Introduction à la Cryptologie – LU3IN024

Cryptologie romantique

Guénaël Renault & Valérie Ménissier-Morain

15 février 2022

Première partie I

Les objets mathématiques de base de la cryptographie

Plan

La cryptologie

Définitions

La cryptologie est la *science du secret*. Elle se divise en deux disciplines :

- La cryptographie qui est l'étude des algorithmes permettant de protéger (cacher) de l'information. Ces algorithmes sont appelés cryptosystèmes ;
- la cryptanalyse qui est l'étude du niveau de sécurité des cryptosystèmes fournis par les cryptographes.

Première partie du cours

- On s'intéresse au problème de confidentialité (chiffrer des messages)
- La clef est la même pour chiffrer et déchiffrer

Terminologie et principe de base de la cryptologie

Terminologie

Chiffrer : l'action de rendre un message clair M (*plaintext*) en un message C appelé cryptogramme ou message chiffré, inintelligible.
Déchiffrer : action inverse du chiffrement : retrouver le clair à partir du chiffré.
Cryptosystème : l'algorithme (ou le dispositif physique) permettant de chiffrer des données.
Attaquer, casser : mettre à mal la sécurité d'un cryptosystème, retrouver M à partir de C sans connaître la clef, voire retrouver la clef.

Principe de Kerckhoffs (Journal des sciences militaires, 1883)

La sécurité d'un cryptosystème ne doit reposer que sur le secret de la clef.
En particulier un attaquant est supposé connaître le cryptosystème (Les cryptosystèmes militaires, vus comme des dispositifs physiques, peuvent tomber aux mains de l'ennemi).

☞La sécurité d'un cryptosystème est attestée par la cryptanalyse.

Cryptographie symétrique : modélisation

Définition

Un cryptosystème est dit symétrique (ou à clef privée) si pour chiffrer et déchiffrer, la même clef secrète est utilisée.

Modélisation d'un cryptosystème

- $\mathcal{P}$ et $\mathcal{C}$ les alphabets pour écrire les messages clairs et les messages chiffrés respectivement.
- $\mathcal{K}$ l'ensemble des clefs possibles.
- Pour tout $K \in \mathcal{K}$ on peut définir deux applications $e_K : \mathcal{P} \to \mathcal{C}$ (*Encryption*) et $d_K : \mathcal{C} \to \mathcal{P}$ (*Decryption*) telles que $d_K(e_K(x)) = x$ pour tout $x \in \mathcal{P}$.

☞Les alphabets peuvent être composés de symboles uniques ou de blocs de symboles. Pour commencer on s'en tiendra aux caractères de l'alphabet usuel.

La cryptographie symétrique en pratique

Cryptosystème symétrique (la même clef K pour dé/chiffrer)

- $\mathcal{P}$ et $\mathcal{C}$ les alphabets clairs et chiffrés.
- $\mathcal{K}$ l'ensemble des clefs possibles.
- $K \in \mathcal{K} : e_K : \mathcal{P} \to \mathcal{C}$ et $d_K : \mathcal{C} \to \mathcal{P}$ avec $\forall x \in \mathcal{P}\, d_K(e_K(x)) = x$.
- Les applications e_K et d_K nécessitent la connaissance complète de K pour être définies.

En pratique : chiffrer avec la clef K

Pour chiffrer un texte P clair on procède comme suit

1. On scinde P en blocs éléments de $\mathcal{P}$
2. On applique e_K sur chacun de ces blocs pour produire des blocs éléments de $\mathcal{C}$
3. On juxtapose les blocs ainsi obtenus pour produire le chiffré C associé à P.

La cryptographie symétrique en pratique

Cryptosystème symétrique (la même clef K pour dé/chiffrer)

- $\mathcal{P}$ et $\mathcal{C}$ les alphabets clairs et chiffrés.
- $\mathcal{K}$ l'ensemble des clefs possibles.
- $K \in \mathcal{K} : e_K : \mathcal{P} \to \mathcal{C}$ et $d_K : \mathcal{C} \to \mathcal{P}$ avec $\forall x \in \mathcal{P}\, d_K(e_K(x)) = x$.
- Les applications e_K et d_K nécessitent la connaissance complète de K pour être définies.

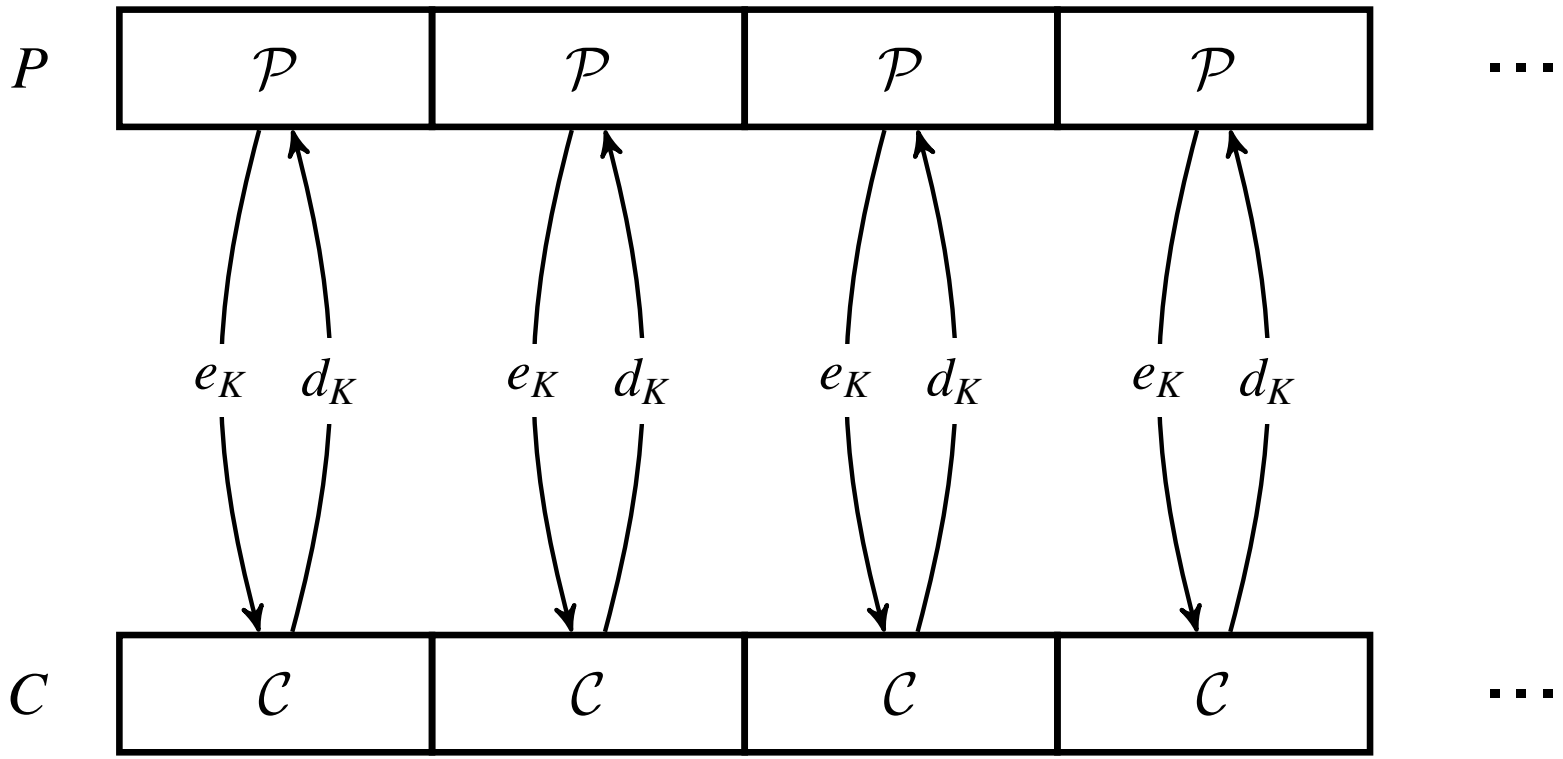
En pratique : déchiffrer avec la clef K

Pour déchiffrer le texte C on procède comme suit

1. On scinde C en blocs éléments de $\mathcal{C}$
2. On applique d_K sur chacun de ces blocs pour produire des blocs éléments de $\mathcal{P}$
3. On juxtapose les blocs ainsi obtenus pour reconstituer P le clair associé à C.

Chiffrement de messages en pratique

☞Découpage par blocs de $\mathcal{P}$



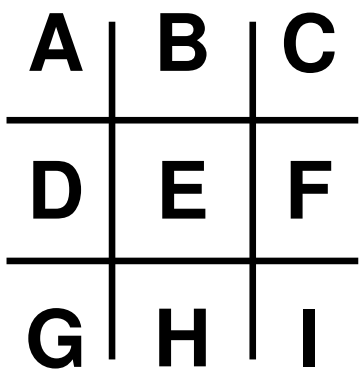
Plan

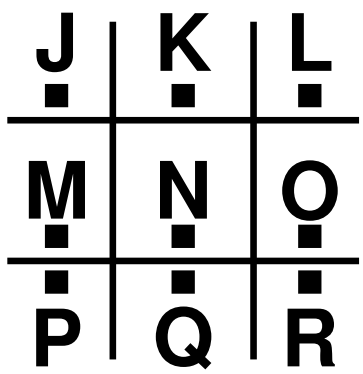
Chiffrement par substitution ou mono-alphabétique

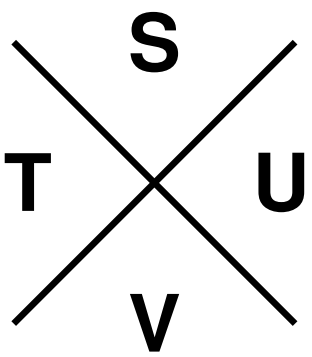
Définition

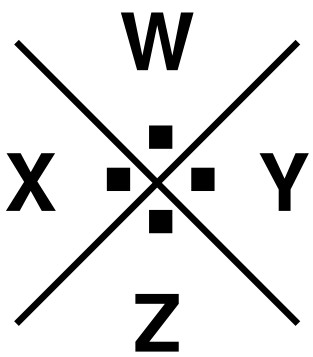
À chaque élément de $\mathcal{P}$ correspond un unique élément de $\mathcal{C}$.

☞Un exemple où l'alphabet d'arrivée est différent de celui de départ
Le chiffrement des Francs-Maçons.

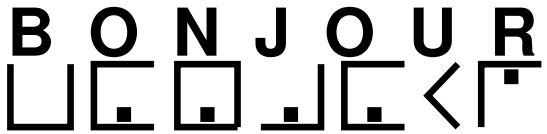








Chiffrement de BONJOUR :



Chiffrement par substitution ou mono-alphabétique

Définition

À chaque élément de $\mathcal{P}$ correspond un unique élément de $\mathcal{C}$.

Cryptosystème par substitution

- $\mathcal{P}, \mathcal{C}$ peuvent être différents mais doivent être de même cardinal
- $\mathcal{K}$ est l'ensemble des bijections de $\mathcal{P}$ dans $\mathcal{C}$.
- Pour tout $K \in \mathcal{K}$ on a
  - $e_K(x) = K(x)$
  - $d_K(y) = K^{-1}(y)$

Chiffrement par substitution ou mono-alphabétique

Définition

À chaque élément de $\mathcal{P}$ correspond un unique élément de $\mathcal{C}$.

☞On peut prendre les alphabets canoniques $\mathcal{P} = \mathcal{C} = \{A, \ldots, Z\}$ mais dans ce cas il faut mélanger cet alphabet.

Cryptosystème par substitution sur alphabets identiques

- $\mathcal{P}, \mathcal{C}$ sont tous les deux égaux à $\mathcal{A} = \{A, \ldots, Z\}$
- $\mathcal{K}$ est l'ensemble des permutations de $\mathcal{P}$ dans $\mathcal{C}$.
- Pour tout $K \in \mathcal{K}$ on a
  - $e_K(x) = K(x)$
  - $d_K(y) = K^{-1}(y)$

Plan

Plan

Groupes

Groupe $(G, \star)$

- Ensemble G muni d'une loi de composition interne $\star : G \times G \mapsto G$
- *Associativité* : $\forall (a, b, c) \in G \times G \times G, \quad a \star (b \star c) = (a \star b) \star c$
- *Neutre* : il existe un élément e neutre pour $\star$, i.e. $\forall a \in G, \quad a \star e = e \star a = a$
- *Inverse* : $\forall a \in G, \exists b \in G \mid \quad a \star b = b \star a = e$

☞Le groupe est dit *commutatif* si de plus $\forall (a, b) \in G \times G, \quad a \star b = b \star a$

Exemples

- L'ensemble $\mathbb{Z}$ des entiers muni de l'addition (neutre 0, commutatif)
- L'ensemble $\mathbb{Q} \setminus \{0\}$ muni de la multiplication (neutre 1, commutatif)
- L'ensemble $\mathrm{Perm}(E)$ des permutations d'un ensemble fini E muni de la composition (neutre Id, non commutatif).
  Exemple : $\sigma_1 : 1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 3$, $\sigma_2 : 1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 2$,
  $\sigma_1 \circ \sigma_2 : 1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1 \neq \sigma_2 \circ \sigma_1 : 1 \mapsto 3, 2 \mapsto 1, 3 \mapsto 2$.


Avoir à l'esprit quelle est l'opération du groupe ! Parfois, l'opération est implicite, par exemple $\mathbb{Z}/n\mathbb{Z}$ est canoniquement un groupe additif.

Groupes et chiffrement par substitution

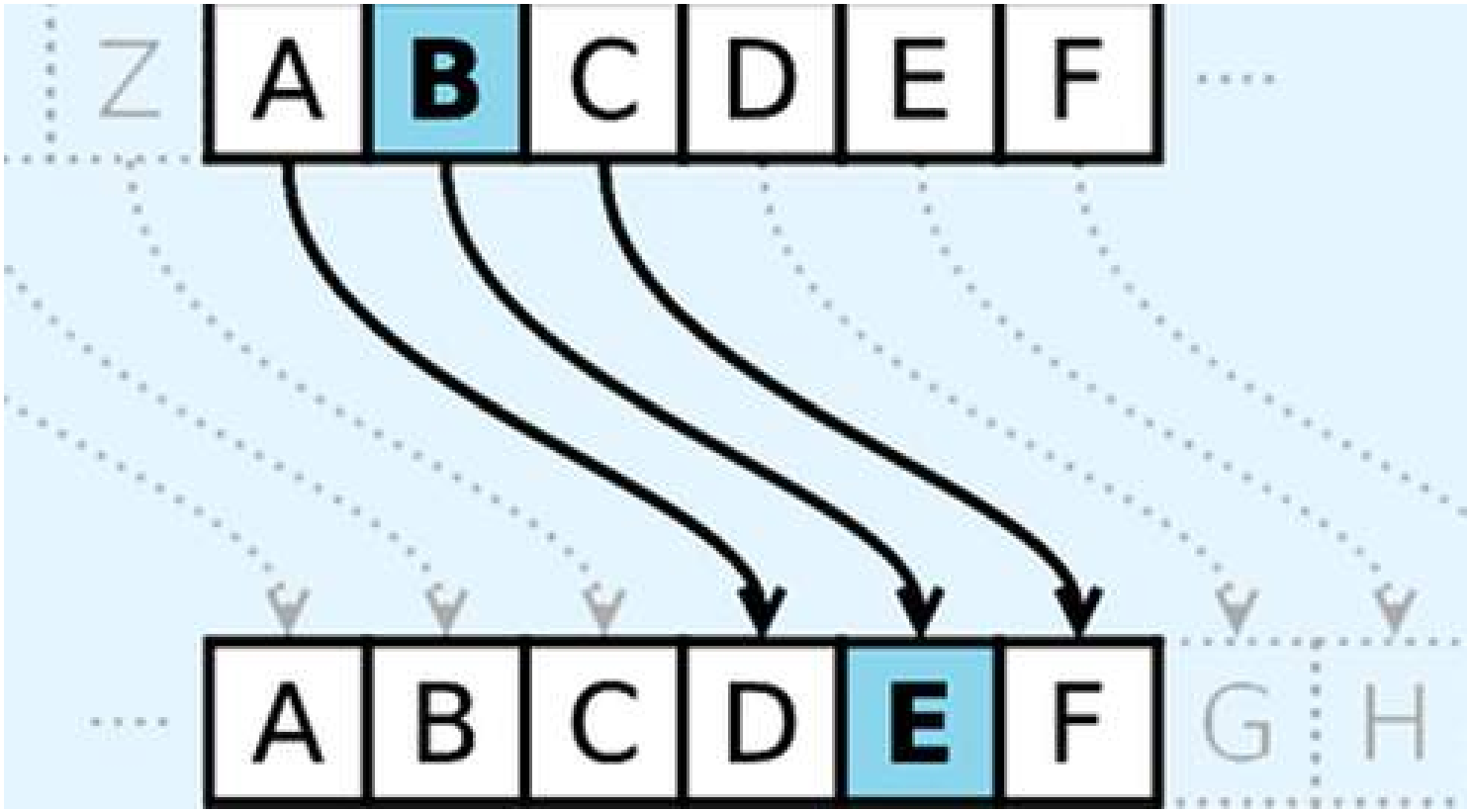
Chiffrement par substitution

À chaque élément de $\mathcal{P}$ correspond un unique élément de $\mathcal{C}$.

Si on choisit de numéroter les lettres de l'alphabet de 0 à 25 on aura alors la modélisation :

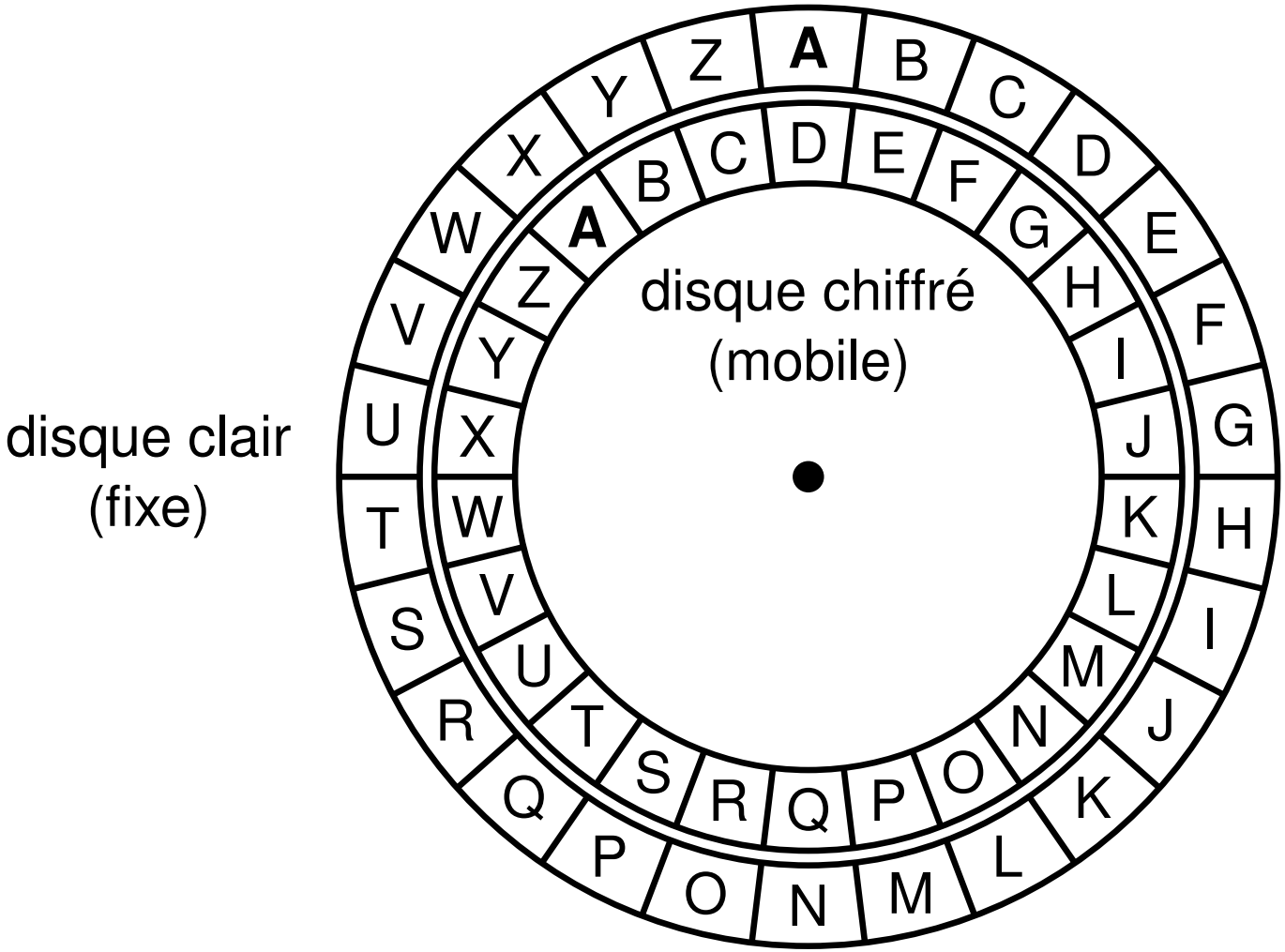
- $\mathcal{P} = \mathcal{C} = \{0, \ldots, 25\}$
- $\mathcal{K} = \mathrm{Perm}(\mathcal{P})$
- $\sigma \in \mathcal{K},\ e_K = \sigma$ et $d_K = \sigma^{-1}$

Cas particulier : le chiffrement de César



Cas particulier : le chiffrement de César

☞Les clefs forment un sous-groupe de $\mathrm{Perm}(\mathcal{P})$, celui des rotations.



Cas particulier : le chiffrement de César

☞Les clefs forment un sous-groupe de $\mathrm{Perm}(\mathcal{P})$, celui des rotations.

Chiffrement de César : modulaire

- $\mathcal{P} = \mathcal{C} = \mathbb{Z}/26\mathbb{Z}$
- $\mathcal{K} = \{0, \ldots, 25\}$
- $\rho \in \mathcal{K},\ e_K : p \mapsto p + \rho \bmod 26$ et $d_K : c \mapsto c - \rho \bmod 26$

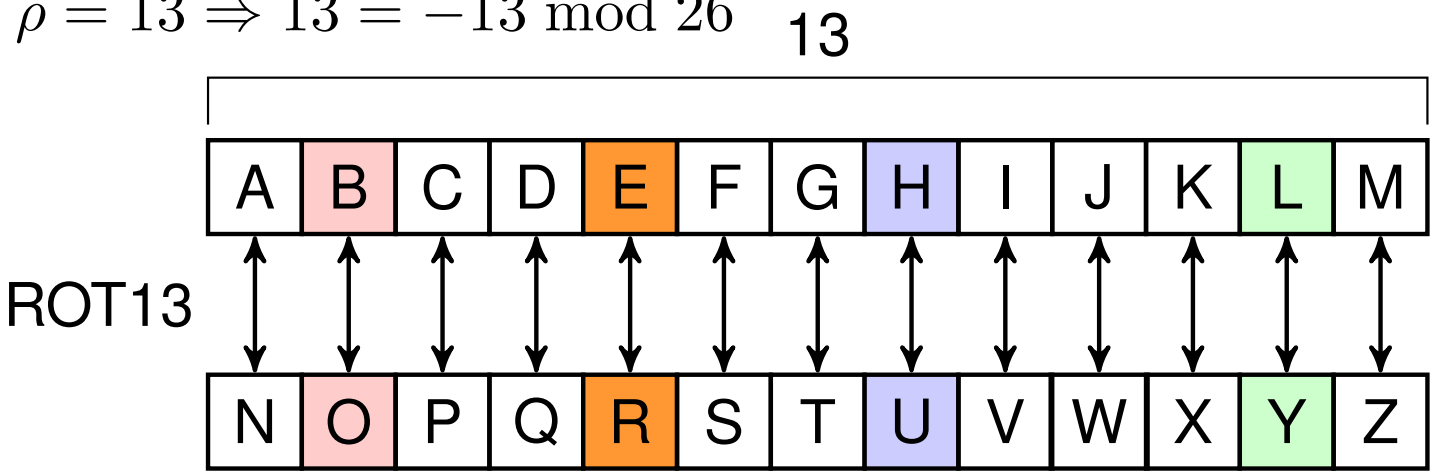
☞Arithmétique modulaire !

