



**Master 1 de Cryptologie et Sécurité Informatique**
**Équipe-projet GRACE**
**Stage estival 2017**

---

# Étude du cryptosystème de Chor-Rivest

Rémi Clarisse

Tuteurs : Daniel Augot et Luca De Feo

# Table des matières


**1 Le problème du sac à dos** **5**
1.1 Densité d'un sac à dos . . . . . 5

**2 Le théorème Bose-Chowla** **6**

**3 Le cryptosystème Chor-Rivest** **7**
3.1 Description . . . . . 7
3.2 Exemple . . . . . 9
3.2.1 Génération des clés . . . . . 9
3.2.2 Chiffrement d'un message . . . . . 10
3.2.3 Déchiffrement du message . . . . . 11
3.3 Transformation des messages . . . . . 12

**4 Les attaques avec divulgation partielle de la clé privée** **12**
4.1 Symétrie dans la clé privée . . . . . 13
4.2 Attaque due à B. Chor et R. Rivest ($g$ et $d$ connus) . . . . . 13
4.3 Attaque due à S. Vaudenay ($g$ connu) . . . . . 14
4.4 Attaque due à B. Chor et R. Rivest ($t$ et $d$ connus) . . . . . 14
4.5 Attaque due à O. Goldreich ($t$ connu) . . . . . 14
4.5.1 Telle que proposée par B. Chor et R. Rivest . . . . . 14
4.5.2 Telle que proposée par S. Vaudenay . . . . . 15
4.6 Attaque due à A. Odlyzko ($\sigma$ connue) . . . . . 15

**5 L'attaque de Brickell** **16**

**6 La cryptanalyse de Vaudenay** **17**
6.1 Attaque sachant $g^{(p^h-1)/(p^r-1)}$ et $\sigma$ . . . . . 18
6.2 Attaque sachant $g^{(p^h-1)/(p^r-1)}$ . . . . . 18
6.3 Test pour trouver les $g^{(p^h-1)/(p^r-1)}$ . . . . . 20
6.4 Sur l'usage de tous les $c_i$ . . . . . 20

**7 Le calcul de logarithmes dans un groupe abélien** **21**
7.1 Recherche exhaustive . . . . . 22
7.2 Méthode de Pohlig et Hellman . . . . . 22
7.3 Algorithme pas de bébé, pas de géant . . . . . 25

**8 La méthode rho de Pollard** **26**
8.1 Discussion liminaire : partitionner, marcher, chercher . . . . . 26
8.2 Méthode rho de Pollard originale dans $(\mathbb{Z}/r\mathbb{Z})^\times$ . . . . . 27
8.3 Méthode rho de Pollard originale généralisée . . . . . 29

**9 La réduction de réseau Lenstra-Lenstra-Lovász** **30**

**10 L'attaque Lagarias-Odlyzko sur les sacs à dos à densité faible** **30**

**A Le b.a.-ba de la théorie des groupes** **31**

**B L'anatomie des corps** **31**
B.1 Anneaux : homomorphismes et caractéristique . . . . . 31
B.2 Endomorphisme de Frobenius . . . . . 32
B.3 Sous-corps premiers . . . . . 33
B.4 Extensions de corps . . . . . 33
B.5 Corps finis . . . . . 35

**C La théorie de Kummer** **35**

**D La théorie d'Artin-Schreier** **35**

**E Le paradoxe des anniversaires** **35**

# Introduction

En 1984, Benny Chor et Ronald L. Rivest ont proposé un cryptosystème [2] à clé publique dont la sécurité repose sur un problème $\mathbb{NP}$-complet : le problème du sac à dos. Il a été cryptanalysé en 1998 par Serge Vaudenay.

Nous nous proposons ici d'étudier ce cryptosystème.

# 1 Le problème du sac à dos

Le problème du sac à dos est le problème $\mathbb{NP}$-complet suivant :

**Définition 1.1** (Problème du sac à dos)**.** Soient $a_0, \ldots, a_{n-1}$, $h$ et $S$ des entiers naturels. Le *problème du sac à dos* est alors le problème de décision : existe-t-il $n$ entiers naturels $x_0, \ldots, x_{n-1}$ de sorte que

$$\sum_{i=0}^{n-1} x_i a_i = S \quad \text{et} \quad \sum_{i=0}^{n-1} x_i \leqslant h \quad ?$$

$h$ est appelé le *poids* du sac à dos.

Un autre problème, toujours $\mathbb{NP}$-complet, demande seulement à ce que les $x_i$ soient 0 ou 1, sans se soucier du poids du sac : c'est le problème *subset sum*.

Les cryptosystèmes se basant sur un problème de sac à dos reposent sur le fait qu'il est difficile de trouver une solution à un tel problème, même quand il en existe une.

Dans de tels cryptosystèmes, l'utilisateur publie un ensemble $a_i$, tel que $0 \leqslant i \leqslant n-1$, et un poids $h$. Un message clair consiste alors en un vecteur d'entiers $\mathbf{m} = (x_0, \ldots, x_{n-1})$ dont le poids est inférieur à $h$. Le chiffré de $\mathbf{m}$ est noté :

$$E(\mathbf{m}) = \sum_{i=0}^{n-1} x_i a_i.$$

L'utilisateur choisi les éléments $a_i$ de telle façon à lui permettre de résoudre assez facilement l'instance du problème du sac à dos pour recouvrer le message clair à partir de $E(\mathbf{m})$. Il s'agit alors d'une *trapdoor* : une instance à laquelle il est aisé de trouver une solution.

Une propriété appréciée des cryptosystèmes de type sac à dos est la facilité de chiffrement : il n'y a qu'à faire des additions. Vu que nous souhaitons coder un message via une instance de sac à dos, il serait préférable qu'il y ait une seule et unique solution au problème, pour pouvoir recouvrer le message de manière univoque.

## 1.1 Densité d'un sac à dos

Pour attaquer les cryptosystèmes reposant sur le problème du sac à dos, Jeffrey C. Lagarias et Andrew M. Odlyzko ont introduit la définition :

**Définition 1.2** (Densité d'un sac à dos)**.** La *densité d'un sac à dos* est le rapport entre le nombre d'éléments dans le sac sur la taille binaire des ces éléments.

Concrètement, si un sac à $n$ éléments, que ceux-ci sont positifs et que son élément le plus grand est $s \in \mathbb{N}^*$, alors la densité de ce sac à dos est $n/\log_2(s)$.

Il s'avère qu'il existe une attaque sur les sac à dos à densité faible ($< 0.94$). Elle a été proposée par J. Lagarias et A. Odlyzko en 1985, et utilise la réduction de réseaux, comme l'algorithme LLL.

B. Chor et R. Rivest se sont donc inspirés du théorème de Bose-Chowla pour construire leur cryptosystème.

## 2 Le théorème Bose-Chowla

Donnons rapidement une définition pour rendre le propos plus clair :

**Définition 2.1** (*h-fold sum*)**.** Soit $h$ un entier naturel. On appelle *h-fold sum* d'une suite, la somme d'exactement $h$ termes de celle-ci, répétition possible.

Nous nous posons alors la question : étant donné $n$ et $h$ deux entiers naturels, existe-t-il une suite d'entiers naturels $(a_0, \ldots, a_{n-1})$ telle que toutes ces *h-fold sums* soient distinctes ?

Il est facilement de construire de telle suite si les $a_i$ croissent exponentiellement en $n$. Par exemple, la suite $(1, h, h^2, \ldots, h^{n-1})$ à cette propriété (mais cela ne marche pas pour $h+1$ termes, puisque $h^2 + h \cdot 1 = (h+1) \cdot h$). Cependant, le sac à dos associé a une densité faible, le rendant vulnérable à l'attaque de J. Lagarias et A. Odlyzko.

En 1962, Raj Chandra Bose et Sarvadaman Chowla ont trouvé une façon de construire une telle suite avec $1 \leqslant a_i \leqslant n^h - 1$ pour tout $0 \leqslant i \leqslant n-1$. Cela permet d'instancier un sac à dos avec une densité acceptablement proche de 1. Nous donnons ici une version modifiée du théorème, qui instruit sur l'inspiration du cryptosystème de Chor-Rivest.

**Théorème 2.2** (Théorème de Bose-Chowla)**.** *Soient $p$ un nombre premier et $h \geqslant 2$ un entier. Il existe une suite $(a_i)_{0 \leqslant i \leqslant p-1}$ d'entiers telle que :*

1. *pour tout $0 \leqslant i \leqslant p-1$,*

$$1 \leqslant a_i \leqslant p^h - 1,$$

2. *si $(x_0, \ldots, x_{p-1})$ et $(y_0, \ldots, y_{p-1})$ sont deux vecteurs distincts d'entiers naturels de poids inférieur à $h$, alors*

$$\sum_{i=0}^{p-1} x_i a_i \neq \sum_{i=0}^{p-1} y_i a_i.$$

*Démonstration.* Plaçons nous dans les corps finis $\mathrm{GF}(p)$ et $\mathrm{GF}(p^h)$, extension de degré $h$ de $\mathrm{GF}(p)$. Numérotons les éléments de $\mathrm{GF}(p) = \{\alpha_0, \ldots, \alpha_{p-1}\}$.

