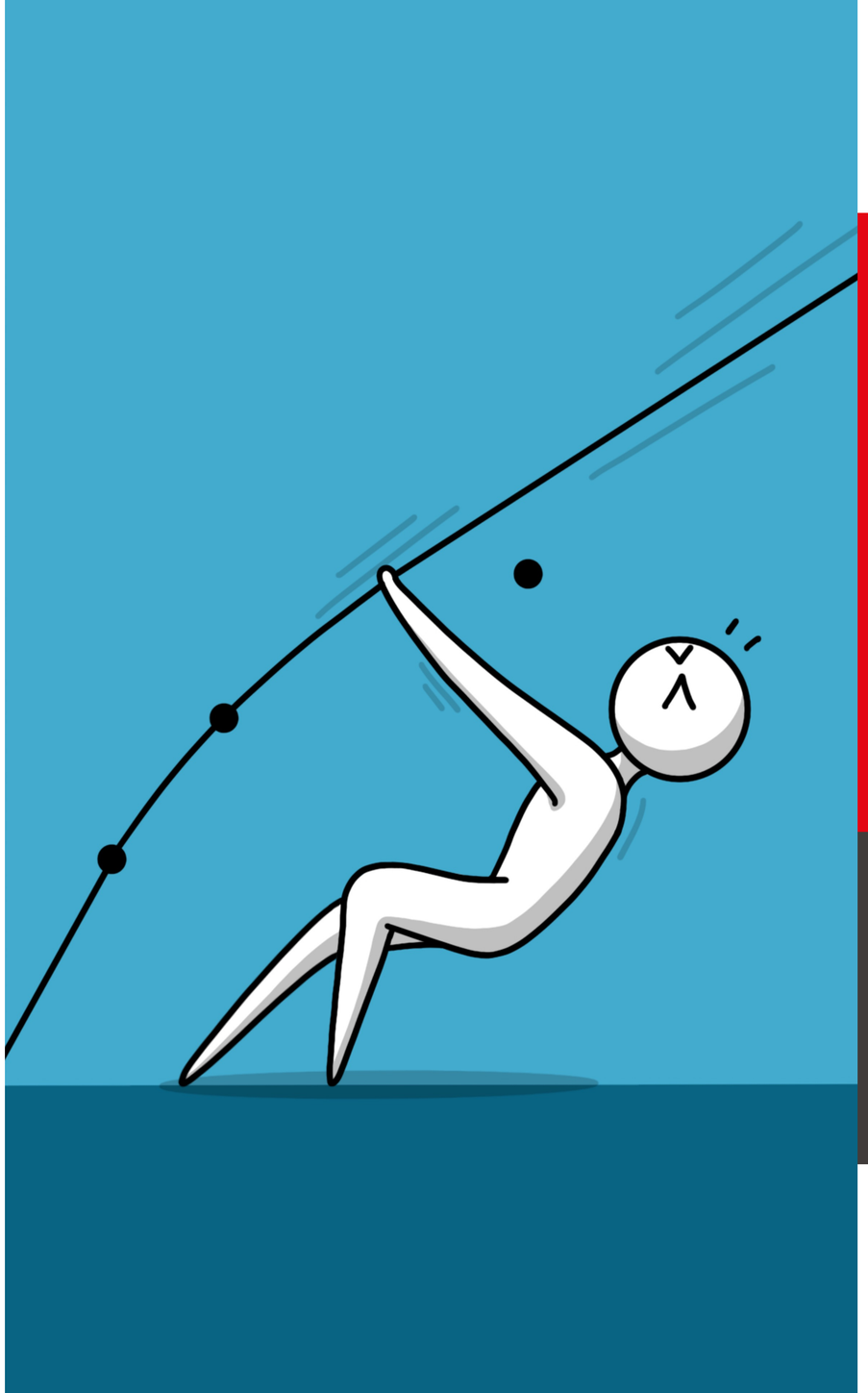
$d = 3$

Limitations

1°) $M \leq 2^n$

$\Uparrow$

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 :

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

Ex : $x = (x_1, x_2, x_3)$ $\quad P(t) = x_1 + x_2 t + x_3 t^2$

Alice génère $y = (y_1 = P(t_1), y_2 = P(t_2), \dots, y_6 = P(t_6))$ et envoie y

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 :