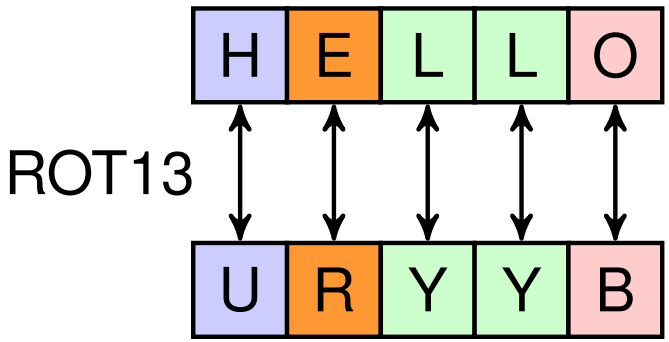
Cas particulier : le chiffrement de César

Chiffrement de César : modulaire

- $\mathcal{P} = \mathcal{C} = \mathbb{Z}/26\mathbb{Z}$
- $\mathcal{K} = \{0, \ldots, 25\}$
- $\rho \in \mathcal{K},\ e_K : p \mapsto p + \rho \bmod 26$ et $d_K : c \mapsto c - \rho \bmod 26$

Exemple $\rho = 13 \Rightarrow 13 = -13 \bmod 26$





Chiffrement de César : propriétés

☞Dans $\mathrm{Perm}(\{0,\ldots,25\})$ une rotation peut être définie comme la composition de plusieurs fois une rotation de longueur 1.

Définition : Groupe cyclique (ou monogène)

C'est un groupe $(G,\star)$ qui peut être engendré par un élément :

$$\exists g \in G \text{ tq } \forall a \in G\ , a = g \star g \star \cdots \star g$$

On peut noter cette propriété sous la forme $G = \langle g \rangle$.

☞Le sous-groupe de $\mathrm{Perm}(\{0,\ldots,25\})$ des rotations est cyclique.

Chiffrement affine

☞En utilisant l'encodage habituel, on se place directement dans un alphabet de la forme $\mathbb{Z}/n\mathbb{Z}$ (ici $n = 26$).

Cryptosystème affine

- $\mathcal{P}, \mathcal{C}$ sont tous les deux égaux à $\mathbb{Z}/26\mathbb{Z}$
- $\mathcal{K}$ est un sous-ensemble de $\mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z}$ tel que
- Pour tout $K = (a, b) \in \mathcal{K}$ on a
  - $e_K(x) = ax + b \bmod 26$
  - et il existe une fonction $d_K : \mathcal{C} \to \mathcal{P}$ inverse de e_K.

☞$\mathcal{K}$ forme ici aussi un sous-groupe de $\mathrm{Perm}(\mathcal{P})$.

Problème

Comment déterminer l'ensemble des clefs possibles ? Ceci revient à déterminer les fonction e_K qui possèdent une fonction inverse.

☞Ici il y a deux opérations : besoin de structures mathématiques plus riches que celle de groupe, en particulier de savoir inverser.

Plan

Objets mathématiques supports de la cryptographie

- Groupe, anneau, corps, ...
- Arithmétique dans $\mathbb{Z}$, ...
- Divisibilité, nombres premiers, ...
- ...

Structures de base : définitions

Anneau commutatif $(A, \star, \circ)$

- *Groupe commutatif pour l'opération* $\star$
- *Associativité et existence d'un élément neutre pour* $\circ$
- *Distributivité de* $\circ$ *par rapport à* $\star$:
  $\forall (a, b, c) \in A^3, \quad c \circ (a \star b) = (c \circ a) \star (c \circ b)$
- *Commutativité de la seconde opération*

Exemples

- L'anneau $\mathbb{Z}$ muni de l'addition (groupe) et de la multiplication.
- L'anneau des fractions $\mathbb{Q}$ avec les mêmes opérations

☞Dans la suite, tous les anneaux seront commutatifs et munis de l'addition et de la multiplication. L'élément neutre est 0 pour l'addition et 1 pour la multiplication.

Structures de base : définitions

Sous-groupe

Un sous-ensemble U (*Untergruppe*) d'un groupe G est un *sous-groupe* de G s'il est lui-même un groupe pour les opérations de G.

Sous-anneau

Un sous-ensemble U (*Unterring*) d'un anneau A est un *sous-anneau* de A s'il est lui-même un anneau pour les opérations de A.

Exemple

- L'anneau $\mathbb{Z}$ muni de l'addition (groupe) et de la multiplication est en fait un sous-anneau de $\mathbb{Q}$.

Divisibilité dans un anneau $(A, +, \times)$

Diviseurs, multiples

Un élément a est un *diviseur* de b dans un anneau A s'il existe $c \in A$ tel que $b = ac$. On note alors $a \mid b$ et b est appelé *multiple* de a.

☞L'ensemble des multiples de a dans A est noté $aA = \{ak \ : \ k \in A\}$

☞L'arithmétique est l'étude de la divisibilité.
☞6 et -6 divisent 12 dans $\mathbb{Z}$.

Si a est un entier divisant $b \in \mathbb{Z}$ alors il en est de même pour $-a$. Ceci vient du fait que -1 est *inversible* dans $\mathbb{Z}$.

☞On ne s'intéresse donc qu'au cas positif (classes d'équivalence modulo la multiplication par un inverse)

Inversibilité dans un anneau $(A, +, \times)$

Inversibles

Un élément a d'un anneau A est dit *inversible* s'il existe $b \in A$ tel que $ab = 1$. L'élément b est appelé *inverse* de a. L'ensemble des inversibles d'un anneau A sera noté $A^\times$ (ou aussi $U(A)$).

- $A^\times$ forme un groupe commutatif dans A pour la multiplication.
- La divisibilité peut être vue à un inversible près.

Association

Pour $a \in A$ nous appelons *associé à* a tout élément b de la forme $b = ua$ avec $u \in A^\times$.

- Pour $A = \mathbb{Z}$ nous avons $A^\times = \{1, -1\}$
- Pour $A = \mathbb{Z}$ les associés de a sont $\{a, -a\}$
- Pour $A = \mathbb{Z}$ on choisira le positif comme représentant la classe d'équivalence pour l'association.

Intégrité

Dans un cadre général, certains éléments d'un anneau A peuvent représentés un obstacle à l'arithmétique. Ces éléments sont les diviseurs de 0.

Diviseur de zéro et anneau intègre

Un élément $a \neq 0 \in A$ est *diviseur de* 0 s'il existe $b \neq 0 \in A$ tel que $ab = 0$. Un anneau A est dit *intègre* s'il n'existe pas de diviseur de 0 dans A.

L'anneau $\mathbb{Z}$ est intègre mais nous rencontrerons plus tard des anneaux qui ne le seront pas, les exemples les plus simples étant certains de ceux construits par quotient de $\mathbb{Z}$.

Division euclidienne

La division euclidienne représente un moyen puissant pour étudier la divisibilité d'un anneau.

Anneau euclidien

Un anneau intègre A équipé d'une fonction $d : A \to \mathbb{N} \cup \{-\infty\}$ est dit *euclidien* si pour tout couple $(a, b) \in A^2$ avec $b \neq 0$, il existe q et r dans A tel que

$$a = qb + r \text{ avec } d(r) < d(b).$$

Cette dernière relation étant appelée *division euclidienne de a par b*, l'élément q le *quotient* et r le *reste* de cette division. d s'appelle un *stathme* sur A.

- Pour $A = \mathbb{Z}$ nous choisissons $d \ : \ x \mapsto |x|$
- $b \mid a$ est équivalent à $a = bq + r$ avec $r = 0$
- Non-unicité, pour $A = \mathbb{Z}$ on prendra comme convention $r \geqslant 0$

Division euclidienne

Dans le langage `C`, la division euclidienne sur les entiers est obtenue en utilisant les opérateurs arithmétiques binaires `/` pour le quotient et `%` pour le reste de la division de deux entiers.
En Python 3, le quotient est calculé avec l'opération `//` et le reste avec l'opération `%`.


D'après la norme ANSI, le résultat correspondra à la convention décrite plus haut (i.e. reste positif ou nul) lorsque les opérandes sont eux-mêmes positifs. Dans le cas contraire le résultat n'est pas spécifié.

☞En particulier le résultat de `(3-14)%26` peut être négatif !

☞Si on divise par 0, le résultat n'a pas de sens.

Éléments irréductibles et premiers

☞Les éléments "*minimaux*" pour la divisibilité

Irréductible

Un élément p d'un anneau A est *irréductible* s'il est ni nul ni inversible et s'il a exactement deux diviseurs (à association près) : 1 et p.

Premiers

Un élément p d'un anneau A est dit *premier* s'il est ni nul ni inversible et s'il vérifie la propriété $p \mid ab \Rightarrow p \mid a \text{ ou } p \mid b$.

Dans $\mathbb{Z}$ les éléments irréductibles et les éléments premiers sont les mêmes (admis ici)

Nombres premiers

Les éléments premiers positifs dans $\mathbb{Z}$ sont appelés *les nombres premiers*.

Décomposition en facteurs premiers dans $\mathbb{Z}$

☞On veut pouvoir décomposer selon ces irréductibles (briques de base)

Théorème fondamental de l'arithmétique

Tout élément qui n'est ni nul ni inversible peut être écrit sous la forme d'un produit $p_1 p_2 \cdots p_n$ d'éléments irréductibles et cette décomposition est unique à association et ordre près des facteurs irréductibles.

qu'on peut énoncer aussi simplement ainsi

Tout entier $n > 1$ s'écrit sous la forme d'un produit de puissances de nombres premiers croissants p_i

$$n = p_1^{e_1} p_2^{e_2} \ \ldots \ p_k^{e_k} \text{ avec } e_i > 0 \text{ et } p_i < p_j \text{ si } i < j$$

Les nombres premiers

☞Les nombres irréductibles sont les nombres premiers !
☞La "*vraie*" notion à retenir en général est celle d'irréductible.
☞Important de connaître les irréductibles car ils définissent tous les éléments.

☞Quel est le nombre de nombres premiers ?

Infinité des nombres premiers

L'ensemble des nombres premiers est infini.

Preuve :

- *On suppose qu'il y a un ensemble fini de nombres premiers* $n_1, \ldots, n_k$.
- *On construit un nouveau nombre premier* $1 + \prod_{i=1}^{k} n_i$, *ce qui contredit l'hypothèse.*

Les nombres premiers

Comment reconnaître un nombre premier ?

Lemme

Un entier $n > 1$ est premier ssi il n'est divisible par aucun entier compris entre 2 et $\sqrt{n}$.

```
#include<stdio.h>
#include<math.h>

int main () {
  int p, d, r;
  printf ("Entrer un entier >2 a tester : ");
  scanf ("%d",&p);

  d=1;
  do {
    d += 1;
    r = p % d;
  } while(r != 0 && d < sqrt((double)p));
  if (r == 0)
    printf("\nL'entier %d n'est pas premier.\n
              Il est divisible par %d.\n", p, d);
  else
    printf("\nL'entier %d est premier.\n", p);
  return 0;
}
```

Les nombres premiers, la factorisation

Comment reconnaître un nombre premier ? (primalité)

- La complexité de ce programme est de l'ordre de $\sqrt{p}$ et est donc exponentielle en la taille de p.
- Agrawal, Kayal et Saxena (2002) décrivent un algorithme de complexité polynomiale en la *taille de l'entrée* non probabiliste et indépendant de toute conjecture.
- Le meilleur logiciel connu est `FastECPP` de F. Morain (basé sur les courbes elliptiques, record 26643 chiffres décimaux, 2011).
- Nombres spéciaux & algorithmes dédiés : record 24 862 048 chiffres décimaux (Mersenne, GIMPS, décembre 2018).

Comment factoriser un nombre entier ? (factorisation)

- C'est beaucoup plus dur !
- Des cryptosystèmes reposent sur cette difficulté.
- Des algorithmes de complexité sous-exponentielle existent (MPQS, NFS).
- Record 250 chiffres décimaux de E. Thomé et al. (SNFS, RSA-250, février 2020)
- Nombres spéciaux & algorithmes dédiés : record 23 703 chiffres décimaux (Mersenne, GIMPS, 2017).

Plan

Plan

Relation d'équivalence

Une relation binaire $\mathcal{R}$ d'un ensemble E est une *relation d'équivalence* lorsqu'elle vérifie les trois propriétés suivantes :

- $\mathcal{R}$ est *réflexive* : tout élément de E peut être mis en relation avec lui-même ($\forall x \in E,\ x\mathcal{R}x$)
- $\mathcal{R}$ est *symétrique* : tout élément de E peut être mis en relation avec les éléments qui lui sont relatifs ($\forall (x,y) \in E^2,\ x\mathcal{R}y \Rightarrow y\mathcal{R}x$)
- $\mathcal{R}$ est *transitive* : une relation d'une relation est une relation ($\forall (x,y,z) \in E^3,\ x\mathcal{R}y \wedge y\mathcal{R}z \Rightarrow x\mathcal{R}z$)

Deux éléments de E qui sont mis en relation par $\mathcal{R}$ sont dits *équivalents* pour cette relation.

Soit x un élément de E, le sous-ensemble de E formé des éléments de E qui sont équivalents à x pour $\mathcal{R}$ est appelé *classe d'équivalence* de x.

Relation d'équivalence : exemple de la relation d'association

Soit $a, b \in A$ un anneau commutatif. On note aA l'ensemble des multiples de a dans A. On définit $\mathcal{R}$ par $a\mathcal{R}b \Leftrightarrow aA = bA$

- $aA = aA$ clair
- $aA = bA \Rightarrow bA = aA$ clair
- $aA = bA$ et $bA = cA \Rightarrow aA = cA$ clair

Provient du fait que $=$ est une relation d'équivalence.

On peut montrer que $aA = bA$ est équivalent à l'assertion :

$$\exists u \text{ inversible dans } A \text{ tq } a = ub.$$

Dans $\mathbb{Z}$ les différentes classes d'équivalence seront alors $\{0\}, \{-1, 1\}, \{-2, 2\}, \ldots$.

Relation d'équivalence : exemple $\mathbb{Z}/n\mathbb{Z}$

(très important)

Soit $a, b \in \mathbb{Z}$ et n un entier positif. $a \mathcal{R}_n b \Leftrightarrow a - b \in n\mathbb{Z}$

- $a - a = n \times 0$
- $a - b \in n\mathbb{Z} \Rightarrow b - a \in (-1) \times n\mathbb{Z} = n\mathbb{Z}$
- $a - b \in n\mathbb{Z}$ et $b - c \in n\mathbb{Z} \Rightarrow (a - b) + (b - c) \in n\mathbb{Z}$

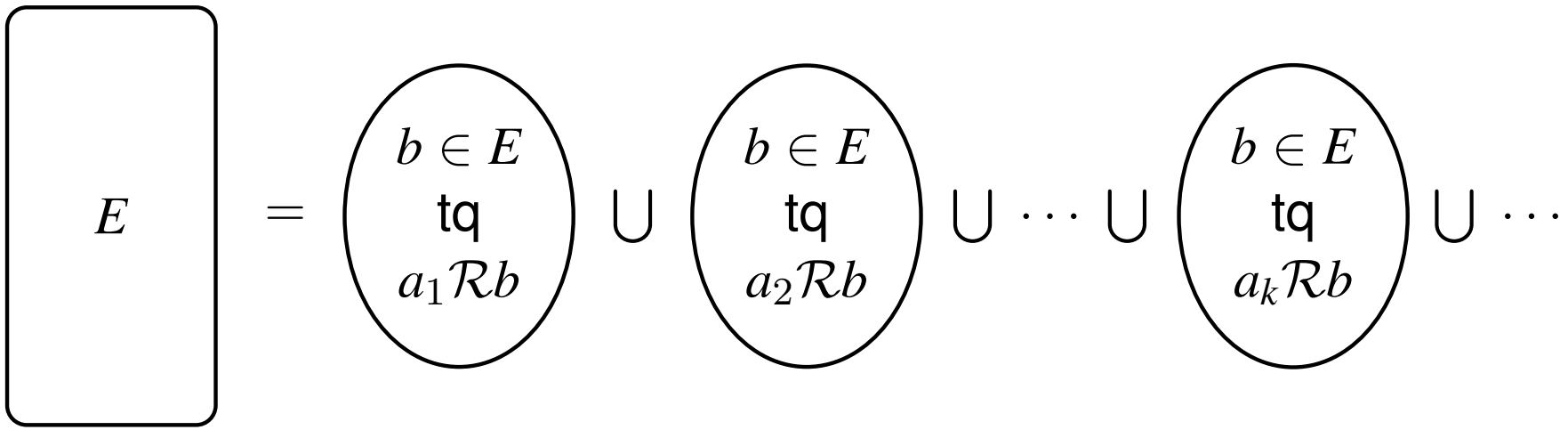
☞Les éléments d'une même classe ont le même reste modulo n (exercice : le montrer)
☞On choisit le plus petit entier positif de la classe (les classes sont donc représentées par $\{0, 1, \ldots, n-1\}$).

Relation d'équivalence : partitionnement

Propriété de partitionnement

Soit E un ensemble et $\mathcal{R}$ une relation d'équivalence sur E. L'ensemble des classes d'équivalence pour $\mathcal{R}$ forme une partition de E, c'est-à-dire un ensemble de sous-ensembles non vides de E, disjoints et dont l'union est E.

☞Preuve laissée en exercice. Idée imagée :



Plan

Relation d'équivalence : application au théorème de Lagrange

Théorème de Lagrange

Soit $(G, \times)$ un groupe fini et H un sous-groupe de G. Alors le cardinal de H est un diviseur du cardinal de G. (Le cardinal d'un groupe est aussi appelé ordre.)

Mathématicien italo-français de la seconde moitié du 18ème siècle. Il développe les idées sur les groupes (sans que la définition soit clairement établie) de permutations pour étudier les solutions des équations polynomiales.

Relation d'équivalence : application au théorème de Lagrange

Théorème de Lagrange

Soit $(G, \times)$ un groupe fini et H un sous-groupe de G. Alors le cardinal de H est un diviseur du cardinal de G. (Le cardinal d'un groupe est aussi appelé ordre.)

Démonstration

- On considère la relation d'équivalence $x\mathcal{R}_H y \Leftrightarrow xy^{-1} \in H$
- On partitionne le groupe G selon les classes de $\mathcal{R}_H$
- On montre que la classe d'équivalence d'un élément x est xH et que H est la classe de 1
- On établit une application bijective entre les classes d'équivalence donc toutes les classes d'équivalence ont même cardinal, celui de H, qui divise donc le cardinal de G.

Ordre d'un élément

Théorème de Lagrange

Soit $(G, \times)$ un groupe fini et H un sous-groupe de G. Alors le cardinal de H est un diviseur du cardinal de G. (Le cardinal d'un groupe est aussi appelé ordre.)

Définition : ordre d'un élément

Soit g un élément d'un groupe fini $(G, \star)$. On appelle ordre de g le plus petit entier strictement positif k tel que

$$\underbrace{g \star g \star \cdots \star g}_{k \text{ fois}} = e$$

Si $g^k = e$ alors k est un multiple de l'ordre de g, l'ordre d'un élément est minimal non seulement pour l'ordre usuel mais surtout pour la relation de divisibilité (démonstration en TD).

☞Soit $g \in G$ on considère le sous-groupe cyclique de G engendré par g. Son cardinal est égal à l'ordre de g et divise le cardinal de G d'après le théorème de Lagrange. C'est pourquoi on nomme ordre le cardinal d'un groupe.

Corollaire

Tout groupe $(G, \star)$ dont l'ordre est un nombre premier est cyclique.

Plan

PGCD

Définition

Le plus grand diviseur commun de deux entiers positifs a et b est l'élément maximal de l'ensemble $Div(a) \cap Div(b)$ avec $Div(a)$ l'ensemble (fini) des diviseurs (positifs) de a.

Propriété

Si on isole les diviseurs premiers communs dans la décomposition de a et b en facteurs premiers

$$a = \underbrace{p_1^{a_1}\, p_2^{a_2}\, \dots\, p_k^{a_k}}_{\text{facteurs premiers communs}} \quad \underbrace{q_1^{a'_1}\, q_2^{a'_2}\, \dots\, q_m^{a'_m}}_{\text{facteurs premiers divisant seulement } a}$$

et

$$b = \underbrace{p_1^{b_1}\, p_2^{b_2}\, \dots\, p_k^{b_k}}_{\text{facteurs premiers communs}} \quad \underbrace{r_1^{b'_1}\, r_2^{b'_2}\, \dots\, r_n^{b'_n}}_{\text{facteurs premiers divisant seulement } b}$$

alors

$$\mathrm{pgcd}(a,b) = p_1^{\min(a_1,b_1)}\, p_2^{\min(a_2,b_2)}\, \dots\, p_k^{\min(a_k,b_k)}.$$

Exemples

$$\begin{aligned} a &= 5 \times 11^2 \\ b &= 2 \times 3^2 \times 11^2 \\ c &= 3 \times 5 \times 11^2 \end{aligned}$$

$\mathrm{pgcd}(a,b) = 11^2$, a divise c donc $\mathrm{pgcd}(a,c) = a$, $\mathrm{pgcd}(b,c) = 3 \times 11^2$.

PGCD

Le PGCD dans $\mathbb{Z}$ est maximal pour la divisibilité comme l'indique son nom, (et aussi selon l'ordre usuel).

Caractérisation

$d = \mathrm{pgcd}(a, b)$ si et seulement si l'assertion suivante est vérifiée :

si c divise a et b alors c'est un diviseur de d

Éléments premiers entre eux

☞ Notion liée à la divisibilité

Premiers entre eux

Deux éléments a et b sont dits premiers entre eux (ou a est premier avec b) dès que l'ensemble des diviseurs communs à a et b est réduit aux inversibles.

☞Dans le cas $A = \mathbb{Z}$ les entiers premiers entre eux sont ceux pour qui l'ensemble de leurs diviseurs positifs communs est réduit à $\{1\}$.

☞Lien avec le PGCD ?

Calculer le PGCD : Euclide

On cherche le plus grand diviseur entre 14 et 8.

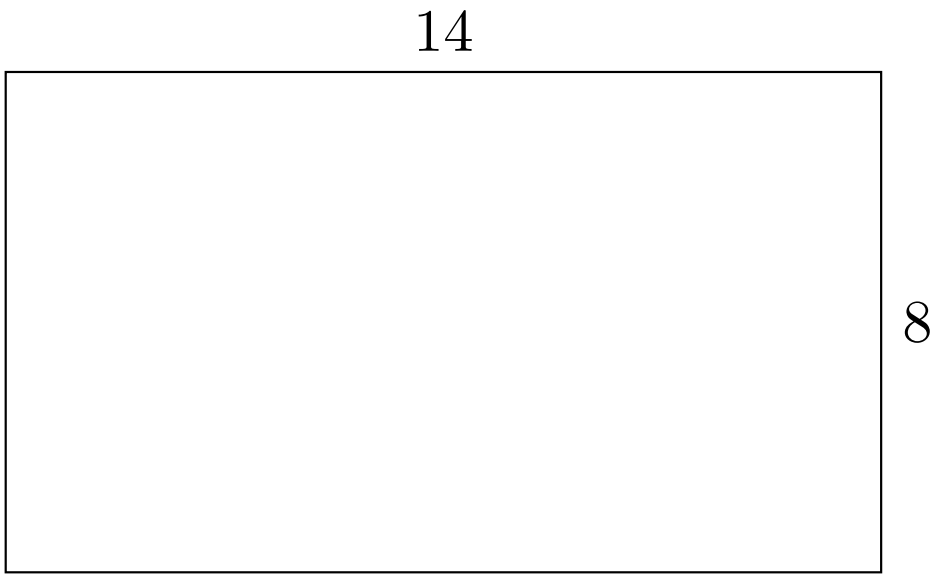
☞On a $14 = 8 + 6$ ainsi comme on cherche d divisant 8 et 14 il suffit de chercher d divisant 8 et 6, on recommence le procédé avec 8 et 6 etc.
☞On s'arrête dès que l'un des entiers considéré est nul.