Soit $t$ un élément de $\mathrm{GF}(p^h)$ algébrique de degré $h$ sur $\mathrm{GF}(p)$, i.e. le polynôme minimal de $t$ dans $\mathrm{GF}(p)[x]$ est de degré $h$.

Soit $g$ un élément primitif de $\mathrm{GF}(p^h)$, i.e. un élément générateur du groupe multiplicatif $\mathrm{GF}(p^h)^\times$, à savoir

$$\mathrm{GF}(p^h)^\times = \{g^e : 0 \leqslant e \leqslant p^h - 1\}.$$

Considérons l'ensemble des translatés de $t$ par un élément de $\mathrm{GF}(p)$, à savoir :

$$t + \mathrm{GF}(p) = \{t + \alpha_i : \alpha_i \in \mathrm{GF}(p)\}.$$

Soit $a_i := \log_g(t + \alpha_i)$, où $\alpha_i \in \mathrm{GF}(p)$, le logarithme de $t + \alpha_i$ en base $g$ dans $\mathrm{GF}(p^h)$. Alors les $a_i$ sont tous des entiers de l'intervalle $[1, p^h - 1]$ et, montrons par l'absurde que, toutes les *h-fold sums* de la suite $(a_i)_{0 \leqslant i \leqslant p-1}$ sont distinctes.

Soient $(x_0, \ldots, x_{p-1})$ et $(y_0, \ldots, y_{p-1})$ deux vecteurs d'entiers naturels satisfaisant à :

$$(x_0, \ldots, x_{p-1}) \neq (y_0, \ldots, y_{p-1}),$$

$$\sum_{i=0}^{p-1} x_i \leqslant h, \qquad \sum_{i=0}^{p-1} y_i \leqslant h,$$
$$\sum_{i=0}^{p-1} x_i a_i = \sum_{i=0}^{p-1} y_i a_i.$$

Il s'en suit l'égalité dans $\mathrm{GF}(p^h)$ :

$$g^{\sum x_i a_i} = g^{\sum y_i a_i},$$

et donc :

$$\prod_{i=0}^{p-1} (g^{a_i})^{x_i} = \prod_{i=0}^{p-1} (g^{a_i})^{y_i}.$$

En utilisant l'égalité $g^{a_i} = t + \alpha_i$ et en ne considérant que les $x_i$ et $y_i$ non-nuls, nous obtenons :

$$(t+\beta_1)^{x_{i_1}} (t+\beta_2)^{x_{i_2}} \cdots (t+\beta_l)^{x_{i_l}} = (t+\gamma_1)^{y_{j_1}} (t+\gamma_2)^{y_{j_2}} \cdots (t+\gamma_m)^{y_{j_m}},$$

où $\{\beta_1, \ldots, \beta_l\}$ et $\{\gamma_1, \ldots, \gamma_m\}$ sont deux sous-ensembles non-vides de $\mathrm{GF}(p)$ de cardinal au plus $h$. Les deux polynômes de part et d'autre de la dernière égalité, sont donc distincts, unitaires, de degré au plus $h$ et à coefficients dans $\mathrm{GF}(p)$. Ainsi, en faisant la différence de ces deux polynômes, nous obtenons que $t$ est racine d'un polynôme non-nul de degré au plus $h-1$ et dont les coefficients sont dans $\mathrm{GF}(p)$. Cela contredit le fait que $t$ est algébrique de degré $h$ sur $\mathrm{GF}(p)$. □

*Remarque* 2.3. D'après la preuve, il est clair que, pour $l \leqslant h$, les *l-fold sums* de la suite $(a_i)_{0 \leqslant i \leqslant p-1}$ sont distinctes dans $\mathbb{Z}$, mais aussi modulo $p^h - 1$.

La supposition "$p$ est un nombre premier" peut être remplacée par "$p$ est une puissance de nombre premier", sans que cela n'affecte le théorème et sa preuve.

Nous exposerons en section 7 diverses façon de calculer les logarithmes dans un corps fini.

# 3 Le cryptosystème Chor-Rivest

## 3.1 Description

Soit $p^h$ une puissance de nombre premier. Considérons le corps fini $\mathrm{GF}(p^h)$ dont il est supposé que sa représentation est publique, i.e. il existe un polynôme unitaire $P(x)$ public de degré $h$ irréductible sur $\mathrm{GF}(p)$ et les éléments de $\mathrm{GF}(p^h)$ sont vus comme des polynômes modulo $P(x)$. Ainsi :

$$\mathrm{GF}(p^h) = \frac{\mathrm{GF}(p)[x]}{(P(x))}.$$

Notons $a = x \pmod{P(x)}$, la classe de $x$ modulo $P(x)$ dans $\mathrm{GF}(p^h)$, et choisissons $(1, a, a^2, \ldots, a^{h-1})$ comme base du $\mathrm{GF}(p)$-espace vectoriel $\mathrm{GF}(p^h)$. Ainsi $\mathrm{GF}(p^h) = \mathrm{GF}(p)[a]$. Considérons aussi une numérotation publique $\alpha$ du sous-corps premier $\mathrm{GF}(p)$, i.e.

$$\{\alpha_0, \ldots, \alpha_{p-1}\} = \mathrm{GF}(p) \subset \mathrm{GF}(p^h).$$

Les éléments, choisis aléatoirement, de la clé privée – à garder secrète – consistent en :

— un élément $t \in \mathrm{GF}(p^h)$ de degré algébrique $h$ sur $\mathrm{GF}(p)$ (nous notons $\mu(x) \in \mathrm{GF}(p)[x]$ le polynôme minimal de $t$ sur $\mathrm{GF}(p)$),
— un générateur $g$ du groupe $\mathrm{GF}(p^h)^\times$,
— un entier $d$ tel que $0 \leqslant d \leqslant p^h - 2$,
— une permutation $\sigma$ de l'ensemble $\{0, \ldots, p-1\}$.

La clé publique – à être publiée – est alors les :

$$c_i = d + \log_g \left(t + \alpha_{\sigma(i)}\right) \pmod{p^h - 1},$$

quel que soit $i$ tel que $0 \leqslant i \leqslant p-1$.

Étant donné qu'il faut calculer des logarithmes discrets pour fabriquer la clé publique, les paramètres choisis doivent permettre d'effectuer ces logarithmes facilement dans $\mathrm{GF}(p^h)$. B. Chor et R. Rivest ont suggéré de prendre un entier premier $p$ relativement petit et un exposant $h$ friable, de sorte que nous puissions appliquer l'algorithme de Pohlig-Hellman (voir section 7). En l'occurrence, ils ont proposé de se placer dans les corps tels que $\mathrm{GF}(197^{24})$, $\mathrm{GF}(211^{24})$, $\mathrm{GF}(243^{24})$ et $\mathrm{GF}(256^{25})$.

L'espace des messages est l'ensemble des chaînes de $p$-bits de poids de Hamming égal à $h$. C'est-à-dire que le message à chiffrer doit d'abord être encodé en une chaîne de bits $m = [m_0 \cdots m_{p-1}]$ telle que $m_0 + \cdots + m_{p-1} = h$.

