Arithmétique

2ième année de DUT Informatique

Version 2.1

3 février 2009

Ph. Roux

Table des matières

Chapitre 1

cours magistral

L'arithmétique est la partie des mathématiques qui étudie les ensembles

$$\mathbb{N} = \{0;\ 1;\ 2;\ \dots\},\ \ \mathbb{N}^* = \{1;\ 2;\ \dots\},\ \ \mathbb{Z} = \{\dots;\ -2;\ -1;\ 0;\ 1;\ 2;\ \dots\}$$

munis des opérations naturelles qui leur sont associées :

- l'addition +
- la multiplication ×
- et la divisibilité |

Ces opérations sont bien sûr indépendantes de la manière dont on représente les nombres entiers. Par défaut on représente les nombres entier en base 10 :

$$x = (a_n a_{n-1} \dots a_2 a_1 a_0)_{10} = \sum_{k=0}^{n} a_k 10^k = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \dots a_2 \times 10^2 + a_1 \times 10 + a_0$$

où les chiffres $a_n \dots a_1, a_0$appartiennent à $\{0;\ 1;\ 2;\ \dots 9\}$ mais on pourrait tout aussi bien travailler dans une base p quelconque, où la notation des entiers est définie par

$$x = (b_n b_{n-1} \dots b_2 b_1 b_0)_p = \sum_{k=0}^{n} b_k p^k = b_n \times p^n + b_{n-1} \times p^{n-1} + \dots b_2 \times p^2 + b_1 \times p + b_0$$

où les chiffres cette fois appartiennent à $\{0;\ 1;\ 2;\ \dots p\}$. Cela pose un problème pour les bases $p > 10$ (les chiffres de 0 à 9 ne suffisent plus). Pour résoudre ce problème on peut utiliser soit des groupements de chiffres soit des lettres pour remplacer les chiffres manquants. Par exemple en Hexadécimal (base 16) on peut utiliser les chiffres :

$$1;\ 2;\ 3;\ 4;\ 5;\ 6;\ 7;\ 8;\ 9;\ A;\ B;\ C;\ D;\ E;\ F$$

ou

$$01;\ 02;\ 03;\ 04;\ 05;\ 06;\ 07;\ 08;\ 09;\ 10;\ 11;\ 12;\ 13;\ 14;\ 15$$

alors

$$(123456789)_{10} = (111010110111100110100010101)_2 = (75BCD15)_{16}\ \textit{ou}\ (07\ 05\ 11\ 12\ 13\ 01\ 05)_{16}$$

La longueur de l'écriture d'un nombre x en base p est la partie entière de

$$\ln_p(x) = \frac{\ln(x)}{\ln(p)}$$

1.1 Divisibilité

1.1.1 L'algorithme d'Euclide

Définition 1.1.1
On dit qu'un entier a divise un autre entier b si et seulement si il existe un entier c tel que $b = a \times c$ ce qu'on note $a|b$. Si a ne divise pas b on le note $a \nmid b$.

Les règles élémentaires de divisibilité :

- $\forall a \in \mathbb{Z}\ \ 1|a,\ \ a|a,\ \ a|0$, et $0 \nmid a$
- si $a|b$ alors $\forall c \in \mathbb{Z},\quad a|bc$
- si $a|b$ et $a|c$ alors$\forall x,y \in \mathbb{Z}\ \ a|bx + cy$
- si $a|b$ et $b|c$ alors $a|c$
- si $a|b$ et $b|a$ alors $a = \pm b$
- si $a > 0$, $b > 0$ et $a|b$ alors $a \leq b$

Définition 1.1.2 *Étant donné deux entiers a et b on note*

- PGCD(a,b) *le plus grand entier naturel qui divise a et b,*
- PPCM(a,b) *le plus petit multiple commun à a et b.*

Théorème 1.1.3 (Division Euclidienne)

$$\forall a \in \mathbb{Z},\ \ b \in \mathbb{N}^*,\ \ \exists!(q,r) \in \mathbb{Z} \times \mathbb{N}^*\ \textit{tel que}\ a = b \times q + r\ \ \textit{avec}\ \ 0 \leq r < b$$

q est appelé le quotient de a par b et r le reste de la division de a par b.

Exemple Pour $a = 17,\ \ b = 9$, on a $17 = 9 \times 1 + 8 \Longrightarrow q = 1,\ \ r = 8$. Les autres égalités $17 = 9 \times 2 - 1 = 9 \times 0 + 17 = 9 \times (-1) + 26$ ne donnent pas les bonnes valeurs de (q,r) car dans chacun des cas qui précèdent le reste $r \notin [0,9[$.

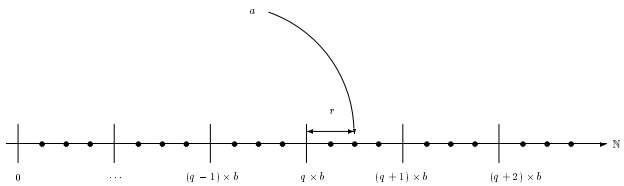

FIG. 1.1 – *Preuve de l'existence d'une division Euclidienne.*

Preuve :

existence: si l'on découpe l'axe $[0, +\infty[$ en "boîtes" de "taille" b le nombre a tombe dans une de ces boites, la $q^{\text{ième}}$. Cela revient à dire que le nombre a/b est compris entre les entiers q et $q+1$, il est alors facile de définir r :

$$\exists q \in \mathbb{N},\ \ q \leq \frac{a}{b} < q + 1 \Rightarrow qb \leq a < (q+1)b \Rightarrow 0 \leq a - qb < b \Rightarrow r = a - qb$$

unicité: elle repose sur une démonstration par l'absurde. Supposons donc qu'il existe 2 solutions (q_1,r_1) et (q_2,r_2) alors on a :

$$a = b \times q_1 + r_1 = b \times q_2 + r_2 \Longrightarrow b(q_1 - q_2) = r_2 - r_1 \tag{1.1}$$

et comme les restes sont compris entre 0 et b

$$0 \leq r_1 < b \quad \text{et} \quad 0 \leq r_2 < b \Longrightarrow \quad -b < r_2 - r_1 < b \Longrightarrow \quad 0 < |r_2 - r_1| < |b| \tag{1.2}$$

Maintenant on a l'alternative suivante:

- soit $q_1 = q_2$ et dans ce cas $r_1 = r_2$ d'après l'équation (1.1)
- soit $q_1 \neq q_2$ et dans ce cas on trouve une contradiction entre les équations (1.1) et (1.2)

$$q_1 - q_2 \neq 0 \Longrightarrow |q_1 - q_2| \geq 1 \Longrightarrow |r_2 - r_1| = |b||q_1 - q_2| \geq |b|.$$

seul le premier cas est donc réalisé.

□

Le calcul du PGCD s'effectue facilement à partir de l'algorithme d'Euclide. Si on suppose que a et $b \geq 0$ alors on peut calculer le PGCD(a,b) par divisions Euclidiennes successives :

$$\begin{array}{rcrcrcrl}
a & = & b & \times & q_1 & + & r_1 & \\
b & = & r_1 & \times & q_2 & + & r_2 & \\
r_1 & = & r_2 & \times & q_3 & + & r_3 & \\
 & \vdots & & & & & & \\
r_{n-4} & = & r_{n-3} & \times & q_{n-2} & + & r_{n-2} & \\
r_{n-3} & = & r_{n-2} & \times & q_{n-1} & + & r_{n-1} & \\
r_{n-2} & = & r_{n-1} & \times & q_n & + & r_n & \Longrightarrow \text{PGCD}(a,b) = r_n \\
r_{n-1} & = & r_n & \times & q_{n+1} & + & 0 &
\end{array}$$

Preuve :

- en remontant la suite d'équations

$$r_n | r_{n-1} \Rightarrow r_{n-1} | r_{n-2} \cdots \Rightarrow r_n | a \text{ et } b \Rightarrow r_n | \text{PGCD}(a,b)$$

- en descendant la suite d'équations

$$\text{PGCD}(a,b) | a \text{ et } b \Rightarrow \text{PGCD}(a,b) | b \text{ et } r_1 \cdots \Rightarrow \text{PGCD}(a,b) | r_n$$

donc $r_n = \text{PGCD}(a,b)$! □

Exemple 1.1.4 [détail d'un exemple avec 48 et 27]

$$\begin{array}{rcrcrcrl}
48 & = & 27 & \times & 1 & + & 21 & \\
27 & = & 21 & \times & 1 & + & 6 & \\
21 & = & 6 & \times & 3 & + & 3 & \\
6 & = & 3 & \times & 2 & + & 0 & \\
 & & & & & \Rightarrow & \text{PGCD}(48,27) = 3 &
\end{array}$$

1.1.2 Identités de Bezout

Théorème 1.1.5 (Identité de Bezout)
Si $a,b \in \mathbb{Z}$ alors il existe $u,v \in \mathbb{Z}$ tels que

$$a \times u + b \times v = \text{PGCD}(a,b).$$

Il faut remarquer que le couple (u,v) dans l'identité de Bezout n'est pas unique! En effet comme $a \times b + b \times (-a) = 0$ on peut ajouter un multiple de b à u et un multiple de a à v. Pour obtenir tous les autres couples solution (u',v') à partir de (u,v) on utilisera les formules suivantes :

$$u' = u + k\frac{\text{PPCM}(a,b)}{a}, \;\; v' = v - k\frac{\text{PPCM}(a,b)}{b}, \;\; k \in \mathbb{Z}$$

on peut exprimer ces formules avec des PGCD car

$$\frac{\text{PPCM}(a,b)}{a} = \frac{\text{PPCM}(a,b) \times \text{PGCD}(a,b)}{a \times \text{PGCD}(a,b)} = \frac{ab}{a \times \text{PGCD}(a,b)} = \frac{b}{\text{PGCD}(a,b)}$$

Exemple 1.1.6 Pour $a = 17, \;\; b = 9$, on a

$$\text{PGCD}(17,9) = 1 = 9 \times 2 + 17 \times (-1) = 9 \times 19 + 17 \times (-10) = \ldots.$$

Pour calculer des identités de Bezout on peut utiliser les différentes équations obtenues dans l'algorithme d'Euclide

$$\begin{array}{rcll}
\Longrightarrow \text{PGCD}(a,b) & = & r_n & \text{étape } 0 \\
 & = & r_{n-2} - r_{n-1}q_n & \text{étape } 1 \\
 & = & r_{n-4} - r_{n-3}q_{n-2} - (r_{n-3} - r_{n-2}q_{n-1})q_n & \text{étape } 2 \\
 & \vdots & & \\
 & = & \text{en fonction de } r_{n-k}, \ldots r_{n-2k} & \text{étape } k \\
 & \vdots & & \\
 & = & au + bv & \text{étape } n
\end{array}$$

Exemple 1.1.7 [détail d'un exemple avec 48 et 27]

$$\begin{array}{rcl|rcl}
48 & = & 27 \times 1 + 21 & 3 & = & -3 \times 6 + 21 \\
27 & = & 21 \times 1 + 6 & & = & -3 \times (27 - 21) + (48 - 27) \\
21 & = & 6 \times 3 + 3 & & = & -3 \times (2 \times 27 - 48) + 48 - 27 \\
6 & = & 3 \times 2 + 0 & & = & -7 \times 27 + 4 \times 48
\end{array}$$

$$\Longrightarrow 27 \times (-7) + 48 \times 4 = 3 = \text{PGCD}(27,48)$$

Résolution de l'équation $ax + by = c$

- Calculer le PGCD de a et b via l'algorithme d'Euclide.
- Si PGCD(a,b) ne divise pas c alors l'équation n'a pas de solutions.
- Si PGCD(a,b)$|c$ on cherche une identité de Bezout

$$au + bv = \text{PGCD}(a,b).$$

- En multipliant l'identité de Bezout par $c/\text{PGCD}(a,b)$ (qui est bien un entier!) on obtient une solution :

$$x_0 = \frac{u \times c}{\text{PGCD}(a,b)},\ \ y_0 = \frac{v \times c}{\text{PGCD}(a,b)}.$$

- À partir de cette solution on obtient toutes les autres solutions (x,y) avec la même méthode que pour les identités de Bezout :

$$x = x_0 + \frac{b \times l}{\text{PGCD}(a,b)},\quad y = y_0 - \frac{a \times l}{\text{PGCD}(a,b)},\quad l \in \mathbb{Z}.$$

1.1.3 Conversion d'un entier en base p

La méthode la plus efficace pour convertir un entier en base p est la *méthode du bit de poids faible*. Elle consiste à calculer le dernier chiffre de la représentation du nombre en base p par une division Euclidienne. en effet si

$$x = (b_n b_{n-1} \dots b_2 b_1 b_0)_p = b_n \times p^n + b_{n-1} \times p^{n-1} + \dots b_2 \times p^2 + b_1 \times p + b_0$$

alors

$$\begin{aligned} x &= (b_n b_{n-1} \dots b_2 b_1 b_0)_p = \sum_{k=0}^{n} b_k p^k \\ &= b_n \times p^n + b_{n-1} \times p^{n-1} + \dots b_2 \times p^2 + b_1 \times p + b_0 \\ &= (b_n \times p^{n-1} + b_{n-1} \times p^{n-2} + \dots b_2 \times p + b_1) \times p + b_0 \\ &= (b_n b_{n-1} \dots b_2 b_1)_p \times p + b_0 \end{aligned}$$

donc le reste de la division de x par p est b_0 et on peut réappliquer la méthode au quotient de x par p qui est $(b_n b_{n-1} \dots b_2 b_1)_p$. On peut récapituler la méthode comme ci-dessous :

écrire x en base p : la méthode du bit de poids faible

- initialisation $x_0 = x$
- à chaque étape on fait la division entière de x_i par p : $x_i = x_{i+1}p + b_i$
  - b_i est donc le reste de la division de x_i par p
  - x_{i+1} est donc le quotient de la division de x_i par p
- il n'est plus nécessaire de continuer dès que $x_i = 0$.

1.2 Nombres premiers

1.2.1 Théorème de décomposition en facteurs premiers

Définition 1.2.1

- *On dit que a,$b \in \mathbb{Z}$ sont* premiers entre eux *si et seulement si* PGCD(a,b) $= 1$.
- *On dit que $p \in \mathbb{N}^*$ est un nombre premier si et seulement si il n'est divisible par aucun entier $1 < k < p$.*
- *On appelle $\mathcal{P}$ l'ensemble des nombres premiers.*

Il faut signaler un cas particulier de cette définition : $p = 1$ est considéré comme premier avec n'importe quel nombre $n > 1$ car PGCD(1,n) $= 1$ comme le montre l'identité de Bezout $n \times 1 + 1 \times (1 - n) = 1$.

La première chose à savoir sur les nombres premiers est qu'il en existe un nombre infini, on pourra donc en trouver d'aussi grands que l'on veut (à condition d'avoir un algorithme efficace pour les générer et les tester!). Ce théorème à été démontré par Euclide.

Théorème 1.2.2 *L'ensemble $\mathcal{P}$ des nombres premiers contient une infinité d'éléments.*

Preuve : Supposons que $\mathcal{P} = \{p_1,p_2,\dots p_n\}$ est fini et considérons $q = p_1p_2 \dots p_n+1$ alors q n'est divisible par aucun p_i donc il est premier et $q \in \mathcal{P}$ ↯ □

Exemple 1.2.3 Cette preuve par l'absurde (donc pas vraiment constructive) permet de trouver une méthode pour engendrer des (facteurs) premiers de plus en plus gros :
$2 \in \mathcal{P}$,$2 + 1 = 3 \in \mathcal{P}$,$2 \times 3 + 1 = 7 \in \mathcal{P}$,$2 \times 3 \times 7 + 1 = 43 \in \mathcal{P}$,
par contre $2 \times 3 \times 7 \times 43 + 1 = 1807 = 13 \times 139 \notin \mathcal{P}$,
mais on abien deux nouveaux facteurs premiers 13 et $139 \in \mathcal{P}$
ensuite $2 \times 3 \times 7 \times 43 \times 13 \times 139 + 1 = 3263443 \in \mathcal{P}$,
et $2{\times}3{\times}7{\times}43{\times}13{\times}139{\times}3263443{+}1 = 10650056950807 = 547{\times}607{\times}1033{\times}31051$
on a donc bien des facteurs premiers nouveaux à chaque étape ... mais pas forcément des nombres premiers et pas forcément en ordre croissant.

Les nombres premiers forment donc une suite $(p_k)_{\mathbb{N}}$ il est de coutume de noter $p_0 = 1$ puis $p_1 = 2$ (premier nombre premier) $p_2 = 3$,$p_4 = 5$,$p_5 = 7$,$p_6 = 11 \dots$. Le théorème suivant est souvent très utile et sa démonstration est caractéristique des raisonnements faisant intervenir les nombres premiers et les relations de divisibilité.

Théorème 1.2.4 (Théorème de Gauss)
si $a|b \times c$ et a est premier avec b alors $a|c$.

Preuve : Traduisons les données du problème en équations :

- $a|b \times c \iff \exists k \in \mathbb{Z},\ \ bc = ak$
- a est premier avec $b \iff \exists u,v \in \mathbb{Z},\ \ au + bv = 1$

multiplions l'identité de Bezout par c puis utilisons la première égalité :

$$c = acu + bcv = acu + akv = a(cu + kv) \Longrightarrow \exists k' = cu + kv \in \mathbb{Z},\ \ c = ak' \iff a|c$$

□

Le théorème fondamental de ce chapitre est le théorème de décomposition en facteurs premier, simple à énoncer il n'est pas facile à traduire en équation. Euclide à son époque ne disposait pas des notations algébriques suffisantes (puissances, suites,...) pour énoncer clairement ce théorème bien qu'il ait certainement compris le théorème de décomposition, puisqu'il en a énoncé toutes les conséquences!

Théorème 1.2.5 (théorème de décomposition en facteurs premier) *Tout entier $n \in \mathbb{N}$, $n \geq 2$ admet une unique décomposition en facteurs premiers :*

$$n = p_1^{\eta_1} p_2^{\eta_2} \dots p_k^{\eta_k}$$

où les entiers p_1,$p_2 \dots p_k$ sont premiers et η_1,η_2,...,$\eta_k \geq 0$

Preuve : le théorème de décomposition nécessite une démonstration par récurrence :

- $\mathcal{P}_n \Leftrightarrow$ tous les nombres $\leq n$ admettent une décomposition unique
- $\mathcal{P}_2$ est vraie.
- $\mathcal{P}_n \Rightarrow \mathcal{P}_{n+1}$
  - soit $n+1$ est premier et sa décomposition est unique
  - soit $n+1$ est divisible par p, comme $(n+1)/p \leq n$ on a $(n+1)/p = p_1^{\eta_1} p_2^{\eta_2} \dots p_k^{\eta_k}$ admet une décomposition unique et $n+1 = p p_1^{\eta_1} p_2^{\eta_2} \dots p_k^{\eta_k}$ aussi

□

cette preuve nous permet de trouver un algorithme pour calculer la décomposition en facteurs premiers d'un entier quelconque :

Décomposition en nombres premiers

$p := 2$;
tant que $n > 1$ **faire**
  si $p|n$ **alors** on cherche l'exposant α de p dans la décomposition de n
    $\alpha := 0$;
    tant que $p|n$ **faire**
      $n := n/p$;
      $\alpha := \alpha + 1$;
    fin faire
    stocker p et α;
  fin si
  $p :=$ nombre premier suivant p;
fin faire

Remarque 1.2.6 On ne sait pas facilement trouver le nombre premier suivant p, alors on fait $p := p+1$. Cependant on est sûr qu'un facteur p trouvé par l'algorithme est à chaque fois un nombre premier. En effet si $p = p_1 \times p_2$ alors p_1 et p_2 auront déjà été testé par l'algorithme (car p_1,$p_2 < p$) et n ne pourra plus être divisible par p.