Lemmes immédiats

- Si d divise a et c et que $(a + b) = c$ alors d divise b.
- Le pgcd d'un élément $a \neq 0$ et de 0 est a.

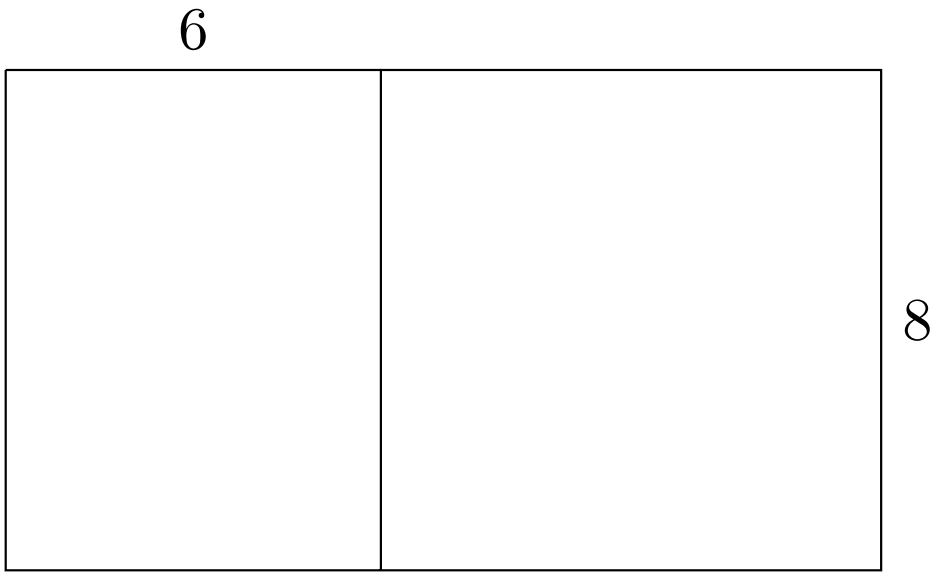
Calculer le PGCD : Euclide

Mathématicien grec (né vers -325, mort vers -265 à Alexandrie, Grèce antique), auteur (supposé) des Éléments, ouvrage fondateur des mathématiques modernes. Il était avant tout géomètre, comme tous les mathématiciens grecs de l'antiquité !



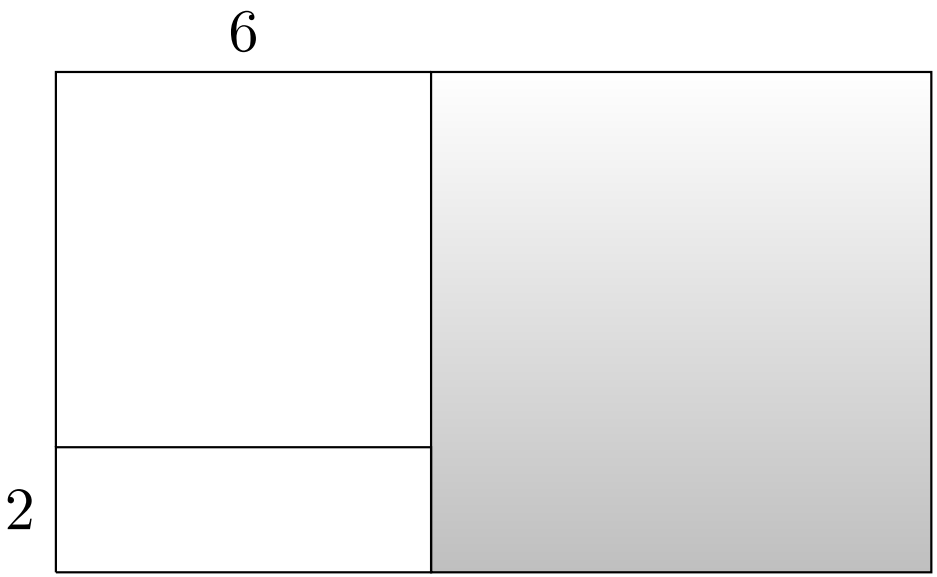
Calculer le PGCD : Euclide

Mathématicien grec (né vers -325, mort vers -265 à Alexandrie, Grèce antique), auteur (supposé) des Éléments, ouvrage fondateur des mathématiques modernes. Il était avant tout géomètre, comme tous les mathématiciens grecs de l'antiquité !



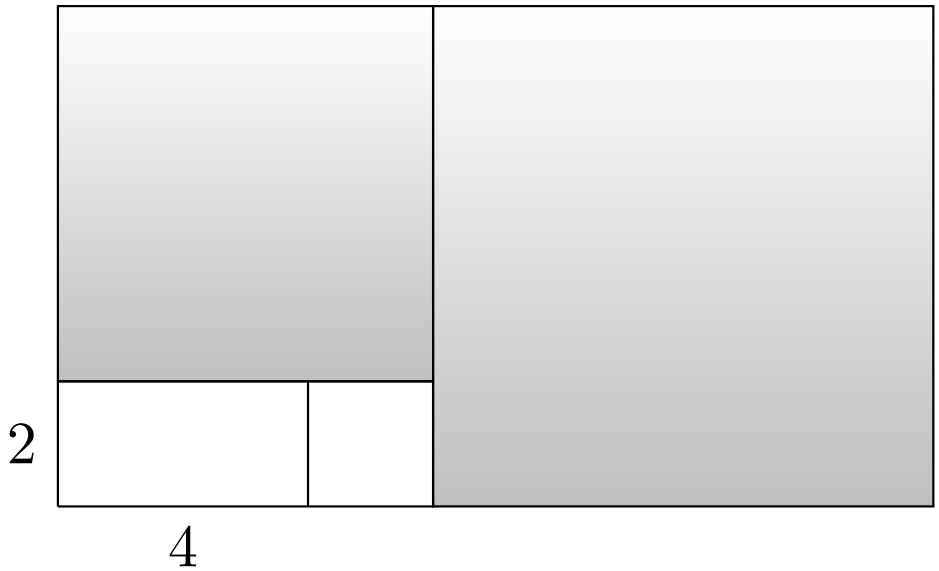
Calculer le PGCD : Euclide

Mathématicien grec (né vers -325, mort vers -265 à Alexandrie, Grèce antique), auteur (supposé) des Éléments, ouvrage fondateur des mathématiques modernes. Il était avant tout géomètre, comme tous les mathématiciens grecs de l'antiquité !



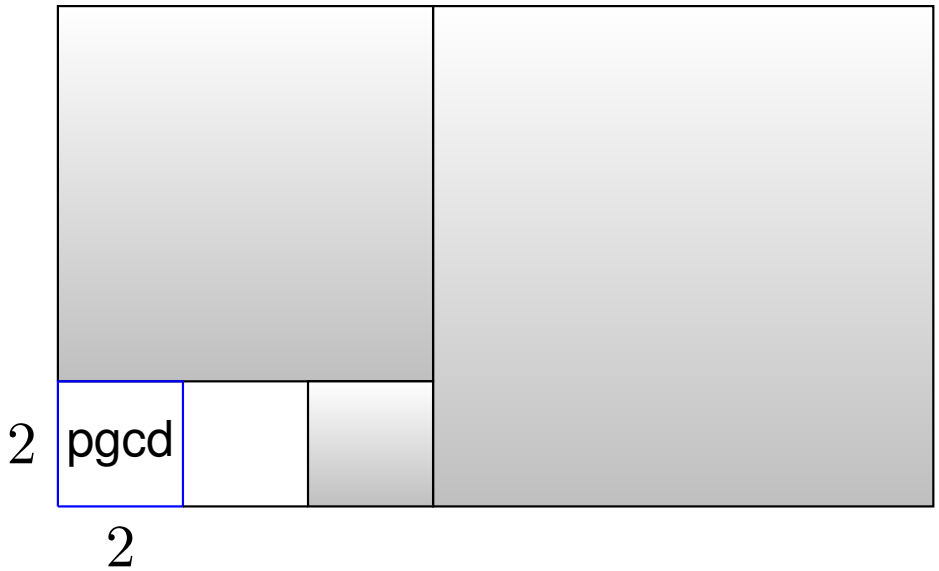
Calculer le PGCD : Euclide

Mathématicien grec (né vers -325, mort vers -265 à Alexandrie, Grèce antique), auteur (supposé) des Éléments, ouvrage fondateur des mathématiques modernes. Il était avant tout géomètre, comme tous les mathématiciens grecs de l'antiquité !



Calculer le PGCD : Euclide

Mathématicien grec (né vers -325, mort vers -265 à Alexandrie, Grèce antique), auteur (supposé) des Éléments, ouvrage fondateur des mathématiques modernes. Il était avant tout géomètre, comme tous les mathématiciens grecs de l'antiquité !



Calculer le PGCD

☞ Plus d'efficacité : division euclidienne dans A

L'anneau A est donc supposé euclidien (de fonction d fixée). Pour calculer le pgcd de deux éléments a et b tels que $d(b) < d(a)$.

Schéma de calcul d'Euclide

Posant $r_0 = a$ et $r_1 = b$ on calcule jusqu'à obtenir un reste nul.

1 si $r_1 \neq 0$ on a $r_0 = r_1 q_1 + r_2$ avec $d(r_2) < d(r_1)$

2 si $r_2 \neq 0$ on a $r_1 = r_2 q_2 + r_3$ avec $d(r_3) < d(r_2)$

$\vdots$

$n-1$ si $r_{n-2} \neq 0$ on a $r_{n-2} = r_{n-1} q_{n-1} + r_n$ avec $d(r_n) < d(r_{n-1})$

n si $r_{n-1} \neq 0$ on a $r_{n-1} = r_n q_n + r_{n+1}$ avec $d(r_{n+1}) < d(r_n)$ et

$$r_{n+1} = 0$$

Algorithme d'Euclide

Le dernier reste non nul r_n est égal au pgcd de a et b.

Algorithme d'Euclide

Ce schéma se traduit immédiatement en C pour les entiers.

```c
#include<stdio.h>
#include<math.h>

int main() {
  int a,b,tmp,r0,r1;
  printf("Entrer deux entiers positifs a > b > 0 : ");
  scanf("%d",&a);
  scanf("%d",&b);

  r0=a;
  r1=b;
  while(r1!=0){
    tmp=r1;
    r1=r0 % r1;
    r0=tmp;
  }
  printf("\nLe pgcd de %d et %d est %d\n",a,b,r0);

  return 0;
}
```

Algorithme d'Euclide : démonstration

☞ Terminaison : c'est immédiat puisque la suite des $d(r_i)$ est une suite strictement décroissante dans $\mathbb{N}$.

☞ Correction : on utilise les deux lemmes immédiats précédemment énoncés pour démontrer les égalités successives :

$$\begin{aligned}\mathrm{pgcd}(a,b) &= \mathrm{pgcd}(r_0,r_1) = \mathrm{pgcd}(r_1q_1+r_2,r_1) = \mathrm{pgcd}(r_2,r_1)\\ &= \mathrm{pgcd}(r_1,r_2) = \ldots = \mathrm{pgcd}(r_n,r_{n+1}) = \mathrm{pgcd}(r_n,0) = r_n\end{aligned}$$

Théorème de Bachet-Bézout

☞ L'algorithme d'Euclide fournit plus d'informations que la seule valeur du pgcd

Théorème de Bachet-Bézout

Soit a et b deux éléments d'un anneau euclidien A. Il existe deux éléments u et v tel que

$$\mathrm{pgcd}(a, b) = ua + vb.$$

Une telle égalité s'appelle une *relation de Bézout*.

☞ Les coefficients u et v sont appelés les *coefficients de Bézout*
☞ Le théorème de Bachet-Bézout a de nombreuses applications.

Théorème de Bachet-Bézout : démonstration

- En utilisant le schéma de calcul d'Euclide, on construit par récurrence des relations linéaires entre r_i, a et b pour $2 \leqslant i \leqslant n$.
- Pour $i = 0$ et $i = 1$, nous avons $r_0 = a = 1 \times a + 0 \times b$ et $r_1 = b = 0 \times a + 1 \times b$. Pour voir comment cela s'amorce pour $i = 2$ nous avons $r_2 = r_0 - r_1 q_1$ avec $r_0 = a$ et $r_1 = b$ et donc $r_2 = a - q_1 b$.
- On suppose que l'on a les relations $r_{i-1} = u_{i-1}a + v_{i-1}b$ et $r_i = u_i a + v_i b$ pour $i \geqslant 2$. On a

$$r_{i+1} = r_{i-1} - q_i r_i = (u_{i-1} - q_i u_i)a + (v_{i-1} - q_i v_i)b$$

- On a donc montré le résultat par récurrence avec $u_0 = v_1 = 1$, $u_1 = v_0 = 0$, $u_{i+1} = u_{i-1} - q_i\, u_i$ et $v_{i+1} = v_{i-1} - q_i\, v_i$.

Algorithme d'Euclide étendu

La démonstration du théorème de Bachet-Bézout permet de déduire une formule de récurrence pour calculer les coefficients de Bézout de a et b et le $\mathrm{pgcd}(a,b)$. Cette formule se traduit sous la forme d'un algorithme :

Algorithme d'Euclide étendu

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \\ u_{i+1} = u_{i-1} - q_i u_i, & v_{i+1} = v_{i-1} - q_i v_i, & r_{i+1} = r_{i-1} - q_i r_i \quad i \geqslant 1 \end{cases}$$

avec $q_i = \left\lfloor \dfrac{r_{i-1}}{r_i} \right\rfloor$ si $r_i \neq 0$ et arrêt sinon.

On a $r_i = u_i\, a + v_i\, b \ \ \forall i$ et en particulier $r_n = u_n\, a + v_n\, b$ où r_n est le dernier reste non nul, est la relation de Bézout recherchée.

☞Nous verrons plus tard dans le cours, le nombre de calculs nécessaires pour obtenir le pgcd et les coefficients de Bézout.

Plan

L'anneau $(\mathbb{Z}/n\mathbb{Z}, +, \times)$

Soit n un entier positif. On considère la relation d'équivalence $a\mathcal{R}_n b \Leftrightarrow a - b \in n\mathbb{Z}$.

☞Les classes d'équivalence dans $\mathbb{Z}$ peuvent être représentées par les $r + n\mathbb{Z} = \{r + nk \ : \ k \in \mathbb{Z}\}$ avec $r \in \{0, \ldots, n-1\}$.

Addition et Multiplication dans $\mathbb{Z}/n\mathbb{Z}$

- $(r_1 + n\mathbb{Z}) + (r_2 + n\mathbb{Z}) = (r_1 + r_2) + n\mathbb{Z} = r_3 + n\mathbb{Z}$ avec r_3 le reste modulo n de $r_1 + r_2$ vu dans $\mathbb{Z}$.
- $(r_1 + n\mathbb{Z}) \times (r_2 + n\mathbb{Z}) = (r_1 \times r_2) + n\mathbb{Z} = r_3 + n\mathbb{Z}$ avec r_3 le reste modulo n de $r_1 \times r_2$ vu dans $\mathbb{Z}$.

☞Muni de ces deux opérations $\mathbb{Z}/n\mathbb{Z}$ est un anneau commutatif.

☞Les calculs se font simplement modulo n.

Lien entre PGCD et multiples

Proposition

Soient $n\mathbb{Z}$ et $m\mathbb{Z}$ deux ensembles de multiples dans $\mathbb{Z}$. On a

$$n\mathbb{Z} + m\mathbb{Z} = \{a_1 + a_2 \ : \ a_1 \in n\mathbb{Z} \text{ et } a_2 \in m\mathbb{Z}\} = \mathrm{pgcd}(n, m)\mathbb{Z}.$$

et

$$n\mathbb{Z} \cap m\mathbb{Z} = \mathrm{ppcm}(n, m)\mathbb{Z}.$$

- L'ensemble des entiers qui sont des multiples de n **ou** de m sont les multiples de $\mathrm{pgcd}(m, n)$.
- L'ensemble des entiers qui sont des multiples de n **et** de m sont les multiples de $\mathrm{ppcm}(m, n)$.

Lien entre PGCD et éléments premiers entre eux

Rappel de la définition

Deux entiers a et b sont premiers entre eux ssi l'ensemble des diviseurs communs à a et b est réduit aux inversibles.

Corollaires du théorème de Bachet-Bézout

Deux entiers a et b sont premiers entre eux ssi

- leur plus grand commun diviseur est inversible et puisqu'il est positif c'est 1 :

$$\mathrm{pgcd}(a,b) = 1$$

- il existe u et v tel que

$$ua + vb = 1$$

- l'ensemble des entiers qui sont multiples de a ou de b forme l'anneau $\mathbb{Z}$ tout entier

$$a\mathbb{Z} + b\mathbb{Z} = 1\mathbb{Z} = \mathbb{Z}$$

$\mathbb{Z}/n\mathbb{Z}$ intégre ?

Intégrité

$\mathbb{Z}/n\mathbb{Z}$ est intègre ssi n est premier.

On démontre l'équivalence entre $\mathbb{Z}/n\mathbb{Z}$ admet des diviseurs de zéro et n non premier dans $\mathbb{Z}$.

Éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$

Éléments inversibles

Un élément $x \in \mathbb{Z}/n\mathbb{Z}$ est inversible ssi il existe $y \in \mathbb{Z}/n\mathbb{Z}$ tq $xy = 1 \bmod n$. Il existe donc u dans $\mathbb{Z}$ tel que $xy + nu = 1$, on pourra donc déterminer y en calculant une relation de Bézout pour x et n par l'algorithme d'Euclide étendu !

Théorème

Les éléments inversibles dans $\mathbb{Z}/n\mathbb{Z}$ sont exactement les classes ayant un représentant premier avec n.

Éléments inversibles dans $\mathbb{Z}/n\mathbb{Z}$

L'ensemble $(\mathbb{Z}/n\mathbb{Z})^\times$ des inversibles de $\mathbb{Z}/n\mathbb{Z}$ est donné par les éléments a vérifiant

$$\mathrm{pgcd}(a, n) = 1$$

L'inverse étant calculé par l'*algorithme d'Euclide étendu*

Exemple : L'élément 10 dans $\mathbb{Z}/21\mathbb{Z}$ (représentant la classe $10 + 21\mathbb{Z}$) est inversible d'inverse $-2 = 19 \bmod 21$. En effet nous avons la relation de Bézout (dans $\mathbb{Z}$) :

$$10 \times (-2) + 21 \times 1 = 1$$

$\mathbb{Z}/n\mathbb{Z}$ corps ?

Corollaire

Si n est irréductible ($\Leftrightarrow$ premier dans $\mathbb{Z}$) alors tous les éléments non nuls de $\mathbb{Z}/n\mathbb{Z}$ sont inversibles.

Définition

Un anneau dont tous les éléments différents de l'unité pour la première opération sont inversibles est un corps.

Théorème

L'anneau $\mathbb{Z}/n\mathbb{Z}$ est un corps ssi n est premier.

Calcul modulaire pratique

Les calculs se font simplement modulo n :

- **on évite soigneusement que les nombres grossissent**
- il est inutile de dépasser n pour une addition et $n^2/4$ pour une multiplication
- pour cela on peut utiliser un représentant non canonique pour simplifier les calculs intermédiaires (typiquement remplacer c par $c - n$ si c est le représentant canonique et $|c - n| < |c|$) et on normalise le résultat à la fin du calcul.

Exemple dans $\mathbb{Z}/7\mathbb{Z}$:
$3 + 6 = 3 - 1 = 2 \bmod 7$,
$2 - 5 = 2 - (-2) = 4 \bmod 7$,
$3 \times 6 = 3 \times (-1) = -3 = 4 \bmod 7$.

Modulo 7 évidemment les résultats sont tout petits mais quand on travaille modulo un nombre plus grand réduire le plus possible la taille des calculs intermédiaires est important.

Calcul modulaire pratique

Principe de base modulo n

Ne jamais laisser grossir le représentant de la classe. On effectue donc des réductions modulo n à la suite de chaque opération.

Comment calculer $15^{362} \bmod 26$?

Calcul modulaire pratique

Principe de base modulo n

Ne jamais laisser grossir le représentant de la classe. On effectue donc des réductions modulo n à la suite de chaque opération.

Comment calculer $15^{362} \bmod 26$?

Ne surtout PAS calculer la puissance !

$15^{362} =$

55595005580040482871625772446775226015821569874302030603328442299724263819359011468269851087775194579110001125944220531149967715499410818881365298654871488787900846061147144666746229316248215432672162996790328783757404986244660505653182529444153101670870110858585193623743922979886472335092027708567006431517665562809292209166977242909844074803420594825320639261901697778149415199151848337999481941551493946462869644165039062 5

Calcul modulaire pratique

Comment calculer $15^{362} \bmod 26$ sans factoriser 15 ?

- on cherche une petite puissance de 15 qui vaudra 1 ou -1 parce que on peut prédire la valeur de 1^k ou $(-1)^k$ quelle que soit la valeur de l'exposant k
- directement $15^2 = 225 = 17 \bmod 26$, $17^2 = 289 = 3 \bmod 26$, $3^3 = 27 = 1 \bmod 26$ [1], donc $1 = 15^{2*2*3} = 15^{12} \bmod 26$,
- ou en passant toujours par le représentant le plus petit en valeur absolue $15 = -11 \bmod 26$, $(-11)^2 = 121 = -9 \bmod 26$, $(-9)^2 = 81 = 3 \bmod 26$ et $3^3 = 27 = 1 \bmod 26$[1], donc on conclut là aussi que $15^{2*2*3} = 1 \bmod 26$,
- on décompose alors $362 = 12 * 30 + 2$,
- il reste à calculer $15^{362} = (15^{12})^{30} \times 15^2 = 1^{30} \times 17 = 17 \bmod 26$.

1. on pousse jusqu'à 3^3 car $3^2 \neq 1 \bmod 26$ et $3^2 \neq -1 \bmod 26$ et 3^3 est la première puissance de 3 qui dépasse 26, qui peut donc conduire à une valeur inférieure par réduction modulo 26

Calcul modulaire pratique

Comment calculer $15^{362} \bmod 26$ en factorisant 15 ?

- $15^{362} = 3^{362} \times 5^{362} \bmod 26$
- $3^3 = 27 = 1 \bmod 26$ et $5^2 = 25 = -1 \bmod 26$ donc $5^4 = 1 \bmod 26$
- $362 = 3 * 120 + 2 = 4 * 90 + 2$
- $3^{362} = (3^3)^{120} \times 3^2 = 1^{120} \times 3^2 = 3^2 = 9 \bmod 26$
- $5^{362} = (5^4)^{90} \times 5^2 = 1^{90} \times (-1) = -1 \bmod 26$
- $15^{362} = 3^{362} \times 5^{362} = 9 \times (-1) = -9 = 17 \bmod 26$

Algorithme Euclide étendu : mise en pratique !

Comment trouver l'inverse de 2345 modulo 30003 s'il existe ?

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \\ u_{i+1} = u_{i-1} - q_i u_i, & v_{i+1} = v_{i-1} - q_i v_i, & r_{i+1} = r_{i-1} - q_i r_i \quad i \geqslant 1 \end{cases}$$

avec $q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor$ si $r_i \neq 0$ et arrêt sinon.

$$r_i = u_i\, a + v_i\, b \quad \forall i$$

$\mathrm{pgcd}(a, b) = r_n = u_n\, a + v_n\, b$ où r_n est le dernier reste non nul.