L'espace des chiffrés est $\mathbb{Z}/(q-1)\mathbb{Z}$ et le chiffré de $m$ est :

$$E(m) = \sum_{i=0}^{p-1} m_i c_i \pmod{q-1}.$$

Pour déchiffrer, nous calculons :

$$G(t) := g^{E(m)-hd}$$

comme un polynôme en $t$ à coefficients dans $\mathrm{GF}(p)$ et de degré au plus $h-1$. Comme $g$ est primitif dans $\mathrm{GF}(p^h)$, i.e. l'ordre de $g$ est $q-1$, nous cherchons à déterminer $E(m) - hd$ modulo $q-1$ :

$$\begin{aligned}
E(m) - hd &\equiv \left(\sum_{i=0}^{p-1} m_i c_i\right) - hd \pmod{q-1} \\
&\equiv \left(\sum_{i=0}^{p-1} m_i \left(d + \log_g \left(t + \alpha_{\sigma(i)}\right)\right)\right) - hd \pmod{q-1} \\
&\equiv \left(hd + \sum_{i=0}^{p-1} m_i \log_g \left(t + \alpha_{\sigma(i)}\right)\right) - hd \pmod{q-1} \\
&\equiv \sum_{i=0}^{p-1} m_i \log_g \left(t + \alpha_{\sigma(i)}\right) \pmod{q-1}
\end{aligned}$$

D'où, l'égalité dans $\mathrm{GF}(p^h)$ :

$$G(t) = g^{E(m)-hd} = \prod_{i=0}^{p-1} \left(t + \alpha_{\sigma(i)}\right)^{m_i} = \prod_{m_i=1} \left(t + \alpha_{\sigma(i)}\right).$$

L'élément $G(t)$ est exprimé dans la base $(1, t, t^2, \ldots, t^{h-1})$ du GF($p$)-espace vectoriel GF($p^h$), où $t = x \pmod{\mu(x)}$. Ainsi :

$$G(x) \equiv \prod_{m_i=1} \left(x + \alpha_{\sigma(i)}\right) \pmod{\mu(x)}.$$

Il existe donc un polynôme $\lambda(x) \in \mathrm{GF}(p)[x]$ tel que :

$$G(x) = \lambda(x)\mu(x) + \prod_{m_i=1} \left(x + \alpha_{\sigma(i)}\right).$$

En résonnant sur les degrés et du fait que $\mu(x)$ et $\prod \left(x + \alpha_{\sigma(i)}\right)^{m_i}$ soient unitaires, nous déduisons que $\lambda(x) = -1$, dont il découle l'égalité de polynômes :

$$G(x) + \mu(x) = \prod_{m_i=1} \left(x + \alpha_{\sigma(i)}\right).$$

La factorisation de ce dernier permet de recouvrer le message $m$.

## 3.2 Exemple

Prenons $p := 17$ et $h := 6$, ainsi $p^h = 24\,137\,569$.

```
sage: p = 17
sage: h = 6
sage: q = p ** h
```

Le corps fini GF($17^6$) construit par `Sage` est, en l'occurrence,

$$\frac{\mathbb{F}_{17}[x]}{(P(x))}, \text{ où } P(x) := x^6 + 2x^4 + 10x^2 + 3x + 3 \in \mathbb{F}_{17}[x].$$

```
sage: K.<a> = FiniteField(q)
sage: P = a.minimal_polynomial()
```

La numérotation $\alpha$ du sous-corps premier choisie est :

$$(\alpha_0, \ldots, \alpha_{16}) = (2, 12, 4, 1, 0, 10, 7, 8, 15, 16, 3, 5, 13, 9, 11, 6, 14)$$

```
sage: alpha = [K(i) for i in range (p)]
sage: shuffle (alpha)
```

### 3.2.1 Génération des clés

Nous prenons un élément $t \in \mathrm{GF}(17^6)$ de degré algébrique 6 sur GF(17). Comme plus haut, nous notons $a = x \pmod{P(x)}$, le $t$ sélectionné est alors :

$$t := 9a^5 + 16a^4 + 10a^3 + 3a^2 + 12a + 12,$$

de polynôme minimal :

$$\mu(x) := x^6 + 9x^5 + 8x^4 + 14x^3 + x^2 + 11x + 6 \in \mathrm{GF}(17)[x].$$

```
sage: while True :
sage:     t = K.random_element()
sage:     if t.minimal_polynomial().degree() == h :
sage:         break
sage: mu = t.minimal_polynomial()
```

Ensuite, comme élément primitif de GF($17^6$), nous prenons :

$$g := 2a^5 + 5a^4 + 14a^3 + 2a^2 + 10a + 16.$$

```
sage: while True :
sage:     g = K.random_element()
sage:     if g.multiplicative_order() == q - 1 :
sage:         break
```

L'entier $d$ est pris égal à 1 530 545.

```
sage: d = randint (0, q - 2)
```

Enfin, nous choisissons une permutation $\sigma$ de l'ensemble $\{0, \ldots, 16\}$ :

$$\sigma := (0, 10, 8, 5, 1, 6, 14)(2, 3, 9, 16, 13)(4, 12, 7, 15)(11).$$

```
sage: s = Permutations([i for i in range (p)]).random_element()
```

Nous avons fini de fabriquer la clé privée ! Reste à construire la clé publique :

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $c_0$ | := | 21 667 185 | $c_1$ | := | 3 210 064 | $c_2$ | := | 6 070 281 |
| $c_3$ | := | 3 093 929 | $c_4$ | := | 19 945 987 | $c_5$ | := | 294 610 |
| $c_6$ | := | 4 230 580 | $c_7$ | := | 18 951 770 | $c_8$ | := | 7 364 695 |
| $c_9$ | := | 23 348 812 | $c_{10}$ | := | 7 918 908 | $c_{11}$ | := | 3 562 855 |
| $c_{12}$ | := | 6 735 636 | $c_{13}$ | := | 13 077 876 | $c_{14}$ | := | 11 303 489 |
| $c_{15}$ | := | 22 106 426 | $c_{16}$ | := | 18 193 975 | | | |

```
sage: c = [mod (d + log (t + alpha[s[i]], g), q - 1)
                                                for i in range (p)]
```

### 3.2.2 Chiffrement d'un message

Maintenant, donnons nous un message à chiffrer de longueur 17 et de poids de Hamming 6 :

$$m := [m_0 \cdots m_{16}] = [00100101100100100].$$

```
sage: m = [1 for i in range (h)] + [0 for i in range (p - h)]
sage: shuffle (m)
```

Chiffrons $m$ :

$$e := 23\,410\,132.$$

```
sage: e = mod (sum ([m[i]*c[i] for i in range (p)]), q-1)
```

### 3.2.3 Déchiffrement du message

Nous souhaitons écrire $g^{e-hd}$ comme un polynôme en $t$. Or `Sage` nous donne $g^{e-hd}$ comme polynôme en $a$ :

$$g^{e-hd} = a^5 + 11a^3 + 9a^2 + 15a + 1.$$

Pour parvenir à exprimer $g$ comme nous le souhaitons, il faut effectuer un changement de base du GF(17)-espace vectoriel GF($17^6$) : passer de la base $\mathcal{A} := (1, a, a^2, \ldots, a^{h-1})$ à la base $\mathcal{T} := (1, t, t^2, \ldots, t^{h-1})$. La matrice de passage facile à calculer est celle qui passe de la base $\mathcal{T}$ à la base $\mathcal{A}$ : il suffit d'écrire dans `Sage` les différentes puissances de $t$, et `Sage` les exprime en fonction des puissances de $a$. Voici cette matrice :

$$\begin{bmatrix} 1 & 12 & 7 & 11 & 14 & 0 \\ 0 & 12 & 9 & 12 & 6 & 12 \\ 0 & 3 & 8 & 3 & 7 & 12 \\ 0 & 10 & 0 & 15 & 15 & 14 \\ 0 & 16 & 4 & 0 & 13 & 3 \\ 0 & 9 & 15 & 10 & 5 & 6 \end{bmatrix}$$

Celle qui nous intéresse est son inverse, la matrice de passage de la base $\mathcal{A}$ à la base $\mathcal{T}$ :

$$\begin{bmatrix} 1 & 5 & 10 & 12 & 15 & 11 \\ 0 & 8 & 8 & 0 & 11 & 5 \\ 0 & 13 & 13 & 9 & 5 & 1 \\ 0 & 9 & 10 & 3 & 11 & 9 \\ 0 & 12 & 6 & 2 & 16 & 8 \\ 0 & 7 & 16 & 2 & 13 & 11 \end{bmatrix}$$

```
sage: V = K.vector_space()
sage: M = Matrix (GF(p),
                      [V(t ** i) for i in range (h)]).transpose()
sage: Minv = M.inverse()
```

Comme $g^{e-hd}$ vaut dans la base $\mathcal{A}$ le vecteur $(1, 15, 9, 11, 0, 1)$, il est facile d'obtenir l'égalité :

$$g^{e-hd} = 10t^5 + 9t^4 + 12t^3 + 4t^2 + 10t + 3.$$

Notons $G(x) := 10x^5 + 9x^4 + 12x^3 + 4x^2 + 10x + 3 \in \mathrm{GF}(17)[x]$. Ainsi le message est recouvré en factorisant le polynôme :

$$G(x) + \mu(x) = x^6 + 2x^5 + 9x^3 + 5x^2 + 4x + 9,$$

ce que nous faisons :

$$G(x) + \mu(x) = (x+1)(x+2)(x+5)(x+6)(x+10)(x+12) \in \mathrm{GF}(17)[x].$$

Nous pouvons alors déterminer les $\alpha_i$ utilisés et nous savons alors que le bit $m_{\sigma^{-1}(i)}$ est à 1, alors que les autres sont nuls.

```
sage: A.<x> = PolynomialRing (GF(p))
sage: Q = A(list (Minv * V(g ** (e - h * d)))) + A(mu)
sage: beta = [p - Q.roots()[i][0] for i in range (h)]
```

Pour facilité l'explication, notons $\beta_i$, pour $i$ tel que $0 \leqslant i \leqslant 5$, les éléments de GF(17) tels que $-\beta_i$ est racine de $G(x) + \mu(x)$, les $\beta_i$ étant tous distincts. Par exemple, $\beta_0 = 1$, $\beta_1 = 2$, ..., $\beta_5 = 12$. Ainsi, pour tout $i$ tel que $0 \leqslant i \leqslant 5$, il faut déterminer l'indice $j$, où $0 \leqslant j \leqslant 16$, tel que $\alpha_j = \beta_i$, et alors nous déduisons qu'il y a un 1 au $\sigma^{-1}(j)$ bit du message.