Il est possible d'améliorer cet algorithme en stoppant la recherche du facteur p dès que $p > \sqrt{n}$ en effet à une étape donné de l'algorithme n n'est pas divisible par

les facteurs premiers $q < p$ donc il n'a que des facteurs $q \geq p$. En conséquence si n à au moins 2 facteurs on a que $n \geq p^2 > \sqrt{n}^2 = n$ ce qui est impossible! Donc n est forcément un nombre premier.

Exemple 1.2.7 En testant si $p < \sqrt{n}$ on peut réduire un peut le temps de calcul, surtout s'il reste un dernier facteur premier de grande taille :

n	p
80360	2
40180	2
20090	2
10045	5
2009	7
287	7
41	arrêt recherche à p =8> $\sqrt{41}$
	41 (premier)
1	

conclusion : $80360 = 2^3 \times 5 \times 7^2 \times 41$.

un autre exemple simple :

n	p
84	2
42	2
21	3
7	7
1	fin

conclusion : $84 = 2^2 \times 3 \times 7$.

On peut signaler que la décomposition en facteurs premiers permet de retrouver les résultats sur les PGCD et PPCM.

Théorème 1.2.8 *Soient a et b deux entiers positifs, on note leurs décompositions en facteurs premiers*

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k} \quad b = p_1^{\beta_1} p_2^{\beta_2} \dots p_k^{\beta_k}$$

alors

$$\mathrm{PGCD}(a,b) = p_1^{\min(\alpha_1,\beta_1)} p_2^{\min(\alpha_2,\beta_2)} \dots p_k^{\min(\alpha_k,\beta_k)},$$

$$\mathrm{PPCM}(a,b) = p_1^{\max(\alpha_1,\beta_1)} p_2^{\max(\alpha_2,\beta_2)} \dots p_k^{\max(\alpha_k,\beta_k)}$$

en particulier $\mathrm{PGCD}(a,b) \times \mathrm{PPCM}(a,b) = a \times b$.

Exemple 1.2.9 Par exemple $48 = 2^4 \times 3$ et $27 = 3^3 = 2^0 \times 3^3$ donc $\mathrm{PGCD}(48,27) = 2^0 \times 3^1 = 3$ et $\mathrm{PPCM}(48,27) = 2^4 \times 3^3 = 432$ et on a bien

$$\mathrm{PGCD}(48,27) \times \mathrm{PPCM}(48,27) = (2^0 \times 3^1) \times (2^4 \times 3^3) = (2^4 \times 3^1) \times (2^0 \times 3^3) = 48 \times 27$$

La fonction suivante sera utilisé pour le crypto-système RSA. Elle joue un grand rôle en arithmétique et son calcul repose entièrement sur la décomposition en facteurs premiers.

Définition 1.2.10 (la fonction d'Euler) *on définit la fonction Φ telle que $\Phi(1) = 1$ et $\Phi(n) =$ le nombre d'entiers premiers avec n et compris entre 1 et $n-1$*

$$\begin{array}{rccc} \Phi & : & \mathbb{N}^* & \longrightarrow & \mathbb{N}^* \\ & & n & \longmapsto & \Phi(n) \end{array}$$

Exemple 1.2.11 Pour comprendre on peut calculer les premières valeurs de $\Phi(n)$:

- $\Phi(7) = 6$ car 7 est premier, donc premier avec 1,2,3,4,5,6
- $\Phi(10) = 4$ car 10 est premier avec 1,3,7,9
- $\Phi(12) = 4$ car 12 est premier avec 1,5,7,11

par contre si n devient grand il devient très lourd de vérifier pour tout $p < n$ si p est premier avec n.

On a donc besoin d'une formule de calcul efficace pour Φ.

Proposition 1.2.12 (calcul de Φ)

- *si p est premier alors* $\Phi(p) = p - 1$
- *si p est premier et $\alpha \geq 1$ alors* $\Phi(p^\alpha) = p^\alpha - p^{\alpha-1} = p^\alpha(1 - \frac{1}{p})$
- *si* PGCD$(a,b) = 1$ *alors* $\Phi(ab) = \Phi(a)\Phi(b)$.
- *si* $n = p_1^{\eta_1} p_2^{\eta_2} \dots p_k^{\eta_k}$ *alors*

$$\Phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\dots\left(1 - \frac{1}{p_k}\right)$$

Preuve :

- tous les nombres inférieurs à p sont premier avec lui donc il y en a $p-1 = \Phi(p)$,
- seuls les nombres $p,2p,3p,\dots p^{\alpha-1}p$ ne sont pas premiers avec p^α et compris entre 1 et p^α. Il y en a donc $p^{\alpha-1}$ et le nombre d'entiers premiers avec p^α est $\Phi(p^\alpha) = p^\alpha - p^{\alpha-1}$.
- On considère un nombre q premier avec ab et $q \in [1,ab[$, et les ensembles :
  $\{r_1,\dots,r_{\Phi(a)}\}$ = nombres premiers avec a et $< a$
  $\{s_1,\dots,s_{\Phi(b)}\}$ = nombres premiers avec b et $< b$
  l'idée de base est que si un nombre q est premier avec ab son reste modulo a est premier avec a (idem avec b). En effet sinon on aurait un diviseur commun à a et r, le reste de q/a (soit $q = ak + r$ avec $k \in \mathbb{N}$), ce qui donnerai :

  $$\exists d > 1,\ d|a \ \ et \ \ d|r \Rightarrow d|ka + r = q$$

  ce qui est impossible car d diviserai q et ab! Donc r est un des restes numérotés r_i (et de même pour le reste modulo b qui doit être un des s_j). Donc on a un système de 2 équations dont il faut compter les solutions :

  $$\left\{\begin{array}{rcl} q &=& r_i + ka \\ q &=& s_j + k'b \end{array}\right. \Rightarrow ka + (-k')b = s_j - r_i$$

  comme PGCD$(a,b) = 1$ il existe des solutions à cette équation diophantienne, solutions qui s'écrivent pour chaque couple (r_i,s_j):

  $$\left\{\begin{array}{rcl} k &=& k_i + b \times l \\ & & \qquad\qquad l \in \mathbb{Z} \implies q = r_i + k_i a + ab \times l \\ -k' &=& -k'_j + a \times l \end{array}\right.$$

  on trouve donc une seule solution q dans l'intervalle $[1,ab]$, associée au couple (r_i,s_j), donc il y a autant de solutions que de couple $\longrightarrow \Phi(a)\Phi(b)$.
- application des 2 résultats précédents.

□

Exemple 1.2.13 on peut reprendre les exemples précédents pour voir l'efficacité de la formule :

- $\Phi(7) = \Phi(7^1) = 7 - 1 = 6$
- $\Phi(10) = \Phi(2 \times 5) = 10\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{5}\right) = 10\frac{1}{2}\frac{4}{5} = 4$
- $\Phi(12) = \Phi(2^2 \times 3) = 12\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 10\frac{1}{2}\frac{2}{3} = 4$
- $\Phi(60) = \Phi(2^2 \times 3 \times 5) = 60\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right) = 60\frac{1}{2}\frac{2}{3}\frac{4}{5} = 16$

attention, dans la formule finale, la puissance d'un facteur p dans la décomposition du nombre n n'intervient pas dans le calcul de $\Phi(n)$!

1.2.2 Recherche de grands nombres premiers

Théorème 1.2.14 (théorème de raréfaction) *Soit $\pi(x)$ le nombre de nombres premiers inférieurs à x alors on a que*

$$\pi(x) \underset{x\to\infty}{\sim} \frac{x}{\ln(x)} \iff \lim_{x\to\infty} \frac{\pi(x)}{x/\ln(x)} = 1.$$

Si on note p_n le $n^{ième}$ nombre premier alors le résultat précédent est équivalent au fait que

$$p_n \underset{n\to\infty}{\sim} n\ln(n).$$

Preuve : La répartition des nombres premiers est l'un des phénomènes les plus complexe. Il est très difficile d'obtenir des informations sur la répartition des nombres premiers, l'approche la plus efficace repose sur l'étude de séries. En effet un bon moyen de savoir s'il y a "beaucoup" d'entiers d'une certaine forme u_n est de regarder la série $\sum \frac{1}{u_n}$. Par exemple on peut dire qu'il y a peu de carrés parfaits ($u_n = n^2$) par rapport aux entiers ($u_n = n$) car $\sum_{k=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ converge alors que $\sum_{k=1}^{\infty} \frac{1}{n} = \infty$ diverge. C'est avec ce type d'idées qu'on peut démontrer le théorème de raréfaction des nombres premiers, on va donner une idée des éléments qui peuvent conduire à une preuve (longue et complexe) de ce théorème.

Dans la suite on note $\mathcal{P}_n$ l'ensemble des n premiers nombres premier (rappel $p_n > n$). La question fondamentale est donc : *"comment estimer la série $\sum \frac{1}{p_n}$ associée à la suite des nombres premier $(p_n)_{\mathbb{N}}$?"*. Tout repose sur les arguments suivants :

- En utilisant la formule de la série géométrique avec p est un facteur premier puis en faisant le produit sur tous les facteurs premiers p_i plus petit que n on obtient :

  $$\left(1 - \frac{1}{p}\right)^{-1} = \sum_{k=0}^{\infty} \frac{1}{p^k} \Rightarrow \prod_{\substack{p_i\in\mathcal{P} \\ p_i\leq n}} \left(1 - \frac{1}{p_i}\right)^{-1} = \prod_{p_i\in\mathcal{P}_n} \left(\sum_{k=0}^{\infty} \frac{1}{p_i^k}\right) = \sum_{k=0}^{\infty} \frac{1}{\prod p_i^k} \geq \sum_{q=1}^{n} \frac{1}{q}$$

  où $\prod\ p_i^k$ représente tous les produits possibles de facteurs premiers $p_i \in \mathcal{P}_n$ à des exposants k (différents). On a donc dans l'avant dernière somme au moins tous les entiers $q \leq n$ (car le $n^{ième}$ nombre premier est $> n$) ce qui justifie la dernière inégalité. On a donc relié un produit de nombres premiers avec la série harmonique $\sum_{q=1}^{n} \frac{1}{q}$.

- Ensuite la série harmonique peut être encadrée par comparaison avec des intégrales (voir le cours d'analyse de première année) ce qui donne :

$$\sum_{q=1}^{n}\frac{1}{q}\geq\int_1^n\frac{1}{x}dx=\ln(n)\Rightarrow\prod_{p_i\in\mathcal{P}_n}\left(1-\frac{1}{p_i}\right)^{-1}\geq\ln(n)$$

- Les produits peuvent être transformés en somme par passage au logarithme, en tenant compte de la croissance de la fonction ln on obtient :

$$\ln\left(\prod_{p_i\in\mathcal{P}_n}\left(1-\frac{1}{p_i}\right)^{-1}\right)=\sum_{p_i\in\mathcal{P}_n}-\ln\left(1-\frac{1}{p_i}\right)\geq\ln(\ln(n))$$

- Enfin on dispose d'encadrements simples du logarithme népérien

$$\forall x\in[0,+\infty[,\ \ x\leq-\ln(1-x)\leq x+\frac{x^2}{2}$$

en les appliquant à l'inégalité précédente on obtient :

$$\sum_{p_i\in\mathcal{P}_n}\frac{1}{p_i}+\frac{1}{2p_i^2}\geq\sum_{p_i\in\mathcal{P}_n}-\ln\left(1-\frac{1}{p_i}\right)=\ln\left(\prod_{p_i\in\mathcal{P}_n}\left(1-\frac{1}{p_i}\right)^{-1}\right)\geq\ln(\ln(n))$$

- En se rappelant que la série $\sum_{q=1}^{\infty}\frac{1}{q^2}=\frac{\pi^2}{6}$ en sommant seulement sur les $q=p_i$ premiers et plus petits que n (donc sur moins de nombres) on obtient que :

$$0\leq\sum_{p_i\in\mathcal{P}_n}\frac{1}{2p_i^2}\leq\frac{\pi^2}{12}\Rightarrow\sum_{p_i\in\mathcal{P}_n}\frac{1}{p_i}\geq\ln(\ln(n))-\sum_{p_i\in\mathcal{P}_n}\frac{1}{2p_i^2}\geq\ln(\ln(n))-\frac{\pi^2}{12}$$

on peut donc en conclure que la série $\sum_{p\in\mathcal{P}}\frac{1}{p}$ diverge ce qui signifie qu'il y a "beaucoup" de nombres premiers. Si on pousse l'analyse plus loin on peut extraire plus d'informations sur la répartition des nombres premiers. D'une manière similaire (mais plus compliquée!) on peut obtenir que $\sum_{p_i\in\mathcal{P}_n}\frac{1}{p_i}\leq\ln(\ln(n))+C^{ste}$ ce qui conduit à dire que :

$$\sum_{p_i\in\mathcal{P}_n}\frac{1}{p_i}\approx\ln(\ln(n))\quad\ quand\ \ n\to\infty$$

de là on peut déduire heuristiquement que :

$$\frac{1}{p_n}=\sum_{p_i\in\mathcal{P}_n}\frac{1}{p_i}-\sum_{p_i\in\mathcal{P}_{n-1}}\frac{1}{p_i}\approx\ln(\ln(n))-\ln(\ln(n-1))=\ln\left(\frac{\ln(n-1)-\ln\left(1-\frac{1}{n}\right)}{\ln(n-1)}\right)$$

et en utilisant que $\ln(1+x)\approx x$ quand $x\to0$ on peut estimer que :

$$\frac{1}{p_n}\approx\ln\left(1+\frac{1}{n\ln(n-1)}\right)\approx\frac{1}{n\ln(n-1)}\approx\frac{1}{n\ln(n)}$$

ce dernier résultat suggère que le $n^{\text{ième}}$ nombre premier p_n vérifie

$$p_n\approx n\ln(n)\quad\ quand\quad n\to\infty$$

le théorème de raréfaction en découle par de simples comparaisons : on pose $f(x)=x\ln(x)$ alors une brève étude de f montre que f est bijective de $[1,+\infty[$ dans $[0,+\infty[$ car $f'(x)=\ln(x)+1>0$ si $x\geq1$ et $f(1)=0$. En particulier f^{-1} sera croissante. Pour montrer que $f^{-1}(y)\approx\frac{y}{\ln(y)}$ quand $y\to\infty$ il suffit d'encadrer convenablement $f(x)$ puis d'appliquer f^{-1} à cette inégalité. La première est assez simple à obtenir :

$$f\left(\frac{y}{\ln(y)}\right)=\frac{y}{\ln(y)}\times(\ln(y)-\ln(\ln(y)))=y\left(1-\frac{\ln(\ln(y))}{\ln(y)}\right)<y$$

donc $\frac{y}{\ln(y)}<f^{-1}(y)$. Pour l'autre sens il faut un peut plus de travail :

$$\begin{aligned}f\left(\frac{y}{\ln(y)}\times\frac{1}{1-\frac{\ln(\ln(y))}{\ln(y)}}\right)&=\frac{y}{\ln(y)}\times\frac{1}{1-\frac{\ln(\ln(y))}{\ln(y)}}\\&\quad\times\left(\ln(y)-\ln(\ln(y))-\ln\left(1-\frac{\ln(\ln(y))}{\ln(y)}\right)\right)\\&=y\times\left(1-\frac{\ln\left(1-\frac{\ln(\ln(y))}{\ln(y)}\right)}{\ln(y)-\ln(\ln(y))}\right)>y\end{aligned}$$

donc $\frac{y}{\ln(y)}\times\frac{1}{1-\frac{\ln(\ln(y))}{\ln(y)}}>f^{-1}(y)$. Au final on a bien

$$\frac{y}{\ln(y)}\times\frac{1}{1-\frac{\ln(\ln(y))}{\ln(y)}}>f^{-1}(y)>\frac{y}{\ln(y)}\Longrightarrow f^{-1}(y)\approx\frac{y}{\ln(y)}$$

donc si on sait que $p_n\approx n\ln(n)=f(n)$ alors $f^{-1}(p_n)\approx n=$ le nombre de nombres premier plus petits que p_n donc $f^{-1}(n)\approx\frac{n}{\ln(n)}\approx$le nombre de nombres premier plus petits que n.

□

Le théorème de raréfaction a été conjecturé par Gauss au début du $XIX^{\text{ième}}$ siècle mais sa démonstration rigoureuse n'a été obtenue qu'à la fin du $XIX^{\text{ième}}$ siècle (par De La Vallée-Poussin et Hadamard en 1896). Il n'est pas possible d'en donner ici une démonstration simple cependant il est nécessaire de bien comprendre toutes les implications de ce théorème en vu des applications que nous en ferons. Le théorème 1.2.14 exprime qu'il est difficile de trouver (en cherchant au hasard) de grand nombres premiers. Le tableau ci-dessous donne la probabilité $\mathbb{P}(a_n\in\mathcal{P})=$ de trouver un nombre premier en tirant au hasard un nombre $a_n\approx n$:

n	100	1000	10000	...	10^8	10^{12}
$\pi(n)$	25	168	1229	...	5 761 455	$3.76\ 10^{10}$
$\frac{n}{\ln(n)}$	21.714724	144.76483	1085.7362	...	5 428 681	$3.619\ 10^{10}$
$\mathbb{P}(a_n\in\mathcal{P})\approx\frac{1}{\ln(n)}$	25%	16.8%	12.3%	...	5.7%	3.7%

Ce fait est d'autant plus contraignant, quand on cherche à générer de grands nombres premiers, qu'il n'existe pas de formule permettant de générer le $n^{\text{ième}}$ nombre premier ou même seulement un nombre premier d'une taille donnée. Pour

se faire une idée de la faiblesse de l'approximation du $n^{\text{ième}}$ nombre premier donnée par le théorème 1.2.14 on peut se référer au tableau suivant dans lequel $p_n = n^{\text{ième}}$ nombre premier et $p'_n = n\ln(n)+n(\ln(\ln(n))-1)$ qui est une approximation un peu plus fine que celle du théorème 1.2.14 (mais aussi peu utile dans la recherche exacte de nombres premiers) :

n	p_n	$n\ln(n)$	p'_n
500	3571	3107.304	3520.7554
1000	7919	6907.7553	7840.4
1500	12553	10969.831	12454.356
2000	17389	15201.805	17258.339
2500	22307	19560.115	22203.12
3000	27449	24019.103	27259.814
3500	32609	28561.814	32409.391
4000	37813	33176.199	37638.352
4500	43051	37853.247	42936.624
5000	48611	42585.966	48296.4

Il n'existe donc aucune formule permettant de calculer directement le $n^{ième}$ nombre premier ou même seulement un nombre premier aussi grand que l'on veut ... pour établir la liste de tous les nombres premiers (inférieur à un certain seuil) on ne peut utiliser que l'algorithme d'Ératosthène :

Le crible d'Ératosthène

- $L = (2,3,\ldots,n)$
- $k = 2$

tant que $k \leq \sqrt{n}$ **faire**
éliminer tous les multiples de k de L;
$k :=$ le premier chiffre supérieur a k restant dans L;
fin faire

remarque: si k est le premier chiffre non souligné et non barré son premier multiple non déjà barré est k^2, donc si $k > \sqrt{n}$ tous les chiffres restant non barrés doivent être souligné et l'algorithme est terminé.

Le théorème suivant explique pour quoi l'algorithme d'Ératosthène est inutilisable pour rechercher des nombres premiers de grande taille (plusieurs milliers de chiffres).

Théorème 1.2.15 *Le calcul de la liste des nombres premiers plus petits que n par l'algorithme d'Ératosthène nécessite un temps de l'ordre de* $n\ln(\ln(n))$.