Calcul pratique du pgcd étendu (à la main ou en machine) : initialisation

i	r_{i-1}	r_i	q_i	r_{i+1}	u_{i+1}	v_{i+1}
-1				30003	1	0
0		30003		2345	0	1
1	30003	2345				

Algorithme Euclide étendu : mise en pratique !

Comment trouver l'inverse de 2345 modulo 30003 s'il existe ?

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \\ u_{i+1} = u_{i-1} - q_i u_i, & v_{i+1} = v_{i-1} - q_i v_i, & r_{i+1} = r_{i-1} - q_i r_i \quad i \geqslant 1 \end{cases}$$

avec $q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor$ si $r_i \neq 0$ et arrêt sinon.

$$r_i = u_i\, a + v_i\, b \quad \forall i$$

$\mathrm{pgcd}(a, b) = r_n = u_n\, a + v_n\, b$ où r_n est le dernier reste non nul.

Calcul pratique du pgcd étendu (à la main ou en machine) : remplissage du tableau

i	r_{i-1}	r_i	q_i	r_{i+1}	u_{i+1}	v_{i+1}
-1				30003	1	0
0		30003		2345	0	1
1	30003	2345	12	1863	1	-12
2	2345	1863	1	482	-1	13
3	1863	482	3	417	4	-51
4	482	417	1	65	-5	64
5	417	65	6	27	34	-435
6	65	27	2	11	-73	934
7	27	11	2	5	180	-2303
8	11	5	2	1	-433	5540
9	5	1	5	**0**	2345	-30003

Algorithme Euclide étendu : mise en pratique !

Comment trouver l'inverse de 2345 modulo 30003 s'il existe ?

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \\ u_{i+1} = u_{i-1} - q_i u_i, & v_{i+1} = v_{i-1} - q_i v_i, & r_{i+1} = r_{i-1} - q_i r_i \quad i \geqslant 1 \end{cases}$$

avec $q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor$ si $r_i \neq 0$ et arrêt sinon.

$$r_i = u_i\, a + v_i\, b \quad \forall i$$

$\mathrm{pgcd}(a, b) = r_n = u_n\, a + v_n\, b$ où r_n est le dernier reste non nul.

Calcul pratique du pgcd étendu (à la main) :

i	r_{i-1}	r_i	q_i	r_{i+1}	u_{i+1}	v_{i+1}
1	30003	2345	12	1863	1	-12
⋮	⋮	⋮	⋮	⋮	⋮	⋮
8	11	5	2	**1**	−433	5540
9	5	1	5	**0**	2345	−30003

☞On en déduit que -433$\times$ 30003 + 5540$\times$ 2345 =1 dans $\mathbb{Z}$

☞et donc 5540=2345^{-1} mod 30003 dans $\mathbb{Z}/30003\mathbb{Z}$.

Exercice inversible / diviseur de zéro

Le problème à résoudre

Vérifier si un nombre a est inversible modulo N et si oui fournir son inverse et sinon exhiber son témoin de diviseur de zéro.

L'algorithme utilisé, le tableau de présentation des calculs

- On va appliquer l'algorithme d'Euclide étendu à N et a
- On présente l'algorithme (initialisation, étape de récurrence, condition d'arrêt, invariant vérifié pour chaque ligne, conclusion à tirer à la fin) pour que le lecteur puisse comprendre le tableau qui va suivre
- On pré-remplit le tableau précédent avec les conditions initiales, puis on applique la formule de récurrence tant que la condition d'arrêt n'est pas remplie
- Enfin on examine les deux dernières lignes du tableau et on tire des conclusions...

Exercice inversible / diviseur de zéro

Les conclusions à tirer des calculs

À la dernière ligne, la ligne n, on a $r_{n+1} = 0$; on regarde la valeur de r_n sur la ligne précédente :

- soit $r_n = 1$ comme c'est le cas dans l'exemple précédent, a est inversible modulo N d'inverse v_n avec $u_n\, N + v_n\, a = 1$,
- soit $r_n > 1$, a et N ont pour pgcd $r_n > 1$, a n'est pas inversible modulo N, a est un diviseur de zéro dans $\mathbb{Z}/N\mathbb{Z}$ et on a deux façons de calculer le témoin de zéro de a
  - N/r_n est témoin de zéro pour a, c'est exact mais cela nécessite d'effectuer une division qui est une opération coûteuse
  - en fait l'invariant pour la dernière ligne nous fournit la relation $0 = u_{n+1}\, N + v_{n+1}\, a$ et $|v_{n+1}|$ est témoin de a comme diviseur de zéro dans $\mathbb{Z}/N\mathbb{Z}$ sans qu'aucun autre calcul ne soit nécessaire.

Exercice inversible / diviseur de zéro

Réflexions sur l'exercice : informations superflues

Il y a des informations superflues dans notre tableau :

- les deux premières colonnes qui nous aident visuellement à suivre la division euclidienne de r_{i-1} par r_i, si vous n'en avez pas besoin. . . le lecteur non plus, à vous de voir
- l'avant-dernière colonne : en effet dans cette colonne il y a le coefficient de Bézout pour N ; or c'est le coefficient pour a qui importe, que a soit inversible ou diviseur de zéro dans $\mathbb{Z}/N\mathbb{Z}$.

  Cette redondance nous permet d'effectuer des vérifications de pertinence : on ne va pas forcément calculer $u_i\,N + v_i\,a$ pour s'assurer que c'est bien r_i, mais on va vérifier que c'est vrai modulo 10 par exemple ou que l'ordre de grandeur est crédible
- et même la dernière ligne ne nous est pas utile si $r_n = 1$, mais si on détermine u_{n+1} et v_{n+1}, on peut vérifier que $|u_{n+1}| = |a|$ et $|v_{n+1} = |N|$ et sont de signes opposés, ce qui valide la suite des calculs intermédiaires et en particulier v_n l'inverse de a modulo N.

Exercice inversible / diviseur de zéro

Réflexions sur l'exercice : vérification de vraisemblance

De façon générale, u_i et v_i sont de signes opposés (sinon comment $u_i N + v_i a$ peut être plus petit que a et N ?) et il y a alternance des signes pour ces deux suites.

Vous avez le droit de vous tromper dans les calculs, par inattention, à cause du stress de l'examen, mais ces vérifications minimales sur les signes ou sur la relation de Bézout modulo 10 permettent de se rassurer rapidement.
Si cela détecte un problème mais que vous n'avez pas le temps d'en trouver l'origine et de le corriger, indiquez-le sur votre copie, montrez que vous en avez conscience, que vous ne faites pas des calculs en fonçant tête baissée, cela sera porté à votre crédit et on suivra plus volontiers la suite de votre calcul à partir de l'erreur.

Et rassurez-vous : dans un exercice où l'on vous fait faire les calculs à la main, les nombres concernés ont 1 à 2 chiffres de moins et le tableau n'a pas plus de 5 ou 6 lignes !

Plan

Chiffrement affine : problème résolu

Cryptosystème affine

- $\mathcal{P}, \mathcal{C}$ sont tous les deux égaux à $\mathbb{Z}/26\mathbb{Z}$
- $\mathcal{K}$ est un sous-ensemble de $\mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z}$ tel que
- Pour tout $K = (a, b) \in \mathcal{K}$ on a
  - $e_K(x) = ax + b \bmod 26$
  - et il existe une fonction $d_K : \mathcal{C} \to \mathcal{P}$ inverse de e_K.

☞Comment déterminer l'ensemble $\mathcal{K}$?

Étant donnés a, b, y il faut résoudre l'équation

$$y = ax + b \bmod 26$$

Ceci n'est possible que si a est inversible modulo 26. On obtient alors $d_K(y) = a^{-1}(y - b)$.

☞On a donc $\mathcal{K} = \mathbb{Z}/26\mathbb{Z}^\times \times \mathbb{Z}/26\mathbb{Z}$. Combien $\mathcal{K}$ a-t-il d'éléments ?

Chiffrement affine : Indicatrice d'Euler

Cryptosystème affine

- $\mathcal{P}, \mathcal{C}$ sont tous les deux égaux à $\mathbb{Z}/26\mathbb{Z}$
- $\mathcal{K}$ est le sous-ensemble des $(a, b) \in \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z}$ tel que a est premier avec 26.

Indicatrice d'Euler

Étant donné un entier n, le nombre d'éléments strictement positifs, inférieurs à n et premiers avec n est noté $\varphi(n)$. Ce nombre est appelé indicatrice d'Euler de n et noté

$$\varphi(a) = \mathrm{card}\{0 < n < a \ : \ \mathrm{pgcd}(a, n) = 1\}$$

Le nombre de clefs possibles est donc $\varphi(26) \times 26$.

Conclusion

Chiffrement de César et chiffrement affine

- Nous avons vu l'utilisation des structures mathématiques de base pour définir des cryptosystèmes.
- Arithmétique modulaire et groupes de permutations.
- Je vous conseille de réviser l'algorithme d'Euclide étendu et son utilisation pour calculer les coefficients de Bézout !

Plan Les objets mathématiques de base de la cryptographie

Deuxième partie II

Cryptanalyse des chiffrements mono-alphabétique et Vigenère

Plan

Analyse

Cryptanalyse

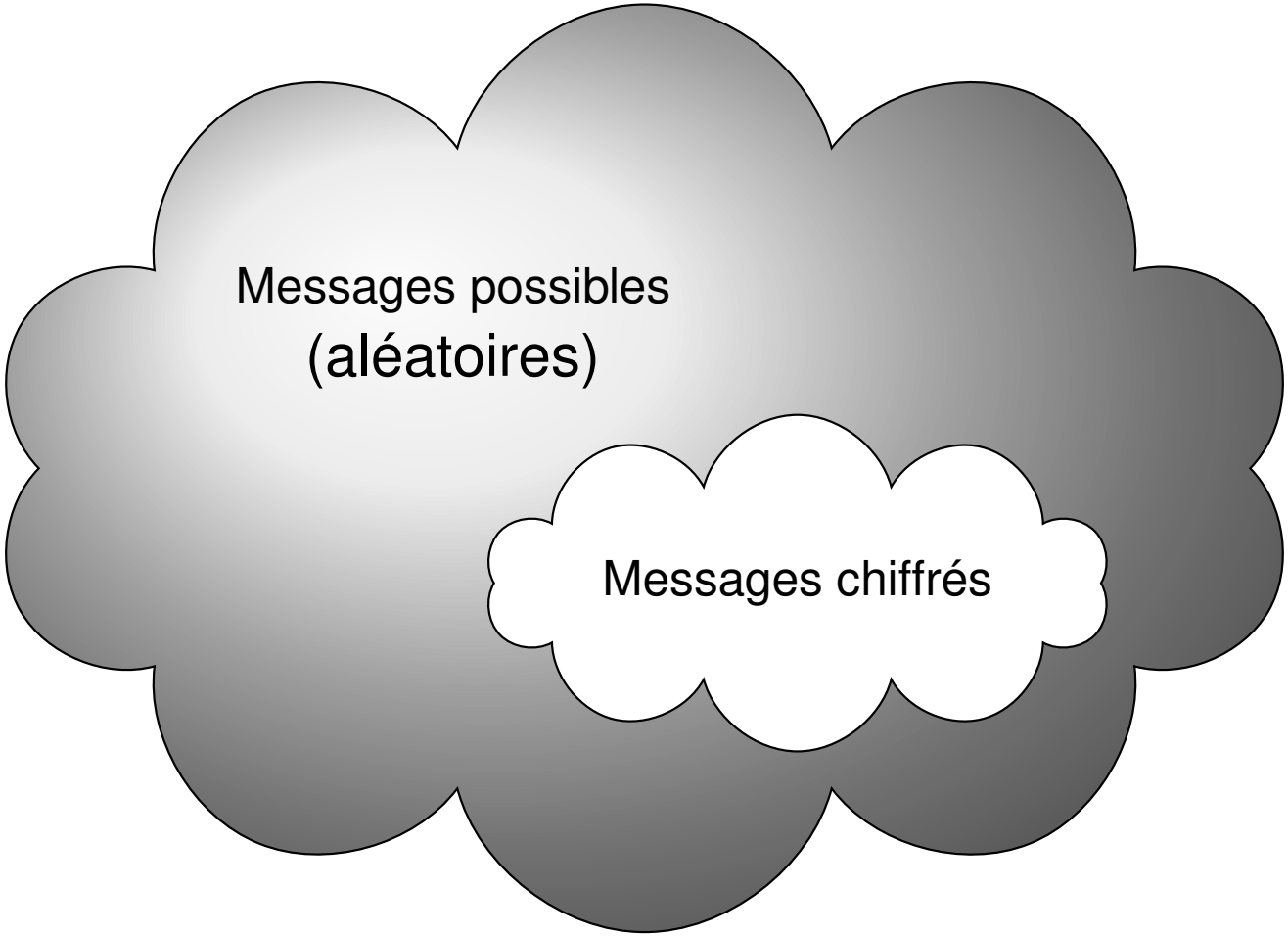
- Analyse de la sécurité d'un cryptosystème (théorique, complexité)
- Attaquer une instance particulière d'un cryptosystème (retrouver la clef, un message clair)

Plusieurs niveaux d'analyse

- Clair/Chiffré inconnu
- Couple Clair/Chiffré connu
- Chiffré choisi (on a accès à un appareil de déchiffrement, boîte noire)
- Clair choisi (on a accès à un appareil de chiffrement, boîte noire)

☞Pour les messages très courts on a recours à l'attaque par mot probable : on a des raisons de penser que certains mots sont présents dans le texte à un endroit prévisible (entête, signature, etc.)

Cryptanalyse : distingueur



☞Comment distinguer les messages chiffrés des messages aléatoires ?

☞Utiliser ce distingueur pour attaquer le cryptosystème.

Distingueur : rappel de probabilités

Définitions

Étant donné un ensemble fini Ω d'*évènements atomiques*. Une fonction $\mathbb{P}$ définie pour tout $\omega \in \Omega$ et à image dans $\mathbb{R}$ est appelée *probabilité* dès que $0 \leqslant \mathbb{P}(\omega) \leqslant 1, \forall \omega \in \Omega$ et $\sum_{\omega \in \Omega} \mathbb{P}(\omega) = 1$.
Un *évènement* E est un sous-ensemble de Ω et sa probabilité d'apparition sera donné par

$$\mathbb{P}(E) = \sum_{\omega \in E} \mathbb{P}(\omega)$$

en particulier $\mathbb{P}(\emptyset) = 0$ et $\mathbb{P}(\Omega) = 1$

Exemples

- Pile ou face : $\Omega = \{P, F\}, \quad \mathbb{P}(F) = \frac{1}{2}, \mathbb{P}(P) = \frac{1}{2}$
- Deux dés à 6 faces :
  $\Omega = \{(1,1), (1,2), \ldots, (6,5), (6,6)\}, \mathbb{P}((m,n)) = \frac{1}{36}, \mathbb{P}((1,*)) = \frac{6}{36}$

Plan

Distingueur : étude statistique des caractères

Soit c un caractère tiré aléatoirement. On cherche à analyser des évènements du genre

{Le caractère c provient d'un texte français et est un A}

☞On ne peut pas faire une analyse sur tous les textes français possibles. Principe de la statistique, on *échantillonne* sur un ensemble fini représentatif et on extrapole les résultats.

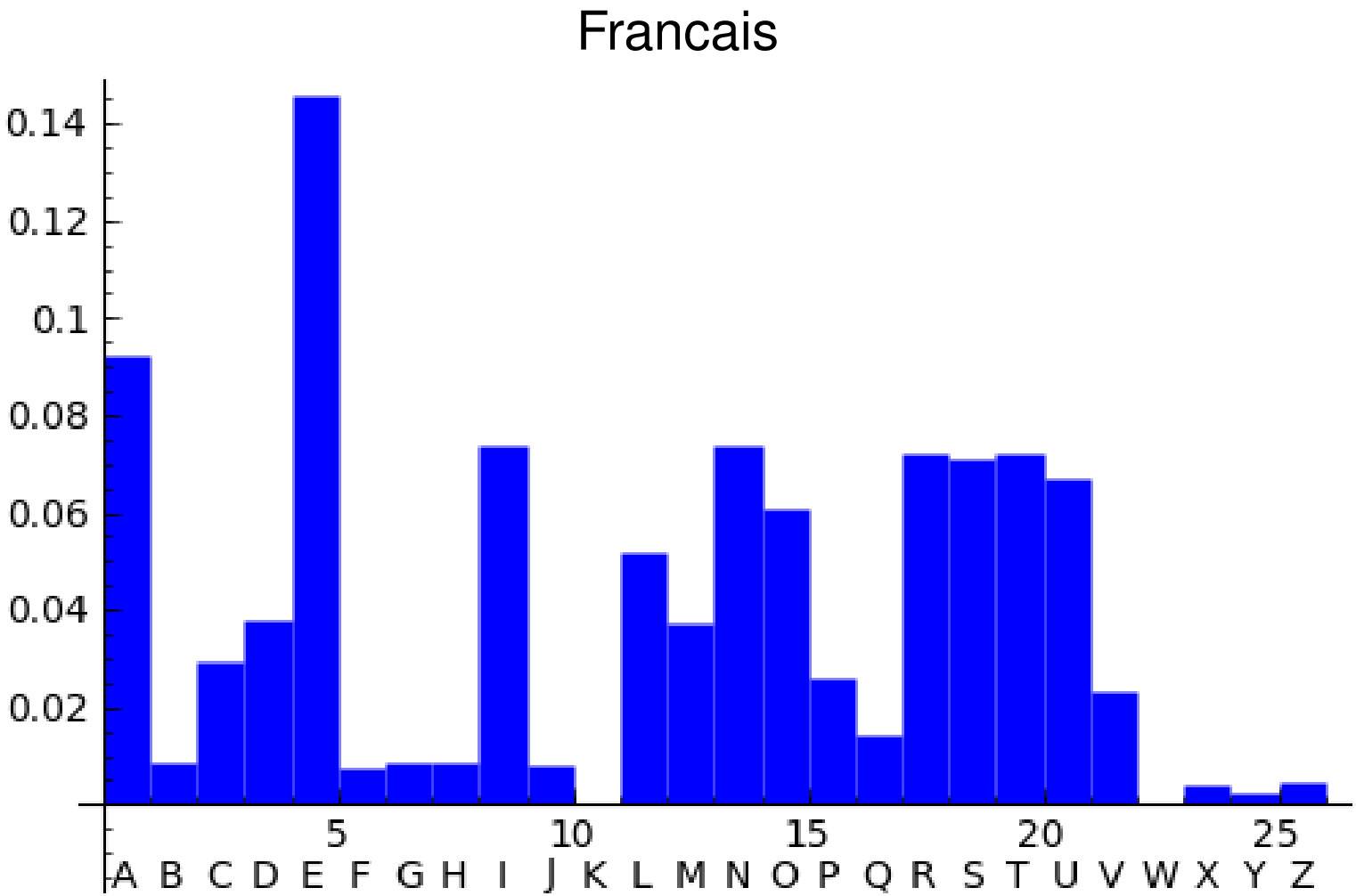
Distingueur

- Dans un texte correspondant à un flux aléatoire de caractères on a $\mathbb{P}(\mathsf{A}) = 1/26 \simeq 3.8\%$
- Dans un texte correspondant à un flux français de caractères on a $\mathbb{P}(\mathsf{A}) \simeq 9.2\%$

Distingueur : la fréquence d'apparition des caractères.

Distingueur : étude statistique des caractères

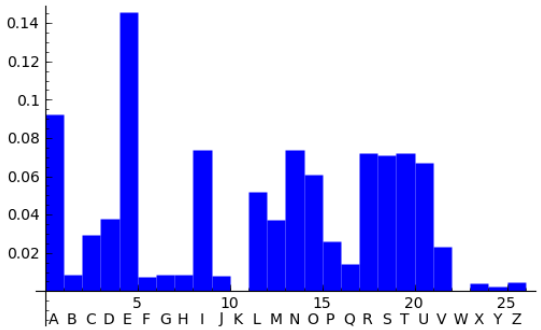
☞La probabilité d'apparition des caractères est un distingueur entre les différentes langues et l'aléa.



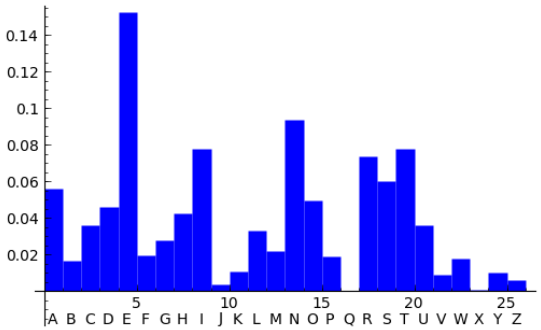
Distingueur : étude statistique des caractères

☞La probabilité d'apparition des caractères est un distingueur entre les différentes langues et l'aléa.

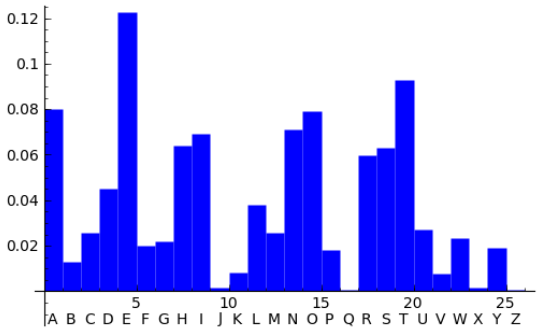
Francais



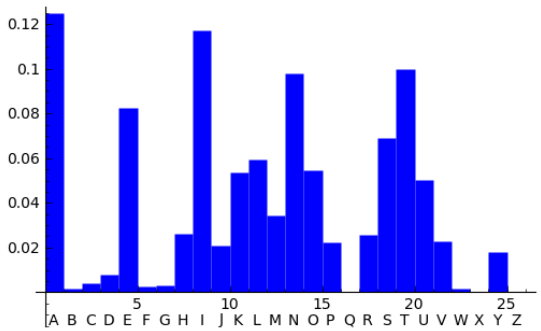
Allemand



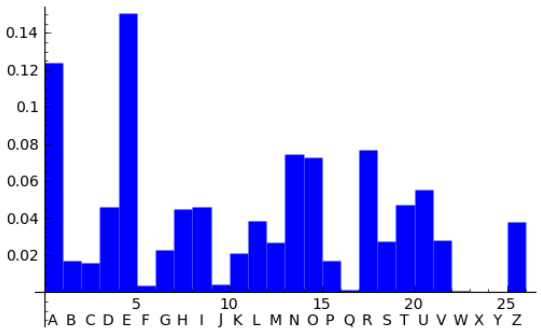
Anglais



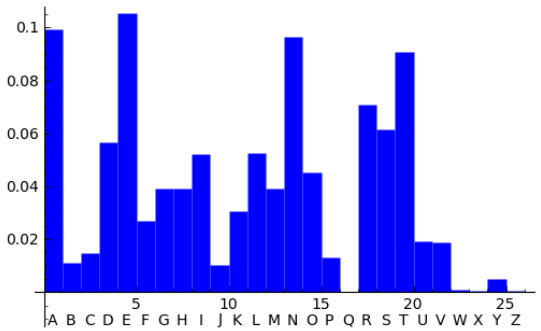
Finnois



Breton



Suedois



Cryptanalyse du chiffrement mono-alphabétique

☞Al Kindi ($\simeq 800$) explique dans son ouvrage de cryptanalyse la méthode de l'étude des fréquences.
Il explique exactement ce que nous venons de voir : la distribution des caractères dans un texte permet de le caractériser.