Par exemple, pour $\beta_0 = 1$ : nous cherchons d'abord l'indice $j$ tel que $\alpha_j = 1$, à savoir $j = 3$. Puis nous déterminons $\sigma^{-1}(3) = 2$. Et nous avons bien $m_2 = 1$. Idem, pour $\beta_2 = 5$ : $\alpha_{11} = 5$, $\sigma^{-1}(11) = 11$ et nous constatons en effet que $m_{11} = 1$.

```
sage: sInv = [s.index(i) for i in range (p)]
sage: message = [0 for i in range (p)]
sage: for k in beta :
sage:     message[sInv [alpha.index(k)]] = 1
```

### 3.3 Transformation des messages

Nous avons supposé que tous les messages chiffrés via le cryptosystème étaient de longueur $p$ et de poids $h$. Cependant, nous pouvons transformer un message binaire quelconque pour qu'il ait cette forme.

Étant donné un message binaire, découpons le en blocs de $\left\lfloor \log_2 \binom{p}{h} \right\rfloor$ bits. Chaque bloc est la représentation binaire d'un nombre $n$ tel que $0 \leqslant n < \binom{p}{h}$. Pour faire correspondre ces nombres à des vecteurs binaires de poids $h$, nous utilisons l'ordre total induit par l'ordre lexicographique sur les vecteurs et l'ordre sur les entiers. Si $n$ est supérieur à $\binom{p-1}{h-1}$, le premier bit correspondant au vecteur est mis à 1, sinon il est mis à 0. Puis $p$ et $h$ sont actualisés, et nous itérons jusqu'à ce que les $p$ bits soient déterminés :

**Algorithme 1** Algorithme de transformation d'un bloc de message en vecteur

**Entrée :** $p$, $h$ et un bloc $n$ du message d'origine tel que $0 \leqslant n < \binom{p}{h}$
**Sortie :** un vecteur $y$ de taille $p$ et de poids $h$

1: **pour** $i$ allant de 1 à $p$ **faire**
2:  **si** $n \geqslant \binom{p-i}{h}$ **alors**
3:  $y_i \leftarrow 1$
4:  $n \leftarrow n - \binom{p-i}{h-1}$
5:  $h \leftarrow h - 1$
6:  **sinon**
7:  $y_i \leftarrow 0$
8:  **fin si**
9: **fin pour**
10: **retourner** $y$

La transformation inverse est donnée par l'algorithme suivant :

## 4 Les attaques avec divulgation partielle de la clé privée

Nous examinons les attaques possibles proposées dans le papier de B. Chor et R. Rivest [2, Section VII, A] et dans celui de S. Vaudenay [8, Section 4].

**Algorithme 2** Algorithme de transformation d'un vecteur en bloc de message

**Entrée :** $p$, $h$ et un vecteur $y$ de taille $p$ et de poids $h$
**Sortie :** un bloc $n$ du message d'origine

```
n ← 0
pour i allant de 1 à p faire
    si y_i = 1 alors
        n ← n + C(p−i, h)
        h ← h − 1
    fin si
fin pour
retourner n
```

Notons aussi que K. Huber [4] propose une attaque plus alambiquée lorsque l'élément primitif $g$ est connu.

Commençons par une observation de Serge Vaudenay [8, Section 3].

## 4.1 Symétrie dans la clé privée

Dans le cryptosystème de Chor-Rivest, nous choisissons la clé privée de façon aléatoire puis, à partir de celle-ci, nous calculons la clé publique. Le système repose sur la difficulté de trouver une clé secrète à partir de la clé publique. Cependant, nous remarquons qu'il y a plusieurs clés privées *équivalentes*, à savoir qu'il existe plusieurs jeux de clés privées qui donnent la même clé publique.

Détaillons. Nous pouvons remplacer $t$ et $g$ par leur puissance $p$-ième, la clé publique reste inchangée car :

$$\log_{g^p}\left(t^p + \alpha_{\sigma(i)}\right) = \frac{1}{p}\log_{g^p}\left(\left(t + \alpha_{\sigma(i)}\right)^p\right) = \log_g\left(t + \alpha_{\sigma(i)}\right).$$

Nous pouvons aussi remplacer $(t, \alpha_\sigma)$ par $(t+u, \alpha_\sigma - u)$, pour tout $u \in \mathrm{GF}(p)$. Et enfin, nous pouvons remplacer $(t, d, \alpha_\sigma)$ par $(ut, d - \log_g(u), u\alpha_\sigma)$, quel que soit $u \in \mathrm{GF}(p)^\times$. Cela donne donc au moins $hp(p-1)$ clés privées équivalentes. Il s'agit alors de déterminer une de ces clés.

*Remarque* 4.1. Ce qu'il faut garder à l'esprit est qu'il est possible de choisir – d'imposer – les valeurs de deux images de la permutation. À savoir, si $\sigma$ est la permutation originale du système, nous déterminons une clé privée équivalente dont la permutation est $\pi$ : pour $i \neq j$, nous construisons $\pi$ telle que

$$\pi(0) = i, \quad \pi(1) = j,$$

$$\forall k : 0 \leqslant k \leqslant p-1, \quad \alpha_{\pi(k)} = u \cdot \alpha_{\sigma(k)} - v,$$

$$\text{où} \quad u = \frac{\alpha_i - \alpha_j}{\alpha_{\sigma(0)} - \alpha_{\sigma(1)}} \in \mathrm{GF}(p)^\times \quad \text{et} \quad v = \frac{\alpha_i\alpha_{\sigma(1)} - \alpha_j\alpha_{\sigma(0)}}{\alpha_{\sigma(0)} - \alpha_{\sigma(1)}} \in \mathrm{GF}(p).$$

## 4.2 Attaque due à B. Chor et R. Rivest ($g$ et $d$ connus)

Sachant $d$, nous pouvons calculer $(b_0, \ldots, b_{p-1}) := (c_0 - d, \ldots, c_{p-1} - d)$. Soit $t' := g^{b_0}$, de sorte que :

$$t' = g^{\log_g(t + \alpha_{\sigma(0)})} = t + \alpha_{\sigma(0)}.$$

D'où $t - t' \in \mathrm{GF}(p)$, et

$$\{t + \alpha_i : \alpha_i \in \mathrm{GF}(p)\} = \{t' + \alpha_i : \alpha_i \in \mathrm{GF}(p)\}.$$

Donc pour chaque $\alpha_i \in \mathrm{GF}(p)$, il existe un unique $\alpha_{\pi(i)} \in \mathrm{GF}(p)$ vérifiant

$$g^{b_{\pi(i)}} = t' + \alpha_i.$$

En utilisant $t'$, $g$, $\pi$ et $d$, l'attaquant peut faire le même déchiffrement que le destinataire légitime.

## 4.3 Attaque due à S. Vaudenay ($g$ connu)

Supposons que $\sigma(0) = i$ et $\sigma(1) = j$, par la symétrie dans la clé privée, nous savons qu'un choix arbitraire de $(i, j)$ convient. Calculons ensuite :

$$g^{c_0 - c_1} = \frac{t + \alpha_i}{t + \alpha_j},$$

et nous pouvons alors déterminer $t$.

## 4.4 Attaque due à B. Chor et R. Rivest ($t$ et $d$ connus)

Choisissons un générateur arbitraire $g'$ de $\mathrm{GF}(p^h)^\times$ et calculons

$$b_i := \log_{g'}(t + \alpha_i).$$

En tant qu'ensembles, nous avons

$$\{c_0 - d \ldots, c_{p-1} - d\} = L\{b_0 \ldots, b_{p-1}\},$$

où l'égalité est modulo $p^h - 1$. Les nombres $L$ et $p^h - 1$ sont premiers entre eux et $L$ satisfait à l'égalité $g = g'^L$. Une fois que nous avons recouvré $L$, nous avons fini, car $g = g'^L$ et nous pouvons reconstruire $\sigma$, obtenant ainsi toutes les pièces de la clé privée.