Preuve : Il faut calculer le nombre d'opérations engendré par l'appel de l'algorithme d'Ératosthène avec le paramètre n. L'ordre de grandeur de ce nombre est donné par la série $\sum_{k=1}^{\sqrt{n}} f(k)$ où $f(k)$ est le nombre d'opérations effectuées lors d'une étape de la boucle **tant que** . Lors de cette étape il y a n/k multiples de k inférieurs

à n à éliminer donc le nombre d'opérations est

$$\sum_{k=1}^{\sqrt{n}} \frac{n}{k} \approx n\ln(\sqrt{n}) = \frac{1}{2}n\ln(n)$$

une analyse plus fine montre qu'il n'y a d'élimination que si k est premier, donc en utilisant le théorème de raréfaction des nombres premiers on a :

$$\sum_{\substack{1\leq k \\ k\ln(k)\leq\sqrt{n}}} \frac{n}{k\ln(k)} \sum_{k=1}^{\frac{\sqrt{n}}{\ln(\sqrt{n})}} \frac{n}{k\ln(k)} \approx n\ln\left(\ln\left(\frac{\sqrt{n}}{\ln(\sqrt{n})}\right)\right) \approx n\ln(\ln(n))$$

□

Pour bien comprendre la démonstration précédente il est utile de tester l'algorithme d'Ératosthène pour de petites valeurs de n. Par exemple pour $n = 40$ on verra qu'il suffit d'appliquer l'algorithme jusqu'à $k > \sqrt{40} = 6.324\cdots > 6$ pour avoir la liste des nombres premiers et qu'on a des éliminations seulement pour $k = 2$,3 et 5.

initialisation

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 27
28 29 30 31 32 33 34 35 36 37 38 39 40

étape 1, k=2

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~~ 27
~~28~~ 29 ~~30~~ 31 ~~32~~ 33 ~~34~~ 35 ~~36~~ 37 ~~38~~ 39 ~~40~~

étape 2, k=3

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~~ ~~27~~
~~28~~ 29 ~~30~~ 31 ~~32~~ ~~33~~ ~~34~~ 35 ~~36~~ 37 ~~38~~ ~~39~~ ~~40~~

étape 3, k=5

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~~ ~~27~~
~~28~~ 29 ~~30~~ 31 ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ 37 ~~38~~ ~~39~~ ~~40~~

fin, k=7

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~~ ~~27~~
~~28~~ 29 ~~30~~ 31 ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ 37 ~~38~~ ~~39~~ ~~40~~

la liste des nombres premiers plus petits que 40 est donc :

2,3,5,7,11,13,17,19,23,29,31,37

à défaut de pouvoir trouver tous les nombres premiers on verra qu'il existe certains nombres d'une forme particulière dont la primalité est plus facile à tester.

Un autre résultat surprenant sur les nombres premiers est le suivant :

Théorème 1.2.16 *La probabilité que deux nombres a et b choisis au hasard soient premiers entre eux est* $\frac{6}{\pi^2} \approx 60.8\%$.

Là encore il n'est pas possible de donner une démonstration rigoureuse de ce résultat dans ce cours mais on peut en donner une euristique convaincante

Preuve : Il faut d'abord définir ce qu'est cette probabilité. Choisir au hasard des nombres entiers signifie qu'on les tire au hasard avec une loi uniforme, mais cela n'a de sens que dans un ensemble fini comme l'intervalle $[1,n]$. On va donc poser que la probabilité que l'on cherche est la limite quand $n \to \infty$ des probabilités de tirer au hasard dans $[1,n]$ deux nombres premiers entre eux :

$$\lim_{n\to\infty} \mathbb{P}\left(\text{PGCD}(a,b) = 1 | a,b \leq n\right)$$

Plus généralement on va considérer $p_k = \mathbb{P}(\text{PGCD}(a,b) = k)$ pour $k \in [1,n]$ alors on a que la somme de ces probabilités est 1 :

$$p_1 + p_2 + \cdots + p_n + \cdots = 1$$

Maintenant si $\text{PGCD}(a,b) = d$ alors a et b sont des multiples de d donc $a = kd$ et $b = ld$. Si on admet que les 3 événements $\text{PGCD}(a,b) = d$ $a = kd$ et $b = ld$ sont indépendants[1] alors on obtient que

$$\begin{aligned} \mathbb{P}(\text{PGCD}(a,b) = d) &= \mathbb{P}([a = kd] et [b = ld] et [\text{PGCD}(k,l) = 1]) \\ &= \mathbb{P}(a = kd) \times \mathbb{P}(b = ld) \times \text{PGCD}(k,l) = 1) \end{aligned}$$

or dans la division Euclidienne par d le nombre de multiples de d plus petit que n est la partie entière de n/d et donc

$$\mathbb{P}(a = kd) = \mathbb{P}(b = ld) = \frac{E\left(\frac{n}{d}\right)}{n} \approx \frac{1}{d}$$

et donc $\mathbb{P}(\text{PGCD}(a,b) = d) = \mathbb{P}([a = kd] et [b = ld] et [\text{PGCD}(k,l) = 1]) \approx \frac{p_1}{d^2}$ donc en utilisant que la somme des p_k vaut 1 on obtient :

$$p_1 \left(\frac{1}{1^2} + \frac{1}{1^2} + \cdots + \frac{1}{n^2}\right) \approx 1 \Longrightarrow p_1 \approx \frac{1}{\sum_{n=1}^{\infty} \frac{1}{n^2}}$$

quand $n \to \infty$, et comme $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ on a bien que $p_1 \approx \mathbb{P}(\text{PGCD}(a,b) = 1) = \frac{6}{\pi^2}$ □

1. ce qui est faux!!! En fait la démonstration ci-dessus n'est pas correcte, mais les "vraies" démonstrations de ce résultat sont très difficiles.

Les nombres de Mersenne et la recherche de grands nombres premiers sur ordinateurs *Marin Mersenne(1588-1648)* étudia les nombres premiers en particuliers ceux de la forme

$$M_p = 2^p - 1, \ \ p \in \mathbb{N}^*$$

Certains de ces nombres sont premiers

$$M_2 = 3, M_3 = 7, M_5 = 31, \ldots M_{127} = 1.701.10^{38}$$

mais pas tous

$$M_{11} = 2047 = 23 \times 89.$$

Mersenne trouva "presque" tous les M_p premiers jusqu'à $p = 257$ en ne faisant que des calculs à la main! En fait il existe des méthodes pour tester la primalité des nombres de la forme M_p sans tester tous les diviseurs inférieurs à M_p. Ces méthodes peuvent être facilement programmées mais demandent beaucoup de temps pour être exécutées. C'est pour cette raison que George Woltman (informaticien américain à la retraite) à imaginé de distribuer (via internet) les calculs à des volontaires acceptant de donner du temps de calcul de leur(s) machine(s) personnelle(s) (lorsqu'elles sont inutilisées) : c'est le projet GIMPS (*Great Internet Mersenne Prime Search*)[5]. C'est de cette manière qu'en 1998 un étudiant de 19 ans Roland Clarkson a découvert le plus grand nombre premier connu à cette date :

$$M_{6\ 972\ 593} \approx 4\ 10^{2098959}$$

le dernier en date a été découvert le 6 septembre 2008 par Edson Smith (ou plutôt les machines du département de mathématique de UCLA!) il s'agit du $45^{\text{ième}}$ nombre de Mersenne premier connu : $2^{43\ 112\ 609} - 1$! Il possède 12 978 189 décimales.

Vous pouvez aussi visiter le site de l'EFF (*Electronic Frontier Foundation* voir [4]), qui offre des récompenses pour la découverte des plus gros nombres premiers. C'est ainsi que 100 000 dollars ont été offerts aux découvreurs du premier nombre premier à plus de 10 millions de chiffres ...

1.3 Congruences

1.3.1 Calcul modulo n

Définition 1.3.1 *Soient trois entiers a,b,n, on dit que a et b sont congrus modulo n (que l'on note $a \equiv b \mod n$) si et seulement si a et b ont le même reste dans la division Euclidienne par n :*

$$a \equiv b \mod n \Longleftrightarrow n|a-b \Longleftrightarrow \exists k \in \mathbb{Z},\ \ a = b + kn$$

Exemple 1.3.2 pour un a et un n fixés on est pas obligé de prendre b égal au reste de a dans la division par n, par exemple :

$$29 \equiv 3 \equiv 16 \equiv -10 \mod 13$$

suivant le cas il sera plus pratique de prendre -10 que 3. Notez aussi qu'il est très pratique de pouvoir écrire plusieurs "congruences" ($\equiv$) sur une même ligne avec un seul "modulo" (mod) en fin de ligne. Pour simplifier il faut bien retenir que le mod ne fait pas partie des nombres qui apparaissent dans les congruences. En particulier pour désigner le carré du reste de a^2 modulo n ne jamais écrire $(a \mod n)^2$!!!

Les règles de calculs avec les principales opérations arithmétiques dans des congruences sont les mêmes que pour les calculs dans $\mathbb{Z}$.

Proposition 1.3.3
Soient cinq entiers a,b,c,d,n $\in \mathbb{Z}$ et e $\in \mathbb{N}$, on a les règles de calcul suivantes :

- $a \equiv b \mod n$ et $c \equiv d \mod n \Longrightarrow a + c \equiv b + d \mod n$
- $a \equiv b \mod n$ et $c \equiv d \mod n \Longrightarrow a \times c \equiv b \times d \mod n$
- $a \equiv b \mod n \Longrightarrow a^e \equiv b^e \mod n$

Preuve : Les bases du calcul modulo n reposent sur les 3 règles de la proposition précédente que l'on peut très facilement démontrer. Commençons par l'addition et la multiplication :

a	$=$	$b+kn$	a	$=$	$b+kn$
c	$=$	$d+k'n$	c	$=$	$d+k'n$
$a+b$	$=$	$b+d+(k+k')n$	ac	$=$	$bd+(bk'+dk+kk')n$

pour les puissances prenons l'exemple $e = 2$:

$$a^2 = (b+kn)^2 = b^2 + 2bkn + k^2n^2 = b^2 + (2bk + k^2n)n$$

le cas général se démontre en utilisant la formule du binôme de Newton :

$$\begin{aligned} a^e &= (b+kn)^e \\ &= \sum_{p=0}^{e} C_e^p b^{e-p}(kn)^p \\ &= b^e + \sum_{p=1}^{e} C_e^p b^{e-p}(kn)^p \\ &= b^e + nk\sum_{p=1}^{e} C_e^p b^{e-p}(kn)^{p-1} \\ &= b^e + nk' \end{aligned}$$

□

Exemple 1.3.4 Pour comprendre comment s'utilisent les règles de calculs de la proposition précédente essayons de résoudre une équation toute simple :

$$15x + 24 \equiv 17 \mod 29$$

Comme on peut utiliser les nombres négatifs dans des congruences il est très facile de se débarrasser du 24 dans le membre de gauche :

$$15x \equiv 17 - 24 \equiv -7 \equiv 22 \mod 29$$

pour éliminer le 15 qui se trouve devant le x on ne peut pas diviser par 15! Il faut donc chercher un u tel que $15 \times u \equiv 1 \mod 29$, c'est possible en cherchant une identité de Bezout :

$$\begin{aligned} 29 &= 15 \times 1 + 14 \\ 15 &= 14 \times 1 + 1 \\ 14 &= 1 \times 14 + 0 \end{aligned}$$

donc

$$15 \times 2 + 29 \times (-1) = 1 \Longrightarrow 15 \times 2 \equiv 1 \mod 29$$

Donc en multipliant par 2 l'équation on obtient

$$\begin{aligned} 15x &\equiv 22 \mod 29 \\ 2 \times 15x &\equiv 2 \times 22 \mod 29 \\ 1 \times x &\equiv 44 \mod 29 \\ x &\equiv 15 \mod 29 \end{aligned}$$

Et les solutions sont de la forme $x = 15 + 29k$ avec un paramètre $k \in \mathbb{Z}$.

Il faudra bien retenir cette technique très importante pour le calcul de l'inverse modulo n

Lemme 1.3.5 (inverse modulo n)
si a et n sont premiers entre eux alors il existe $u \in \mathbb{Z}$ tel que $a \times u \equiv 1 \mod n$

Preuve : si a et n sont premiers entre eux alors on peut trouver une identité de Bezout

$$\exists u,v \in \mathbb{Z}, \quad au + vn = 1 \Longrightarrow au = 1 - vn \Longrightarrow au \equiv 1 \mod n$$

□

Le fait que $a \equiv b \mod n \Longrightarrow a^e \equiv b^e \mod n$ est très important, ce type de calcul apparaît partout dans le crypto-système RSA, et pour des nombres a,b,e

de très grande taille. Nous avons donc besoin d'étudier comment calculer de telles puissances pour des nombres assez grands.

Lemme 1.3.6 *pour tout a,b,c,n* $\in \mathbb{Z}$ *on a :*

$$n^{a+b} = n^a \times n^b,\ \ n^{a\times b} = (n^a)^b,\ \ n^{a\times b+c} = (n^a)^b \times n^c$$

Exemple 1.3.7 ce lemme est très utile pour le calcul de puissances modulo n. Calculons par exemple 153^{100} mod 29. D'abord, plutôt que de calculer les puissances de 153, il vaut mieux calculer les puissances de son reste mod 29 qui sera un nombre plus petit :

$$153 = 5 \times 29 + 8 \Longrightarrow 153^{100} \equiv 8^{100} \equiv (2^3)^{100} \equiv 2^{300} \bmod 29$$

maintenant calculons les premières puissances de 2 modulo 29 :

$$2^3 = 8,\ \ 2^4 = 16,\ \ 2^5 = 32 \equiv 3 \bmod 29$$

donc on peut réduire l'exposant en le divisant par 5 et chaque 2^5 sera remplacé par 3 (modulo 29) :

$$300 = 5 \times 60 \Longrightarrow 153^{100} \equiv 2^{5\times 60} \equiv (2^5)^{60} \equiv 3^{60} \bmod 29$$

on recommence ensuite avec 3 :

$$3^3 = 27 \equiv -2 \bmod 29 \Longrightarrow 153^{100} \equiv 3^{3\times 20} \equiv (3^3)^{20} \equiv (-2)^{20} \equiv 2^{20} \bmod 29$$

ainsi de suite on va réduire l'exposant jusqu'à arriver à quelquechose de calculable de tête :

$$153^{100} \equiv 2^{5\times 4} \equiv (2^5)^4 \equiv 3^4 \equiv 3^3 \times 3 \equiv -2 \times 3 \equiv -6 \equiv 23 \bmod 29$$

ceci permet de calculer de tête le reste de 153^{100} mais cette méthode est aussi très importante du point de vue algorithmique. En effet pour calculer le reste de 153^{100} modulo 29 cette méthode est beaucoup plus efficace que celle qui consisterait à calculer d'abord le développement de 153^{100} (218 chiffres!) puis d'en faire la division par 29. Nous verrons que c'est d'une manière très similaire que sont calculés les restes de puissances modulo n dans le crypto-système RSA (algorithme d'exponentiation modulaire rapide).

Pour simplifier des puissances élevées nous avons dû chercher (au hasard) une relation donnant une simplification de base ($2^5 \equiv 3 \bmod 29$ et $3^3 \equiv -2 \bmod 29$). Le théorème suivant, dû à Fermat, permet de trouver de telles relations pour simplifier a^e mod n dans le cas où a et n sont premiers entre eux.

Théorème 1.3.8 (petit théorème de Fermat)
Soient a et n deux entiers premiers entre eux alors

$$a^{\Phi(n)} \equiv 1 \bmod n$$

où Φ *est la fonction d'Euler. En particulier si n est premier alors*

$$a^{n-1} \equiv 1 \bmod n,\ \ \forall a \in [1,n-1].$$

ou encore

$$a^n \equiv a \bmod n,\ \ \forall a \in \mathbb{Z}.$$

Preuve : On peut immédiatement remarquer que si $PGCD(a,n) = 1$ et $\mathrm{PGCD}(x,n) = 1$ alors $\mathrm{PGCD}(a \times x,n) = 1$. Ceci signifie que la fonction

$$\begin{array}{rccl} f: & E & \longrightarrow & E = \{a_1, \ldots a_{\Phi(n)}\} \\ & x & \longmapsto & ax \end{array}$$

ne prend ses valeurs que parmi les $\Phi(n)$ entiers premiers plus petits que n (cet ensemble est noté E et $a_1 = a \in E$). Chacun de ces entiers a_i étant premier avec n on peut leur trouver un inverse modulo n, comme dit dans le lemme 1.3.5, à partir d'identités de Bezout

$$\mathrm{PGCD}(a_i,n) = 1 \Longleftrightarrow \exists u_i \in \mathbb{Z},\ \ a_i u_i + kn = 1 \Longrightarrow a_i u_i \equiv 1 \bmod n$$

d'où on obtient que

$$a \times a_i \equiv a \times a_j \bmod n \Longrightarrow u_1 a \times a_i \equiv u_1 a \times a_j \bmod n \Longrightarrow a_i \equiv a_j \bmod n$$

c'est à dire que f permute les valeurs de l'ensemble E. Maintenant le résultat s'obtient en faisant le produit des valeurs $f(a_i)$:

$$\prod_{i=1}^{\Phi(n)} a_i = \prod_{i=1}^{\Phi(n)} f(a_i) \Longrightarrow \prod_{i=1}^{\Phi(n)} a \times a_i = a^{\Phi(n)} \prod_{i=1}^{\Phi(n)} a_i \equiv \prod_{i=1}^{\Phi(n)} a_i \bmod n$$

il reste à multiplier les deux membres par $\prod_{i=1}^{\Phi(n)} u_i$ pour éliminer les facteurs $\prod_{i=1}^{\Phi(n)} a_i$ ensuite il reste $a^{\Phi(n)} \equiv 1 \bmod n$ □

Exemple 1.3.9 On peut facilement vérifier le théorème de Fermat pour de petites valeurs de n, par exemple dans le cas $n = 7$:

a	a^2	a^3	a^4	a^5	a^6
1	1	1	1	1	1
2	4	1	2	4	1
3	2	6	4	5	1
4	2	1	4	2	1
5	4	6	2	3	1
6	1	6	1	6	1

Mais l'intérêt du théorème est justement de l'utiliser pour de grandes valeurs de n, pour simplifier des calculs de puissances. Reprenons le calcul de 153^{100} mod 29 comme 153 et 29 sont premier entre eux et que $\Phi(29) = 28$ on peut dire que $153^{28} \equiv 1 \bmod n$ ce qui permet de réduire déjà pas mal le calcul :

$$153^{100} \equiv 153^{28\times 3+16} \equiv (153^{28})^3 \times 153^{16} \equiv (1)^3 \times 153^{16} \equiv (2^3)^{16} \equiv 2^{48} \bmod 29$$

On peut alors recommencer avec 2, comme 2 et 29 sont premiers entre eux :

$$2^{48} \equiv 2^{28} \times 2^{20} \equiv 1 \times 2^{20} \equiv 2^{20} \bmod 29$$

Calculer 2^{20} de tête est envisageable et, se faisant, on tombera rapidement sur les simplifications déjà vues :

$$2^{20} = (2^{10})^2 = 1024^2 \equiv (9)^2 \equiv 81 \equiv 23 \bmod 29$$

Autre exemple le calcul de $17^{1717} \bmod 10$ se ramène à :

- $\Phi(10) = 10 \times (1-1/2) \times (1-1/5) = 4$
- $17^4 \equiv 1 \bmod 10$ (vérifier)
- $1717 = 4 \times 429 + 1 \Longrightarrow 17^{1717} = (17^4)^{429} \times 17 \equiv 17 \equiv 7 \bmod 10$

La réciproque du théorème de Fermat n'est pas vraie, mais le théorème suivant permet de résoudre partiellement le problème.