Exemple : Cryptanalyse du décalage

Soit à retrouver le texte clair français correspondant au chiffré suivant

PITVIWMHIRXIWXQEMRXIRERXIRZEGERGIWEY

QEVSGMPIWXEGGSQTSKRIHIWEJIQQIIXHIWIW

XVSMWIRJERXWXSYXIWPIWGSQQYRMGEXMSRWH

IZVSRXIXVIGLMJJVIIWTEVPIGSHIBXVYREKI

YMTIHYRIZEPMWIHIGSQQYRMGEXMSRGLMJJVE

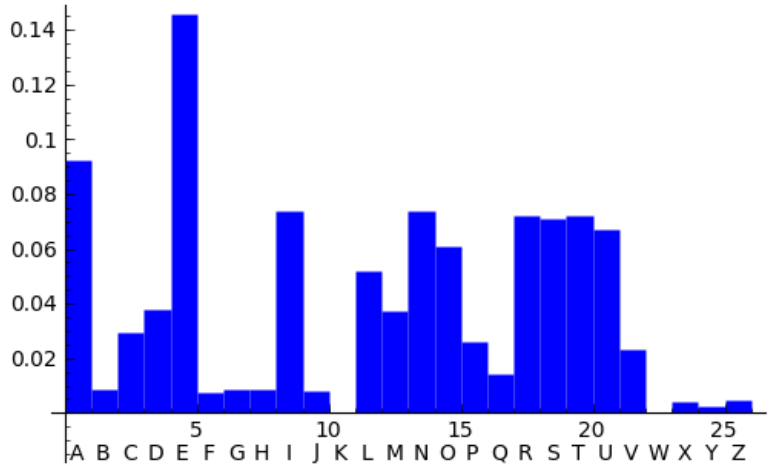
RXPIWYMZVEXSYXEYPSRKHYWINSYVIXWIVEIU

RXIMPSGGYTIVEPITEPEGIJPSXXERXJEGIEPE

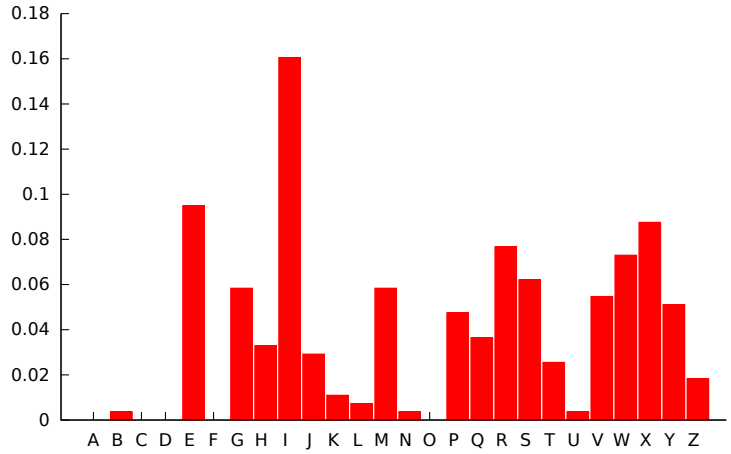
ZIRYIHIWXVSMWTEPQMIVW

Exemple : Cryptanalyse du décalage

Francais



Chiffré



On lit le décalage 4 et on obtient

Exemple : Cryptanalyse du décalage

PITVIWMHIRXIWXQEMRXIRERXIRZEGERGIWEY
LEPRESIDENTESTMAINTENANTENVACANCESAU
QEVSGMPIWXEGGSQTSKRIHIWEJIQQIIXHIWIW
MAROCILESTACCOMPOGNEDESAFEMMEETDESES
XVSMWIRJERXWXSYXIWPIWGSQQYRMGEXMSRWH
TROISENFANTSTOUTESLESCOMMUNICATIONSD
IZVSRXIXVIGLMJJVIIWTEVPIGSHIBXVYREKI
EVRONTETRECHIFFREESPARLECODEXTRUNAGE
RXPIWYMZVEXSYXEYPSRKHYWINSYVIXWIVEIU
NTLESUIVRATOUTAULONGDUSEJOURETSERAEQ
YMTIHYRIZEPMWIHIGSQQYRMGEXMSRGLMJJVE
UIPEDUNEVALISEDECOMMUNICATIONCHIFFRA
RXIMPSGGYTIVEPITEPEGIJPSXXERXJEGIEPE
NTEILOCCUPERALEPALACEFLOTTANTFACEALA
ZIRYIHIWXVSMWTEPQMIVW
VENUEDESTROISPALMIERS

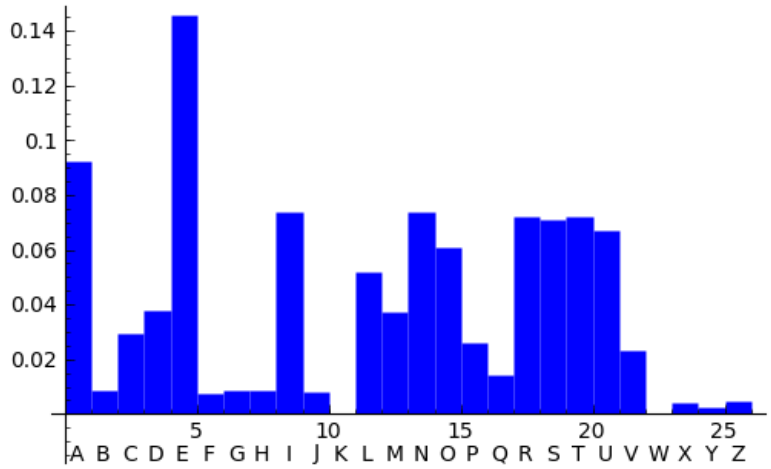
Exemple : Cryptanalyse du mono-alphabétique

Soit à retrouver le texte clair francais correspondant au chiffré suivant

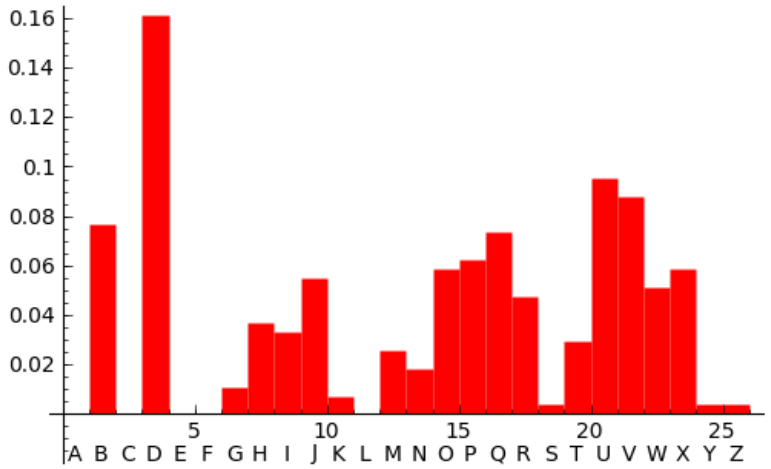
```
RDMJDQXIDBVDQVHUXBVDBUBVDBNUOUBODQUW

HUJPOXRDQVUOOPHMPGBDIDQUTDHHDDVIDQDQ

VJPXQDBTUBVQVPWVDQRDQOPHHWBXOUVXPBQI

DNJPBVDVJDOKXTTJDDQMUJRDOPIDSVJWBUGD

BVRDQWXNJUVPWVUWRPBGIWQDYPWJDVQDJUDZ

WXMDIWBDNURXQDIDOPHHWBXOUVXPBOKXTTJU

BVDXRPOOWMDJURDMURUODTRPVVUBVTUODURU

NDBWDIDQVJPXQMURHXDJQ
```

Exemple : Cryptanalyse du chiffrement mono-alphabétique

Francais



Chiffré



La clef est la permutation :

[ULOIDTGKXYCRHBPMZJQVWNFSAE]

Exemple : Cryptanalyse du mono-alphabétique

- Méthode force brute assistée avec des hypothèses à faire tout au long de l'attaque.
- Besoin de plus que la fréquence des lettres pour y arriver efficacement, il faut s'intéresser à la fréquence de groupes de lettres, à la façon dont les caractères se succèdent, s'articulent.
- En étudiant les fréquences des bigrammes `E?` et `?E` on peut distinguer parmi des lettres de fréquences individuelles similaires telles que `S`, `N`, `A` et `I` les consonnes plus souvent appariées à `E` que les voyelles (assistance technique à une cryptanalyse manuelle).

Bigrammes	Pourcentages	Bigrammes	Pourcentages
ES	2,9	SE	1,4
EN	2,3	NE	1,2
ET	1,5	TE	1,5
ER	1,5	RE	2,1
EA	0.53	AE	0.58
EI	0.29	IE	1,2

- De même on peut utiliser la fréquence des tétragrammes dans une stratégie itérative de hill climbing (marche aléatoire, automatisation quasi complète).

Hill climbing sur le texte précédent

```
% python3 crypt_auto_mono.py texte
texte chiffré
RDMJDQXIDBVDQVHUXBVDBUBVDBNUOUBODQUWHUJPOXRDQVUOOPHMPGBDIDQ
UTDHHDDVIDQDQVJPXQDBTUBVQVPWVDQRDQOPHHWBXOUVXPBQIDNJPBVDVJD
OKXTTJDDQMUJRDOPIDSVJWBUGDBVRDQWXNJUVPWVUWRPBGIWQDYPWJDVQDJ
UDZWXMDIWBDNURXQDIDOPHHWBXOUVXPBOKXTTJUBVDXRPOOWMDJURDMURUO
DTRPVVUBVTUODURUNDBWDIDQVJPXQMURHXDJQ
évaluation -1189.3970275440793

Après 5000 itérations, texte déchiffré
LEPRESIDENTESTMAINTENANTENVACANCESAUMAROCILESTACCOMPOGNEDES
AFEMMEETDESESTROISENFANTSTOUTESLESCOMMUNICATIONSDEVRONTETRE
CHIFFREESPARLECODEXTRUNAGENTLESUIVRATOUTAULONGDUSEJOURETSER
AEQUIPEDUNEVALISEDECOMMUNICATIONCHIFFRANTEILOCCUPERALEPALAC
EFLOTTANTFACEALAVENUEDESTROISPALMIERS
subst appliquée au texte fourni BJWEKYGMDRHZPVCOSLXFATUINQ
clef UAOIDTGKXBERHYPMZJQVWNCSFL
évaluation 2480.916654812422
```

Conclusion 1

☞Le chiffrement mono-alphabétique est très facile à attaquer !

Comment le sécuriser ?

- Substitution homophonique (15ème siècle)

Steno (doge de Venise) 1411

Période d'utilisation intense entre 1500 et 1750, notamment variante Babou pour François Ier

Conclusion 1

☞Le chiffrement mono-alphabétique est très facile à attaquer !

Comment le sécuriser ?

- Substitution homophonique
- Passer au poly-alphabétique (Alberti, Vigenère)

Plan

Plan

Le chiffrement poly-alphabétique par Alberti (15ème siècle)

Leon Battista Alberti
(1404 - 1472)

☞Chiffrement multi-mono-alphabétique

- On chiffre des blocs de texte de longueur ℓ (ligne)
- Pour la lettre d'indice i d'un bloc on applique un chiffrement mono-alphabétique e_{K_i}.

Autrement dit on écrit le texte sur ℓ colonnes et on utilise une permutation différente pour chaque colonne.

☞La clef secrète correspond à ℓ clefs secrètes de chiffrements mono-alphabétiques (permutation d'un alphabet).
☞Trop difficile à utiliser, nécessité de le simplifier.

Le chiffrement poly-alphabétique : un siècle d'évolution lente

- Idée : reprendre le chiffrement d'Alberti en se restreignant aux chiffrements par décalage dans chaque colonne.
- Alberti 1466 : cadran pour (dé)chiffrer
- En Allemagne l'abbé Trithème 1518 : tableau de Trithème qui permet de chiffrer un texte par décalage en augmentant le décalage d'un cran à chaque caractère
- En Italie : Bellaso 1553, repris par Porta 1563 : introduit la notion de mot-clef avec une autre table
- En France : Vigenère 1586 : mot-clef + tableau de Trithème

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

Utilisation historique

- Messages de Marie-Antoinette dans la prison du Temple
- Œuvres de fiction (la plus ancienne : la Jangada de Jules Verne, 1881)
- Pendant la guerre de Sécession (mise en œuvre déplorable).

Chiffrement de Vigenère 1586

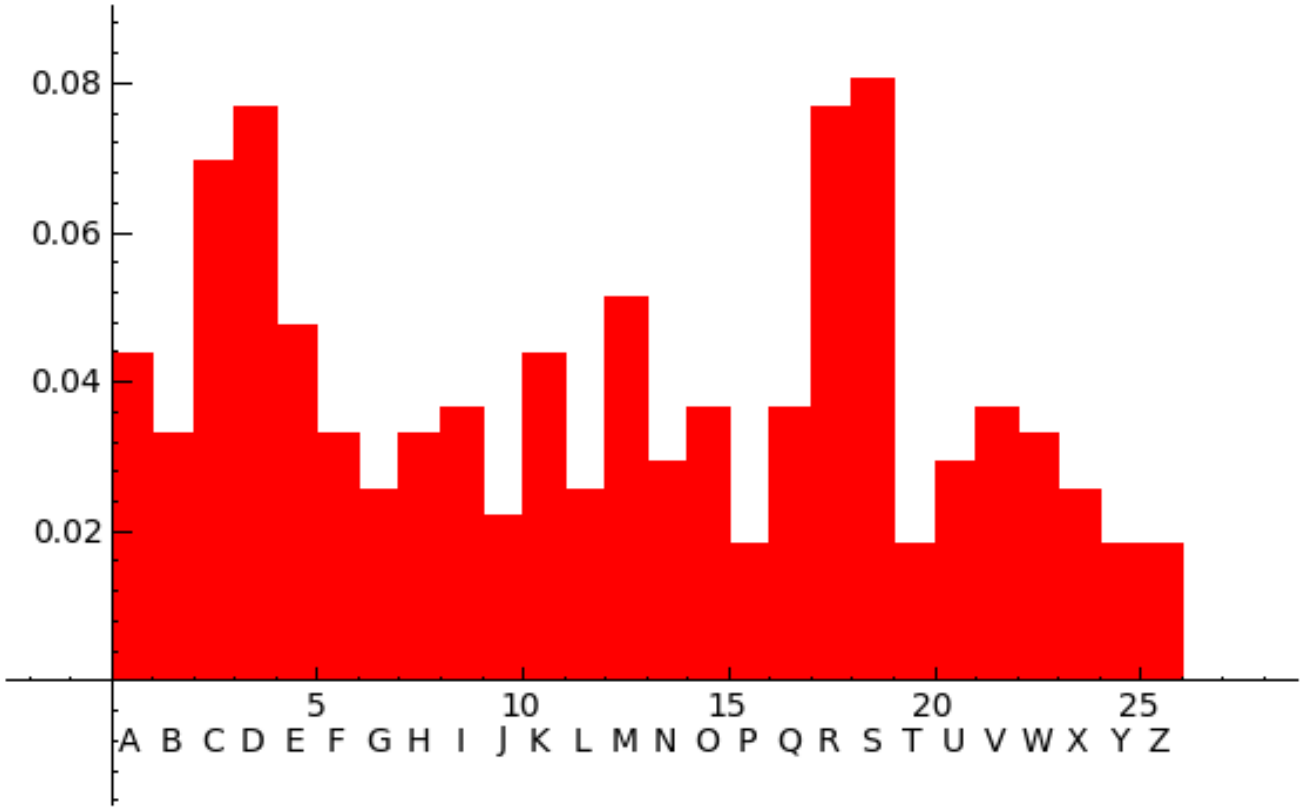
A	T	T	A	Q	U
C	R	Y	P	T	O
C	K	R	P	J	I
E	N	N	E	M	I
C	R	Y	P	T	O
G	E	L	T	F	W
D	E	M	A	I	N
C	R	Y	P	T	O
F	V	K	P	B	B

Chiffrement de Vigenère 1586

☞Idée : reprendre le chiffrement de Alberti en se restreignant aux chiffrements par décalage (en France Vigenère).

☞La clef secrète correspond à un mot de ℓ caractères.

☞Ne conserve pas les propriétés statistiques du chiffrement mono-alphabétique.



Plan

Cryptanalyse étape 1 : longueur de la clef

- Charles Babbage (1792-1871)
- Friedrich Wilhelm Kasiski (1805-1881)

Une des techniques pour calculer la longueur de la clef est le test de Kasiski : il s'agit de repérer les motifs qui se répètent

KQOWEFVJPUJUUNUKGLMEKJINMWUXFQMKJBGWRLFNFGHUD**WUU**MBSVLPS
NCMUEKQCTESWR**EEK**OYSSIWCTUAXYOTAPXPLWPNTCGOJBGFQHTD**WXIZA**
YGFFNSXCSEYNCTSSPNTUJNYTGGWZGR**WUU**NEJUUQEAPYMEKQHUIDUXFP
GUYTSMTFFSH**NUOCZGM**RUWEYTRGKMEEDCTVRECFBDJQCUSWVBPNLGOYL
SKMTEFVJJTWWMFMWPNMEMTMHRSPXFSSKFFST**NUOCZGMDOEOYEEK**CPJR
GPMURSKHFRSEIUEVGOYC**WXIZAY**GOSAANY**DOEOY**JLWUNHAMEBFELXYVL
WNOJNSIOFRWUCCESWKVID**GMU**CGOCRUWGNMAAFFVNSIUDEKQHCEUCPFC
MPVSUDGAVEMNYMAMVLFMAOYFNTQCUAFVFJNXKLNEIWCWODCCULWRIFT
W**GMU**SWOVMATNYBUHTCOCWFYTNMGYTQMKBBNLGFBTWOJFTWGNTEJKNEE
DCLDHWTVBUVGFBIJG

Cryptanalyse étape 1 : longueur de la clef

Principe du test de Kasiski

- Une répétition dans le message clair si elle se produit à la même position dans la clef (la même colonne) produit une répétition dans le message chiffré,
- on repère les répétitions dans le chiffré en espérant qu'elles correspondent à une répétition dans le clair
- si c'est le cas la distance (le nombre de caractères entre le début de chaque occurrence) entre elles fournit un multiple de la longueur de la clef.

☞Le pgcd des distances entre les répétitions de motif est probablement un multiple de la longueur de la clef

Nous verrons par la suite une méthode plus fiable et plus facile à automatiser pour calculer la longueur de la clef

Test de Kasiski

Distances entre deux occurrences

Répétition	Distance	Factorisation
WUU	95	$2^0 * 3^0 * 5^1 * 19^1$
EEK	200	$2^3 * 3^2 * 5^2 * 19^0$
WXISAY	190	$2^1 * 3^1 * 5^1 * 19^1$
NUOCZGM	80	$2^4 * 3^0 * 5^1 * 19^0$
DOEOY	40	$2^3 * 3^0 * 5^1 * 19^0$
GMU	90	$2^1 * 3^2 * 5^1 * 19^0$

Longueur de la clef

Le pgcd des distances est 5.

Pour info : la clef est bien de longueur 5 et il s'agit du texte complet de *L'albatros* de Charles Baudelaire.

Cryptanalyse étape 2 : finalisation de l'attaque

Principe

Si les colonnes sont suffisamment longues on peut espérer que la distribution des fréquences des caractères de l'alphabet dans chaque colonne sera à peu près celle du texte complet et surtout celle de la langue.

☞On découpe le texte chiffré en bloc de ℓ caractères et on applique une cryptanalyse par décalage sur les colonnes !

Cryptanalyse étape 2 : finalisation de l'attaque

Attaque du chiffrement par décalage : Basée sur les fréquences

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

☞Sous-textes plus courts $\Rightarrow$ l'analyse des fréquences fournit moins d'informations

☞Plus difficile de deviner la clef, 26^ℓ tests où ℓ est la longueur de la clef

Cryptanalyse automatique : indice de coïncidence

Définition

L'indice de coïncidence d'un texte est la probabilité de tirer un couple de lettres identiques au hasard.

$$IC = \sum_{i=0}^{25} \frac{C_{n_i}^2}{C_n^2} = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$$

où n_i est le nombre de caractère c_i dans le texte et n est la longueur totale de ce dernier.

William F. Friedman
(1891 - 1969)

Cryptanalyse automatique : indice de coïncidence

☞$IC = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$ distingue l'aléatoire $\Rightarrow$ attaque longueur de la clef

- Lorsque le texte est suffisamment long ($n \to \infty$) l'indice IC est donné par

$$IC \simeq \sum_{i=0}^{25} p_i^2 = 0.074 \text{ pour l'alphabet français}$$

où p_i est la probabilité d'apparition de la lettre numérotée i dans un texte en français.

- Lorsque les lettres sont distribuées aléatoirement, l'indice de coïncidence est faible

$$IC \simeq \sum_{i=0}^{25} \left(\frac{1}{26}\right)^2 = \frac{1}{26} \simeq 0.038$$

Cryptanalyse automatique : indice de coïncidence

☞$IC = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$ distingue l'aléatoire $\Rightarrow$ attaque longueur de la clef

Key Length	Average Index	Individual Indices of Coincidence
4	0.038	0.034, 0.042, 0.039, 0.035
5	0.037	0.038, 0.039, 0.043, 0.027, 0.036
6	0.036	0.038, 0.038, 0.039, 0.038, 0.032, 0.033
7	0.062	0.062, 0.057, 0.065, 0.059, 0.060, 0.064, 0.064
8	0.038	0.037, 0.029, 0.038, 0.030, 0.034, 0.057, 0.040, 0.039
9	0.037	0.032, 0.036, 0.028, 0.030, 0.026, 0.032, 0.045, 0.047, 0.056

- On parcourt les longueurs de clefs possibles, pour chacune d'elle on découpe le texte en colonnes et on calcule la moyenne des IC de ces colonnes.
- Lorsque la longueur de clef n'est pas la bonne, on mélange du texte de plusieurs des vraies colonnes donc des caractères du clair qui ont été chiffré avec des décalages différents, ce qui brouille la distribution des caractères et donne un IC voisin de celui d'un texte aléatoire.
- En revanche si la longueur de la clef est bonne, tous les caractères d'une colonne subissent le même décalage, l'IC de la colonne chiffrée est le même que celui de la colonne claire qui lui correspond et leur moyenne est voisine de l'IC de la langue.
- On repère donc une moyenne d'IC supérieure à un certain seuil pour déduire la longueur de la clef.

Cryptanalyse automatique : IC Mutuelle

☞IC mutuelle distingue l'aléatoire sur deux textes $\Rightarrow$ attaque sur la clef.

Définition

L'indice de coïncidence mutuelle (ICM) entre deux textes t_1 et t_2 est la probabilité de tirer au hasard la même lettre dans t_1 et t_2.

$$ICM = \sum_{i=0}^{25} \frac{m_i n_i}{mn}$$

où m_i (resp. n_i) est le nombre de caractères c_i dans le texte t_1 (resp. t_2) et m (resp. n) la taille de ce dernier.