Si un des $b_i$, disons $b_0$, est premier avec $p^h - 1$, alors $L$ est l'un des

$$(d - c_j)b_0^{-1} \pmod{p^h - 1},$$

pour un $0 \leqslant j \leqslant p-1$. Sinon, l'attaquant peut calculer $L$ modulo chaque facteur puissance de nombre premier et les combiner en utilisant le théorème des restes chinois.

## 4.5 Attaque due à O. Goldreich ($t$ connu)

### 4.5.1 Telle que proposée par B. Chor et R. Rivest

Choisissons un générateur arbitraire $g'$ de $\mathrm{GF}(p^h)^\times$ et calculons

$$b_i := \log_{g'}(t + \alpha_i).$$

En tant qu'ensembles, nous avons

$$\{c_0 - c_0, c_1 - c_0, \ldots, c_{p-1} - c_0\} = L\{b_0 - b_0, b_1 - b_0, \ldots, b_{p-1} - b_0\},$$

où l'égalité est modulo $p^h - 1$. Ensuite, nous procédons comme dans l'attaque précédente, sous-section 4.4.

### 4.5.2 Telle que proposée par S. Vaudenay

Supposons que $\sigma(0) = i$ et $\sigma(1) = j$, par la symétrie dans la clé privée, nous savons qu'un choix arbitraire de $(i, j)$ convient. Choisissons $g'$ un élément primitif de GF($p^h$). Calculons $\log_{g'}(t+\alpha_i)$ et $\log_{g'}(t+\alpha_j)$, et alors les équations :

$$c_0 = d + \frac{\log_{g'}(t+\alpha_i)}{\log_{g'}(g)},$$

$$c_1 = d + \frac{\log_{g'}(t+\alpha_j)}{\log_{g'}(g)},$$

permettent de déterminer $d$ et $\log_{g'}(g)$.

## 4.6 Attaque due à A. Odlyzko ($\sigma$ connue)

Il faut trouver une combinaison linéaire de la forme

$$\sum_{i=1}^{p-1} x_i \left(c_{\sigma^{-1}(i)} - c_{\sigma^{-1}(0)}\right) = 0,$$

avec des entiers $x_i$ relativement petits. Cela peut être fait via l'algorithme LLL. Nous pouvons nous attendre à $|x_i| \leqslant B$, où $B \sim p^{h/(p-1)}$.

Soit $\alpha$ un élément primitif de GF($p^h$) :

$$\begin{aligned}
&\sum_{i=1}^{p-1} x_i \left(c_{\sigma^{-1}(i)} - c_{\sigma^{-1}(0)}\right) = 0 \\
\Leftrightarrow &\prod_{i=1}^{p-1} \alpha^{x_i\left(c_{\sigma^{-1}(i)} - c_{\sigma^{-1}(0)}\right)} = 1 \\
\Leftrightarrow &\prod_{i=1}^{p-1} \alpha^{x_i(\log_g(t-\alpha_i) - \log_g(t-\alpha_0))} = 1 \\
\Leftrightarrow &\prod_{i=1}^{p-1} \alpha^{x_i \log_g(t-\alpha_i)} = \prod_{i=1}^{p-1} \alpha^{x_i \log_g(t-\alpha_0)} \\
\Leftrightarrow &\prod_{i=1}^{p-1} (t-\alpha_i)^{x_i} = \prod_{i=1}^{p-1} (t-\alpha_0)^{x_i}
\end{aligned}$$

Ainsi $t$ est racine du polynôme

$$\prod_{i=1}^{p-1} (y-\alpha_i)^{x_i} - \prod_{i=1}^{p-1} (y-\alpha_0)^{x_i} \in \mathrm{GF}(p)[y],$$

dont un algorithme comme celui de Rabin, permet de trouver les racines.

# 5 L'attaque de Brickell

Dans le papier de B. Chor et R. Rivest [2, Section VII, A.5], Ernest Brickell propose une attaque où rien de la clé privée n'est connu.

Le but de cette attaque est de trouver une équation polynomiale de petit degré satisfaite par $g$. En utilisant un réseau bien choisi, il est possible de trouver des entiers $x_i$, la plus part nuls, tels que les deux équations

$$\sum_{i=0}^{p-1} x_i c_i \equiv 0 \pmod{p^h - 1},$$

$$\sum_{i=0}^{p-1} x_i \equiv 0 \pmod{p^h - 1},$$

soient satisfaites. Notons $b_i := \log_g(t + \alpha_{\sigma(i)})$. La deuxième équation garantie :

$$\begin{aligned} g^{\sum_{i=0}^{p-1} x_i c_i} &= g^{\sum_{i=0}^{p-1} x_i (d+b_i)} \\ &= g^{\sum_{i=0}^{p-1} x_i b_i} g^{d \sum_{i=0}^{p-1} x_i} \\ &= g^{\sum_{i=0}^{p-1} x_i b_i} \end{aligned}$$

et donc par la première égalité :

$$g^{\sum_{i=0}^{p-1} x_i b_i} = \prod_{i=0}^{p-1} \left(t + \alpha_{\sigma(i)}\right)^{x_i} = 1.$$

Définissons, pour tout $0 \leqslant i \leqslant p-1$, les entiers $x_i^+$ et $x_i^-$ par :

$$x_i^+ := \begin{cases} x_i & \text{si } x_i \geqslant 0, \\ 0 & \text{sinon.} \end{cases}$$

$$x_i^- := \begin{cases} -x_i & \text{si } x_i \leqslant 0, \\ 0 & \text{sinon.} \end{cases}$$

Pour $\pi$ une permutation de l'ensemble $\{0, \ldots, p-1\}$, soit $r_\pi(y) \in \mathrm{GF}(p)[y]$ le polynôme suivant :

$$r_\pi(y) := \prod_{i=0}^{p-1} (y + \alpha_{\pi(i)})^{x_i^+} - \prod_{i=0}^{p-1} (y + \alpha_{\pi(i)})^{x_i^-}.$$

Notons $\ell$ le nombre de $x_i$ non-nuls. Chaque application $\{i : x_i \neq 0\} \to \mathrm{GF}(p)$ bijective donne un polynôme $r_\pi(y)$ différent. Seul le "bon" polynôme aura $t$ comme racine. Donc en moyenne il faut essayer $(1/2)(p!/(p-\ell)!)$ applications pour obtenir le bon polynôme. Pour chaque telle application, il faut trouver dans $\mathrm{GF}(p^h)$ les racines de $r_\pi(y)$, puis nous les testons grâce à l'attaque de O. Goldreich exposée en sous-section 4.5.

**Proposition 5.1.** *Pour effectuer l'attaque de Ernest Brickell, il existe un algorithme de complexité* $\mathcal{O}(p^{2\sqrt{h}} h^2 \log_2(p))$ *opérations arithmétiques dans* $\mathrm{GF}(p)$.

# 6 La cryptanalyse de Vaudenay

Nous présentons ici l'attaque de Serge Vaudenay [8]. Elle s'appuie sur la forte friabilité de $h$, i.e. sur l'existence de nombreux sous-corps de $\mathrm{GF}(p^h)$.

Commençons par donner quelques outils pour l'explication de l'attaque.

**Proposition 6.1.** *Pour tout facteur $r$ de $h$, il existe un générateur $g_{p^r}$ du groupe multiplicatif du sous-corps $\mathrm{GF}(p^r)$ de $\mathrm{GF}(p^h)$ et un polynôme $Q(x) \in \mathrm{GF}(p^r)[x]$ de degré $h/r$ et tel que $-t$ en est une racine, et pour tout $0 \leqslant i \leqslant p-1$, nous avons :*

$$Q\left(\alpha_{\sigma(i)}\right) = (g_{p^r})^{c_i}.$$

*Démonstration.* Soit

$$Q(x) = (g_{p^r})^d \prod_{i=0}^{h/r-1} \left(x + t^{p^{ri}}\right), \quad \text{où } g_{p^r} := \prod_{i=0}^{h/r-1} g^{p^{ri}},$$