Théorème 1.3.10 (de l'ordre) *Soient a et n deux entiers premiers entre eux. On appelle $k_a \in \mathbb{N}^*$ le plus petit entier tel que*

$$a^{k_a} \equiv 1 \bmod n$$

alors

$$k_a | \Phi(n)$$

et

$$a^k \equiv 1 \bmod n \Longrightarrow k_a | k$$

Preuve : si $a^k \equiv 1 \bmod n$ et que $k_a \nmid k$ alors soit r le reste de la division Euclidienne de k par k_a :

$$k = k_a \times q + r \quad \text{avec} \quad 0 \leq r < k_a \Longrightarrow 1 \equiv a^k = (a^{k_a})^q \times a^r \equiv a^r \bmod n$$

donc r est une solution plus petite que k_a ↯. □

Le petit théorème de Fermat est la source d'un test de non-primalité très efficace : le test de Fermat. L'idée de Pierre de Fermat est que pour un nombre $n \geq 3$ composé on doit trouver un a tel que le petit théorème de Fermat soit mis en défaut :

$$\boxed{n \geq 3 \ \text{ premier} \quad \Longrightarrow a^{n-1} \equiv 1 \bmod n, \quad \forall a \in \mathbb{N}^*, \ 2 \leq a < n.}$$

$$\Downarrow$$

Test de Fermat

procédure *fermat*(n)
choisir un entier a compris entre 2 et $n-1$
calculer $x = a^{n-1} \bmod n$
si $x \neq 1$ **alors** n est composé
sinon n est *probablement* premier
fin si

Un nombre n qui passe (victorieusement) le test de Fermat pour un a donné est dit *a-pseudo-premier*. Le test de Fermat est un test de non-primalité c'est à dire que si le test renvoie faux on est sûr que n est composé alors que s'il renvoie vrai n peut être composé ... mais un nombre qui passe le test de Fermat plusieurs a différents aura de moins en moins de chances d'être composé comme le montre le résultat suivant.

Lemme 1.3.11 *Soient les événements : $F_{n,k}$ = "le nombre n passe k fois le test de Fermat" et P_n = "le nombre n est premier" alors la probabilité que n soit premier s'il passe k fois le test de Fermat vérifie :*

$$\mathbb{P}(P_n | F_{k,n}) \geq 1 - \ln(n) \left(1 - \frac{6}{\pi^2}\right)^k$$

Preuve : On rappelle que le choix de a peut être considéré comme le tirage d'un nombre au hasard entre 1 et n, donc la probabilité que a et n soit premiers entre eux est

$$\mathbb{P}(A_{a,n}) = \frac{6}{\pi^2} \approx 60{,}8\% \text{ avec } A_{a,n} = \text{“ } a \text{ et } n \text{ sont premiers entre eux”}$$

si n passe le test de Fermat pour un certain a alors a et n doivent être premiers entre eux car :

$$a^{n-1} \equiv 1 \bmod n \Longleftrightarrow \exists k \in \mathbb{Z}, \ \ a \times a^{n-2} - n \times k = 1 \Longrightarrow \text{PGCD}(a,n) = 1$$

donc si n n'est pas premier il ne peut passer le test de Fermat que si le a choisit est au moins premier avec lui. Si l'on admet que les choix des différents a sont indépendants entre eux, on obtient que

$$\mathbb{P}(F_{k,n} | \overline{P_n}) \leq \mathbb{P}(A_{a,n})^k \leq \left(1 - \frac{6}{\pi^2}\right)^k$$

la probabilité qu'un nombre composé passe k fois le test de Fermat vérifie donc :

$$\mathbb{P}(\overline{P_n} \cap F_{n,k}) = \mathbb{P}(F_{n,k} | \overline{P_n}) \mathbb{P}(\overline{P_n}) \leq \left(1 - \frac{6}{\pi^2}\right)^k \mathbb{P}(\overline{P_n})$$

d'un autre coté si n est premier il passe forcément le test de Fermat donc :

$$\mathbb{P}(F_{n,k} | P_n) = 1 \text{ et } \mathbb{P}(P_n \cap F_{n,k}) = \mathbb{P}(F_{n,k} | P_n) \mathbb{P}(P_n) = \mathbb{P}(P_n)$$

ce qui permet d'évaluer la probabilité qu'un nombre n quelconque (premier ou pas) passe k fois le test de Fermat :

$$\mathbb{P}(F_{n,k}) = \mathbb{P}(P_n \cap F_{n,k}) + \mathbb{P}(\overline{P_n} \cap F_{n,k}) \leq \mathbb{P}(P_n) + \mathbb{P}(\overline{P_n}) \left(1 - \frac{6}{\pi^2}\right)^k$$

on en déduit la probabilité qu'un nombre n qui passe k fois le test de Fermat soit premier est :

$$\mathbb{P}(P_n | F_{n,k}) = \frac{\mathbb{P}(P_n \cap F_{n,k})}{\mathbb{P}(F_{n,k})} \geq \frac{\mathbb{P}(P_n)}{\mathbb{P}(P_n) + \mathbb{P}(\overline{P_n}) \left(1 - \frac{6}{\pi^2}\right)^k} = \frac{1}{1 + \frac{\mathbb{P}(\overline{P_n})}{\mathbb{P}(P_n)} \left(1 - \frac{6}{\pi^2}\right)^k}$$

comme pour $u \geq 0$ on a $\frac{1}{1+u} \geq 1 - u$ et $\mathbb{P}(P_n) \approx 1/\ln(n)$ on peut dire que :

$$\mathbb{P}(P_n | F_{k,n}) \geq f(n) \approx 1 - \ln(n) \left(1 - \frac{6}{\pi^2}\right)^k$$

Comme $f(n) \to 0$ quand k tends vers ∞ plus un nombre passe de tests de Fermat plus la probabilité qu'il soit premier sera grande. □

1.3.2 Le Théorème Chinois

Pour comprendre le fonctionnement de plusieurs crypto-systèmes nous aurons besoin de résoudre des systèmes d'équations aux congruence. Dans le cas général, ce type de problème est très compliqué à résoudre. Nous allons seulement nous intéresser aux cas les plus simples. La théorie de ces systèmes d'équations repose sur le Théorème Chinois :

Théorème 1.3.12 (Théorème Chinois) *Soient* $m_1,m_2,\dots m_k \in \mathbb{N}^*$ *et* $a_1,a_2,\dots a_k \in \mathbb{Z}$ *et le système d'équations :*

$$(\mathcal{E}) \Longleftrightarrow \begin{cases} x & \equiv & a_1 \bmod m_1 \\ x & \equiv & a_2 \bmod m_2 \\ \vdots & \equiv & \vdots \\ x & \equiv & a_k \bmod m_k \end{cases}$$

- *si* $(\mathcal{E})$ *possède une solution* x_0 *alors il en possède une infinité données par la formule :*

  $$x = x_0 + k \times \text{PPCM}(m_1,m_2,\dots m_k) \Longleftrightarrow x \equiv x_0 \bmod \text{PPCM}(m_1,m_2,\dots,m_k)$$

- *si* $m_1,m_2,\dots m_k$ *sont premiers entre eux (*$\text{PGCD}(m_1,m_2,\dots,m_k) = 1$*) alors il existe une solution* x_0*, cette solution s'obtient à partir d'une identité de Bezout entre les* $M_i = \prod_{i\neq j} m_j$ *par*

  $$\sum_{i=1}^{k} u_i M_i = 1 \Longrightarrow x_0 = \sum_{i=1}^{k} a_i u_i M_i$$

Preuve : La première assertion est très simple. Dès qu'on a une solution particulière x_0 on trouve facilement toutes les autres puisque

$$\begin{cases} x - x_0 & \equiv & 0 \bmod m_1 \\ x - x_0 & \equiv & 0 \bmod m_2 \\ \vdots & \equiv & \vdots \\ x - x_0 & \equiv & 0 \bmod m_k \end{cases}$$

donc $x - x_0$ est un multiple du $\text{PPCM}(m_1,\dots,m_k) = m_1 \times \cdots \times m_k$. Ensuite il faut trouver la solution particulière dans le cas où les m_i sont premiers entre eux. Dans ce cas les M_i sont aussi premiers entre eux donc on doit avoir une identité de Bezout $u_1M_1 + u_2M_2 + \dots u_kM_k = 1$ en particulier

$$\begin{array}{rcll} 1 & \equiv & u_1M_1 \bmod m_1 & \text{car seul } M_1 \text{ n'est pas divisible par } m_1 \\ & \equiv & u_2M_2 \bmod m_2 & \text{car seul } M_2 \text{ n'est pas divisible par } m_2 \\ & & \dots & \\ & \equiv & u_kM_k \bmod m_k & \text{car seul } M_k \text{ n'est pas divisible par } m_k \end{array}$$

donc $x_0 = a_1u_1M_1 + a_2u_2M_2 + \dots a_ku_kM_k$ est bien une solution particulière car

$$\begin{array}{rcll} x_0 & \equiv & a_1u_1M_1 \equiv a_1 \bmod m_1 & \text{car seul } M_1 \text{ n'est pas divisible par } m_1 \\ & \equiv & a_2u_2M_2 \equiv a_2 \bmod m_2 & \text{car seul } M_2 \text{ n'est pas divisible par } m_2 \\ & & \dots & \\ & \equiv & a_ku_kM_k \equiv a_k \bmod m_k & \text{car seul } M_k \text{ n'est pas divisible par } m_k \end{array}$$

□

Regardons ce que donne concrètement le théorème Chinois dans le cas d'un système de 2 équations :

Exemple 1.3.13 Résoudre

$$\begin{array}{rcl} x & \equiv & 5 \bmod 17 \\ x & \equiv & 8 \bmod 29 \end{array}$$

il faut résoudre

$$17 \times k_1 - 29 \times k_2 = 3$$

on en déduit une identité de Bezout :

$$17 \times 12 + 29 \times (-7) = 1$$

comme $1|3$ il y a des solutions en multipliant par 3 on obtient une solution

$$17 \times 36 + 29 \times (-21) = 3$$

donc $x_0 = 36 \times 17 + 5 = 21 \times 29 + 8 = 617$ et

$$x_0 \equiv 617 \equiv 124 \bmod 493$$

La méthode de résolution repose donc sur celle vue au premier chapitre. On peut la résumer ainsi :

résolution du système de deux congruences

$$(\mathcal{E}) \Leftrightarrow \left\{ \begin{array}{rcl} x & \equiv & a_1 \bmod m_1 \\ x & \equiv & a_2 \bmod m_2 \end{array} \right.$$

• **Trouver toutes les solutions :** On peut déduire TOUTES les solutions à partir d'une solution particulière x_0 en remarquant que

$$(\mathcal{E}) \Longleftrightarrow \left\{ \begin{array}{rcl} x & = & x_0 \bmod m_1 \\ x & = & x_0 \bmod m_2 \end{array} \right. \Longleftrightarrow \left\{ \begin{array}{rcl} x - x_0 & = & 0 \bmod m_1 \\ x - x_0 & = & 0 \bmod m_2 \end{array} \right.$$

donc $x - x_0$ est divisible par PPCM(m_1,m_2) et

$$x = x_0 + k\text{PPCM}(m_1,m_2), \quad k \in \mathbb{Z}$$

• **Trouver une solution particulière :**
traduire les congruences en égalités dans $\mathbb{Z}$:

$$(\mathcal{E}) \Longleftrightarrow \left\{ \begin{array}{rcll} x & = & a_1 + m_1 \times k_1 & k_1 \in \mathbb{Z} \\ x & = & a_2 + m_2 \times k_2 & k_2 \in \mathbb{Z} \end{array} \right.$$

en éliminant x entre les deux équations on obtient une équation Diophantienne

$$a_1 + m_1 \times k_1 = x = a_2 + m_2 \times k_2 \Rightarrow m_1 \times k_1 + m_2 \times (-k_2) = a_2 - a_1$$

on est ramené à chercher les solutions $(x_1,x_2) \in \mathbb{Z}^2$ de

$$m_1 x_1 + m_2 x_2 = a_2 - a_1$$

- si PGCD(m_1,m_2) $\nmid a_1 - a_2$ alors le système n'a pas de solution :
- sinon le système à des solutions pour les trouver :
  - on cherche une identité de Bezout pour m_1 et m_2 :

$$m_1 \times u_1 + m_2 \times u_2 = \text{PGCD}(m_1,m_2)$$

  - en multipliant par $(a_2 - a_1)/\text{PGCD}(m_1,m_2)$ on obtient une égalité:

$$x_1 \times m_1 + x_2 \times m_2 = a_2 - a_1$$

$$\left\{ \begin{array}{rcl} k_1 & = & x_1 = u_1 \times (a_2 - a_1)/\text{PGCD}(m_1,m_2), \\ k_2 & = & -x_2 = -u_2 \times (a_2 - a_1)/\text{PGCD}(m_1,m_2) \end{array} \right.$$

  - on obtient la solution particulière en écrivant :

$$x_0 = a_1 + m_1 \times k_1 = a_2 + m_2 \times k_2$$

On peut étendre cette méthode pour résoudre des systèmes de plus de deux équations mais cela devient vite très compliqué. Dans le cas général où les m_i ne sont pas premiers entre eux l'existence d'une solution particulière n'est pas assurée,

si on prend deux équations du système

$$\left\{ \begin{array}{rcl} x & \equiv & a_i \bmod m_i \\ x & \equiv & a_j \bmod m_j \end{array} \right. \Longleftrightarrow \left\{ \begin{array}{rcll} x & = & a_i + m_i \times k_i & k_i \in \mathbb{Z} \\ x & = & a_j + m_j \times k_j & k_j \in \mathbb{Z} \end{array} \right.$$

on doit avoir que $m_j \times k_j - m_i \times k_i = a_i - a_j$ et on obtient une condition nécessaire :

$$\forall i,j = 1,2 \ldots k, \quad \text{PGCD}(m_i,m_j) | a_i - a_j$$

mais cette condition ne dit pas comment trouver une solution particulière. La seule manière générale d'y arriver est de résoudre les 2 premières équations

$$\left\{ \begin{array}{rcl} x & \equiv & a_1 \bmod m_1 \\ x & \equiv & a_2 \bmod m_2 \end{array} \right.$$

ce qui donne une nouvelle équation $x \equiv x_1 \bmod \text{PPCM}(m_1,m_2)$ puis recommencer avec la troisième équation :

$$\left\{ \begin{array}{rcl} x & \equiv & x_1 \bmod \text{PPCM}(m_1,m_2) \\ x & \equiv & a_3 \bmod m_3 \end{array} \right.$$

et ainsi de suite ... un exemple pour comprendre :

Exemple 1.3.14 Résoudre

$$\begin{array}{rcl} x & \equiv & 5 \bmod 17 \\ x & \equiv & 8 \bmod 29 \\ x & \equiv & 11 \bmod 9 \end{array}$$

on a déjà obtenu pour les deux premières équations que

$$x \equiv 617 \equiv 124 \bmod 493$$

ensuite il faut donc résoudre

$$\begin{array}{rcl} x & \equiv & 124 \bmod 493 \\ x & \equiv & 11 \bmod 9 \end{array}$$

il faut résoudre

$$493 \times k_1 - 9 \times k_2 = -113$$

on en déduit une identité de Bezout :

$$493 \times 4 + 9 \times (-219) = 1$$

comme $1 | -113$ il y a des solutions en multipliant par -113 on obtient une solution

$$493 \times (-452) + 9 \times 24747 = -113$$

donc $x_0 = -452 \times 493 + 124 = -24747 \times 9 + 11 = -222712$ et

$$x \equiv -222712 \equiv 3575 \bmod 4437$$

1.4 L'ensemble $\mathbb{Z}/n\mathbb{Z}$

1.4.1 Structure d'anneau et de corps

Dans le chapitre sur l'algèbre linéaire nous avons vu un certain nombres de méthodes générales de calculs (pivot de Gauss) permettant de résoudre de nombreux problèmes d'apparence différents. Ces méthodes sont les mêmes quelque soit le corps de nombres que l'on utilise : généralement les réels ($\mathbb{R}$) parfois les complexes ($\mathbb{C}$). Dans cette partie nous allons découvrir que l'arithmétique nous permet de construire des corps de nombres finis, c'est à dire contenant un nombre fini d'éléments contrairement à $\mathbb{R}$ ou $\mathbb{C}$.

Définition 1.4.1 *On appelle $\mathbb{Z}/n\mathbb{Z}$ l'ensemble $\{0,1,2,\ldots,n-1\}$ que l'on munit des deux opérations internes suivantes :*

- *l'addition définie par :*

$$\forall a,b \in \mathbb{Z}/n\mathbb{Z},\ \ a+b = (a+b \bmod n)$$

- *la multiplication définie par :*

$$\forall a,b \in \mathbb{Z}/n\mathbb{Z},\ \ a\times b = (a\times b \bmod n)$$

dans la pratique on retiendra que :

faire des calculs dans $\mathbb{Z}/n\mathbb{Z}$ $\Leftrightarrow$ calculer des congruences en enlevant $\bmod n$ **et en remplaçant $\equiv$ par $=$**

Exemple 1.4.2 quelques exemples de calculs simples :

- Pour $\mathbb{Z}/3\mathbb{Z} = \{0;\ 1;\ 2\}$ on a

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

- par contre pour $\mathbb{Z}/4\mathbb{Z} = \{0;\ 1;\ 2;\ 3\}$ on a

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

- exemple de calcul dans $\mathbb{Z}/7\mathbb{Z}$:

$$\begin{aligned} 3\times 5^2 + 2\times 5 - 3 &= 3\times 4 + 3 + 4 \\ &= 5 + 0 \\ &= 5 \end{aligned}$$

Étant donné la définition de $\mathbb{Z}/n\mathbb{Z}$ les opérations $+$ et $\times$ possèdent au minimum les propriétés des opérations $+$ et $\times$ dans $\mathbb{Z}$.

Proposition 1.4.3 *On a les propriétés suivantes :*

- *L'opération $+$ est commutative, associative et possède un élément neutre 0.*
- *l'opération $\times$ est commutative, associative et possède un élément neutre 1.*
- *$\times$ est distributive par rapport à $+$ dans $\mathbb{Z}/n\mathbb{Z}$*
- *tout $a \in \mathbb{Z}/n\mathbb{Z}$ possède un opposé (symétrique pour $+$), noté $-a$, vérifiant :*

$$-a \equiv (n-a) \bmod n$$

$\mathbb{Z}/n\mathbb{Z}$ a une structure d'anneau commutatif.

Preuve : on peut repasser par les définitions de $+$ et $\times$ en termes de congruences pour vérifier chacune des affirmations de la proposition :

- par exemple pour l'addition on a :
  - $a+b = b+a$ dans $\mathbb{Z}$ donc $a+b \equiv b+a \bmod n$ et donc $a+b = b+a$ dans $\mathbb{Z}/n\mathbb{Z}$
  - $(a+b)+c = a+(b+c)$ dans $\mathbb{Z}$ donc $(a+b)+c \equiv a+(b+c) \bmod n$ et donc $(a+b)+c = a+(b+c)$ dans $\mathbb{Z}/n\mathbb{Z}$
  - $a+0 = 0+a = a$ dans $\mathbb{Z}$ donc $a \equiv a+0 \equiv 0+a \bmod n$ et donc $a+0 = 0+a = a$ dans $\mathbb{Z}/n\mathbb{Z}$
- par exemple pour la multiplication on a :
  - $a\times b = b\times a$ dans $\mathbb{Z}$ donc $a\times b \equiv b\times a \bmod n$ et donc $a\times b = b\times a$ dans $\mathbb{Z}/n\mathbb{Z}$
  - $(a\times b)\times c = a\times(b\times c)$ dans $\mathbb{Z}$ donc $(a\times b)\times c \equiv a\times(b\times c) \bmod n$ et donc $(a\times b)\times c = a\times(b\times c)$ dans $\mathbb{Z}/n\mathbb{Z}$
  - $a\times 1 = 1\times a = a$ dans $\mathbb{Z}$ donc $a \equiv a\times 1 \equiv 1\times a \bmod n$ et donc $a\times 1 = 1\times a = a$ dans $\mathbb{Z}/n\mathbb{Z}$
- pour l'opposé : $(n-a)+a = n+(-a+a) = n+0 \equiv 0 \bmod n$ donc $n-a \in \mathbb{Z}/n\mathbb{Z}$ est l'opposé de a

□

Comme toujours le problème vient de l'existence d'un symétrique pour la multiplication.