Cryptanalyse automatique : IC Mutuelle

☞$ICM = \sum_{i=0}^{25} \frac{m_i n_i}{mn}$ pour distinguer l'aléatoire, attaque du décalage

- Propriété identique à l'indice de coïncidence
- Attaque des chiffrements par décalage par analyse successive de décalés

Cryptanalyse automatique : IC Mutuelle

Blocks		Shift Amount												
i	j	0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	.025	.034	.045	.049	.025	.032	.037	.042	.049	.031	.032	.037	.043
1	3	.023	**.067**	.055	.022	.034	.049	.036	.040	.040	.046	.025	.031	.046
1	4	.032	.041	.027	.040	.045	.037	.045	.028	.049	.042	.042	.030	.039
1	5	.043	.021	.031	.052	.027	.049	.037	.050	.033	.033	.035	.044	.030
1	6	.037	.036	.030	.037	.037	.055	.046	.038	.035	.031	.032	.037	.032
1	7	.054	.063	.034	.030	.034	.040	.035	.032	.042	.025	.019	.061	.054
2	3	.041	.029	.036	.041	.045	.038	.060	.031	.020	.045	.056	.029	.030
2	4	.028	.043	.042	.032	.032	.047	.035	.048	.037	.040	.028	.051	.037
2	5	.047	.037	.032	.044	.059	.029	.017	.044	.060	.034	.037	.046	.039
2	6	.033	.035	.052	.040	.032	.031	.031	.029	.055	.052	.043	.028	.023
2	7	.038	.037	.035	.046	.046	.054	.037	.018	.029	.052	.041	.026	.037
3	4	.029	.039	.033	.048	.044	.043	.030	.051	.033	.034	.034	.040	.038
3	5	.021	.041	.041	.037	.051	.035	.036	.038	.025	.043	.034	.039	.036
3	6	.037	.034	.042	.034	.051	.029	.027	.041	.034	.040	.037	.046	.036
3	7	.046	.023	.028	.040	.031	.040	.045	.039	.020	.030	**.069**	.042	.037
4	5	.041	.033	.041	.038	.036	.031	.056	.032	.026	.034	.049	.029	.054
4	6	.035	.037	.032	.039	.041	.033	.032	.039	.042	.031	.049	.039	.058
4	7	.031	.032	.046	.038	.039	.042	.033	.056	.046	.027	.027	.036	.036
5	6	.048	.036	.026	.031	.033	.039	.037	.027	.037	.045	.032	.040	.041
5	7	.030	.051	.043	.031	.034	.041	.048	.032	.053	.037	.024	.029	.045
6	7	.032	.033	.030	.038	.032	.035	.047	.050	.049	.033	.057	.050	.021

Cryptanalyse automatique : IC Mutuelle

Blocks		Shift Amount												
i	j	13	14	15	16	17	18	19	20	21	22	23	24	25
1	2	.034	.052	.037	.030	.037	.054	.021	.018	.052	.052	.043	.042	.046
1	3	.031	.037	.038	.050	.039	.040	.026	.037	.044	.043	.023	.045	.032
1	4	.039	.040	.032	.041	.028	.019	**.071**	.038	.040	.034	.045	.026	.052
1	5	.042	.032	.038	.037	.032	.045	.045	.033	.041	.043	.035	.028	.063
1	6	.040	.030	.028	**.071**	.051	.033	.036	.047	.029	.037	.046	.041	.027
1	7	.040	.032	.049	.037	.035	.035	.039	.023	.043	.035	.041	.042	.027
2	3	.054	.040	.028	.031	.039	.033	.052	.046	.037	.026	.028	.036	.048
2	4	.047	.034	.027	.038	.047	.042	.026	.038	.029	.046	.040	.061	.025
2	5	.034	.026	.035	.038	.048	.035	.033	.032	.040	.041	.045	.033	.036
2	6	.033	.034	.036	.036	.048	.040	.041	.049	.058	.028	.021	.043	.049
2	7	.042	.037	.041	.059	.031	.027	.043	.046	.028	.021	.044	.048	.040
3	4	.037	.045	.033	.028	.029	**.073**	.026	.040	.040	.026	.043	.042	.043
3	5	.035	.029	.036	.044	.055	.034	.033	.046	.041	.024	.041	**.067**	.037
3	6	.023	.043	**.074**	.047	.033	.043	.030	.026	.042	.045	.032	.035	.040
3	7	.035	.035	.035	.028	.048	.033	.035	.041	.038	.052	.038	.029	.062
4	5	.032	.041	.036	.032	.046	.035	.039	.042	.038	.034	.043	.036	.048
4	6	.034	.034	.036	.029	.043	.037	.039	.036	.039	.033	**.066**	.037	.028
4	7	.043	.032	.039	.034	.029	**.071**	.037	.039	.030	.044	.037	.030	.041
5	6	.052	.035	.019	.036	.063	.045	.030	.039	.049	.029	.036	.052	.041
5	7	.040	.031	.034	.052	.026	.034	.051	.044	.041	.039	.034	.046	.029
6	7	.029	.035	.039	.032	.028	.039	.026	.036	**.069**	.052	.035	.034	.038

Cryptanalyse automatique : IC Mutuelle

Blocks		Shift Amount												
i	j	13	14	15	16	17	18	19	20	21	22	23	24	25
1	2	.034	.052	.037	.030	.037	.054	.021	.018	.052	.052	.043	.042	.046
1	3	.031	.037	.038	.050	.039	.040	.026	.037	.044	.043	.023	.045	.032
1	4	.039	.040	.032	.041	.028	.019	**.071**	.038	.040	.034	.045	.026	.052
1	5	.042	.032	.038	.037	.032	.045	.045	.033	.041	.043	.035	.028	.063
1	6	.040	.030	.028	**.071**	.051	.033	.036	.047	.029	.037	.046	.041	.027
1	7	.040	.032	.049	.037	.035	.035	.039	.023	.043	.035	.041	.042	.027
2	3	.054	.040	.028	.031	.039	.033	.052	.046	.037	.026	.028	.036	.048
2	4	.047	.034	.027	.038	.047	.042	.026	.038	.029	.046	.040	.061	.025
2	5	.034	.026	.035	.038	.048	.035	.033	.032	.040	.041	.045	.033	.036
2	6	.033	.034	.036	.036	.048	.040	.041	.049	.058	.028	.021	.043	.049
2	7	.042	.037	.041	.059	.031	.027	.043	.046	.028	.021	.044	.048	.040
3	4	.037	.045	.033	.028	.029	**.073**	.026	.040	.040	.026	.043	.042	.043
3	5	.035	.029	.036	.044	.055	.034	.033	.046	.041	.024	.041	**.067**	.037
3	6	.023	.043	**.074**	.047	.033	.043	.030	.026	.042	.045	.032	.035	.040
3	7	.035	.035	.035	.028	.048	.033	.035	.041	.038	.052	.038	.029	.062
4	5	.032	.041	.036	.032	.046	.035	.039	.042	.038	.034	.043	.036	.048
4	6	.034	.034	.036	.029	.043	.037	.039	.036	.039	.033	**.066**	.037	.028
4	7	.043	.032	.039	.034	.029	**.071**	.037	.039	.030	.044	.037	.030	.041
5	6	.052	.035	.019	.036	.063	.045	.030	.039	.049	.029	.036	.052	.041
5	7	.040	.031	.034	.052	.026	.034	.051	.044	.041	.039	.034	.046	.029
6	7	.029	.035	.039	.032	.028	.039	.026	.036	**.069**	.052	.035	.034	.038

☞On termine en résolvant un système linéaire.

$$\begin{cases} \delta_3 = \delta_4 + 18 \\ \delta_3 = \delta_6 + 15 \\ \delta_4 = \delta_7 + 18 \\ \vdots \end{cases}$$

La cryptanalyse de Vigenère (avec les IC) : trouver la longueur de clef

Essayer des longueurs $\ell = 4..20$ et pour chaque longueur ℓ possible :

- Découper le texte en colonnes $C_{0 \leqslant i < \ell}$,
- Calculer

$$m = \underset{0 \leqslant i < \ell}{\text{moyenne}}(\text{IC}(C_i))$$

- Si $m > 0.06$ alors ℓ est la longueur de la clef, sinon on passe à la valeur de ℓ suivante

La cryptanalyse de Vigenère (avec les IC) : décalage de chaque colonne I

Première solution :

Analyse de fréquences directe

Par analyse de fréquences, trouver la position $f(C_i)$ de la lettre la plus fréquente de la colonne, compte tenu de la position $f(langue)$ de la lettre la plus fréquente de la langue du message, déduire le décalage $d_i = f(C_i) - f(langue)$ de la colonne C_i.

La cryptanalyse de Vigenère (avec les IC) : décalage de chaque colonne II

Seconde solution :

Utilisation des ICM puis analyse de fréquences

- Calculer les ICM des C_i et les stocker dans un tableau de différences de décalage $d_j - d_i$ entre les colonnes i et j
- Résoudre le système linéaire correspondant pour exprimer le décalage $d_{j \neq i}$ de toutes les colonnes sauf la colonne i en fonction de d_i
- Procéder par analyse de fréquences
  - soit sur la colonne i et de d_i déduire les $d_{j \neq i}$,
  - soit sur tout le texte après avoir aligné les colonnes $j \neq i$ sur le décalage d_i

Puis déduire la clef et le déchiffré du texte.

Corrélation de Pearson : pourquoi s'y intéresse-t-on ?

- Lorsqu'on pense avoir trouvé une substitution s entre deux alphabets A et B pour déchiffrer un chiffré C
- La distribution des fréquences des caractères
  - de c dans un texte de référence de la langue
  - de $s(c)$ dans le chiffré C

  doivent être à peu près les mêmes
- Les points du nuage de points $\mathrm{freq}(s(c), C)$ en fonction de $\mathrm{freq}(c, \mathit{langue})$ pour $c \in A$, doivent être à peu près alignés.

Le coefficient de corrélation de Pearson mesure la qualité de cet alignement et peut donc servir de distingueur pour déterminer si la substitution s effectuée sur le chiffré C est la bonne.

Coefficient de corrélation de Pearson de deux variables aléatoires X et Y

$$\rho_{X,Y} = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2}\sqrt{\sum_i (Y_i - \bar{Y})^2}}$$

avec $\bar{X}$ la moyenne de la variable aléatoire X.

Une corrélation parfaite des variables X et Y produit $\rho_{X,Y} = 1$.

Plus $\rho_{X,Y}$ est proche de 1 et plus on peut considérer que la fréquence des caractères $s(c)$ dans le chiffré C et celle de c dans la langue sont corrélées.

☞ Évidemment ici X représente la fréquence des caractères dans le texte chiffré *en cours de transformation* et Y la fréquence des caractères dans la langue, donc dans un texte de référence.

Cryptanalyse de Vigenère (corrélation de Pearson)

Essayer des longueurs $\ell = 4..20$ et pour chaque longueur ℓ possible :

- Découper le texte en colonnes $C_{0 \leqslant i < \ell}$,
- Pour chaque colonne C_k, calculer la corrélation maximale m_k entre les fréquences de référence de la langue et C_k décalé de i pour $i = 0..|A| - 1$, cette corrélation maximale est atteinte pour un décalage d_k
- On calcule $m = \underset{k}{\text{moyenne}}(m_k)$,
- Si $m > 0.6$, alors ℓ est la longueur de la clef, $(d_o, \ldots, d_{\ell-1})$ est la clef et on déduit le message clair.

On obtient de très bons résultats très rapidement et pour des textes bien plus courts que pour la cryptanalyse par IC.

Conclusion 2

☞Le chiffrement mono-alphabétique est très facile à attaquer !
☞Le chiffrement de Vigenère ne semble pas beaucoup plus sûr !

Comment le sécuriser ?

- Substitution homophonique (1500-1750)
- Substitution poly-alphabétique (Vigenère)
- Substitution polygrammique (Playfair) et combinaisons substitution+transposition (ADFGVX)

Plan

Substitution polygrammique

Polygrammique : chiffrer non plus des caractères individuels mais des blocs de caractères (des polygrammes).

Ce qu'on gagne

- Pour des bigrammes, on travaille sur un alphabet à $26^2 = 676$ caractères donc le nombre de substitutions possibles augmente énormément ($26! \sim 2^{88} \longrightarrow 26^2! \sim 2^{5386}$)
- Précédemment on effectuait manuellement la cryptanalyse en complétant l'analyse de la fréquence des 26 caractères individuels par celle de la fréquence des bigrammes
- Mais ici l'analyse de fréquences sur ces 26 caractères individuels ne sert à rien et partir de l'analyse sur les 676 bigrammes est très ardu : on ne peut pas s'intéresser à autant de valeurs aussi peu différenciées.

Pour une attaque *brute-force*, si on suppose qu'on fait un essai par seconde, il faut 2^{80} secondes et c'est très supérieur à l'âge de l'univers...

Un exemple de substitution polygrammique : le chiffrement de Playfair

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

Wheatstone (1802 - 1875)

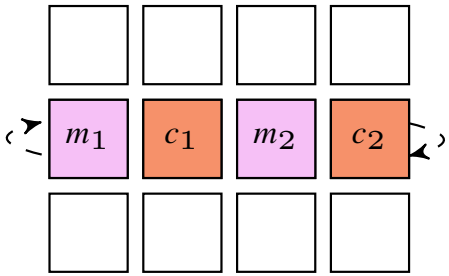
Inventé par Wheatstone en 1854 et popularisé par Lord Playfair, utilisé pendant la Première Guerre Mondiale par les anglais et la Seconde Guerre Mondiale par les australiens, bien que ce chiffrement ait été analysé en 1914 par Mauborgne.

La clef secrète se présente sous la forme d'un carré 5x5 qui contient toutes les lettres de l'alphabet, en en confondant 2, telles que I et J ou V et W.

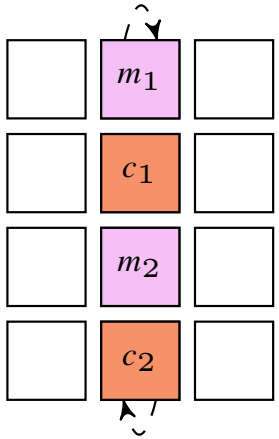
Principe du chiffrement de Playfair

On analyse le texte par groupe de 2 caractères et on applique selon ces caractères l'une des 3 règles suivantes :

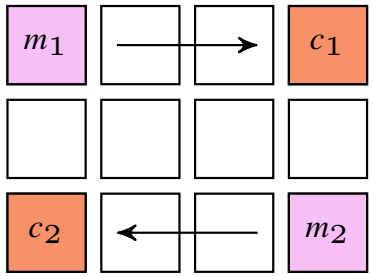
2 caractères distincts m_1 m_2



Règle ligne
wrap si nécessaire



Règle colonne
wrap si nécessaire



Règle rectangle

S'il s'agit d'une lettre double $m\,m$: on glisse une lettre neutre entre les deux m et on applique à nouveau les règles à partir du premier m (oui tout l'appariement des caractères se trouve décalé et peut produire de nouveaux doubles plus loin).

S'il ne reste qu'un seul caractère m à chiffrer on ajoute derrière une lettre neutre et on chiffre avec les règles précédentes.

Lettre neutre : une lettre peu fréquente, donc facilement identifiable comme neutre dans le clair après déchiffrement, telle que X si $m \neq$X évidemment.

Exemple de chiffrement de Playfair

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

ATTAQUE PARIS DEMAIN AVEC PETIT TRAIN BLEU

AT TA QU EP AR IS DE MA IN AV EC PE TI **TT** RA IN BL EU

AT TA QU EP AR IS DE MA IN AV EC PE TI TX TR AI NB LE **U**X

RK KR ZD RG EA RI YI KM EF MC AN GR WR XL XP ER BF NR DV

RKKRZ DRGEA RIYIK MEFMC ANGRW RXLXP ERBFN RDV

On verra en TD un autre chiffrement polygrammique : le chiffrement de Hill.

Surchiffrement

Principe

Chiffrer avec un autre système de chiffrement un message déjà chiffré

Le chiffre du Che : échange entre le Che en Bolivie et Fidel Castro à Cuba

Substitution mono-alphabétique + substitution poly-alphabétique :

- un premier chiffrement associe au hasard à chaque lettre de l'alphabet un nombre de 1 ou 2 chiffres, un nombre de 1 chiffre n'étant pas le chiffre des dizaines d'un nombre de 2 chiffres pour pouvoir déchiffrer sans ambiguïté,
- le second chiffrement consiste à ajouter un multi-décalage (Vigenère) représenté par un nombre (cette clef change à chaque message).

Surchiffrement

Dans les œuvres de fiction

Le chiffre de l'épisode 2.15 de Mission impossible

Mot-clef `PHOTOGRAPHER` date du jour `08/23`
Utilisation de la clef :

```
P H O T O G R A P H E R B C  D  F  I  J  K  L  M  N  Q  S  U  V  W  X  Y  Z
1 2 3 4   5 6 7    8   9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
```

annuaire 1er numéro de téléphone local de la page `823` : `509-2138`, etc.

```
  T    E    X    T    E
  4    8    24   4    8
+ 5    0    9    2    1
--------------------------
  9    8    7    6    9 mod 26
  B    E    A    R    B
```

Surchiffrement

ADFGVX

Substitution mono-alphabétique + transposition :

- substitution avec pour alphabet d'arrivée le carré de Polybe dont les lignes et les colonnes sont indexées par les lettres `A, D, F, G, V` et `X`,
- puis transposition.

Inventé par Nebel (mars 1918) . . . et cryptanalysé par Painvin (juin 1918).

Carré de Polybe

IIe siècle av. JC

Les caractères (25 lettres en confondant deux caractères tels que I et J, 36 avec lettres et chiffres, etc.) sont placés dans un carré avec des chiffres ou des lettres pour les coordonnées des lignes et des colonnnes. Un caractère est remplacé par son abscisse et son ordonnée dans le tableau.

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

Original

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

Avec chiffres
et mot de passe
`MOTDEPASSE`

	1	2	3	4	5	6	7
1	we	a	ya	ra	yo	chi	i
2	hi	sa	ma	mu	ta	ri	ro
3	mo	ki	ke	u	re	nu	ha
4	se	yu	fu	wi	so	ru	ni
5	su	me	ko	no	tsu	wo	ho
6	n	mi	e	o	ne	wa	he
7	shi	te	ku	na	ka	to	

Variante de Uesugi (XVIe Japon)

Cette variante utilise l'alphabet japonais traditionnel de 48 lettres (tiré du poème iroha-uta).
Si les caractères ne sont pas ordonnés alphabétiquement le carré est la clef du chiffrement sinon c'est un simple code.
Chiffrement de `BONJOUR` avec le carré avec mot de passe `MOTDEPASSE` :

```
B O N J O U R
2312363312 4342
```

Transposition

- Dès l'Antiquité avec la scytale
- Le plus souvent en combinaison avec d'autres systèmes de chiffrement.
- On écrit le message sur des lignes de longueur fixe, on permute les colonnes obtenues puis on relit le texte colonne par colonne.
- La clef est évidemment la permutation de colonnes.
- Elle peut être décrite
  - par un nombre composé des numéros des colonnes relus dans l'ordre où il faut les lire pour chiffrer (cf. TD)
  - ou par un mot-clef (cf. exemple suivant) : à chaque caractère du mot-clef est associée sa position dans ce mot puis on réordonne alphabétiquement les lettres du mot, ce qui induit une permutation des colonnes.

Exemple de chiffrement par transposition

Pour chiffrer la phrase :

`Mais dis donc, on n'est quand même pas venus pour beurrer les sandwichs!`

avec le mot-clef `MONTAUBAN`, on numérote les colonnes

```
MONTAUBAN
123456789
```

puis on les réordonne

```
AABMNNOTU
587139246
```

on écrit le message en majuscules sans accents ni ponctuation

`MAISDISDONCONNESTQUANDMEMEPASVENUSPOURBEURRERLESSANDWICHS`

sur 9 colonnes (la longueur de la permutation), en complètant la dernière ligne par des `X` (du *padding*)

Exemple de chiffrement par transposition (suite)

```
123456789
MAISDISDO
NCONNESTQ
UANDMEMEP
ASVENUSPO
URBEURRER
LESSANDWI
CHSXXXXXX
```

puis on relit le texte en commençant par la colonne 5 puis la colonne 8, etc. (587139246)

DNMNUAXDTEPEWXSSMSRDXMNUAULCIONVBSSOQPORIXACASREHSNDEESXIEEURNX

Cryptanalyse de la transposition :

- retrouver la longueur de la permutation ℓ (la longueur n du chiffré est égale au produit du nombre de colonnes ℓ par leur hauteur h, donc ℓ divise n et $h = n/\ell$ divise la distance entre deux caractères de padding),
- s'aider des fréquences des bigrammes pour replacer les colonnes l'une par rapport à l'autre.

Cryptanalyse du chiffrement polygrammique et du surchiffrement par transposition

Méthodes de cryptanalyse

- Utilisation des statistiques d'apparition des bigrammes, tétragrammes, etc. (attaque manuelle plus compliquée que pour le chiffrement poly-alphabétique)
- Utilisation de caractérisations propres au cryptosystème (par exemple pour Playfair : RE $\rightarrow$ GH alors ER $\rightarrow$ HG)
- Cryptosystème plus long à casser mais on y arrive, surtout si on a un couple clair-chiffré (Painvin)

☞Idée reprise pour Enigma en mélangeant plus radicalement
☞Attaque mécanisée de Turing

Résistance d'un chiffrement

Cryptosystème réputé cryptanalysé mais toujours utilisé ?!

Playfair, double Playfair, ou double transposition, bien que cryptanalysés pendant la première guerre mondiale, ont été utilisés pendant la seconde guerre mondiale pour des communications importantes mais non-critiques sur les champs de bataille.

Résistance : temps de cryptanalyse / durée de vie de l'information

Principe

On peut utiliser un cryptosystème pour chiffrer une information tant que le temps pour effectuer la cryptanalyse du système dépasse la durée de la pertinence de l'information.
Évidemment on ne sait pas évaluer a priori quand la cryptanalyse devient assez rapide pour fournir le clair avant la péremption de l'information donc on essaie de prévoir large pour ne pas être pris au dépourvu.

Notion cruciale pour comprendre l'intérêt de la crypto post-quantique alors qu'on ne sait pas si et quand les ordinateurs quantiques existeront et seront largement disponibles.

☞Existe-t-il un chiffrement inattaquable ? Est-ce une panacée ?

Plan

Shannon

Claude Shannon (1916 - 2001) a publié deux articles de recherche en 1948 et 1949 donnant les fondations de la théorie de l'information et, plus généralement, de la cryptologie moderne. Il donne les premières preuves de sécurité d'un cryptosystème en se basant sur des principes de probabilité et de statistique.

Notions importantes

- Théorie de l'information
- Entropie d'un langage
- Chiffrement parfait

Chiffrement parfait

Intuition

Un cryptosystème sera dit un chiffrement parfait lorsque la donnée d'un message chiffré ne révèle aucune fuite d'information sur la clef ou le message clair correspondant et aucune information non plus sur les textes chiffrés futurs.

Caractérisation

Supposons qu'un cryptosystème vérifie

$$\#\mathcal{K} = \#\mathcal{P} = \#\mathcal{C}$$

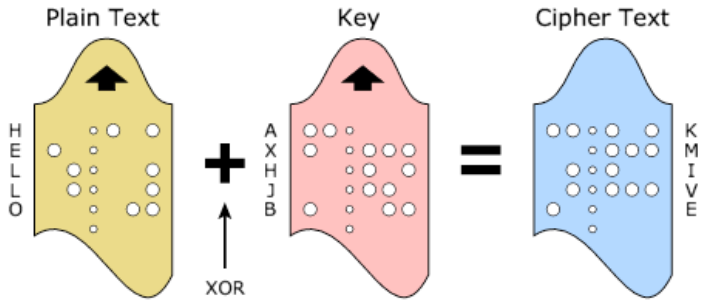
alors il sera un chiffrement parfait ssi on a

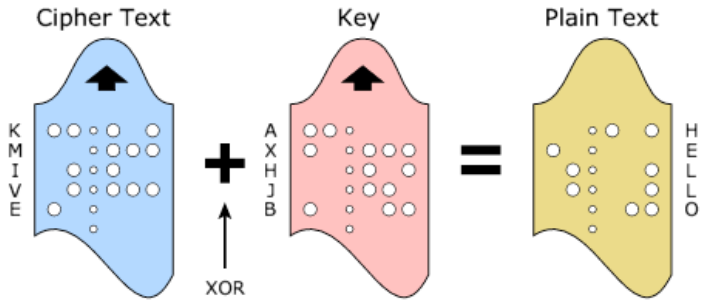
- Toutes les clefs sont utilisées avec la même probabilité
- Pour tout couple $(m, c) \in \mathcal{P} \times \mathcal{C}$ il existe une unique clef k telle que $e_k(m) = c$.