$g_{p^r}$ peut être considérer comme la norme de $g$ considéré dans l'extension de corps $\mathrm{GF}(p^r) \subseteq \mathrm{GF}(p^h)$, ainsi $(g_{p^r})^{p^r} = g_{p^r}$ et $g_{p^r}$ est générateur de $\mathrm{GF}(p^r)^\times$ car $g$ est primitif dans $\mathrm{GF}(p^h)$. Nous remarquons que $Q\left(x^{p^r}\right) = Q(x)^{p^r}$, ainsi $Q(x) \in \mathrm{GF}(p^r)[x]$, en effet :

$$\begin{aligned}
Q\left(x\right)^{p^r} &= (g_{p^r})^{dp^r} \prod_{i=0}^{h/r-1} \left(x + t^{p^{ri}}\right)^{p^r} \\
&= (g_{p^r})^d \prod_{i=0}^{h/r-1} \left(x^{p^r} + t^{p^{ri} p^r}\right) \\
&= (g_{p^r})^d \prod_{i=1}^{h/r} \left(x^{p^r} + t^{p^{ri}}\right) \\
&= Q\left(x^{p^r}\right),
\end{aligned}$$

car $t^{p^{r(h/r)}} = t^{p^h} = t = t^{p^0}$. Cela montre aussi que $Q(-t) = 0$. Nous avons aussi que $Q(x)$ est de degré $h/r$. Et enfin, pour $0 \leqslant j \leqslant p-1$, calculons $(g_{p^r})^{c_i}$ :

$$\begin{aligned}
(g_{p^r})^{c_j} &= \left(\prod_{i=0}^{h/r-1} g^{p^{ri}}\right)^{c_j} \\
&= \prod_{i=0}^{h/r-1} (g^{c_j})^{p^{ri}} \\
&= \prod_{i=0}^{h/r-1} g^{dp^{ri}} \left(\alpha_{\sigma(j)} + t\right)^{p^{ri}} \\
&= (g_{p^r})^d \prod_{i=0}^{h/r-1} \left(\alpha_{\sigma(j)} + t^{p^{ri}}\right) \\
&= Q\left(\alpha_{\sigma(j)}\right)
\end{aligned}$$

Cela conclut la démonstration de la proposition. □

Comme $h/r$ est assez petit, il est peu probable qu'il existe d'autres solutions $(g_{p^r}, Q)$, et $g_{p^r}$ est donc essentiellement unique.

**Notation :** Pour $r$ divisant $h$, notons

$$g_{p^r} := g^{(p^h-1)/(p^r-1)}.$$

## 6.1 Attaque sachant $g^{(p^h-1)/(p^r-1)}$ et $\sigma$

Supposons connus $g_{p^r} = g^{(p^h-1)/(p^r-1)}$, pour $r$ divisant $h$ (comme dans la proposition 6.1), ainsi que la permutation $\sigma$. Nous pouvons alors interpoler le polynôme $Q(x)$ de la proposition 6.1, avec $h/r+1$ pairs $(\alpha_{\sigma(i)}, (g_{p^r})^{c_i})_i$. Cela donne un polynôme de degré $h/r$ dont les racines sont les conjugués de $-t$. Nous résolvons l'attaque en testant l'opposé chaque racine grâce à l'attaque de O. Goldreich exposée en sous-section 4.5.

## 6.2 Attaque sachant $g^{(p^h-1)/(p^r-1)}$

Supposons connu l'élément $g_{p^r}$, la norme de l'élément primitif $g$ considéré dans l'extension $\mathrm{GF}(p^r) \subseteq \mathrm{GF}(p^h)$, comme dans la proposition 6.1.

Soient $(i_0, \ldots, i_{h/r})$ une famille de $h/r+1$ indices distincts entre 0 et $p-1$. Par la proposition 6.1, nous pouvons interpoler le polynôme $Q(x)$ en les $\alpha_{\sigma(i_j)}$ :

$$\begin{aligned} Q(x) &= \sum_{j=0}^{h/r} Q\left(\alpha_{\sigma(i_j)}\right) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}} \\ &= \sum_{j=0}^{h/r} (g_{p^r})^{c_{i_j}} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}}, \end{aligned}$$

ce qui mène aux égalités : quel que soit $i$ tel que $0 \leqslant i \leqslant p-1$,

$$(\dagger) \qquad (g_{p^r})^{c_i} = \sum_{j=0}^{h/r} (g_{p^r})^{c_{i_j}} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}}.$$

En fait, nous pouvons même écrire : quel que soit $i$ tel que $0 \leqslant i \leqslant p-1$,

$$(\ddagger) \qquad (g_{p^r})^{c_i} - (g_{p^r})^{c_{i_0}} = \sum_{j=1}^{h/r} \left((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}}\right) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}}.$$

En effet, soit $i$ parmi $i_0, \ldots, i_{h/r}$ :

$$(g_{p^r})^{c_i} - (g_{p^r})^{c_{i_0}} = \sum_{j=1}^{h/r} \left((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}}\right) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}},$$

car

$$\sum_{j=0}^{h/r} \left( \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}} \right) = 1.$$

Ainsi, nous avons l'égalité de polynôme :

$$Q(x) - (g_{p^r})^{c_{i_0}} = \sum_{j=1}^{h/r} ((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}}) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}},$$

car les deux polynômes sont de degré $h/r$ et sont égaux sur un ensemble de $h/r+1$ éléments : d'où l'égalité lorsque $x$ est substitué par $\alpha_{\sigma(i)}$, où $0 \leqslant i \leqslant p-1$.

À cause de la symétrie de la clé privée, indiquée en sous-section 4.1, nous pouvons choisir arbitrairement $\sigma(i_1)$ et $\sigma(i_2)$. Un algorithme naïf pour trouver la permutation $\sigma$ est de chercher exhaustivement les valeurs $\sigma(i_j)$ pour $j = 0$ et $3 \leqslant j \leqslant h/r$, jusqu'à ce que (†) donne une permutation consistante.

**Algorithme 3** Algorithme pour recouvrer $\sigma$ sachant $g_{p^r}$

**Entrée :** GF($p^h$), $(c_0, \ldots, c_{p-1})$, $r$ divisant $h$ et $g_{p^r}$
**Sortie :** une permutation $\sigma$ d'une clé privée équivalente

1: choisir des $i_0, \ldots, i_{h/r}$ deux à deux distincts dans $\{0, \ldots, p-1\}$
2: choisir arbitrairement $\sigma(i_1)$ et $\sigma(i_2)$ distincts dans $\{0, \ldots, p-1\}$
3: **pour tout** $\sigma(i_0), \sigma(i_3), \ldots, \sigma(i_{h/r})$ distincts deux à deux **faire**
4: $S \leftarrow [\sigma(i_0), \sigma(i_3), \ldots, \sigma(i_{h/r})]$
5: **pour tout** $\ell \notin S$ **faire**
6: calculer le membre de droite de (†) avec $\alpha_\ell$ au lieu de $\alpha_{\sigma(i)}$ :

$$\mathbf{res} \leftarrow \sum_{j=0}^{h/r} (g_{p^r})^{c_{i_j}} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_\ell - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}}$$

7: **si** il existe $i$ tel que $\mathbf{res} = (g_{p^r})^{c_i}$ et $\sigma(i)$ n'est pas définie **alors**
8: $\sigma(i) \leftarrow j$
9: $S \leftarrow S + [\sigma(i)]$
10: **sinon**
11: continuer la boucle ligne 3
12: **fin si**
13: **fin pour**
14: **retourner** $\sigma$
15: **fin pour**

La complexité de cet algorithme est grossièrement $\mathcal{O}(rp^{h/r})$ opérations dans GF($p$) : la boucle à la ligne 5 fait en moyenne $\mathcal{O}(pr/h)$ itérations, chacune avec une complexité $\mathcal{O}(h)$, et il nous avons besoin de $\mathcal{O}(p^{h/r} - 1)$ itérations de cette boucle.

Quand $r$ est suffisamment grand, il existe un meilleur algorithme. En fait, si $r \geqslant h/r$, la famille $((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}})_{1 \leqslant j \leqslant h/r}$ est libre (voir plus loin). Cela

signifie que les coefficients dans (†) sont les seuls coefficients dans GF($p$) de l'écriture de $(g_{p^r})^{c_i} - (g_{p^r})^{c_{i_0}}$, pour $0 \leqslant i \leqslant p-1$, comme combinaison linéaire des vecteurs :

$$(g_{p^r})^{c_{i_0}} - (g_{p^r})^{c_{i_0}}, \quad (g_{p^r})^{c_{i_1}} - (g_{p^r})^{c_{i_0}}, \quad \ldots, \quad (g_{p^r})^{c_{i_{h/r}}} - (g_{p^r})^{c_{i_0}}.$$