Théorème 1.4.4 *Soit $a \in \mathbb{Z}/n\mathbb{Z}$ alors a possède un inverse (symétrique pour $\times$), noté a^{-1}, si et seulement si* $\mathrm{PGCD}(a,n) = 1$. *En particulier si n est premier tout $a \in \mathbb{Z}/n\mathbb{Z}$ $a \neq 0$ possède un inverse et donc $\mathbb{Z}/n\mathbb{Z}$ est un corps si et seulement si n est premier.*

Preuve : Ce problème de l'inversion a déjà été traité dans la partie sur les congruences. Il faut commencer par chercher une identité de Bezout dans $\mathbb{Z}$, ce qui n'est possible que si a et n sont premiers entre eux :

$$\mathrm{PGCD}(a,n) = 1 \Longrightarrow \exists u,v \in \mathbb{Z},\ \ a\times u + n\times v = 1 \Longrightarrow au = 1 - nv$$

ensuite on voit que les coefficients de cette identité de Bezout fournissent un inverse pour a :

$$\exists u,v \in \mathbb{Z},\ au \equiv 1 \bmod n \Longrightarrow a \times u = 1 \ \text{ dans } \ \mathbb{Z}/n\mathbb{Z}$$

inversement si a à un inverse u alors

$a \times u = 1$ dans $\mathbb{Z}/n\mathbb{Z}$ alors $a \times u \equiv 1 \bmod n \Longrightarrow \exists k \in \mathbb{Z},\ au = 1 + kn \Longrightarrow au + n(-k) = 1$

donc $PGCD(a,n) = 1$. □

On se rappellera que trouver l'opposé d'un élément de $\mathbb{Z}/n\mathbb{Z}$ est simple par contre trouver l'inverse nécessite plus de travail car repose sur le calcul d'une identité de Bezout

trouver un inverse dans $\mathbb{Z}/n\mathbb{Z}$ revient à chercher une identité de Bezout :

$$au + nv = 1 \Longleftrightarrow au \equiv 1 \bmod n \Longleftrightarrow u = a^{-1} \ \text{ dans } \ \mathbb{Z}/n\mathbb{Z}$$

Exemple 1.4.5 On peut faire des calculs dans $\mathbb{Z}/n\mathbb{Z}$ comme dans n'importe quel autre corps, par exemple pour résoudre une équation ou un système d'équations dans $\mathbb{Z}/7\mathbb{Z}$:

$$\begin{aligned} 3x + 2 = 0 &\Longleftrightarrow 3x = -2 \\ &\Longleftrightarrow 3x = 5 \\ &\Longleftrightarrow 5 \times 3x = 5 \times 5 \\ &\Longleftrightarrow 1 \times x = 4 \\ &\Longleftrightarrow x = 4 \end{aligned}$$

car l'inverse de 3 est 5 puisqu'une identité de Bezout entre 3 et 7 est donnée par :

$$3 \times 5 - 2 \times 7 = 1$$

si on tombe sur une autre identité de Bezout on doit savoir retrouver le résultat :

$$3 \times 12 - 5 \times 7 = 1 \Longleftrightarrow 3 \times 12 \equiv 3 \times 5 \equiv 1 \bmod 7$$

Les éléments inversibles forment un sous ensemble important de $\mathbb{Z}/n\mathbb{Z}$:

Définition 1.4.6 (éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$)
On notera $(\mathbb{Z}/n\mathbb{Z})^*$ l'ensemble des éléments inversibles dans $\mathbb{Z}/n\mathbb{Z}$:

$$(\mathbb{Z}/n\mathbb{Z})^* = \left\{ a \in \mathbb{Z}/n\mathbb{Z} \ \mid \ \exists a^{-1} \in \mathbb{Z}/n\mathbb{Z},\ a \times a^{-1} = 1 \right\}$$

on a $\text{Card}\left((\mathbb{Z}/n\mathbb{Z})^*\right) = \Phi(n)$.

Mis à part la difficulté du calcul de l'inverse les règles de calculs dans $\mathbb{Z}/n\mathbb{Z}$ sont les mêmes que dans $\mathbb{R}$.

Proposition 1.4.7 *soit $x \in (\mathbb{Z}/n\mathbb{Z})^*$ et $a,b \in \mathbb{N}$ alors :*

- $x^0 = 1$ *l'élément neutre de* $\times$,
- $(-x)^a = (-1)^a \times (x^a)$,
- $(-x)^{-1} = -(x^{-1})$,
- $(x^a)^{-1} = (x^{-1})^a$ *qu'on pourra noter* x^{-a},
- $x^{a-b} = x^a \times x^{-b}$

Preuve : Par abus de notation on confond $x \in \mathbb{Z}/n\mathbb{Z}$ et sont "représentant" dans $x \in \mathbb{Z}$:

- pour $a > 0$ dans $\mathbb{Z}$ on a $x^0 \times x^a = x^{0+a} = x^a = x^{a+0} = x^a \times x^0$ ce qui reste vrai $\bmod n$ et donc aussi dans $\mathbb{Z}/n\mathbb{Z}$, x^0 est donc bien l'élément neutre de $\times$ d'où $x^0 = 1$.
- la relation $(-x)^a = (-1)^a \times (x^a)$ étant vraie dans $\mathbb{Z}$ et reste vraie en $\bmod n$ et donc dans $\mathbb{Z}/n\mathbb{Z}$,
- si x est premier avec n alors $-x (= n - x \bmod n)$ est aussi premier avec n et on peut chercher l'inverse de $-x$. ensuite on joue sur la commutativité de $\times$ pour montrer que $-(x^{-1})$ est l'inverse de $-x$:

  $$-(x^{-1}) \times (-x) = (-1) \times (x^{-1}) \times (-x) = (x^{-1}) \times (-1) \times (-x) = (x^{-1}) \times x = 1$$

  on ne peut pas passer par $\mathbb{Z}$ car $x^{-1} \notin \mathbb{Z}$!!! Sans compter que en fait x^{-1} diffère de $\frac{1}{x}$ (l'inverse de x au sens des $\mathbb{R}$ ou $\mathbb{Q}$).
- même type de démonstration que ci-dessus pour montrer que $(x^{-1})^a$ est l'inverse de x^a :

  $$(x^{-1})^a \times x^a = (x^{-1} \times x) \times \ldots (x^{-1} \times x) = 1 \times \cdots \times 1 = 1$$

- plus difficile à comprendre, on le fait par récurrence sur b : pour $n = 1$ on a l'implication évidente (multiplier l'équation de gauche par x^{-1} :

  $$x \times x^{a-1} = x^a \Longrightarrow x^{a-1} = x^a \times x^{-1}$$

  pour passer de b à $b + 1$ on fait de même en utilisant en plus l'hypothèse de récurrence

  $$x \times x^{a-b-1} = x^{a-b} \Longrightarrow x^{a-b-1} = x^{a-b} \times x^{-1} = x^a \times x^{-b} \times x^{-1} = x^a \times x^{-b-1}$$

□

La non-inversibilité de certains élément de $\mathbb{Z}/n\mathbb{Z}$, quand n n'est pas premier, fait apparaître des phénomènes qui peuvent sembler étrange dans certains calculs et qui sont liés à l'existence de diviseurs de 0 :

Proposition 1.4.8 *On considère l'équation $a \times b = 0$ dans $\mathbb{Z}/n\mathbb{Z}$ alors :*

- *si n est premier alors $a \times b = 0 \Longrightarrow a = 0$ ou $b = 0$*
- *si n est composé alors $\exists a,b \in \mathbb{Z}/n\mathbb{Z} \setminus \{0\},\ a \times b = 0$, a et b sont appelés des diviseurs de* 0.

Preuve :

- si n est premier alors soit $a = 0$ et la conclusion est vraie, soit $a \neq 0$ et dans ce cas on peut trouver un inverse à a, en multipliant par cet inverse on obtient :
$$a^{-1} \times a \times b = 0 \Longrightarrow 1 \times b = b = 0$$
et la conclusion de la proposition est aussi vraie.
- si n est composé alors $\exists 1 < a,b < n,\ \ n = a \times b$ donc $a \times b = 0$ avec $a,b \in \mathbb{Z}/n\mathbb{Z} \setminus \{0\}$.

□

Exemple 1.4.9 En faisant les tables de multiplications on peut vérifier que dans un $\mathbb{Z}/11\mathbb{Z}$ il n'y a pas de diviseurs de 0 :

×	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	1	3	5	7	9
3	0	3	6	9	1	4	7	10	2	5	8
4	0	4	8	1	5	9	2	6	10	3	7
5	0	5	10	4	9	3	8	2	7	1	6
6	0	6	1	7	2	8	3	9	4	10	5
7	0	7	3	10	6	2	9	5	1	8	4
8	0	8	5	2	10	7	4	1	9	6	3
9	0	9	7	5	3	1	10	8	6	4	2
10	0	10	9	8	7	6	5	4	3	2	1

alors que dans $\mathbb{Z}/15\mathbb{Z}$ il y en a plusieurs :

×	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	0	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	0	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	0	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	0	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	0	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	0	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	0	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	0	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	0	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	0	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	0	14	13	12	11	10	9	8	7	6	5	4	3	2	1

par exemple :
$$3 \times 5 = 6 \times 5 = 9 \times 10 = 12 \times 5 = 0$$

On retiendra qu'il est beaucoup plus facile de travailler dans $\mathbb{Z}/n\mathbb{Z}$ avec n premier que'avec n composé.

1.4.2 Polynômes et résidus quadratiques

Nous aurons besoin du théorème suivant valable dans n'importe quel corps de nombre $\mathbb{K}$.

Théorème 1.4.10 *Soit un polynôme $P(x) = a_k X^k + \cdots + a_0 \in \mathbb{K}[X]$ de degré k alors P possède au plus k racines dans $\mathbb{K}$*

Pour le démontrer on a besoin du lemme suivant :

Lemme 1.4.11 *pour tout a,b*
$$a^n - b^n = (a-b)(a^{n-1} + a^{n-2}b + \cdots + ab^{n-2} + b^{n-1})$$

Preuve : le lemme est très simple à démontrer :

$$\begin{aligned}
&(a-b)(a^{n-1} + a^{n-2}b + \cdots + ab^{n-2} + b^{n-1}) \\
&= \ a^n + a^{n-1}b + \cdots + a^2b^{n-2} + ab^{n-1} \\
&\qquad - a^{n-1}b - a^{n-2}b^2 - \cdots - ab^{n-1} - b^n \\
&= \ a^n - b^n
\end{aligned}$$

ensuite le théorème se démontre par récurrence sur le degré k du polynôme :

- on pose $\mathcal{P}_k$: { un polynôme de degré k à au plus k racines }
- rang initial $k = 1$: $P(X) = a_1 X + a_0 = 0 \Longrightarrow X = -a_1^{-1} a_0$ si a_1 est inversible sinon il n'y a pas forcément de solution. Donc il y a bien au plus une solution.
- passage de $\mathcal{P}_k \Rightarrow \mathcal{P}_{k+1}$:
soit P n'a aucune racine sinon on en a une r et en utilisant que $P(r) = 0$ et que $x^i - r^i$ se factorise par $x - r$ et un polynôme de degré $i-1$ on a alors
$$P(X) = P(X) - P(r) = \sum_{i=0}^{k} a_i (X^i - r^i) = (X - r)Q(X)$$
avec Q un polynôme de degré $k-1$ qui a donc au plus $k-1$ racine. Au total P a au plus k racine.

□

Comme dans $\mathbb{R}$ on aimerait pouvoir trouver les racines d'un trinôme du second degré. Mais on va voir que c'est plus difficile que dans $\mathbb{R}$.

Théorème 1.4.12
Si n est premier l'équation $x^2 = 1$ a deux solution Dans $\mathbb{Z}/n\mathbb{Z}$: $x = 1$ ou $x = -1$.

Preuve : on travaille comme avec les nombres réels :
$$x^2 = 1 \Longleftrightarrow x^2 - 1 = 0 \Longleftrightarrow (x-1)(x+1) = 0$$
comme il n'y a pas de diviseurs de 0 on a
$$[x + 1 = 0 \ \ ou \ \ x - 1 = 0] \Leftrightarrow [x = 1 \ \ ou \ \ x = -1]$$

□

Exemple 1.4.13 En ne gardant que la diagonale des tables de Pythagore de $\mathbb{Z}/11\mathbb{Z}$ on voit bien qu'il n'y a que les deux solutions annoncées 1 et $-1 = 10$:

x	0	1	2	3	4	5	6	7	8	9	10
x^2	0	1	4	9	5	3	3	5	9	4	1

Par contre dans $\mathbb{Z}/15\mathbb{Z}$ d'autres solutions, comme 4 et 11, peuvent apparaître, en plus de 1 et $-1 = 14$, car 15 est composé :

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x^2	0	1	4	9	1	10	6	4	4	6	10	1	9	4	1

On peut voir dans ces exemples que tous les nombres ne sont pas des carrés dans $\mathbb{Z}/n\mathbb{Z}$. Savoir si un nombre est un carré ou pas est une question cruciale en arithmétique, résolue par le théorème suivant :

Proposition 1.4.14 *Si n est premier il y a $\frac{n+1}{2}$ carrés dans $\mathbb{Z}/n\mathbb{Z}$ (dont 0).*

Preuve : si $x^2 = y^2$ alors $x^2 - y^2 = (x-y)(x+y) = 0$ donc $x = y$ ou $x = -y$. On peut donc regrouper les valeurs de $\mathbb{Z}/n\mathbb{Z}$ par 2 $\{x;\ -x\}$ qui donnent le même carré plus 0 qui est un carré. Conclusion le nombre de carrés différents est la moitié de $n-1$ plus le 0 donc $\frac{n-1}{2} + 1 = \frac{n+1}{2}$ □

Exemple 1.4.15 Il y a bien 6 carrés dans $\mathbb{Z}/11\mathbb{Z}$ qui sont 0; 1; 4; 9; 5; 3. La formule ne marche pas dans $\mathbb{Z}/15\mathbb{Z}$ où il n'y a que 6 carrés 0; 1; 4; 6; 9; 10.

Le théorème suivant permet de tester si un x donné est un carré ou pas (sans calculer les carrés de tous les éléments de $\mathbb{Z}/n\mathbb{Z}$).

Théorème 1.4.16
Si $n > 2$ est premier alors $x \in (\mathbb{Z}/n\mathbb{Z})^*$ est un carré si et seulement si $x^{\frac{n-1}{2}} = 1$.

Preuve : on peut d'abord remarquer que comme n est impair $\frac{n-1}{2}$ est bien un entier. si $\exists y \in \mathbb{Z}/n\mathbb{Z},\ \ y^2 = x$ alors en utilisant le petit théorème de Fermat ($\forall 0 < y < n,\ \ y^{n-1} \equiv 1 \bmod n$) on a $(y^2)^{\frac{n-1}{2}} = y^{n-1} = 1 = x^{\frac{n-1}{2}}$ dans $\mathbb{Z}/n\mathbb{Z}$. Inversement si $x^{\frac{n-1}{2}} = 1$ alors x est racine de $P(X) = X^{\frac{n-1}{2}} - 1$ qui a au plus $\frac{n-1}{2}$ racines dans $\mathbb{Z}/n\mathbb{Z}$ or on sait déjà qu'il n'y a que $\frac{n-1}{2}$ carrés non nuls qui sont déjà racines de P. Conclusion il n'y en a pas d'autres. □

En général (pour n grand) il est difficile de calculer $x^{\frac{n-1}{2}}$ pour vérifier si n est un carré (avec l'algorithme d'exponentiation modulaire rapide et un ordinateur ça devient plus facile) sauf dans la cas où $x = -1$:

Théorème 1.4.17
Si $n > 2$ est premier alors $-1 \in (\mathbb{Z}/n\mathbb{Z})^*$ est un carré si et seulement si $n \equiv 1 \bmod 4$

Preuve : $(-1)^{\frac{n-1}{2}} = 1$ si et seulement si $\frac{n-1}{2} = 2k,\ \ k \in \mathbb{Z}$ (*i.e.* est pair!) donc $n = 1 + 4k \iff n \equiv 1 \bmod 4$. □

Pour aller plus loin il faut introduire le symbole de Legendre :

Théorème 1.4.18 (Symbole de Legendre)
Soient $n > 2$ premier et $x \in \mathbb{Z}$ alors on définit le symbole de Legendre par :

$$\left(\frac{x}{n}\right) = x^{\frac{n-1}{2}} = \begin{cases} 0 & si \quad n|x \\ 1 & si \quad x \text{ est un carré } \bmod n \\ -1 & si \quad x \text{ n'est pas un carré } \bmod n \end{cases}$$

le symbole de Legendre est multiplicatif : $\forall x,y \in \mathbb{Z},\ \ \left(\frac{xy}{n}\right) = \left(\frac{x}{n}\right)\left(\frac{y}{n}\right)$.

Preuve : On commence par se ramener à $x \in \mathbb{Z}/n\mathbb{Z}$, en particulier si $n|x$ on est ramené à calculer $x^{\frac{n-1}{2}} = 0^{\frac{n-1}{2}} = 0$. Ensuite comme $x^{n-1} = 1$ (petit théorème de Fermat on a que $x^{\frac{n-1}{2}}$ est une racine carrée de 1 donc vaut ± 1 selon que x est un carré ou pas. La dernière égalité vient juste de $\left(\frac{xy}{n}\right) = (xy)^{\frac{n-1}{2}} = (x)^{\frac{n-1}{2}}(y)^{\frac{n-1}{2}} = \left(\frac{x}{n}\right)\left(\frac{y}{n}\right)$. □

Lemme 1.4.19 (résidus minimaux)
Si on appelle résidu minimal de x modulo n le plus petit entier congru à x dans l'intervalle $]-\frac{n-1}{2};\ \frac{n-1}{2}]$ alors $\left(\frac{x}{n}\right) = (-1)^{\mu}$ où

$$\mu = \textit{nombre de résidu minimaux} < 0 \textit{ dans } \{x;\ 2x;\ 3x;\ \ldots \tfrac{n-1}{2}x\}$$

Preuve : On commence par écrire que

$$\left\{x;\ 2x;\ 3x;\ \ldots \frac{n-1}{2}x\right\} = \{-s_1;\ -s_2;\ \ldots;\ -s_\mu;\ r_1;\ r_2;\ \ldots r_{\frac{n-1}{2}-\mu}\}$$

où les s_i,$r_j > 0$ sont (au signe près) les résidus minimaux. Le résultat essentiel est que les r_i et s_j prennent $\frac{n-1}{2}$ valeurs différentes :

$$\{s_1;\ s_2;\ \ldots;\ s_\mu;\ r_1;\ r_2;\ \ldots r_{\frac{n-1}{2}-\mu}\} = \left\{1;\ 2;\ 3;\ \ldots \frac{n-1}{2}\right\}$$

en effet si

$$s_i = s_j \Longrightarrow m_i x = m_j x \Longrightarrow m_i = m_j$$

de même pour $r_i = r_j$ mais aussi pour $s_i = r_j$ car

$$s_i = r_j \Longrightarrow -m_i x = m_j x \Longrightarrow -m_i = m_j \Longrightarrow m_i + m_j \equiv 0 \bmod n$$

or $0 < m_i$,$m_j < n/2 \Longrightarrow 0 < m_i + m_j < n$. Maintenant on note P le produit des éléments de $\{x;\ 2x;\ 3x;\ \ldots \frac{n-1}{2}x\}$ on a alors :

$$P = x \times 2x \times \cdots \times \frac{n-1}{2}x = \underbrace{x \times x \times \cdots \times x}_{\frac{n-1}{2} \text{ fois}} \times 1 \times 2 \times \cdots \times \frac{n-1}{2} = x^{\frac{n-1}{2}} \times \left(\frac{n-1}{2}\right)!$$

d'un autre coté en notant r_i les résidus positifs et $-s_j$ ceux qui sont négatifs :

$$P = x \times 2x \times \cdots \times \frac{n-1}{2}x = (-1)^{\mu} s_1 \times \cdots \times s_\mu \times r_1 \times \cdots \times r_{\frac{n-1}{2}-\mu} = (-1)^{\mu}\left(\frac{n-1}{2}\right)!$$

il ne reste plus qu'à multiplier par l'inverse de $\left(\frac{n-1}{2}\right)!$ (qui n'est pas divisible par p donc inversible dans $\mathbb{Z}/p\mathbb{Z}$) pour obtenir $x^{\frac{n-1}{2}} = (-1)^{\mu}$. □

Maintenant on peut calculer le symbole de Legendre pour d'autres valeurs :

Théorème 1.4.20 *Pour n premier > 2 on a : $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$*

Preuve : il faut donc compter le nombre de résidus minimaux < 0 dans

$$\{2;\ 4;\ 6;\ \dots n-1\} = \{2;\ 4;\ 6;\ \cdots -3;\ -1\}$$

il sont donc positifs si $2x \leq \frac{n-1}{2}$ donc il y a au plus $E\left(\frac{n-1}{4}\right)$ résidus positifs (partie entière de $\frac{n-1}{4}$). Si $n \equiv 1 \bmod 4$ alors $E\left(\frac{n-1}{4}\right) = \frac{n-1}{4}$ et $\mu = \frac{n-1}{2} - \frac{n-1}{4} = \frac{n-1}{4}$.
Par contre si $n \equiv 3 \equiv -1 \bmod 4$ $E\left(\frac{n-1}{4}\right) = \frac{n-3}{4}$ et $\mu = \frac{n-1}{2} - \frac{n-3}{4} = \frac{n+1}{4}$.