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

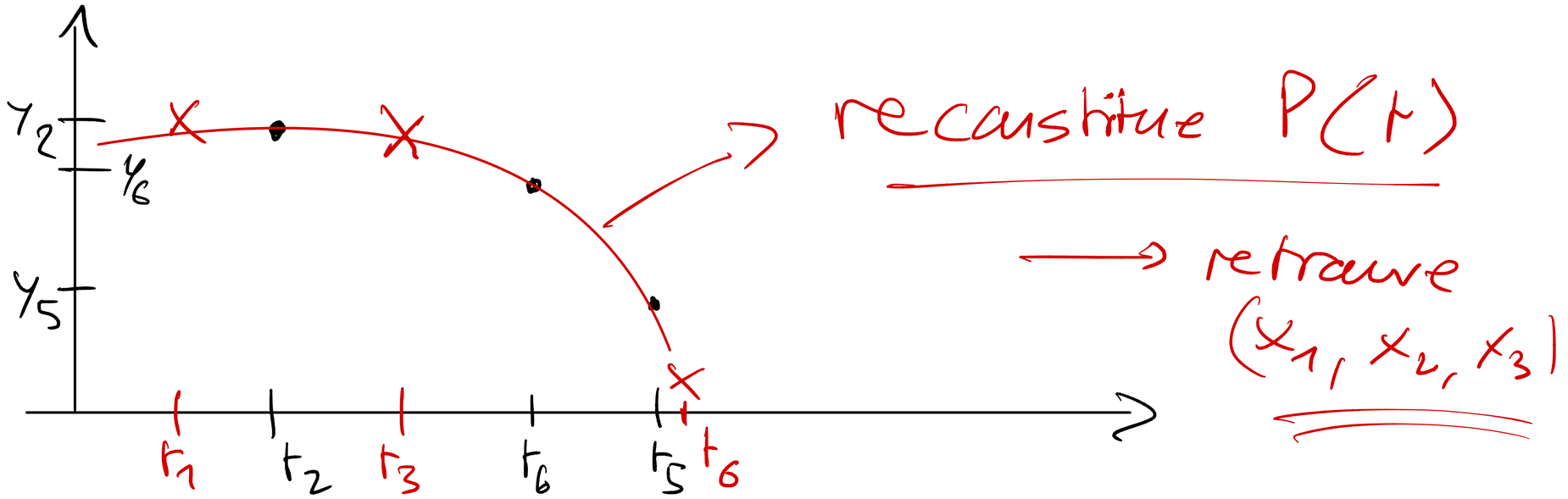
- et envoie finalement le message :

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

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.

Ex: Bob reçoit seulement $y_2 = P(t_2)$, $y_5 = P(t_5)$, $y_6 = P(t_6)$ et effectue :



Reed-Solomon

Alice veut envoyer $x = (x_1, \dots, x_k)$

$\longrightarrow$ elle envoie $y = (P(t_1) \dots P(t_n))$

Si $\leq n-k$ effacements surviennent, alors Bob retrouve $\geq k$ points de y et comme $\deg(P) \leq k-1$, Bob est capable de reconstruire P, donc x.

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 $\{y_1, y_2, y_3, \dots, y_k\}$

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

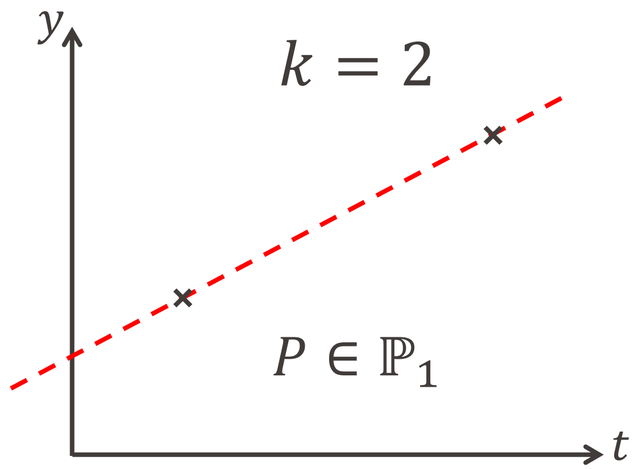
- Il s'agit donc de résoudre un système linéaire de k équations à k inconnues!

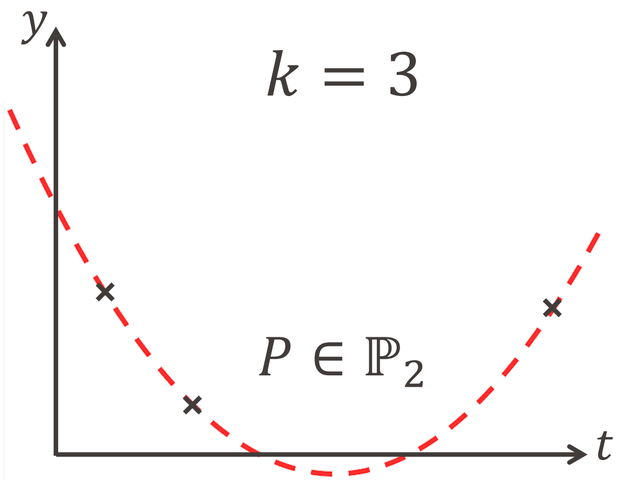
$$\text{Exemple pour } k = 3 \text{ et } t = \{1, 2, 3\} : \begin{cases} x_1 \cdot 1 + x_2 \cdot 1 + x_3 \cdot 1 = y_1 \\ x_1 \cdot 1 + x_2 \cdot 2 + x_3 \cdot 4 = y_2 \\ x_1 \cdot 1 + x_2 \cdot 3 + x_3 \cdot 9 = y_3 \end{cases}$$

$P(t_i) = y_i$

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.