Notons $a_i^j$ le coefficient de $(g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}}$ pour $(g_{p^r})^{c_i} - (g_{p^r})^{c_{i_0}}$. Par (‡), nous avons :

$$\begin{aligned}
\frac{a_2^i}{a_1^i} &= \left( \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq 2}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_2)} - \alpha_{\sigma(i_k)}} \right) \left( \prod_{\substack{0 \leqslant k' \leqslant h/r \\ k' \neq 1}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_{k'})}}{\alpha_{\sigma(i_1)} - \alpha_{\sigma(i_{k'})}} \right)^{-1} \\
&= \left( \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq 2}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_2)} - \alpha_{\sigma(i_k)}} \right) \left( \prod_{\substack{0 \leqslant k' \leqslant h/r \\ k' \neq 1}} \frac{\alpha_{\sigma(i_1)} - \alpha_{\sigma(i_{k'})}}{\alpha_{\sigma(i)} - \alpha_{\sigma(i_{k'})}} \right) \\
&= \left( \prod_{\substack{0 \leqslant k,k' \leqslant h/r \\ k \neq 2, k' \neq 1}} \frac{\alpha_{\sigma(i_1)} - \alpha_{\sigma(i_{k'})}}{\alpha_{\sigma(i_2)} - \alpha_{\sigma(i_k)}} \right) \left( \prod_{\substack{0 \leqslant k,k' \leqslant h/r \\ k \neq 2, k' \neq 1}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i)} - \alpha_{\sigma(i_{k'})}} \right) \\
&= \left( \prod_{\substack{0 \leqslant k,k' \leqslant h/r \\ k \neq 2, k' \neq 1}} \frac{\alpha_{\sigma(i_1)} - \alpha_{\sigma(i_{k'})}}{\alpha_{\sigma(i_2)} - \alpha_{\sigma(i_k)}} \right) \left( \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_1)}}{\alpha_{\sigma(i)} - \alpha_{\sigma(i_2)}} \right)
\end{aligned}$$

Ainsi, il existe $u$ dans GF($p$), indépendant de $i$, tel que

$$(\star) \qquad \frac{a_2^i}{a_1^i} = u \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_1)}}{\alpha_{\sigma(i)} - \alpha_{\sigma(i_2)}}.$$

En passant toute les valeurs de $u$ en revue, nous pouvons obtenir $\sigma(i)$ de l'équation ($\star$). Cette remarque donne naissance à l'algorithme suivant :

## 6.3 Test pour trouver les $g^{(p^h-1)/(p^r-1)}$

## 6.4 Sur l'usage de tous les $c_i$

Nous allons améliorer l'attaque précédente en utilisant la connaissance de tous les $c_i$. Nous allons nous servir du fait suivant :

**Proposition 6.2.** *Soit $Q(x)$ un polynôme de* GF($p^r$)$[x]$ *de degré $d$ et soit $e$ un entier tel que $1 \leqslant e \leqslant (p-1)/d$. Nous avons*

$$\sum_{a \in \mathrm{GF}(p)} Q(a)^e = 0.$$

Le lemme qui suit va permettre de démontrer cette proposition.

**Algorithme 4** Algorithme pour recouvrer $\sigma$ sachant $g_{p^r}$ lorsque $r \geqslant \sqrt{h}$

**Entrée :** $\mathrm{GF}(p^h)$, $(c_0, \ldots, c_{p-1})$, $r$ divisant $h$, $r \geqslant \sqrt{h}$ et $g_{p^r}$
**Sortie :** une permutation $\sigma$ d'une clé privée équivalente

choisir des $i_0, \ldots, i_{h/r}$ deux à deux distincts dans $\{0, \ldots, p-1\}$
pré-calculer la matrice de changement de bases de la base "classique" vers la "base" $\left((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}}\right)_{1 \leqslant j \leqslant h/r}$
choisir arbitrairement $\sigma(i_1)$ et $\sigma(i_2)$ distincts dans $\{0, \ldots, p-1\}$
**pour tout** $u$ dans $\mathrm{GF}(p)$ **faire**
  **pour tout** $i$ tel que $0 \leqslant i \leqslant p-1$ et $i \neq i_0, i_1, i_2$ **faire**
    écrire $(g_{p^r})^{c_i} - (g_{p^r})^{c_{i_0}}$ dans la "base" $\left((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}}\right)_{1 \leqslant j \leqslant h/r}$
    récupérer les coefficients $a_1^i$ et $a_2^i$
    trouver la valeur de $\sigma(i)$ grâce à $(\star)$
  **fin pour**
  **si** $\sigma$ est consistante **alors**
    compléter $\sigma(i_0)$
    **retourner** $\sigma$
  **fin si**
**fin pour**

**Lemme 6.3.** *Soit $k$ un entier tel que $1 \leqslant k \leqslant p-1$. Alors*

$$\sum_{a \in \mathrm{GF}(p)} a^i = 0.$$

*Ainsi, si $P(x) \in \mathrm{GF}(p^n)[x]$ est de degré inférieur à $p-1$, alors :*

$$\sum_{a \in \mathrm{GF}(p)} P(a) = 0.$$

*Démonstration du lemme.* Tout d'abord remarquons que $0 \in \mathrm{GF}(p)$ ne contribue pas à la somme, et ensuite, notons $g$ un élément primitif du corps $\mathrm{GF}(p)$. Alors

$$\sum_{a \in \mathrm{GF}(p)} a^i = \sum_{j=0}^{p-2} g^i = \frac{g^{i(p-1)} - 1}{g^i - 1} = 0.$$

□

# 7 Le calcul de logarithmes dans un groupe abélien

Les résultats proposés ici sont essentiellement tous issus du manuel de Steven Galbraith [3, section 13 et 14, pp. 246 à 297].

**Définition 7.1.** Soit $G$ un groupe noté multiplicativement. Le *problème du logarithme discret* est : étant donné $g \in G$ et $h \in \langle g \rangle$, trouver $a$, tel que $h = g^a$.

Nous notons alors $a = \log_g(h)$, et $g$ est appelé la base du logarithme $a$.

*Exemple* 7.2. Considérons le problème du logarithme discret dans le groupe additif des entiers modulo $p$, où $p$ est un nombre premier. C'est-à-dire qu'étant donné $g \neq 0$ et $h$ dans $\mathbb{Z}/p\mathbb{Z}$, nous cherchons $a \in \mathbb{Z}$ tel que $ag = h$. Comme $\mathbb{Z}/p\mathbb{Z}$ est un corps, il suffit de prendre $a \equiv g^{-1}h \pmod{p}$. Nous pouvons déterminer $g^{-1}$ grâce à une identité de Bézout. Ainsi, résoudre ce cas de logarithme discret se fait en temps polynomial.

## 7.1 Recherche exhaustive

Soient $G$ un groupe, $g \in G$ et $h \in \langle g \rangle$. La façon la plus simple de résoudre ce problème du logarithme discret est de calculer à la suite les $g^a$ pour $0 \leqslant a < r$, où $r$ est l'ordre de $g$, et de comparer la valeur obtenue avec $h$. Cela s'effectue en au plus $r - 2$ opérations dans le groupe et $r$ comparaisons.

**Algorithme 5** Algorithme naïf : recherche exhaustive

**Entrée :** $g$ et $h \in \langle g \rangle$
**Sortie :** $a$ tel que $g^a = h$ et $1 \leqslant a \leqslant r$

1: $a \leftarrow 1$
2: $t \leftarrow g$
3: **tant que** $t \neq h$ **faire**
4: $\quad a \leftarrow a + 1$
5: $\quad t \leftarrow tg$
6: **fin tant que**
7: **retourner** $a$

## 7.2 Méthode de Pohlig et Hellman

Soient $G$ un groupe, $g \in G$ et $h \in \langle g \rangle$. Supposons que $g$ est d'ordre $n$. Écrivons

$$n = \prod_{i=1}^{r} p_i^{\alpha_i},$$

où les $p_i$ sont premiers et les $\alpha_i$ sont tels que $p_i$ ne divise pas $n/p_i^{\alpha_i}$. L'idée de la méthode de Stephen C. Pohlig et Martin E.Hellman [5] est de calculer $a$ modulo les puissances de premier $p_i^{\alpha_i}$ et de recouvrer la solution en utilisant le théorème des restes chinois.

Pour cela, nous utilisons l'homomorphisme de groupes suivant, qui permet de réduire la recherche du logarithme discret aux sous-groupes d'ordre une puissance de nombre premier.

**Lemme 7.3.** *Soient $G$ un groupe cyclique d'ordre $n$, $g$ un générateur de $G$, i.e. $G = \langle g \rangle$, et $q$ un diviseur de $n$. L'application $\varphi_q : x \mapsto x^{n/q}$ est un homomorphisme de groupes de $G$ dans l'unique sous-groupe de $G$ d'ordre $q$. Ainsi, si $h = g^a$, alors*

$$\varphi_q(h) = \varphi_q(g)^{a \bmod q}.$$

*Démonstration.* Soit $H$ un sous-groupe de $G$ d'ordre $q$. Montrons que nous avons l'égalité $H = \langle g^{n/q} \rangle$. Le groupe engendré par $g^{n/q}$ est d'ordre $q$ car $g$ est d'ordre $n$, i.e. pour tout $1 \leqslant k \leqslant q-1$, l'élément $(g^{n/q})^k$ est distinct de l'élément neutre. Considérons, pour $\alpha$ un entier, $g^\alpha$ dans $H$ et distinct de l'élément neutre. Ainsi,

$(g^\alpha)^q$ est l'élément neutre du groupe. Cela implique que l'ordre de $g$ divise l'entier $\alpha q$, ce qui est équivalent à $n/q$ divise $\alpha$. Donc, tout élément de $H$ est une puissance de $g^{n/q}$. Nous concluons grâce à la cardinalité des sous-groupes : l'unique sous-groupe de $G$ de cardinal $q$ est $H = \langle g^{n/q} \rangle$.