Exemple : Vernam's One Time Pad

- Gilbert Vernam proposa le *One Time Pad* en 1917. Amélioration Joseph Mauborgne (USA). Idée initiale banquier américain Frank Miller 1882.
- Principe : un Vigenère avec une clef aléatoire aussi longue que le message à chiffrer, utilisée une seule fois.
- En pratique on utilise un outil extrêmement simple et rapide, le XOR ! Les messages clairs et chiffrés ainsi que les clefs seront des suites de bits de même longueur (ce qu'on peut voir comme du Vigenère bit à bit).

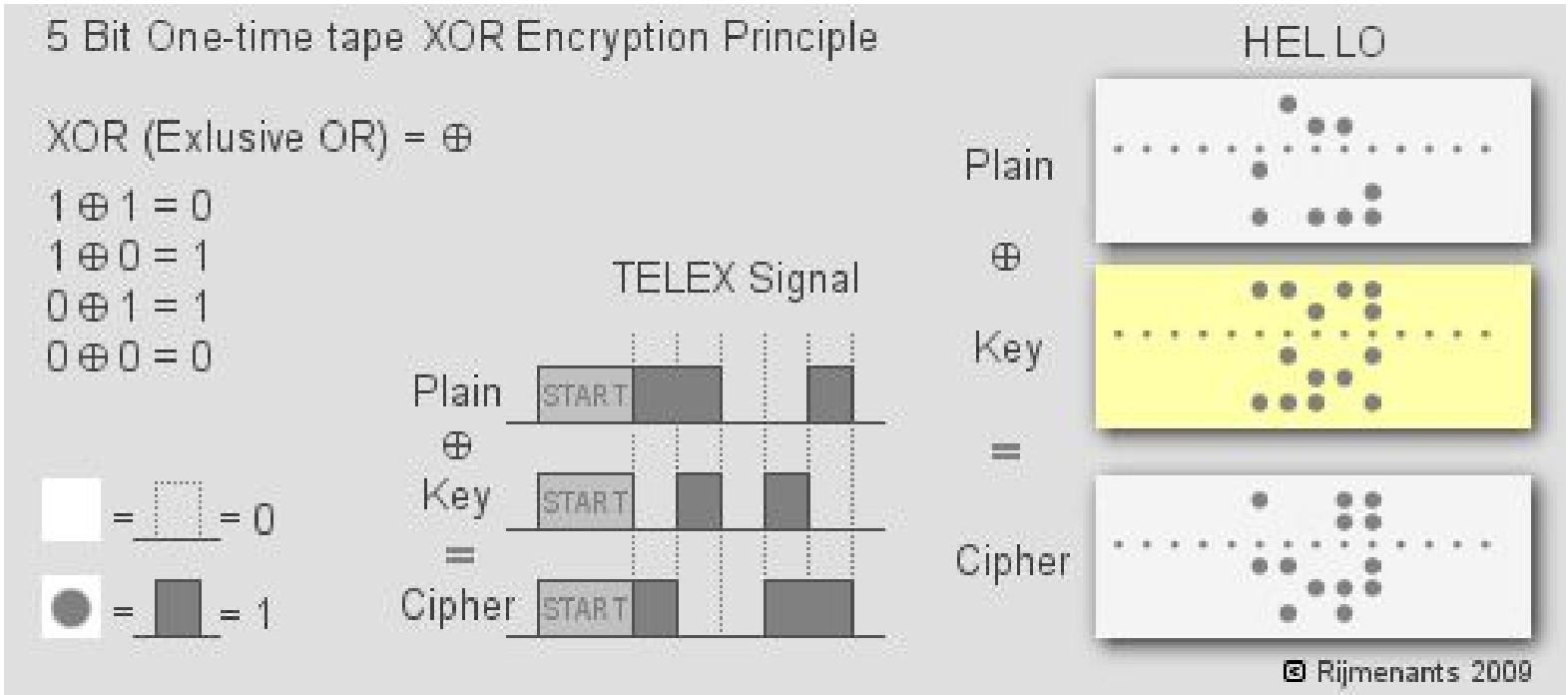
$$C[i] = M[i] \oplus K[i] \text{ et } M[i] = C[i] \oplus K[i]$$





Exemple : Vernam's One Time Pad

- C'est le seul cryptosystème à chiffrement parfait !
- Très peu pratique !
- Utilisé dans la cryptographie Top Secrète (téléphone rouge, valise diplomatique, militaire (Atomique)).
- Le principe est utilisé pour faire des chiffrements symétriques dépendant de générateur aléatoire.



Exemple : Vernam's One Time Pad

- La clef doit être aussi longue que le message
- Elle doit être aléatoire
- Elle doit être utilisée une unique fois
- Projet VENONA des USA pour écouter les discussions Russes utilisant un Two-Time Pad $\Rightarrow$ faiblesse !

La panacée ?

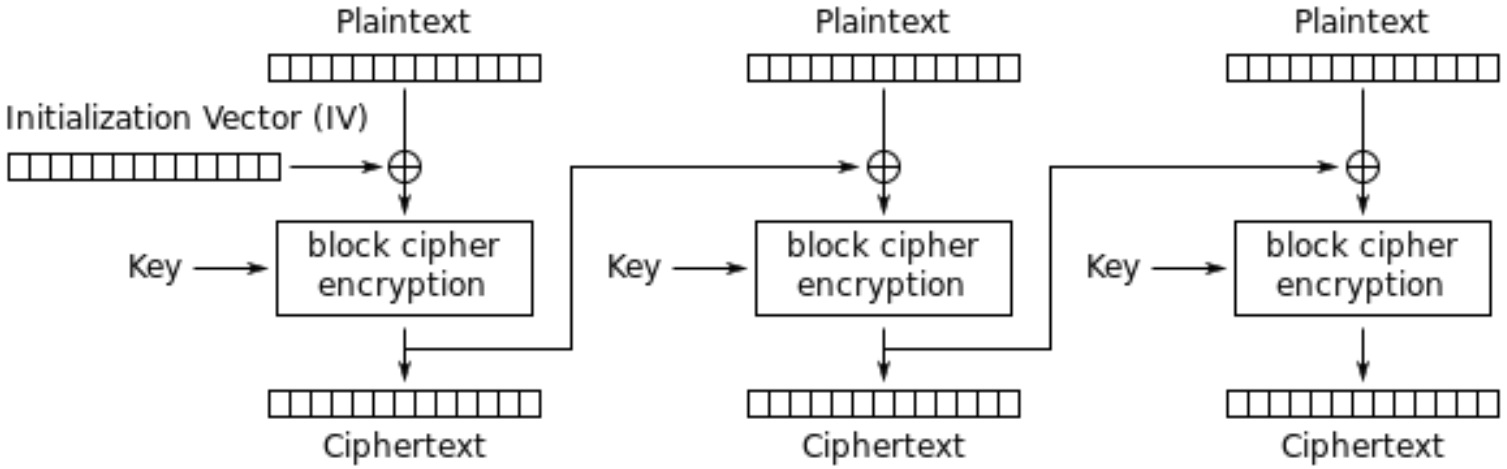
Évidemment non si ça ne s'est pas imposé en 1 siècle !

Impraticable

- Usage unique d'une clef : casse-tête de la transmission sécurisée de la nouvelle clef à chaque transmission de message
- Usage unique d'une clef : il faut être sûr même après des années de ne pas réutiliser une clef
- Difficile de produire des clefs vraiment aléatoires

Conclusion finale !

- ☞Chiffrement parfait existe mais impraticable
- ☞Partir de ce principe pour définir des *Block Ciphers* standardisés
- ☞Idée clef : randomiser la clef de chiffrement suivant des tours (complexifier la tâche des attaquants)



Cipher Block Chaining (CBC) mode encryption

- ☞Voir la suite en crypto/sécu en M1

Cryptanalyse des chiffrements mono-alphabétique et Vigenère

8. Généralités

9. Cryptanalyse du chiffrement mono-alphabétique

10. Chiffrement de Vigenère et sa cryptanalyse
    - Définition du chiffrement de Vigenère
    - Cryptanalyse du Chiffrement de Vigenère
        - Longueur de la clef et principe de cryptanalyse
        - Indices de coïncidence
        - Corrélation de Pearson

11. Substitution polygrammique et surchiffrement

12. Chiffrement Parfait

Troisième partie III

Complexité pour mesurer la sécurité

Plan

Recherche exhaustive et ordres de grandeur

- Chiffrement de Vigenère avec une clef de longueur ℓ sur un alphabet de t caractères.
- Test de tous les t décalages possibles pour chacune des ℓ colonnes.
- Pour une recherche exhaustive il faut donc effectuer $t^\ell = 2^{\ell \log_2(t)}$ tests.
- Dans le cadre de l'alphabet usuel, $t = 26 > 16$ donc $\log_2(t) > 4$.
- Donc si ℓ est égale à 20, nous avons au moins 2^{80} tests à réaliser.

☞2^{80} est une constante cryptographique. Au delà de cette barrière, l'humanité ne peut pas réaliser cette quantité de calculs.

Âge de l'univers (au minimum) : 2^{58} secondes !!!

☞Il y a généralement plus intelligent que la recherche exhaustive !

Difficulté pour la cryptanalyse

- On a vu comment la théorie des probabilités permet de définir un cryptosystème incassable.
- On cherche d'autres moyens plus généraux pour estimer la difficulté de cryptanalyse.
- On va faire reposer la difficulté sur un problème mathématique (Shannon a montré qu'une cryptanalyse équivaut toujours à la résolution d'un système algébrique) :
  - La difficulté de cryptanalyse doit reposer uniquement sur l'ignorance de la clef gardée secrète (sa taille doit donc être conséquente) alors que le cryptosystème est connu de tous (principe de Kerckhoffs)
  - Le chiffrement et le déchiffrement doivent être faciles
  - La cryptanalyse doit être difficile

Mesure de la difficulté d'un problème

Difficulté d'un problème

La difficulté d'un problème est la complexité du meilleur algorithme connu pour le résoudre.

☞Contextuel : on se base sur les connaissances algorithmiques actuelles.

☞L'efficacité de l'algorithme (et de ses implantations) dépend du nombre d'opérations nécessaires pour résoudre le problème défini par l'entrée de taille n.

Complexité d'un algorithme

Définition

Étant donné un algorithme résolvant un problème défini à partir d'une entrée de taille n. La complexité de l'algorithme représente le nombre d'opérations nécessaires pour résoudre le problème en fonction de n.

☞Généralement les complexités sont asymptotiques ($n \to \infty$).

Notation $\mathcal{O}$ pour f et g positives

On note $f = \mathcal{O}(g)$ lorsqu'il existe une constante $C > 0$ et un entier N tels que

$$\forall n > N, \quad f(n) \leqslant Cg(n)$$

Échelle de complexité

Classes importantes de complexité

- Constante en t : $C = \mathcal{O}(1)$ pour tout constante C.
- Complexité polynomiale en t : $P(t)$ opérations où P est un polynôme de degré $k \geqslant 1$; on a $P(t) = \mathcal{O}(t^k) = 2^{\mathcal{O}(\log(t))}$.
- Complexité exponentielle en t : $2^{\mathcal{O}(t^k)}$ opérations pour une constante $k \geqslant 1$ fixée.
- Entre les deux : complexité sous-exponentielle $L[\alpha, c] = \exp(ct^\alpha \log(t)^{1-\alpha})$ n'est ni polynomiale ni exponentielle pour $\alpha \in]0, 1[$ et c une constante fixée.

Problème facile ? Problème difficile ?

- Un problème est facile s'il se résout au plus en temps polynomial en la taille de son entrée.
- Un problème est difficile si l'on ne sait pas mieux faire qu'en temps exponentiel !

Classes de complexité $\mathcal{P}$ et $\mathcal{NP}$

Définitions

- Un problème appartient à $\mathcal{P}$ s'il existe un algorithme polynomial (en la taille n décrivant le problème) pour résoudre ce problème.
- Un problème appartient à $\mathcal{NP}$ s'il existe un algorithme polynomial (en la taille n décrivant le problème) pour vérifier une solution de ce problème.

- Il a été montré que $\mathcal{P} \subset \mathcal{NP}$
- 1 000 000$ de récompense à qui montrera que $\mathcal{P} = \mathcal{NP}$ ou $\mathcal{P} \neq \mathcal{NP}$

☞Attention ces définitions sont informelles. Normalement, elles se basent sur des problèmes décisionnels.

Difficulté de problèmes en cryptologie

☞En crypto on va s'intéresser à des problèmes du type $\mathcal{NP}$ mais rien ne nous dit qu'un jour un attaquant trouvera un moyen de les faire passer dans $\mathcal{P}$.

Exemples : factorisation d'entier, log discret, résolution de systèmes polynomiaux, etc.

☞Il y a des problèmes encore plus difficiles du point de vue de la complexité sur lesquels on peut aussi s'appuyer.

Par exemple les problèmes $\mathcal{NP}$-complets : tous les algorithmes connus pour résoudre un tel problème ont un temps d'exécution exponentiel en la taille des données d'entrée dans le pire des cas, et sont donc inexploitables en pratique même pour des instances de taille modérée.

Complexité et crypto

☞Les problèmes sur lesquels on se base relèvent en général du calcul scientifique ($\subset$ mathématiques et informatique).

Pour estimer leur complexité il faut commencer par déterminer le coût des opérations de base.

Complexité et crypto

☞Problèmes difficiles à résoudre : si ce problème repose sur de l'arithmétique des entiers, il vaut mieux pouvoir maîtriser la complexité de ces opérations !

Ce coût est calculé en fonction de la taille des entrées.

Taille des entrées

Pour un entier a sa taille est $\ell(a) = \log_2(a) = \mathcal{O}(\log(a))$

☞Un algorithme prenant a en entrée est de complexité polynomiale s'il nécessite $\mathcal{O}(\log(a)^k)$ opérations sur des mots machine.

Problème

Comment calculer avec de grands entiers sur un ordinateur qui ne possède qu'un espace fini (registres) pour stocker les données dans le processeur ?

Plan

L'arithmétique entière sur machine : opérations de base

- Opposé, addition, soustraction, multiplication, quotient et reste (signé), valeur absolue
- Opérations bit à bit : NON, ET, OU, OU EXCLUSIF, décalages gauche et droite
- Comparaisons ($=$, $\neq$,$<$,$>$,$\leqslant$,$\geqslant$)
- Et quelques autres en assembleur.

Et en C, Python... ou d'autres langages informatiques

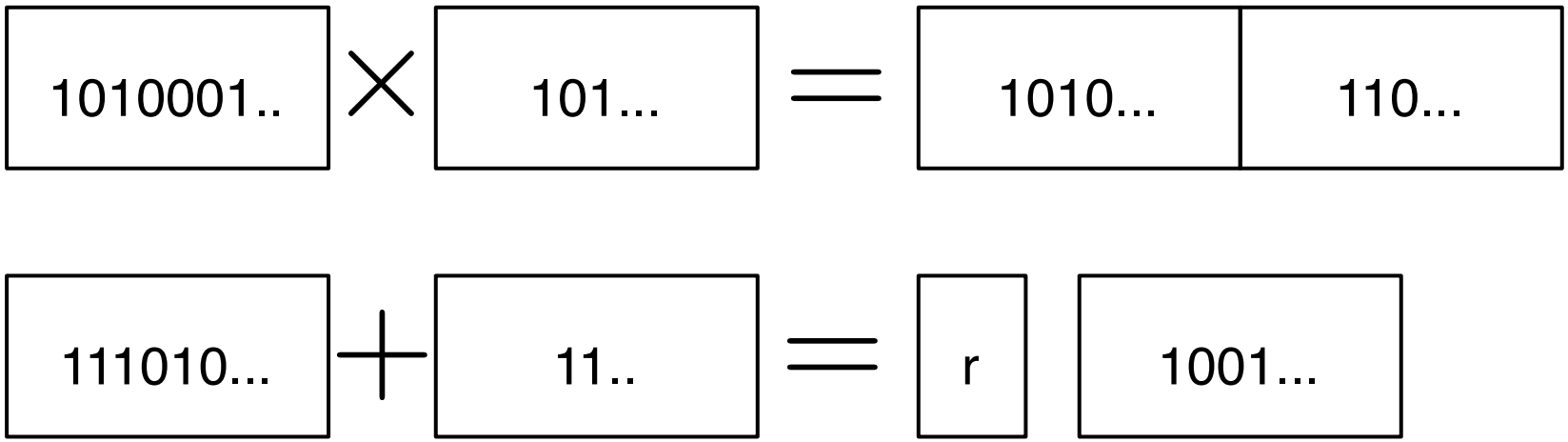
- Entre `INT_MIN` et `INT_MAX` (entiers signés)
- `+`, `-`, `*`, `/`, `//`, `%`, `abs`,
- `~`, `&`, `|`, `^`, `<<`, `>>`,
- `==`, `!=`, `<`, `>`, `<=`, `>=`.

L'arithmétique entière sur machine : précision limitée

α est la longueur d'un mot machine. C'est un multiple de 8 bits (nombre entier d'octets) :

- 32 ou 64 sur les ordinateurs les plus courants,
- mais 8 ou 16 bits sur cartes à puce ou ad hoc, etc.

Précision limitée donc opérations avec débordement (*overflow*)



Représentation des entiers en multiprécision

Décomposition d'un nombre entier naturel a en base B :

$$a = \sum_{i=0}^{\ell(a)-1} a_i B^i$$

où a_i sont les chiffres en base B et vérifient $0 \leqslant a_i \leqslant B-1$.

☞La base B est une puissance de 2 (correspond à la taille d'un mot machine ou d'un demi-mot ce qu'on appelera un *limb*)

☞On prend généralement $B = 2^\alpha$ ou $2^{\alpha/2}$

Représentation des entiers naturels : stockage des chiffres

- Tableau contenant ses chiffres dans la base B
- Chiffres de poids faible en début (a_0 est rangé dans la case d'indice 0, etc.) : c'est plus pratique pour l'addition et la multiplication, moins pour la division mais opération moins fréquente.

- La taille renseigne le nombre de chiffres et son signe est le signe de l'entier représenté.
- C'est la représentation utilisée par GMP, la bibliothèque de référence actuelle pour l'arithmétique multi-précision.

Opérations multi-précision : *schoolbook*

☞Les algorithmes que nous présentons maintenant ne sont pas les plus efficaces, surtout pour la multiplication :

- Ensemble de fonctions de multiplication où l'on choisit la mieux adaptée à la taille des données sur lesquelles on travaille (naïve, Karatsuba, Toom-Cook, FFT, Schönhage-Strassen, van der Hoeven) (cf. bibliothèques)
- Ici, on se contente de voir la multiplication naïve, telle qu'on la fait à la main et qui est rentable pour les nombres de petite taille et l'on verra l'intuition de la méthode de Karatsuba.
- La *transformée de Fourier rapide*, FFT, est encore plus rapide mais ne sera pas vue ici.
- Pour en savoir plus : `http://cr.yp.to/papers/m3.pdf`

Plan

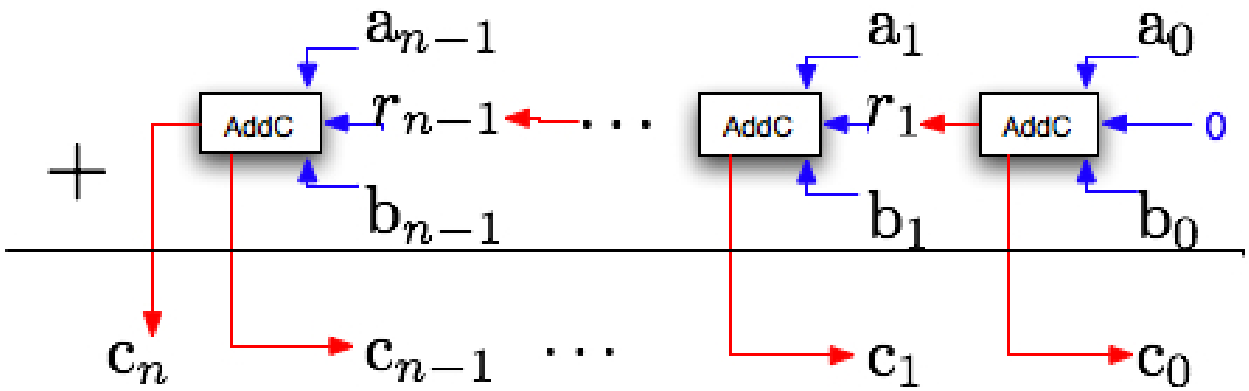
Addition : algorithme de base

À la main, additionner deux chiffres (en base 2 ou 10) et une retenue est immédiat.

☞C'est une opération de base qui servira d'unité pour compter la complexité.

☞On la note `AddC`.

Pour additionner deux entiers positifs arbitrairement grands on a le schéma suivant



Pour additionner deux nombres entiers $a \geqslant b$ de $\ell(a) = \ell(b) = n$ chiffres, le nombre d'opérations de base est n. On dit alors que la complexité est de l'ordre de $\mathcal{O}(n) = \mathcal{O}(\log(a))$ ou encore linéaire en la taille de l'entrée.

Multiplication : algorithme de base

$$a = \sum_{i=0}^{\ell(a)-1} a_i B^i \quad b = \sum_{i=0}^{\ell(b)-1} b_i B^i$$

Le produit peut se faire comme on l'apprend à l'école

$$c = \sum_{i=0}^{\ell(b)-1} a \times b_i B^i$$

- $\ell(b)$ multiplications de a par un chiffre b_i
- Une multiplication de a par un chiffre coûte $\ell(a)$ multiplications d'un chiffre par un chiffre
- Les chiffres sont des mots-machine, les additions sont moins nombreuses que les multiplications.

Complexité pour $a \geqslant b$ de tailles respectives $\ell(a) = n$ et $\ell(b) = m$, $\mathcal{M}(m, n) = \mathcal{O}(mn)$ et $\mathcal{M}(n) = \mathcal{O}(n^2)$ (complexité quadratique).

Division : algorithme de base

☞Résultat admis (voir le cours de *Représentation et Méthodes Numériques* en L2)

Complexité pour $a \geqslant b$ de tailles respectives $n+m$ et n,

$$\mathcal{D}(n+m, n) = \mathcal{O}(mn)$$

soit $\mathcal{D}(n) = \mathcal{O}(n^2)$ si $\ell(b) = n$ et $\ell(a) = 2n$.

m est la taille du quotient.

Complexité des algorithmes de base dans $\mathbb{Z}$

Complexité naïve

Pour $a \geqslant b > 0$

- Addition : $\mathcal{O}(\log(a))$
- Multiplication : $\mathcal{O}(\log(a)\log(b))$
- Division : $\mathcal{O}(\log(\frac{a}{b})\log(b)) = \mathcal{O}(\log(q)\log(b))$

☞Les complexités sont au plus quadratiques en la taille des entrées

Complexité des algorithmes de base dans $\mathbb{Z}$

Complexité naïve

Pour $a \geqslant b > 0$

- Addition : $\mathcal{O}(\log(a))$
- Multiplication : $\mathcal{O}(\log(a)\log(b))$
- Division : $\mathcal{O}(\log(\frac{a}{b})\log(b)) = \mathcal{O}(\log(q)\log(b))$

☞Les complexités sont au plus quadratiques en la taille des entrées

☞Le cas du calcul modulaire ?

Plan

Complexités modulo N (algorithmes de base)

☞Tous les calculs se font modulo N donc les opérandes sont tous majorés par N et la complexité s'exprime en fonction de $\log(N)$.

Complexité naïve

- Addition : $\mathcal{O}(\log(N))$
- Multiplication : $\mathcal{O}(\log(N)^2)$
- Division : comment divise-t-on modulo N ? en multipliant par l'inverse modulo $N^\dagger$.

† uniquement si nécessaire : si b divise a dans $\mathbb{Z}$, alors $a = bq$ dans $\mathbb{Z}$ mais aussi $a = bq \mod N$ et $q \in \mathbb{Z}/N\mathbb{Z}$. On ne calcule l'inverse de b que si b ne divise pas a.