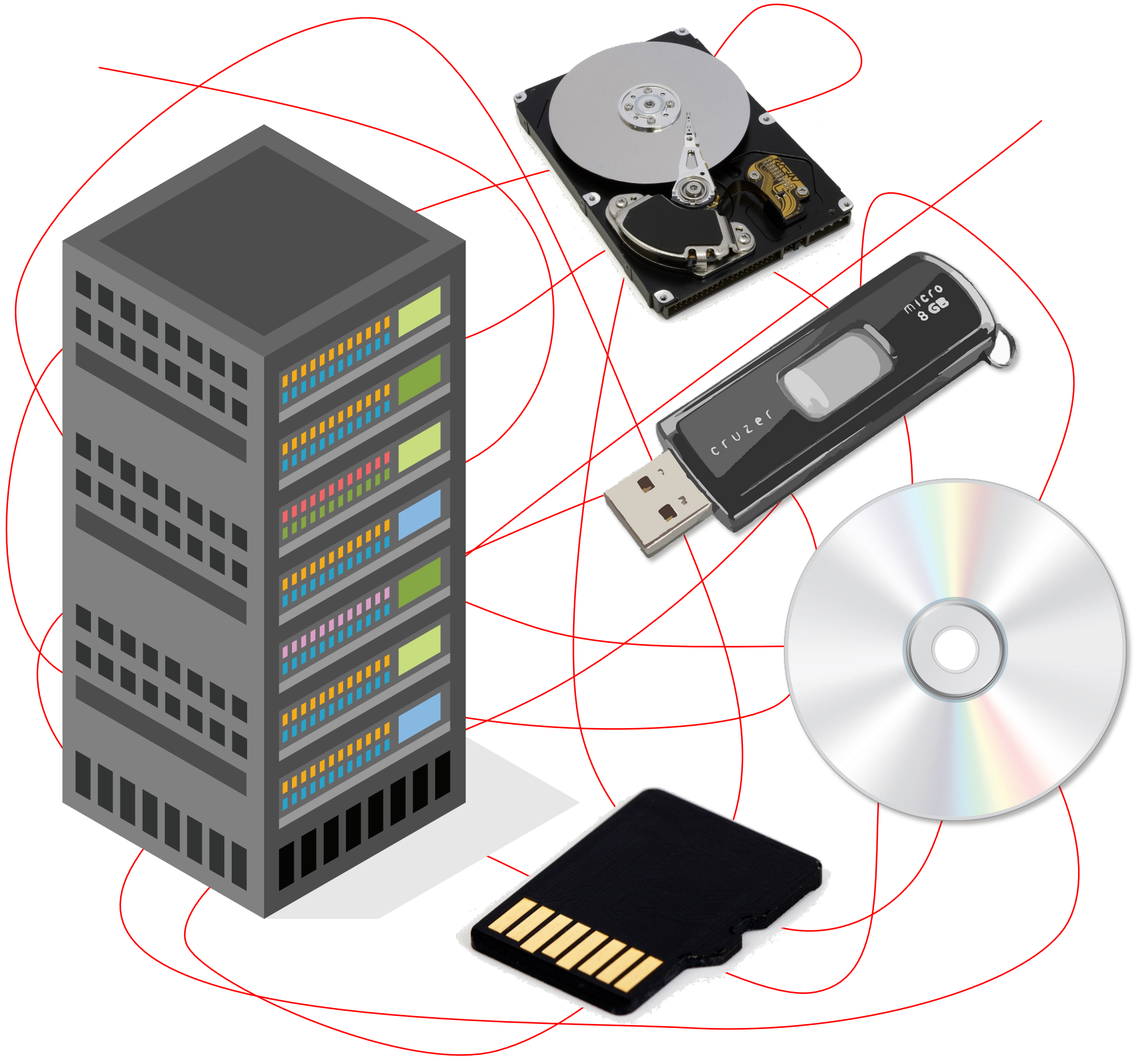


- Ainsi, Bob retrouve le message envoyé x (composé des coefficients du polynôme P).

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.

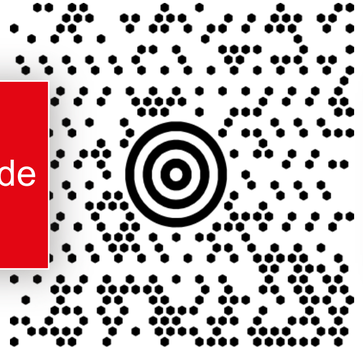
Application : Stockage de données et codes-barres 2D



QR code

Aztec Code

MaxiCode



Data Matrix

PDF-417