Il ne reste plus à remarquer que :

$$\frac{n^2-1}{8} = 2\frac{n+1}{4}\frac{n-1}{4} = \begin{cases} \frac{n-1}{2}\frac{n+1}{4} & si \quad n \equiv -1 \bmod 4 \\ \frac{n+1}{2}\frac{n-1}{4} & si \quad n \equiv 1 \bmod 4 \end{cases}$$

à donc la même parité que $E\left(\frac{n-1}{4}\right)$ ce qui permet d'écrire que :

$$(-1)^{\mu} = (-1)^{\frac{n^2-1}{8}} = \begin{cases} (-1)^{\frac{n-1}{2}\frac{n+1}{4}} & si \quad n \equiv -1 \bmod 4 \\ (-1)^{\frac{n+1}{2}\frac{n-1}{4}} & si \quad n \equiv 1 \bmod 4 \end{cases}$$

□

le théorème suivant du à Gauss est très pratique pour savoir si un nombre est un carré ou pas.

Théorème 1.4.21 (loi de réciprocité quadratique)
si p et q sont premiers et impairs alors

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\times\frac{q-1}{2}}\left(\frac{q}{p}\right)$$

Exemple 1.4.22 Voici quelques exemples d'utilisation des règles précédentes pour savoir si des entiers sont ou pas des carrées modulo 86477 :

- Pour 2 on utilise la formule du théorème 1.4.20 :

  $$\left(\frac{2}{86477}\right) = (-1)^{\frac{86477^2-1}{8}} = (-1)^{934783941} = -1$$

  donc 2 n'est pas un carré dans $\mathbb{Z}/86477\mathbb{Z}$.
- pour 5 on utilise la loi de réciprocité quadratique pour réduire les calculs au minimum. Il faut donc calculer :

  $$\left(\frac{5}{86477}\right) = (-1)^{2\times 43238}\left(\frac{86477}{5}\right) = \left(\frac{2}{5}\right)$$

  ensuite on peut calculer si 2 est un carré modulo 5 en utilisant la définition :

  $$\left(\frac{2}{5}\right) = 2^{\frac{5-1}{2}} = 2^2 = 4 = -1$$

  ou le théorème 1.4.20 :

  $$\left(\frac{2}{5}\right) = (-1)^{\frac{5^2-1}{8}} = (-1)^{24} = 1$$

  dans tous les cas 5 n'est pas un carré dans $\mathbb{Z}/86477\mathbb{Z}$!
- Par contre 10 est un carré dans $\mathbb{Z}/86477\mathbb{Z}$ car en utilisant le fait que le symbole de Legendre est multiplicatif on obtient :

  $$\left(\frac{10}{n}\right) = \left(\frac{2}{n}\right)\left(\frac{5}{n}\right) = (-1)\times(-1) = 1$$

1.4.3 Générateurs de $\mathbb{Z}/n\mathbb{Z}$

Les puissances successives d'un élément de $\mathbb{Z}/n\mathbb{Z}$ jouent un rôle très important (comme dans le chapitre précédent).

Définition 1.4.23 *Un $a \in (\mathbb{Z}/n\mathbb{Z})^*$ est générateur de $(\mathbb{Z}/n\mathbb{Z})^*$ si et seulement si les puissances successives de a permettent de générer tous les éléments de $(\mathbb{Z}/n\mathbb{Z})^*$:*

$$(\mathbb{Z}/n\mathbb{Z})^* = \left\{a^k | k = 1, \dots, \Phi(n)\right\}$$

Pour qu'un élément soit générateur de $\mathbb{Z}/n\mathbb{Z}$ on a une condition nécessaire :

Théorème 1.4.24
Soit $n > 2$ est premier, si $a \in (\mathbb{Z}/n\mathbb{Z})^*$ est générateur de $(\mathbb{Z}/n\mathbb{Z})^*$ alors $x^{\frac{n-1}{2}} = -1$.

Preuve : Les puissances successives de a se répètent forcément au bout de la $n^{\text{ième}}$ au maximum puisqu'il n'y a que $n-1$ valeurs différentes dans $(\mathbb{Z}/n\mathbb{Z})^*$. Si $a^{\frac{n-1}{2}} = 1$ alors $a^{\frac{n+1}{2}} = a$ donc les valeurs se repètent au bout de seulement $\frac{n+1}{2}$ valeurs et a ne peut générer $(\mathbb{Z}/n\mathbb{Z})^*$ donc $a^{\frac{n-1}{2}} = -1$. □

Exemple 1.4.25 Hélas cette condition n'est pas suffisante! Prenons l'exemple de $\mathbb{Z}/n\mathbb{Z}$ avec $n = 11$ si on fait la table des puissances on trouve que :

x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}
1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	3	6	12	11	9	5	10	7	1
3	9	1	3	9	1	3	9	1	3	9	1
4	3	12	9	10	1	4	3	12	9	10	1
5	12	8	1	5	12	8	1	5	12	8	1
6	10	8	9	2	12	7	3	5	4	11	1
7	10	5	9	11	12	6	3	8	4	2	1
8	12	5	1	8	12	5	1	8	12	5	1
9	3	1	9	3	1	9	3	1	9	3	1
10	9	12	3	4	1	10	9	12	3	4	1
11	4	5	3	7	12	2	9	8	10	6	1
12	1	12	1	12	1	12	1	12	1	12	1

les générateurs de $(\mathbb{Z}/n\mathbb{Z})^*$ sont donc $a = 2;\ 6;\ 7;\ 11$ alors que $a = 2;\ 5;\ 6;\ 7;\ 8;\ 11$ vérifient le critère ($-1 = 12$ dans $\mathbb{Z}/13\mathbb{Z}$) :

x	1	2	3	4	5	6	7	8	9	10	11	12
$x^{\frac{n-1}{2}}$	1	12	1	1	12	12	12	12	1	1	12	1

Conclusion : comme les problèmes précédents (factorisation, calcul de résidu,...) trouver a un générateur de $\mathbb{Z}/n\mathbb{Z}$ ou pour un b trouver x tel que $a^x = b$ (problème du logarithme discret) sont des problèmes difficiles ... qui vont servir de base pour la construction de cryptosystèmes efficaces.

1.5 Cryptographie

1.5.1 Codage d'un texte ou d'un document

Pour comprendre les notions de bases de la cryptographie il faut commencer par définir ce à quoi elle s'applique. Les données que nous avons besoin déchiffrer peuvent être assimilés à des *textes* c'est à dire une suite de caractères sont choisis dans un ensemble fini (qu'on appelle couramment un alphabet) de longueur finie qui n'est limitée que par la capacité de stockage du support utilisé pour conserver ou transmettre le texte. La transmission d'un texte (clair ou crypté) repose sur son découpage en paquets qui peuvent être transmis les uns à la suite des autres, par exemple nous pouvons découper un texte en syllabes, en mots, en phrases, en chapitres ...

Cette définition d'un *texte* s'applique sans problème au fichiers informatiques que nous pouvons avoir besoin de chiffrer. Examinons le cas particulier d'un fichier texte. Ils sont écrit dans à partir d'un alphabet de 256 caractères *le code ASCII*, chaque caractère peut être représenté par un code, un nombre entre 0 et 255 (un entier sur 8 bits) donc en binaire par une suite de 8 nombres valant 0 ou 1. En mettant bout à bout les codes des différents caractères on obtient une suite de 0 et de 1 représentant le texte (un nombre binaire en fait) quel'on peut découper en tranches de taille arbitraire (128 bits, 512 bits ...) et que l'on peut ensuite transmettre.

message original = | 1011011010000111 ... |

↓

texte prêt à coder =

1011	0110	1000	0111	...

nombre de 4 bits

Ces groupes peuvent donc être considérés comme représentant des nombres binaires à k chiffres ou, ce qui revient au même, des éléments de $\mathbb{Z}/n\mathbb{Z}$ (avec $n = 2^k$). Si on utilise une autre base p pour représenter les codes des différents caractères et que l'on fait des blocs de k chiffres on obtiendra une représentation du texte par une suite d'éléments de $\mathbb{Z}/p^k\mathbb{Z}$. Pendant ce cours (et en particulier en TP) pour continuer à travailler avec les nombres écrit en base 10 (qui nous sont plus familiers) nous allons utiliser la table de 100 caractères suivante :

code	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
chaine	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
code	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
chaine	u	v	w	x	y	z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
code	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
chaine	O	P	Q	R	S	T	U	V	W	X	Y	Z	à	ç	é	ê	è	î	ï	ô
code	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
chaine	ù	(	)	[	]	{	}	<	>	/	\	=	+	-	*		.	,	;	:
code	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
chaine	!	?	&	|	$	%	@	^	~	'	0	1	2	3	4	5	6	7	8	9

Pour comprendre prenons un exemple. Si l'on a la texte *Un exemple !*

46	13	75	4	23	4	12	15	11	4	80

Pour découper ce texte en groupe de 4 caractères il faut qu'il ait un nombre de caractères multiple de 4, quitte à ajouter des espaces (caractère 75) à la fin. Ici le texte n'ayant que 11 caractères on ajoute un espace pour obtenir :

46	13	75	4	23	4	12	15	11	4	80	75

puis on regroupe les termes par 4 (notez bien le caractère e de code 4 devient 04):

20137504	23041215	11048075

Dans la réalité c'est de cette manière que sont découpés les textes mais en utilisant le codage ASCII et en regroupant les chiffres de leur représentation binaire. Pour information voici la table des caractères ASCII de code 0 à 127 (qui fut ensuite étendue à 256 caractères).

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

FIG. 1.2 – *Table des caractères ASCII*

1.5.2 Les principes de la cryptographie moderne

Maintenant nous pouvons expliquer en quoi consiste le chiffrement d'un tel *texte*. Le chiffrement repose sur plusieurs règles

- La modification de l'alphabet utilisée (par exemple : le a est systématiquement remplacé par la lettre u, le b par m *etc.*...)
- un mélange des caractères (par exemple : écrire chaque bloc de texte codé de droite à gauche au lieu du sens habituel)
- le tout doit dépendre d'une information simple *la clé* permettant le chiffrement ou le déchiffrement.

Un exemple pour illustrer qu'il est nécessaire de combiner ces 3 règles pour obtenir un chiffrement le *code de César*. Ce code consister à décaler les lettres de l'alphabet d'un certain rang a. Par exemple pour $a = 3$ on a la table de conversion suivante :

clair	a	b	c	d	e	f	g	h	i	j	k	l	m	...
codé	d	e	f	g	h	i	j	k	l	m	n	o	p	...

on crypte donc un texte en décalant les lettres de 3 rangs :

`un exemple simple ---> xq hahpsoh vlpsoh`

Du point de vue mathématique on travaille avec un alphabet de 26 lettres, on peut donc se ramener à manipuler des éléments de $\mathbb{Z}/26\mathbb{Z}$ (`a->0, b->1,` ...) et dans ce cas la fonction de chiffrement qui à x (le code d'une lettre) associe $f(x)$ (sont équivalent chiffré) est très simple à écrire : $f(x) = x + a$. Elle est donc entièrement définie par la donnée a qui est la *clé* du cryptosystème. Déchiffrer un texte nécessite de pouvoir inverser la fonction f c'est à dire calculer la fonction g telle que :

$$f(x) = y \Longleftrightarrow x = g(y) = f^{-1}(y)$$

Dans le cas du code de César l'équation $f(x) = x + a$ est très simple à inverser :

$$f(x) = x + a = y \Longleftrightarrow x = y - a = g(y) = f^{-1}(y)$$

Ce crytosystème n'assure pas de grande sécurité. En effet si on code un texte assez long on pourra chercher la lettre apparaissant le plus fréquemment et identifier par quelle lettre est remplacé le `e` ce qui permet de déterminer la valeur de a et de décoder tout le texte. Il s'agit d'une attaque "statistique" du cryptosystème.

Un cas particulier du code de César est celui ou la clé $a = 13$ qui est connu sous le nom de *ROT13*. Dans ce cas on a : $f(x) = x + 13 = g(x)$ car $13 = -13$ dans $\mathbb{Z}/26\mathbb{Z}$! Ce système est utilisé dans certains forums ou sur certains newsgroups pour éviter que l'on ne lise une information involontairement (pour la lire il faut la décoder). On l'appelle *ROT13* car il consiste en une permutation circulaire des lettres de l'alphabet qu'on peut représenter par la table suivante :

clair	a	b	c	d	e	f	g	h	i	j	k	l	m	codé
codé	n	o	p	q	r	s	t	u	v	w	x	y	z	clair

On sait que César utilisait ce code car Suétone (écrivain Romain, 70-127) en fait une description dans *"La vie des 12 Césars"*, une biographie des 12 premiers empereurs de Rome. César employait également une autre méthode de stéganographie très originale. Pour transmettre un message, il rasait la tête d'un esclave et inscrivait le message sur son crâne! Ensuite il attendait la repousse des cheveux et envoyait l'esclave. Malgré (ou à cause de) sa simplicité, ce chiffre fut encore employé par des officiers sudistes pendant la guerre de Sécession et par l'armée russe en 1915.

Ce sont les mêmes principes qui ont été utilisés par les ingénieurs allemands pour définir le cryptosystème ENIGMA (en 1919) qui fut largement utilisé par les armées allemandes pour protéger leur communications à partir de 1926 puis par les nazis pendant toute la seconde guerre mondiale. La machine ENIGMA est basé sur une technologie électromécanique (sans rapport avec celle utilisée maintenant) mais selon les même principes que nous avons énoncés. Elle se présente comme une machine à écrire où l'on saisit le texte à chiffrer.

FIG. 1.3 – *la machine ENIGMA*

Un système complexes de rotors permet d'associer à une lettre frappée sur le clavier à une autre lettre. C'est la position des rotors au début de la frappe du texte qui détermine les règle de conversions : c'est donc la clé du cryptosystème. S'il l'on veut déchiffrer un message intercepté sans aucune autre information alors il faut essayer chacune des clés possibles! La sécurité du système reposait donc sur le nombre de clés différentes à tester (contrairement au code de César) car sur les machines Enigma on pouvait chiffrer un texte selon 10^{16} combinaisons différentes !

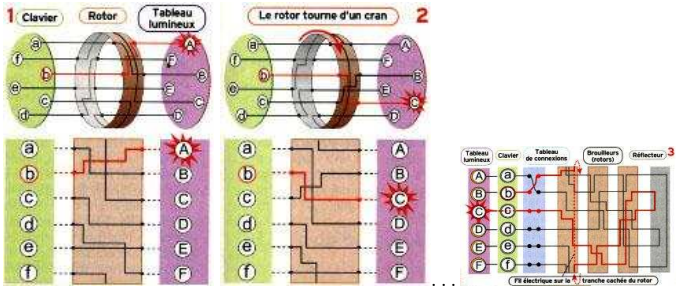

FIG. 1.4 – *Fonctionnement d'ENIGMA : la clé (privée) est déterminée par la position initiale des rotors*

C'est un mathématicien polonais de 27 ans, Marian Rejewski, qui trouvera dès les années 30 une première manière de trouver rapidement la clé, idées qui seront ensuite développées sous la direction d'Alan Turing à Bletchley Park, un manoir proche de Londres où se sont retranchés tous les cryptologues et mathématiciens alliés, et qui permettrons de casser tout le système cryptographique de l'armée nazi.

Pour appliquer ces règles (en particulier le changement d'alphabet), à un texte codé sous forme d'une suite de chiffres, il est très pratique d'utiliser des fonctions arithmétiques telles que l'élévation à la puissance modulo n. En effet le comportement irrégulier (mais totalement déterministe) de ces fonctions semble totalement aléatoire!Dans la suite nous allons voir qu'il existe 2 grands types de cryptosystèmes :

cryptosystème à clé privée : le chiffrement et le déchiffrement reposent sur la connaissance d'une même clé qui doit rester secrète pour conserver la sûreté du cryptosystème,

cryptosystème à clé publique : le chiffrement repose sur une méthode publique (accessible à tous) mais le déchiffrement repose sur la connaissance d'une clé secrète.

1.5.3 Cryptosystème à clé symétrique : Le DES

Le DES (data encryption system) est l'exemple type des cryptosystèmes à clé privée. Dans une version très simplifiée le principe de l'algorithme DES consiste à répéter un grand nombre de fois une opération très simple (que l'on peut inverser) qui "mélange" peu à peu les bits d'un document. L'opération de base s'appelle *La méthode de Feistel* (cf. figure 1.5) et consiste à échanger/combiner deux groupes de bits adjacents. Étant donné une fonction $f : \mathbb{Z}/n\mathbb{Z} \longrightarrow \mathbb{Z}/n\mathbb{Z}$, (pas forcément bijective) la fonction de chiffrement sera définie par :

$$(A,B) \underset{\text{chiffrement}}{\longmapsto} (B,A+f(B))$$

Il est alors très facile de décoder en utilisant les formules suivantes

$$(A',B') \underset{\text{déchiffrement}}{\longmapsto} (B'-f(A'),A')$$

en effet on peut vérifier que

$$(A,B) \underset{\text{chiffrement}}{\longmapsto} (B,A+f(B)) \underset{\text{déchiffrement}}{\longmapsto} (A+f(B)-f(B),B) = (A,B)$$

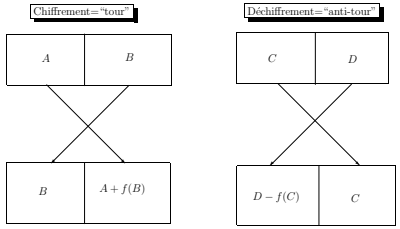

FIG. 1.5 – *La méthode de Fiestel*

Le DES consiste à répéter de nombreuses fois de suite la méthode de Fiestel avec différentes fonctions à chaque fois (cf. figure 1.6), ce qui complique la recherche de clés pour quelqu'un qui voudrait déchiffrer le message sans y être autorisé. La clé secrète est donc la suite de fonctions $f_j : \mathbb{Z}/n\mathbb{Z} \longrightarrow \mathbb{Z}/n\mathbb{Z}$, $j = 1,\dots k$. Dans la pratique, chacune de ces clés peut se ramener à un entier dans $\mathbb{Z}/n\mathbb{Z}$ (de grande taille). On peut facilement implémenter cet algorithme en prenant des fonctions du type :

$$f_j(x) = x_j^a \bmod n \Longrightarrow (a_1,\dots,a_k) = \text{clé secrète}$$

dont le comportement "aléatoire" permettra d'obtenir un chiffrement de bonne qualité.