Exemple dans $\mathbb{Z}/9\mathbb{Z}$, $a = 6$ et $b = 2$: $a/b = 6/2 = 3$
(en passant par l'inverse on trouve évidemment le même résultat : $2 \times 5 = 10 = 1 \mod 9$ donc $b^{-1} = 5 \mod 9$ et $a \times b^{-1} = 6 * 5 = 3 \mod 9$).

Complexités modulo N (algorithmes de base)

☞Tous les calculs se font modulo N donc les opérandes sont tous majorés par N et la complexité s'exprime en fonction de $\log(N)$.

Complexité naïve

- Addition : $\mathcal{O}(\log(N))$
- Multiplication : $\mathcal{O}(\log(N)^2)$
- Division : comment divise-t-on modulo N ? en multipliant par l'inverse modulo $N^\dagger$.

☞Les complexités sont au plus quadratiques en la taille de N mais que coûte le calcul d'un inverse ?

Complexités modulo N (algorithmes de base)

☞Tous les calculs se font modulo N donc les opérandes sont tous majorés par N et la complexité s'exprime en fonction de $\log(N)$.

Complexité naïve

- Addition : $\mathcal{O}(\log(N))$
- Multiplication : $\mathcal{O}(\log(N)^2)$
- Division : comment divise-t-on modulo N ? en multipliant par l'inverse modulo $N^\dagger$.

☞Les complexités sont au plus quadratiques en la taille de N mais que coûte le calcul d'un inverse ?

☞Coût du calcul du PGCD et de l'algorithme d'Euclide étendu ?

Plan

Calculer le PGCD

Si on ne sait pas calculer $\mathrm{pgcd}(a, b)$ sans factoriser a et b...

☞Le meilleure algorithme permettant de factoriser des entiers est de complexité sous-exponentielle. Le problème de la factorisation est donc assez difficile.

☞Si l'on ne connaissait pas d'autres algorithmes permettant le calcul du PGCD il en serait de même pour la complexité du PGCD...

Calculer le PGCD dans un anneau euclidien

Algorithme d'Euclide étendu

Initialisation :

$$\begin{cases} u_0 = 1, & v_0 = 0, & r_0 = a \\ u_1 = 0, & v_1 = 1, & r_1 = b \end{cases}$$

Calcul $i \geqslant 1$ tant que $r_i \neq 0$:

- $q_i = r_{i-1} \operatorname{div} r_i$
- $u_{i+1} = u_{i-1} - q_i u_i, v_{i+1} = v_{i-1} - q_i v_i, r_{i+1} = r_{i-1} - q_i r_i$

$\operatorname{pgcd}(a, b) = r_n$ le dernier reste non nul.

Plan

Nombre d'itérations : le pire cas (pour une taille donnée) !

☞Quand le quotient vaut 1 lors d'une étape, cela revient à utiliser la version simple d'Euclide (géométrique) : dans ce cas la division euclidienne est remplacée par une soustraction.

Exemple : pour $a = 13$ et $b = 8$:

$$\begin{aligned} 13 &= 1 \times 8 + 5 \\ 8 &= 1 \times 5 + 3 \\ 5 &= 1 \times 3 + 2 \\ 3 &= 1 \times 2 + 1 \\ 2 &= 2 \times 1 + 0 \end{aligned}$$

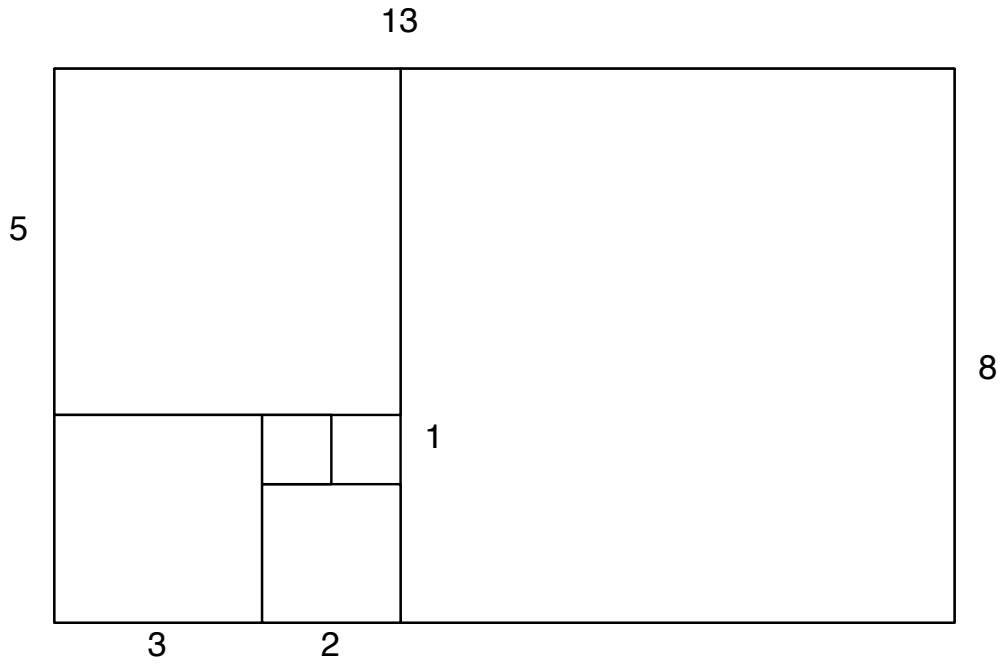
Avec 13 et 7 ou 14 et 8 on aurait 3 étapes seulement, on a vraiment un extrêmum local.

☞ Le nombre de calculs est maximal car les quotients sont tous (sauf le dernier) réduits à 1.

Nombre d'itérations : le pire cas (pour une taille donnée) !

☞Quand le quotient vaut 1 lors d'une étape, cela revient à utiliser la version simple d'Euclide (géométrique) : dans ce cas la division euclidienne est remplacée par une soustraction.

Plus généralement : cette propriété est vraie pour tout couple F_{n+2}, F_{n+1} de nombres de Fibonacci successifs



☞ Par définition, ces couples sont les pires cas.

Nombre d'itérations : les nombres de Fibonacci

Définition

$$\begin{cases} F_0 & = & 0 \\ F_1 & = & 1 \\ F_{n+2} & = & F_{n+1} + F_n \end{cases}$$

Propriétés

- Formule close et vitesse de croissance : $F_n = \frac{(\phi^n - \hat{\phi}^n)}{\sqrt{5}}$ avec $\phi = \frac{1+\sqrt{5}}{2}$ et $\hat{\phi} = \frac{1-\sqrt{5}}{2}$.
- Le calcul de $\mathrm{pgcd}(F_{n+2}, F_{n+1})$ par l'algorithme d'Euclide demande exactement n itérations.
- De plus $\mathrm{pgcd}(F_{n+2}, F_{n+1}) = 1$

Nombre d'itérations : le cas général

Théorème

Soit $a > b > 0$. Pour que le calcul de $\mathrm{pgcd}(a, b)$ nécessite n itérations il faut nécessairement que

$$a \geqslant F_{n+2} \text{ et } b \geqslant F_{n+1}$$

Corollaire (Théorème de Lamé)

$$n + 2 < \mathcal{O}(\log(a)) \text{ et } n + 1 < \mathcal{O}(\log(b))$$

☞ Le théorème de Lamé fournit la meilleure constante possible dans le $\mathcal{O}$ (5 dans le cas du logarithme décimal).
☞ On se rappellera que le nombre d'itérations est de l'ordre de $\log(b)$ i.e la taille du plus petit opérande.

Algorithme d'Euclide : coût de la i^{e} itération

- Le coût d'une itération est celui de la division
- En notant $t_i = \log(r_i)$ la taille de r_i, le coût de la i^{e} itération est

$$\mathcal{O}(\log\left(\frac{r_i}{r_{i+1}}\right)\log(r_{i+1})) = c_i\, t_{i+1}\,(t_i - t_{i+1})$$

Algorithme d'Euclide : coût total

Pour $a \geqslant b > 0$

- Il y a $\mathcal{O}(\log(b))$ itérations de la boucle.
- En notant n le nombre d'itérations, le coût total du pgcd est

$$\sum_{i=0}^{n-1} c_i t_{i+1}(t_i - t_{i+1})$$

- En utilisant $c_i \leqslant C = \max_i(c_i)$, $t_1 \geqslant t_{i+1}$ et $t_0 \geqslant t_0 - t_n$, le coût total du pgcd est majoré par la somme suivante

$$\sum_{i=0}^{n-1} c_i t_{i+1}(t_i - t_{i+1}) \leqslant C\, t_1 \sum_{i=0}^{n-1}(t_i - t_{i+1}) \leqslant C\, t_1\, t_0 = \mathcal{O}(\log(a)\log(b))$$

☞complexité au plus quadratique en la taille des entrées

Récapitulatif des complexités pour les algorithmes de base

Complexités sur les entiers

Pour $a \geqslant b > 0$

- Addition : $\mathcal{O}(\log(a))$
- Multiplication : $\mathcal{O}(\log(a)\log(b))$
- Division : $\mathcal{O}(\log(\frac{a}{b})\log(b))$

Complexités modulo N

- Addition : $\mathcal{O}(\log(N))$
- Multiplication : $\mathcal{O}(\log(N)^2)$
- Inverse (algorithme d'Euclide étendu) : $\mathcal{O}(\log(N)^2)$

☞complexité quadratique en la taille des entrées

Plan

Plan

Algorithmes un peu moins naïfs

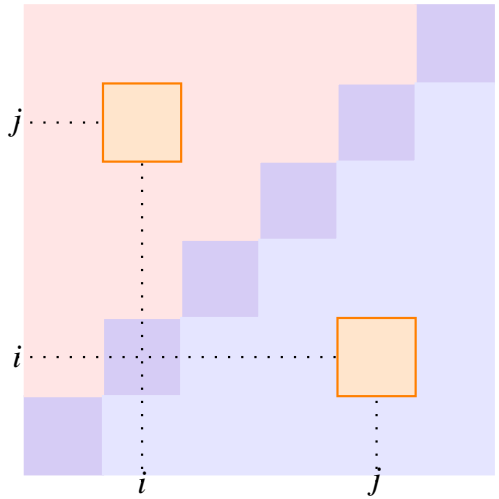
Tous les algorithmes vus précédemment reposent sur des techniques très simples n'utilisant pas forcément les symétries intrinsèques au problème pour calculer plus rapidement. Nous améliorons cela sur trois exemples :

- Le calcul du carré
- La multiplication (Karatsuba)
- L'exponentiation modulaire (*square and multiply*)

☞Il y a aussi des améliorations possibles de l'algorithme d'Euclide étendu mais ceci ne sera pas vu dans ce cours.

Plan

Carré



Pour calculer le carré de $n = \sum_{i=0}^{\ell(N)-1} n_i B^i$, on veut utiliser la symétrie et ne calculer qu'une seule fois le double produit

$$\left(\sum_{i=0}^{\ell(N)-1} n_i B^i\right)^2 = \sum_{i=0}^{\ell(N)-1} n_i^2 B^{2i} + 2 \sum_{0 \leqslant i < j \leqslant \ell(N)-1} n_i\, n_j B^{i+j}.$$

☞La complexité asymptotique reste la même mais on ne calcule qu'une seule fois le double produit, ce qui fait gagner un facteur entre 1,5 et 2.

Plan

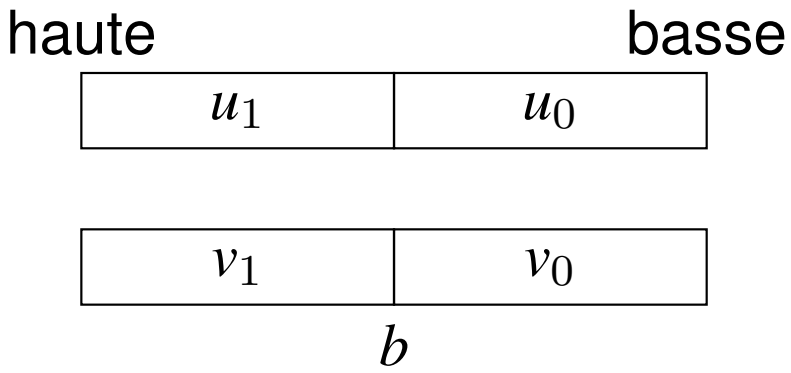
Multiplication selon Karatsuba

☞Nous avons vu que la complexité de la multiplication classique est quadratique en la taille des entrées.

Karatsuba

Procédé *Diviser pour régner* pour la multiplication de grands entiers.

$$u = u_1 b + u_0 \quad v = v_1 b + v_0$$



Multiplication selon Karatsuba

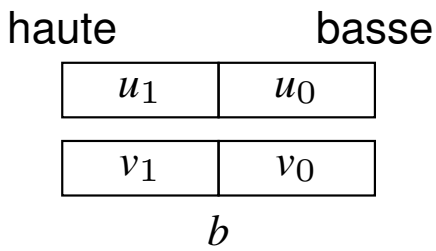
Karatsuba

Procédé *Diviser pour régner* pour la multiplication de grands entiers.

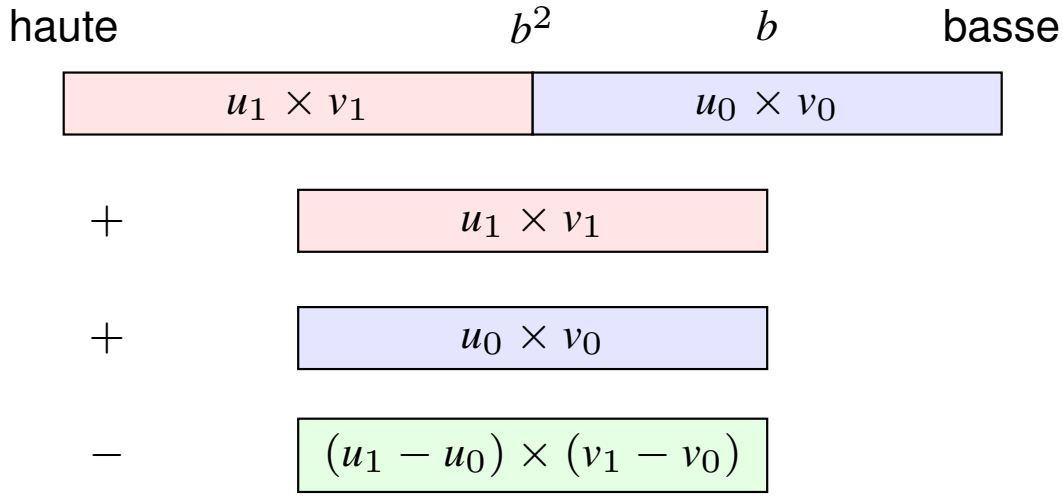
$$u \times v = (u_1 \times v_1)\, b^2 + (u_1 \times v_0 + u_0 \times v_1)\, b + (u_0 \times v_0)$$

classique $$u \times v = (u_1 \times v_1)\,(b^2 - b) + (u_1 + u_0) \times (v_1 + v_0)\, b + (u_0 \times v_0)\,(-b + 1)$$

optimisé $$u \times v = (u_1 \times v_1)\,(b^2 + b) - (u_1 - u_0) \times (v_1 - v_0)\, b + (u_0 \times v_0)\,(b + 1)$$



- $u_1 \times v_1$
- $(u_1 - u_0) \times (v_1 - v_0)$
- $u_0 \times v_0$



☞Récursivement

Karatsuba : complexité et pratique

Complexité

- Multiplication classique $\mathcal{O}(n^2)$
- Multiplication Karatsuba $\mathcal{O}(n^{\log_2(3)}) = \mathcal{O}(n^{1.585})$ (Théorème Maître avec $T(l) = 3T(l/2) + O(l)$)

- On remplace une multiplication par plusieurs additions et shifts
- $\mathcal{M}(\ell) = \mathcal{O}(\ell^2)$ et $\mathcal{A}(\ell) = \mathcal{O}(\ell)$ donc l'algorithme est rentable à partir d'une certaine taille MAIS cela dépend des constantes dans les $\mathcal{O}$, en particulier de l'architecture de la machine.

Plan

Exponentiation modulaire : algorithme naïf par $k-1$ multiplications

- Entrée $a \in \{0, \ldots, n-1\}$ et $k \in \mathbb{N}$.
- Sortie $a^k \bmod n$

Réduction finale

- On fait $k-1$ multiplications de a par $a, \ldots, a^{k-1}$ dans $\mathbb{Z}$ en utilisant de l'arithmétique sur les grands entiers et on réduit $\bmod n$ un nombre de l'ordre de n^k
- Le coût est

$$\sum_{i=1}^{k-1} \underbrace{1 * i * \mathcal{M}(\log(n))}_{\text{produit de } a \text{ par } a^i} + \underbrace{\mathcal{M}(k\log(n))}_{\text{réduction de } a^k \text{ modulo } n}$$

$$= \left(\sum_{i=1}^{k-1} i\right) \mathcal{M}(\log(n)) + \mathcal{O}(k^2)\mathcal{M}(\log(n)) = \mathcal{O}(k^2)\mathcal{M}(\log(n)) = \mathcal{O}(k^2)$$

donc est exponentiel en la taille de k

Exponentiation modulaire : algorithme naïf par $k-1$ multiplications

- Entrée $a \in \{0, \ldots, n-1\}$ et $k \in \mathbb{N}$.
- Sortie $a^k \mod n$

Réduction au fur et à mesure

- On fait $k-1$ multiplications de a, donc d'au plus n par un nombre de même taille dans $\mathbb{Z}/n\mathbb{Z}$ et on réduit à chaque étape un nombre inférieur à $n^2 \mod n$
- Le coût est $2(k-1)\mathcal{M}(\log(n))$ (1 pour la multiplication, 1 pour la réduction à chaque étape) donc en $\mathcal{O}(k)$, c'est mieux mais c'est toujours exponentiel en la taille de k

☞Il faut conserver la réduction modulaire mais il faut effectuer moins de multiplications. Dans la suite on oublie le caractère modulaire, on commence par s'occuper de l'exposant k.

Exponentiation : square-and-multiply

Écrivons le début de la décomposition de k en base 2 : $k = 2k' + k_0$.
Si $k_0 = 0$

$$a^k = a^{2k'} = (a^{k'})^2 \qquad \text{Square}$$

Si $k_0 = 1$

$$a^k = a^{2k'} \times a \qquad \text{Multiply}$$

- Algorithme récursif
- On ne va faire que $\log_2(k)$ multiplications, c'est beaucoup mieux !
- On peut évidemment le faire pour des multiplications de grands nombres ou modulo n...

Dans le cas modulaire

On fait les calculs modulo un entier n.

☞On peut considérer $k \leq n-1$, pourquoi ? ☞TD

☞On fait tous les calculs modulo n donc toutes les opérandes sont de taille $\log(n)$

Conclusion

Complexité de l'exponentiation modulaire
$\mathcal{O}(\log(n)\mathcal{M}(\log(n))) = \mathcal{O}(\log(n)^3)$ dans le cas de la multiplication naïve

Plan

Conclusion : problèmes $\mathcal{NP}$ identifiés

Vérifications faciles

- Étant donné n, a, b on peut vérifier $n = a \times b$ en multipliant !
- Étant donné n, a, b, k on peut vérifier facilement $b = a^k \bmod n$ en calculant une exponentiation modulaire !

Problèmes difficiles correspondants

- FACT : étant donné un grand entier n, donner sa factorisation ?
- DLP : étant donné b, a, n, comment retrouver l'exposant k tel que $b = a^k \bmod n$?
- Racine k-ième modulo n : étant donné b, k, n, comment retrouver a tel que $b = a^k \bmod n$?

☞On ne connaît pas d'algorithme de complexité polynomiale résolvant ces problèmes !

☞Peut-on utiliser ces problèmes pour de la cryptographie ?

Plan

Plan

Des bibliothèques d'arithmétique de grands entiers

- Historique (MP (1970), Bignum (1987), GMP (1991), dans logiciels dédiés tels que Maple, Mathematica (calcul formel), Pari (théorie des nombres), NTL (corps finis), I2P (réseau d'anonymisation de connexion), etc.)
- Avec le temps, les logiciels dédiés ont petit à petit délégué leur arithmétique entière à des bibliothèques spécialisées, le développement de Bignum est arrêté, il reste essentiellement GMP sur lequel porte tout l'effort de développement et qui est devenu *de facto* un standard.
- Contrairement à ce que pourrait laisser croire les dates de publications des algorithmes, les premières implémentations sont souvent récentes et c'est encore un domaine de recherche très actif.
- Algorithmes à la pointe en GMP, tout nouvel algorithme est codé/testé dans GMP ou presque et s'il vaut le coup est intégré dans une version ultérieure.

Plan

GMP : ses caractéristiques

- Multiplication optimisée conjuguant la multiplication naïve, Karatsuba(1962), n-way Toom-Cook (Toom 1963 Russe, Cook 1966 Américain, $O(n^{1.465})$ pour $n = 3$, $O(n^{1.404})$ pour $n = 4$, $O(n^{1.328})$ pour $n = 6.5$, $O(n^{1.295})$ pour $n = 8.5$), FFT (Gauß 1805, Cooley&Tukey 1965, Winograd 1978, Schönhage 1986, etc. $O(n \log(n) \log(\log(n)))$)
- Multiples algorithmes de division : par un chiffre, de base, diviser pour règner (idée : calculer récursivement le quotient des $\ell(v)$ premiers chiffres de u par v comme si on appliquait l'algorithme de division de base en base $B^{\ell(v)}$ puis le reste en utilisant Karatsuba pour calculer le produit du diviseur par le début du quotient de même taille, $O(\mathcal{M}(n)\ \log(n))$), exacte, etc.
- Pgcd binaire, *à la Jebelean*, étendu ; exponentielle, exponentielle modulaire, calcul de racines, etc.
- Les seuils de rentabilité des algorithmes sont disponibles sous forme de constantes instanciées pour chaque processeur/compilateur

Voir `http://gmplib.org/manual/Algorithms.html`

GMP : son installation

Package source disponible ici : `http://gmplib.org`
Installation standard :

```
% ./configure
% make
% make check
% make install
#Dans /usr/local par défaut, mais cela peut être spécifié
```

Entête de fichier source :

```
#include <gmp.h>
...
```

Édition de liens :

```
% gcc -o monprog monprog.c -lgmp
```

Attention la compilation MAC OS X nécessite l'option -m64.

GMP : les variables entières

☞Les variables GMP sont des structures contenant des pointeurs

Déclaration, initialisation, libération

- Déclaration `mpz_t n`
- Initialisation `mpz_init(n)`
- Libération `mpz_clear(n)`

Exemple :

```
void foo (void) {
  mpz_t  n;
  int    i;
  mpz_init (n);
  for (i = 1; i < 100; i++) {
    mpz_mul (n, ...);
    mpz_fdiv_q (n, ...);
    ...
  }
  mpz_clear (n);
}
```

GMP : les fonctions

☞Les valeurs calculées sont généralement renvoyées par le premier paramètre
☞Les valeurs de retour renseignent sur l'état, le signe, etc. du calcul
☞Les fonctions sont spécifiques à la taille des entrées pour une meilleure efficacité.

Exemple de l'affectation, `rop` reçoit le résultat :

- `mpz_set (mpz_t rop, mpz_t op)`
- `mpz_set_ui (mpz_t rop, unsigned long int op)`
- `mpz_set_str (mpz_t rop, char *str, int base)`

`rop` $\leftarrow$ `op`

Exemple de la multiplication, `rop` reçoit le résultat :

- `mpz_mul (mpz_t rop, mpz_t op1, mpz_t op2)`
- `void mpz_mul_si (mpz_t rop, mpz_t op1, long int op2)`
- `void mpz_mul_ui (mpz_t rop, mpz_t op1, unsigned long int op2)`

`rop` $\leftarrow$ `op1` $\times$ `op2`

Complexité pour mesurer la sécurité