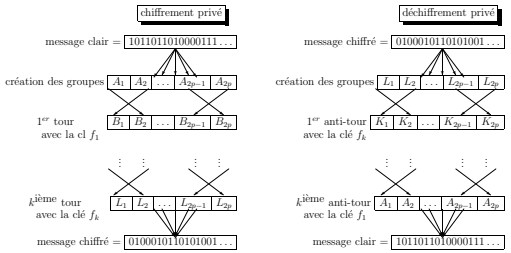

FIG. 1.6 – *Le cryptosystème DES*

1.5.4 Cryptosystème à clé asymétrique : le RSA

Au contraire du DES, le RSA est un cryptosystème à clé publique. La clé d'un tel cryptosystème est un triplet d'entiers (n,e,d) qui doit être généré de la manière suivante :

- générer aléatoirement deux nombres premiers (distincts) p et q
  - calculer $n = p \times q$
  - calculer $\Phi(n) = (p-1) \times (q-1)$
- générer aléatoirement un nombre e premier avec $\Phi(n)$
  - calculer d tel que $e \times d \equiv 1 \bmod \Phi(n)$

Au final les parties publiques et privées de la clé et les fonctions de chiffrement/déchiffrement s'obtiennent de la manière suivante :

(n,e) clé publique $\downarrow$ $f(A) = A^e \bmod n$ fonction de chiffrement	(n,d) clé secrète $\downarrow$ $f^{-1}(B) = B^d \bmod n$ fonction de déchiffrement

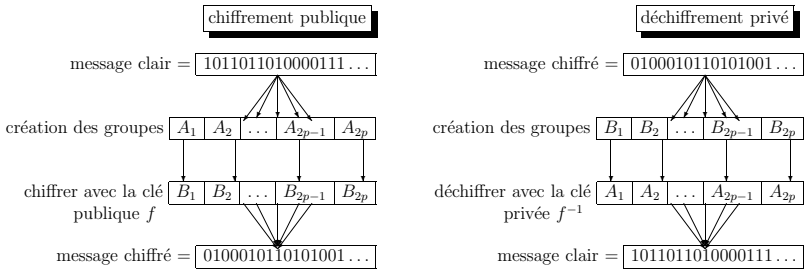

FIG. 1.7 – *Le cryptosystème RSA*

La mise en œuvre du cryptosystème à clé publique RSA est plus simple que celle du DES (cf. figure 1.7), par contre la preuve du bon fonctionnement du RSA est plus complexe. Pour montrer que les fonction f et f^{-1} sont bien inverses l'une de l'autre on a besoin du petit théorème de Fermat et du théorème chinois :

Preuve : Il faut montrer que

$$\forall \ x \in [0,n-1], \quad f(f^{-1}(x)) = f^{-1}(f(x)) = x^{ed} \equiv x \bmod n.$$

car

$$x \underset{\text{chiffrement}}{\longmapsto} x^e \bmod n \underset{\text{déchiffrement}}{\longmapsto} (x^e)^d = x^{e\times d} \bmod n$$

Remarquons d'abord que quelque soit $x \in [0,n-1]$ (avec $n = pq$) alors PGCD$(x,p) =$ 1 ou PGCD$(x,q) = 1$. Les rôles de p et q étant totalement symétriques il suffit de considérer le cas PGCD$(x,p) = 1$. On a alors l'alternative suivante :

- PGCD$(x,q) = 1$ d'après le théorème de Fermat on a

  $$x^{\Phi(n)} = (x^{p-1})^{q-1} \equiv 1^{q-1} \equiv 1 \bmod p$$

  et

  $$x^{\Phi(n)} = (x^{q-1})^{p-1} \equiv 1^{p-1} \equiv 1 \bmod q$$

  le théorème Chinois permet alors de conclure que

  $$x^{\Phi(n)} \equiv 1 \bmod pq.$$

  ce qui donne

  $$x^{e\times d} = x^{1+k\times\Phi(n)} = x \times (x^{\Phi(n)})^k \equiv x \bmod n$$

- PGCD$(x,q) \neq 1$ dans ce cas on a toujours

  $$x = \alpha \times q \equiv 0 \bmod q, \quad 0 \leq \alpha < p.$$

  On a alors

  $$x^{ed} = 0^{ed} \equiv 0 \equiv x \bmod q$$

  et aussi

  $$x^{ed} = x((x^{p-1})^{q-1})^k \equiv x \bmod p$$

  donc $x^{ed} \equiv x \bmod pq$.

□

Il reste à montrer la sûreté du cryptosystème RSA. Pour comprendre cherchons à trouver la clé secrète d à partir de la clé publique (n,e). Pour cela il faut :

- décomposer n en 2 facteurs premiers $p \times q$
- Calculer $\Phi(n) = (p-1) \times (q-1)$
- Calculer une identité de Bezout $e \times d + k \times \Phi(n) = 1$ pour obtenir d

Toutes ces opérations peuvent être programmées sur machine (cf. les TP) et l'on pourrait croire que le cryptosystème est facilement "cassable". En fait la sécurité du cryptosystème RSA repose sur un choix des entiers p et q qui rende très difficile la factorisation de n. Il est possible de générer rapidement des entiers p, q et e premiers qui soient très grands (plusieurs centaines de chiffres décimaux) qui rendent impossible la factorisation de n en un temps raisonnable! En effet si on compare les temps de calcul des différents algorithmes à mettre en oeuvre en fonction de la taille des données en entrée on obtient le tableau qui suit. Si a,b,$e \approx n$ sont des entiers de taille maximale n et

$$L = \log(n) \approx \text{nombre de chiffres en écriture décimale}$$

alors

opération	temps de calcul nécessaire $\leq$
$a+b$	$c_1 \times L$
$a \times b$	$c_2 \times L^2$
PGCD(a,b)	$c_2 \times L^3$
`Bezout`(a,b)	$c_4 \times L^3$
`powermod`(a,e,n)	$c_5 \times L^3$
`prime`(k)	$c_7 \times L^5$
`eratosthene`(n)	$c_8 \times \log(L) \times 10^L$
`decomp(n)`	$c_9 \times 10^{L/2}$
$\Phi(n)$	$c_{10} \times 10^{L/2}$

Les algorithmes à utiliser pour générer des clés ont un temps de calcul (on parle de complexité) proportionnel à L^k (*i.e.* polynômial) alors que les algorithmes pour factoriser des entiers ont une complexité en $10^{k \times L}$ donc exponentielle. Bref pour des entrées suffisamment grandes le temps de calcul pour générer des entiers premiers reste raisonnable alors que le temps de calcul pour factoriser un entier devient irréaliste (plusieurs fois l'age supposé de l'univers ... cf figure 1.8).

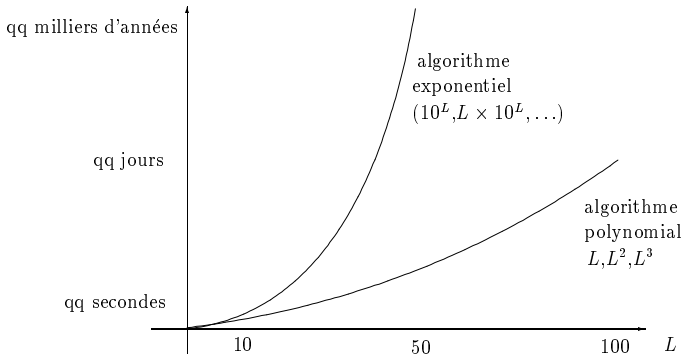

FIG. 1.8 – *Comparaison des complexités des différents algorithmes*

La taille des clés de chiffrement est d'ailleurs clairement indiquée dans les textes de loi en France. Ces textes concernant la cryptographie ont été révisés par les décrets 99-199 et 99-200 du 17 mars 1999 :

- la création et l'utilisation de logiciels dont l'algorithme utilise une la clé inférieure à 40bits est totalement libre
- l'utilisation de logiciels dont l'algorithme utilise une clé inférieure à 128 bits est libre à condition que:
  - l'éditeur du logiciel ait fait une déclaration
  - logiciels utilisés dans un cadre privé exclusivement
- les logiciels de cryptographie utilisant une clé à plus de 128 bits doivent faire l'objet d'une déclaration pour pouvoir être utilisés, par exemple :
  - Pour les cartes bancaires les clés RSA (à 700 bits) doivent être communiqué à un organisme public (un tiert de confiance)
  - L'utilisation de GNUPG (version libre de RSA) pour l'échange de mails et la signature électronique à été autorisée

1.5.5 Protection des cartes bancaires

Que se passe-t-il lorsqu'on introduit sa carte bleue dans un distributeur automatique ou un terminal de paiement chez un commerçant? Ce système[2] repose sur plusieurs protocoles cryptographiques bien plus complexes que ne le laisse imaginer la simple saisie de votre code personnel à 4 chiffres ...

La carte à puce française est une sorte de petit ordinateur, elle possède :

- un processeur (assez peu puissant) qui permet d'effectuer des calculs
- une mémoire "vive" pour enregistrer l'historique des transactions
- une mémoire "morte" où sont stockées les informations nécessaires à l'authentification et au chiffrement des données (dont une partie est en lecture seule et l'autre en lecture cachée)

Dans les autres pays, Japon ou Etats-Unis par exemple, les cartes de paiement sont toujours dépourvues de puces. Examinons maintenant ce qui se passe lors d'un paiement (en France) :

- Tout commence par **l'authentification de la carte** qui repose sur le système **RSA** : une valeur de signature VS a été calculée lors de la fabrication de la carte à partir d'informations relatives au propriétaire (nom, numéro de carte, date de validité ...) d'un fonction de hachage (voir chapitre suivant) et de (S,N) la partie secrète de la clé RSA associée à la carte à partir de la formule :

  $$VS = RSA(hash(infos),S,N)$$

  de telle sorte que si (P,N) est la partie publique de la clé RSA associée à la carte on a :

  $$hash(infos) = RSA(VS,P,N)$$

2. inventé par deux français (Roland Moreno et Michel Ugon) vers la fin des années 1970!

Il faut savoir que pour les CB le modulo N est un entier possédant entre 768 et 1024 bits (produit de 2 facteurs premiers) et $P = 3$. L'authentification de la carte se fait donc simplement par :

$infos$ = informations personnelles (nom, numéro de carte, date de validité ...)
$y_1 = hash(infos)$ (calcul d'une empreinte de taille fixée)
VS = valeur de signature de la carte
(P,N) = clé RSA (partie Publique) associé à la carte
$y_2 = RSA(VS,P,N)$
VS = valeur de signature
si $y_1 = y_2$ **alors** OK
sinon $FAILED$
fin si

La sécurité de la carte repose sur le fait que VS est écrit sur la partie non-lisible de sa mémoire morte. Il n'est donc pas possible de cloner la carte

Il est très important de remarquer aussi que cette partie est donc faite *hors-ligne* !

- Ensuite on procède à **l'identification du porteur de la carte** : c'est là que l'on saisit son *code confidentiel* à 4 chiffres. Ce code étant stocké[3] sur la partie lisible de la mémoire "morte" le processeur de la carte peut le comparer à la valeur saisie et transmet sa réponse au terminal (si tout se passe bien le terminal affiche "code bon").
- Au delà d'un certain montant une troisième protection est ajoutée il s'agit d'une **Authentification en ligne**[4] : un serveur distant envoie à la carte une valeur aléatoire x. La carte calcule $y = DES(x,K)$ avec une clé secrète K, inscrite dans la partie cachée de la mémoire "morte" de la carte, et renvoie le résultat au serveur distant qui donne ou pas l'autorisation[5].

L'affaire Humpich En 1998, Serge Humpich a fabriqué (à partir de pièces en vente libre) une fausse carte bancaire capable de passer les contrôles d'un terminal de paiement. C'est ce qu'on a appelé *"l'affaire Serge Humpich"* qui a fait la une des journaux pendant quelques semaines. En fait Serge Humpich n'avait contourné que deux des trois systèmes de sécurité :

- à partir d'un lecteur de carte à puce il a extrait les informations publique d'une véritable carte à puce : informations personnelles ($infos$), la partie publique N (et $P = 3$) de la clé RSA.
- en 1998, le clés publiques N utilisé par le Groupement Inter-Bancaire (GIE) avait pour taille 320 bits (valeur fixée depuis 1990). A cette époque, factoriser un tel entier n'était plus impossible (le record se situait à 512 bits), et Humpich, en utilisant simplement un logiciel japonais de factorisation, a réussi à

3. ce code est aussi stocké sur la piste magnétique et les terminaux de paiement étrangers qui n'utilisent pas la puce de nos cartes authentifient le porteur de carte à partir de la piste magnétique

4. le terminal de paiement indique alors qu'il demande une "autorisation" au serveur de la banque

5. la clé secrète K est donc un secret partagé entre le possesseur de la carte et le serveur distant.

factoriser le N, à calculer la clé secrète S et à en déduire la valeur de signature VS de la carte!

- Il ne lui restait plus qu'à fabriquer une *"yes card"* c'est à dire une carte à puce qui répond OK au premier protocole d'identification du porteur de la carte quelque soit le code à 4 chiffre saisit au terminal (c'est possible puisque cette vérification est faite sur le processeur de la carte, il suffit donc de modifier l'opération de contrôle qui est codée sur cette puce) et à partir de là il pouvait cloner la carte bancaire en utilisant la valeur de signature VS calculée (pour un coût faible car un graveur de carte à puce vaut 50 Euros et une carte vierge s'achète dans les 15 Euros).

La dernière sécurité, elle, est toujours restée valide, mais comme cette vérification DES n'est faites qui si l'achat dépasse un certain montant la carte pouvait fonction dans beaucoup de cas. Le GIE aurait du relever bien plus tôt la taille des clés, les 320 bits étaient déjà suffisants en 1990. l est désormais de 768 bits). Il existe bien d'autres moyens de falsifier une carte bleu, le plus simple étant de calculer un faux numéro à 16 chiffres[6] (et le cryptogramme qui va avec) et d'effectuer des paiement sur internet avec ce seul numéro.

6. Avant 2001, ce numéro était indiqué sur les tickets de carte bancaire! Depuis qu'il en a été supprimé il faut en généré un qui soit valide car tous les nombres à 16 chiffres ne donnent pas un numéro de carte bancaire valide ... il existe des sites Internet où on explique comment calculer un numéro de carte bancaire valable!

1.6 Autres Applications

1.6.1 les fonctions de Hachage

Définition 1.6.1 *Une fonction de Hachage est une fonction qui prend en entrée un texte (de longueur quelconque) et renvoie un nombre dans $\mathbb{Z}/n\mathbb{Z}$, appelé* empreinte*, de telle sorte qu'étant donné une empreinte il soit très difficile de trouver un texte ayant cette empreinte.*

Par exemple les fonctions qui associent à un texte la valeur du premier bit (ou des n premiers bits) ou du dernier bit (ou des n dernier bits) ne sont pas des fonctions de hachage car il est facile de trouver un texte qui redonne une empreinte donnée.

Proposition 1.6.2 *une fonction de hachage à valeur dans $\mathbb{Z}/n\mathbb{Z}$ est une fonction f de $\mathbb{N}$ dans $\mathbb{Z}/n\mathbb{Z}$, cette fonction est donc forcément non-injective, c'est à dire que :*

$$\exists x,y \in \mathbb{N},\ \ x \neq y \ \ et \ \ f(x) = f(y)$$

Étant donné une fonction de hachage, trouver deux textes ayant la même empreinte s'appelle trouver une collision*.*

Puisqu'il y a plus de n textes différents dans l'ensemble de départ la fonction ne peut être injective. L'intérêt d'une fonction de hachage est que lorsque l'on change ne serait ce qu'un seul caractère du texte son empreinte est complètement modifiée. Prenons l'exemple de la fonction *md5* (nommée ici *md5sum*), l'empreinte d'un texte simple est donnée ci-dessous sous forme hexadécimale (128 bits $\rightarrow$ 32 chiffres en base 16 car $2^4 = 16$ et $128 = 32 \times 16$) :

```
philippe@pc1 ~
$ echo 'Calcul d'une empreinte avec l'algorithme md5.' > testmd5

philippe@pc1 ~
$ md5sum testmd5
af23f039a3770dfbf43c311b5208f892 *testmd5

philippe@pc1 ~
$ echo 'calcul d'une empreinte avec l'algorithme md5.' > testmd5

philippe@pc1 ~
$ md5sum testmd5
a0096eb83b1422e146f3701c7036b616 *testmd5
```

On voit que le simple fait de changer le 'c' en 'C' change totalement l'empreinte du message. L'une des applications principales des fonctions de hachage concerne la transmission sécurisée des mots de passe. En effet les mots de passe ne doivent pas être conservés dans une une base de donnée (sinon quelqu'un peut éventuellement y accéder) mais il faut absolument pouvoir vérifier si quelqu'un connaît ou pas un mot de passe donné. Pour cela il suffit de conserver seulement l'empreinte du mot de passe (celle-ci ne permet pas de retrouver le mot de passe) et de demander

à un utilisateur d'envoyer seulement l'empreinte de son mot de passe (si elle est interceptée ce n'est pas grave elle ne permet pas de retrouver le mot de passe) et de comparer que les deux empreintes correspondent. Cependant il faut être sûr qu'on ne peut pas à partir de l'empreinte trouver un mot de passe donnant cette empreinte (une collision). Hélas pour le *md5*, en 2005 une méthode a été trouvé pour produire à partir de l'empreinte (et en un temps raisonnable) un mot de passe donnant la même empreinte.

```
/* gentil.c */

#include <stdlib.h>
#include <stdio.h>


int main(){
    printf("Je suis gentil !\n");
}
```

```
/* mechant.c */

#include <stdlib.h>
#include <stdio.h>


int main(){
    printf("Je suis mechant !\n");
}
```

après les avoir compilés on obtient deux fichiers qui ont des signatures différents :

```
philippe@pc1 ~
$ gcc -o gentil ./gentil.c

philippe@pc1 ~
$ ./gentil
Je suis gentil !

philippe@pc1 ~
$ gcc -o mechant ./mechant.c
philippe@pc1 ~
$./mechant
Je suis mechant !

philippe@pc1 ~
$ md5sum gentil.exe
c4ff40f0ec801b9b0d41832396bbcfe8 *gentil.exe

philippe@pc1 ~
$ md5sum mechant.exe
```

```
c92f90479e7f1f247108b3cfc4895874 *mechant.exe
```

mais en utilisant le crack disponible à l'adresse [7] on peut créer 2 archives `*.bin` qui ont la même empreinte et la même taille, seul un `diff` permet de voir que les deux fichiers sont différents :

```
philippe@pc1 ~
$ md5sum gentil.bin
41916ea01aaf54d222a6d538d5bb65a8 *gentil.bin

philippe@pc1 ~
$ md5sum mechant.bin
41916ea01aaf54d222a6d538d5bb65a8 *mechant.bin

philippe@pc1 ~
$ ls -l *.bin
-rw-r--r-- 1 philippe Aucun 17882 Feb 14 18:23 gentil.bin
-rw-r--r-- 1 philippe Aucun 17882 Feb 14 18:23 mechant.bin

philippe@pc1 ~
$ diff gentil.bin mechant.bin
Files gentil.bin and mechant.bin differ
```

on peut donc distribuer une archive vérolée à la place de l'archive saine car affichant le md5 certifié! Il en est de même pour d'autre algorithmes de hachage comme le SHA, SHA1. Il en existe d'autres sûr (SHA256 et SHA512) mais plus lourds à utiliser. Des méthodes comme celle du *grain de sel* permettent cependant de contrer ce genre de faille du MD5. Pour information on trouvera la description du fonctionnement du MD5 en fin de cours.

1.6.2 Codes correcteurs d'erreurs

Voir exercice de TD.

La fonction de hashage MD5

Soit un message W qui va être le message à chiffrer composé de l bits, $MD5(W)$ va produire une empreinte à 128 bits (dans $\mathbb{Z}/2^{128}\mathbb{Z}$) :

- Complétion du message :
  - On complète le message par un 1 et autant de 0 que nécessaires pour obtenir une longueur l' congrue à 448 modulo 512 (même si la longueur du message d'origine est congrue a 448 modulo 512) .
  - On ajoute à ce message la valeur de l codée sur 64 bits, ce qui donne un message de longueur $l'' = l' + 64 \equiv 448 + 64 \equiv 512 \equiv 0 \bmod 512$.
- On découpe le message en N blocs de 16 mots de 32 bits (car $16 \times 32 = 512$ et l'' est un multiple de 512) puis on calcule l'empreinte :

remarque : on commence par initialiser certaines valeurs (sur 64 bits) :
$r[0..15] := 7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22$
$r[16..31] := 5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20$
$r[32..47] := 4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23$
$r[48..63] := 6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21$
$h0 := 0x67452301;\ h1 := 0xEFCDAB89;\ h2 := 0x98BADCFE;\ h3 := 0x10325476$
$a := h0;\ b := h1;\ c := h2;\ d := h3$
remarque : MD5 utilise des sinus d'entiers pour ses constantes
pour $i = 0$ **jusqu'à** 63 **faire**
  $k[i] := floor(abs(sin(i + 1)) \times 2^{32})$
fin faire
pour chaque bloc w de 512 bits du message W **faire**
  pour $i = 0$ **jusqu'à** 63 **faire**
    si $0 \leq i \leq 15$
      alors $f := (b \wedge c) \vee ((\neg b) \wedge d);\ g := i$
      sinon si $16 \leq i \leq 31$
        alors $f := (d \wedge b) \vee ((\neg d) \wedge c);\ g := (5 \times i + 1) \bmod 16$
        sinon si $32 \leq i \leq 47$
          alors $f := b \oplus c \oplus d;\ g := (3 \times i + 5) \bmod 16$
          sinon si $48 \leq i \leq 63$
            alors $f := c \oplus (b \vee (\neg d));\ g := (7 \times i) \bmod 16$
          fin si
        fin si
      fin si
    fin si
    $temp := d;\ d := c;\ c := b;$
    $b := ((a + f + k[i] + w[g])\ leftrotate\ r[i]) + b$
    remarque : leftrotate fait un décalage de $r[i]$ bits vers la gauche
    $a := temp$
  fin faire
  $h0 := h0 + a;\ h1 := h1 + b;\ h2 := h2 + c;\ h3 := h3 + d$
fin faire
$empreinte := (h0;\ h1;\ h2;\ h3)$

1.7 Aspect historiques

Les éléments historiques cités ci-après proviennent essentiellement de [2] du très bon livre [1].

1.7.1 l'arithmétique dans l'antiquité

Euclide : il vécu de -330 à -275 (environ) à Alexandrie. Il est considéré comme le père des mathématiques moderne. Il rédigea *Les éléments*, synthèse exhaustive des connaissances des mathématiques de son époque ou apparaissent clairement les notions d'axiomes, définitions, démonstrations .... Les volumes VII, VIII et IX des *éléments* sont consacrés à l'arithmétique et aux nombres premiers et il faudra près de 20 siècles pour voir apparaître des progrès par rapport au travail d'Euclide!
Ératosthène : il naquit en -276 à Cyrène (Libye) et fit ses études de mathématiques à Alexandrie et Athènes avant de devenir directeur de la grande bibliothèque d'Alexandrie et précepteur du fil du roi Ptolémé III. Outre le crible qui porte son nom il est connu pour avoir calculé la circonférence de la Terre ($\approx$ 40000 km) avec une extrême précision pour les moyens de l'époque (mesure de la longueur de l'ombre d'un mat!!). Devenu aveugle il se suicide vers -194.

1.7.2 Pierre de Fermat

Pierre de Fermat (1601-1665) est issu d'une famille bourgeoise aisée, il travaille comme conseiller au parlement de Toulouse. Fermat semble contredire tous les clichés sur les génies : il n'est pas précoce (il commence à s'intéresser aux mathématiques après 30 ans), il travaille en amateur et ne s'intéresse pas à la publication de ses résultats. C'est dans ses loisirs qu'il aime lire la traduction en latin des œuvres de Diophante ($III^{\text{ième}}$ siècle après J-C) qu'il annote de démonstrations ou de généralisations. Ses communications avec les mathématiciens de son époque se limitent à des échanges épistolaires avec Mersenne, Pascal, Huygens. C'est son fils qui publira les annotations laissées par son père dans un livre, *arithmetica*, publié après sa mort. On y trouve en particulier l'énoncé du *grand théorème de Fermat*:

l'équation $a^n + b^n = c^n$ n'a pas de solution entières $(a,b,c) \neq (0,0,0)$ si $n \geq 3$

avec l'annotation : *"J'ai découvert une démonstration merveilleuse mais je n'ai pas la place de la mettre dans la marge"*[7]... on mettra plus de trois siècles avant de trouver une démonstration valable (A. Wiles 1993).

L'idée du crible quadratique est aussi due à Fermat qui l'utilise conjointement au test basé sur le petit théorème de Fermat pour trouver des facteurs premiers de grands entiers. C'est au début des années 80 que le crible quadratique à pu être amélioré grace à une méthode qui "accélère" la recherche des "bons" x à tester. À partir de 1980 ce type de crible a permi de fatoriser des clés jusqu'à une taille de 10^{140}.

7. méthode à déconseiller pour le DS!

1.7.3 Marin Mersenne

Marin Mersenne (1588-1648) moine de l'ordre Minimes enseigne la philosophie au couvent de l'Annonciade à Paris. Opposant à l'alchimie, l'astrologie et autres sciences mystiques il défend les théories des grands scientifiques de l'époque (Descartes, Galilée,...) dont il traduit et diffuse les œuvres. Il fait une recherche exhaustive des entiers p , compris entre 2 et 257, tels $2^p - 1$ soit premier et dont la liste est : $p = 2;\ 3;\ 5;\ 7;\ 13;\ 17;\ 19;\ 31;\ 61;\ 89;\ 107;\ 127$. Il se trompa seulement pour $p = 67{,}257$ (qui ne conduisent pas à des nombres premiers) et ne trouva pas ceux correspondant à $p = 61{,}89$ et 107[8]. Sans autre moyen que le calcul manuel il réussit la factorisation $2^{37} - 1 = 223 \times 616318177$. La popularité des nombres de Mersenne vient de leur lien avec les nombres parfaits[9] cités dans la bible. Euclide à montré à leur sujet que si $2^p - 1$ est premier alors $2^{p-1}(2^p - 1)$ est nombre parfait.

1.7.4 Cryptographie

Le DES fût utilisé par l'administration des États Unis pour le chiffrement de toutes les données non liées à la défense nationale à partir de 1977. L'intérêt du DES était qu'à condition de répéter un nombre de fois suffisant la méthode Feistel on pouvait se permettre d'utiliser des clés relativement petites (56 bits dans les années 80!) assurant à la fois une grande rapidité et un bon degré de sûreté. Techniquement on peut préciser que le texte était découpé en blocs de 32 bits, chacun de ces bloc était transformé en blocs de 48 bits (par une méthode de duplication de bits) auquel on appliquait la méthode de Feistel avant de retransformer les blocs de 48 bits obtenus en blocs de 32 bits. Le tout était répété 16 fois de suite avec 16 clés différentes! Dans les années 90 la puissance des ordinateurs progressant, les autorités Américaines ont du relever la norme de sécurité du DES (qui devint le TDES avec des clés de taille 112 bits), mais avec cette nouvelle norme l'algorithme était devenu trop lent. Il dut être remplacé, il y a quelques années, par un nouveau cryptosystème le AES.

On entend par cryptosystème à *c*lé publique un système de chiffrement/déchiffrement où le chiffrement reposerai sur une clé publique pouvant être divulguée (par exemple dans un annuaire ouvert à tout le monde) le déchiffrement reposant quant à lui sur une clé privée connue seulement de celui qui décode! Le risque de voir la sécurité du cryptosystème mise à mal par un "transfuge" devient alors nulle. L'idée d'un cryptosystème à *c*lé publique est apparue dans les années 60. les agences de renseignement Britannique focalisèrent leur efforts sur cette question vers 1969 (sous l'impulsion de Ellis) et la solution, trouvée en 1973 (par un certain Cocks), resta classée "secret défense". C'est en 1977 que trois Américains (Rivest, Shamir et Adi) ont proposés la première mise en œuvre d'un cryptosystème à clé publique : Le RSA. Les Américains déposèrent un brevet et créèrent la société *R*SA security pour l'exploiter. Les Britannique ayant conservé tout le secret de leur découverte perdirent donc leur avantage dans l'exploitation commerciale de cette découverte ...

8. Quoique certains prétendent que la confusion entre $p = 61$ et $p = 67$ proviennent d'une coquille dans le texte de Mersenne!
9. égaux à la somme de leurs diviseurs comme $6 = 1 + 2 + 3$ ou $28 = 1 + 2 + 4 + 7 + 14$.

1.7.5 RSA data-security

Comme nous l'avons vu dans ce cours, bien que la multiplication $a.b$ de deux nombres a et b constitue une opération simple, la factorisation, qui consiste à retrouver les nombres a et b à partir du produit $a.b$, peut se révéler très délicate, en particulier lorsque a et b sont des nombres premiers de plusieurs dizaines de chiffres. La sécurité du système de chiffrement RSA (du nom de ses concepteurs R. Rivest, A. Shamir, L. Adleman) repose précisément sur cette difficulté puisqu'un message codé selon ce principe ne peut être lu que si l'on connaît une des clés, c'est-à-dire l'un des nombres a ou b.

Afin de s'assurer que la longueur des clefs de chiffrement demeure suffisamment grande pour garantir une sécurité optimale, les laboratoires RSA Security lancèrent en 1991 des défis RSA en soumettant de grands nombres à factoriser. Ceux-ci ont été baptisés RSA suivi du nombres de décimales correspondant. RSA-100, et RSA-110, de 100 et 110 chiffres respectivement, furent par exemple les premiers nombres RSA à être factorisés en 1991 et 1992. On pensait à la fin des années 70 qu'il faudrait des millions d'années pour y parvenir. Mais de nouvelles méthodes (comme le crible quadratique ou les cribles sur des corps de nombres généralisés) alliées à la puissance de calcul accrue des ordinateurs ont changé la donne.

Pour encourager la recherche, une récompense était offerte à la première équipe qui trouverait la factorisation d'un nombre RSA :

- 10 000 dollars pour une clé inférieure à 768 bits ($n \approx 1.553\ 10^{231}$)
- ...
- 200 000 dollars pour des clés de 2048 bits ($n \approx 3.232\ 10^{616}$) .

Pour plus d'informations voir :*http://www.rsa.com/rsalabs/node.asp?id=2094*.
L'opération s'est arrêtée fin 2007, mais il reste encore de nombreux défis ouverts ... citons le nombre RSA-193 ou le monstre RSA-2048 pour lequel 200 000 dollars étaient promis. Pour comprendre la complexité du problème examinons les moyens mis en œuvre pour factoriser le **RSA-155**. Le 22 août 1999, une équipe du CWI (*Institut national de recherche en mathématiques et en science informatique d'Amsterdam*) annonça qu'elle avait cassé une clé numérique de 512 bits (155 chiffres décimaux) du système cryptographique RSA. Il s'agit de la factorisation d'un nombre de 155 chiffres en deux nombres premiers de 78 chiffres :

```
     10941 7386415705 2742180970 7322040357
6120037329 4544920599 0913842131 4763499842
8893478471 7997257891 2673324976 2575289978
1833797076 5372440271 4674353159 3354333897
=
  10263959 2829741105 7720541965 7399167590
0716567808 0380668033 4193352179 0711307779
x
  10660348 8380168454 8209272203 6001287867
9207958575 9892915222 7060823719 3062808643
```


Quelques détails technique sur le RSA-155 :

- Initialisation du calcul 9 semaines (sur une seule machine!)
- Calcul du système d'équations à résoudre (répartit sur 300 machines)
- Tri des équations pendant 1 mois (sur une seule machine encore) jusqu'à obtenir le système d'équations à 6 699 191 lignes et 6 711 336 colonnes.
- Résolution du système d'équations sur un ordinateur *CrayC916* durée 224heures, espace disque: 2 gigaoctets .
- Simplification du résultat 39 heures.

La dernière factorisation en date est celle du RSA-200. Une équipe de la "Bundesamt für Sicherheit in der Informationstechnik" (BSI, agence fédérale pour la sécurité des techniques de l'information) a annoncé le 9 mai 2005 la factorisation d'un nombre à 200 chiffres, connu sous le nom de RSA-200. Cette équipe s'était déjà illustrée en décembre 2003 pour la factorisation d'un nombre à 174 chiffres (RSA-174). La factorisation de RSA-200 trouvée par l'équipe allemande est la suivante:

```
2799783391 1221327870 8294676387 2260162107 0446786955
4285375600 0992932612 8400107609 3456710529 5536085606
1822351910 9513657886 3710595448 2006576775 0985805576
1357909873 4950144178 8631789462 9518723786 9221823983
=
3532461934 4027701212 7260497819 8464368671 1974001976
2502364930 3468776121 2536794232 0005854795 6528088349
x
7925869954 4783330333 4708584148 0059687737 9758573642
1996073433 0341455767 8728181521 3538140930 4740185467
```

1.7.6 Le bug du P*ntium

T. Nicely est professeur au Lynchburg College, en Virginie. Il travaille sur les nombres premiers et cherche à obtenir des tables précises des nombres premiers inférieurs à 10^{15} pour évaluer la validité de certaines conjectures (sur les écarts entre 2 nombres premiers consécutifs).

Pour faire ce travail il utilisa le réseau informatique de son université lorsque celui-ci est peu ou pas utilisé (nuit, vacances ...) en distribuant les calculs sur une centaine de machines : des PC équipés de "vieux" processeurs 486 ou du "flambant neuf" Pentium.

Pour éviter toute erreur de calcul tous les programmes étaient exécutés en "double" sur 2 machines différentes ... et le 13 juin 1994 : un décompte incorrect des nombres premiers inférieurs à 20 10^{12} est identifié par comparaison avec des tables!

Juin-septembre 1994 : T. Nicely réécrit l'ensemble de ses programmes et les teste un à un. Il est alors sûr de son programme et identifie le compilateur C de Borland comme responsable de l'erreur de décompte.

4 octobre 1994 : T. Nicely note une nouvelle erreur. Comme le programme est sûr, il localise rapidement l'erreur (en qq jours!). Il établit que c'est le calcul de 1/824 633 702 441 qui est faux en langage C++ avec le Pentium. L'erreur provient donc soit du compilateur C++, soit du Pentium.

18 octobre 1994 : L'erreur est reproduite avec le langage Power Basic et le logiciel Quatro-Pro sur un ordinateur équipé d'un Pentium (ce qui met le compilateur C++ hors de cause), mais disparaît quand le processeur mathématique du Pentium est désactivé : c'est donc lui le coupable.

21-22 octobre 1994 : Des tests complémentaires confirment le défaut du Pentium.

Le 24 octobre 1994 : Intel est prévenu. L'affaire commence...

...durant les mois qui suivirent le problème du Pentium a été décortiqué, et des cas d'erreurs très simples ont été repérés. Si vous souhaitez tester votre machine le pire cas d'erreur a été découvert par Tim Coe, de la Société Vitesses Semiconductors, faite la division suivante :

$$4195835.0/3145727.0 = \begin{cases} 1.333820449136... & OK \\ 1.333739068902... & \text{Pentium défectueux} \end{cases}$$

L'erreur du Pentium est 10000 fois supérieure à celle d'une calculette valant 50 francs : il ne s'agit pas d'une erreur normale d'arrondi.

Une analyse détaillée montre que c'est avec des nombres proches de l'unité que le Pentium se trompe or ce sont ce type de divisions qui ont la plus grande probabilité d'apparaître dans les calculs. Au total le Pentium se trompe avec une probabilité de 1/3000!

1.7.7 l'histoire de GNUPG

GNUPG est le premier logiciel libre publié donnant accès au moyens moderne de cryptographie. Il est le fruit du travail d'un ingénieur informaticien et militant pacifiste à ses heures : *Philip R. Zimmermann* (surnommé sur Internet "PRZ"). C'est le 5 Juin 1991 que la version PGP 1.0 est diffusée en urgence sur des BBS téléphoniques (l'ancêtre du réseau Internet) répartis dans tous les USA alors qu'un projet de loi US menace d'interdire la cryptographie libre. Peu après, la société RSA Data Security Inc. somme Phil Zimmermann de stopper la diffusion de PGP 1.0 (qui utilise l'algorithme breveté RSA sans avoir acheté de licence). Malgré tout P. Zimmerman continue le développement et publie PGP2.0 en septembre 1992. Mais le 14 Septembre 1993 le bureau des Douanes américaines de San Jose, Californie (USA) délivre une assignation contre "PGP, Philip Zimmermann, and anyone or any entity acting on behalf of Philip Zimmermann for the time period June 1, 1991 to the present" au sujet d'une violation de la réglementation sur les ventes d'armes et l'exportation de matériel technologique! En effet la réglementation US assimile les logiciels de cryptographie à une "arme" non-exportable. L'affaire durera 2 ans et demi, jusqu'au 11 Janvier 1996 où les Douanes US annonce que les poursuites contre Phil Zimmermann sont arrêtées. Aucune explication n'est fournie (le communiqué fait 7 lignes!). Le créateur de PGP n'est plus menacé, mais comme tous les logiciels de cryptographie, son programme reste interdit d'exportation des USA. Malgré tout le

10 août 1997 Ståle Schumacher publie en Norvège une version internationale freeware (PGP 5.0i) pour Unix, basée pour la première fois sur une exportation légale des USA du code-source sous la forme ... *d'un livre imprimé*! Ce livre a ensuite été scanné, puis passé à la reconnaissance de caractères pour reconstituer le code-source en dehors des USA et compiler légalement cette version de GNUPG. Voir le cite [8] pour plus de détails. Vous pouvez télécharger et installer GNUPG sur votre machine depuis le site [9].

Bibliographie

[1] Jean-Paul Delahaye, *Merveilleux nombres premiers*, Belin-pour la science.
[2] Bertrand Hauchecorne, Daniel Surrateau *Des mathématiciens de A à Z*
[3] *http://www.rsa-security.org*
[4] *http://w2.eff.org/awards/coop-prime-rules.php*
[5] *http://www.mersenne.org*
[6] *http://www.utm.edu/research/primes/index.html*
[7] *http://www.ghostsinthestack.org/article-22-exploitation-des-collisions-md5.html*
[8] *http://openpgp.vie-privee.org/histoire.htm*
[9] *http://gnupg.org/*