Montrons que $\varphi_q$ est un homomorphisme de $G$ dans $H$. Tout élément de $G$ à la puissance $n/q$ est d'ordre divisant $q$. Ainsi, l'image de $G$ par $\varphi_q$ est incluse dans $H$. Et pour tous $\alpha$ et $\beta$ entiers :

$$\varphi_q(g^\alpha g^\beta) = \left(g^\alpha g^\beta\right)^{n/q} = (g^\alpha)^{n/q} \left(g^\beta\right)^{n/q} = \varphi_q(g^\alpha)\varphi_q(g^\beta).$$

Donc $\varphi_q$ est un homomorphisme du groupe $G$ dans le groupe $H$.

Soit $a$ un entier. Notons $a = qb + r$ la division euclidienne de $a$ par $q$, en d'autres termes $r \equiv a \pmod q$. En sachant que $\varphi_q$ est un homomorphisme et que $\varphi_q(g)$ est d'ordre $q$, nous pouvons écrire :

$$\varphi_q(g^a) = \varphi_q(g)^a = \varphi_q(g)^{qb+r} = \left(\varphi_q(g)^q\right)^b \varphi_q(g)^r = \varphi_q(g)^r = \varphi_q(g)^{a \bmod q}.$$

□

Dans les sous-groupes cycliques d'ordre une puissance de nombre premier, voici comment nous pouvons procéder : supposons que $g_0$ est d'ordre $p^\alpha$ et que $h_0 := g_0^a$, pour un entier. Nous pouvons écrire $a = a_0 + a_1 p + \cdots + a_{\alpha-1}p^{\alpha-1}$, où les $a_i$ sont tels que $0 \leqslant a_i < p$. Soit $g_1 := g_0^{p^{\alpha-1}}$. Élever à la puissance $p^{\alpha-1}$ donne :

$$h_0^{p^{\alpha-1}} = (g_0^a)^{p^{\alpha-1}} = \left(g_0^{p^{\alpha-1}}\right)^{a_0+a_1p+\cdots+a_{\alpha-1}p^{\alpha-1}} = \prod_{i=0}^{\alpha-1} g^{a_i p^{\alpha-1+i}} = g_1^{a_0}.$$

Nous pouvons ensuite déterminer $a_0$, en cherchant de manière exhaustive ou avec une autre méthode, voir sous-section 7.3.

Pour avoir $a_1$, définissons $h_1 := h_0 g_0^{-a_0}$ de sorte que

$$h_1 = g_0^{a_1 p+\cdots+a_{\alpha-1}p^{\alpha-1}}.$$

Puis, en élevant à la puissance $p^{\alpha-2}$ :

$$h_1^{p^{\alpha-2}} = g_1^{a_1}.$$

Pour obtenir $a_2$, nous posons $h_2 := h_1 g_0^{-a_1 l}$, et ainsi de suite pour les autres valeurs. Cela donne la solution modulo $p^\alpha$.

Une fois que $a$ est connu pour tout $p^\alpha$ divisant $n$, nous pouvons reconstituer $a$ à l'aide du théorème des restes chinois.

*Exemple* 7.4. Soit $p := 61$, $g := 7$ et $h := 25$. Mettons en œuvre l'algorithme Pohlig-Hellman pour trouver $a$ tel que $g^a \equiv h \pmod p$. Notons que $\mathrm{GF}(61)^\times$ est d'ordre $2^2 \cdot 3 \cdot 5$.

Trouvons d'abord $a$ modulo 4. Comme $(p-1)/4 = 15$, nous posons :

$$g_0 := g^{15} \equiv 11 \pmod{61} \quad \text{et} \quad h_0 := h^{15} \equiv 1 \pmod{61}.$$

Pour obtenir de l'information modulo 2, nous calculons :

$$g_1 := g_0^2 \equiv -1 \pmod{61} \quad \text{et} \quad h_0^2 \equiv 1 \pmod{61}.$$

**Algorithme 6** Algorithme de Pohlig-Hellman avec recherche exhaustive

**Entrée :** $g$ d'ordre $n$, $h \in \langle g \rangle$ et $(p_i, \alpha_i)_{1 \leqslant i \leqslant r}$ tel que $n = \prod p_i^{\alpha_i}$ et $p_i$ premier ne divisant pas $n/p_i^{\alpha_i}$

**Sortie :** $a$ tel que $g^a = h$

**pour** $i$ allant de 1 à $r$ **faire**
  $a_i \leftarrow 0$
  **pour** $j$ allant de 1 à $\alpha_i$ **faire**
    $g_0 \leftarrow g^{n/p_i^j}$
    $h_0 \leftarrow g_0^{-a_i} h^{n/p_i^j}$
    **si** $h_0 \neq 1$ **alors**
      $g_0 \leftarrow g^{n/p_i}$
      $b \leftarrow 1$
      $t \leftarrow g_0$
      **tant que** $h_0 \neq t$ **faire**
        $b \leftarrow b + 1$
        $t \leftarrow t g_0$
      **fin tant que**
      $a_i \leftarrow a_i + b p_i^{j-1}$
    **fin si**
  **fin pour**
**fin pour**
calculer $a \equiv a_i \pmod{p_i^{\alpha_i}}$, pour $1 \leqslant i \leqslant r$, grâce à l'isomorphisme du théorème des restes chinois
**retourner** $a$

Il s'en suit que $a \equiv 0 \pmod 2$. Comme $a$ est nul modulo 2, nous résolvons pour $b$ modulo 4, l'équation modulaire $h_0 \equiv g_1^b \pmod{61}$, ce qui implique que $b$ est pair et ainsi $a \equiv 0 \pmod 4$.

Ensuite, calculons $a$ modulo 3. Comme $(p-1)/3 = 20$, nous redéfinissons :

$$g_0 := g^{20} \equiv 47 \pmod{61} \quad \text{et} \quad h_0 := h^{20} \equiv 13 \pmod{61}.$$

Résolvons pour $a$ modulo 3, l'équation modulaire $h_0 \equiv g_0^a \pmod{61}$, ce qui donne $a \equiv 2 \pmod 3$.

Enfin, déterminons $a$ modulo 5. Comme $(p-1)/5 = 12$, nous redéfinissons :

$$g_0 := g^{12} \equiv 34 \pmod{61} \quad \text{et} \quad h_0 := h^{34} \equiv 13 \pmod{61}.$$

Dont nous obtenons $a \equiv 1 \pmod 5$.

Le théorème des restes chinois donne :

$$a \equiv 2 \cdot 4 \cdot 5 \cdot (4^{-1} \cdot 5^{-1} \bmod 3) + 4 \cdot 3 \cdot (4^{-1} \cdot 3^{-1} \bmod 5) \pmod{60},$$

ce qui fait $a = 2 \cdot 4 \cdot 5 \cdot 2 - 4 \cdot 3 \cdot 2 \equiv 56 \pmod{60}$, i.e. $7^{56} \equiv 25 \pmod{61}$.

**Définition 7.5** (Entier $B$-friable)**.** Un entier est dit *$B$-friable* si tous ses diviseurs premiers sont inférieurs ou égaux à $B$.

**Théorème 7.6.** *Soit $g \in G$ d'ordre $n$. Soit $B \in \mathbb{N}$ tel que $n$ est $B$-friable. Alors l'algorithme de Pohlig-Hellman résout le problème du logarithme discret en $\mathcal{O}(\log(n)^2 + B\log(n))$ opérations dans le groupe.*

À cause de cette méthode, les nombres $B$-friables pour un petit $B$ n'opposent pas de difficulté majeure à la recherche de logarithmes discrets. Donc, en général, en cryptographie, il est préférable d'utiliser des éléments d'ordre un grand nombre premier.

Dans leur papier, S. Pohlig et M. Hellman proposent d'utiliser des entiers premiers de la forme $2p+1$ avec $p$ premier : dans la littérature, $2p+1$ est appelé nombre premier sûr et $p$ est appelé nombre premier de Sophie Germain.

## 7.3 Algorithme pas de bébé, pas de géant

Cet algorithme, souvent attribué à Shanks, est un incarnation du principe de *time/memory tradeoff*. Supposons $g$ d'ordre un nombre premier $r$ et $h = g^a$ pour un certain $0 \leqslant a < r$. Soit $m := \lceil \sqrt{r} \rceil$. Il existe alors deux entiers $a_0$ et $a_1$ tels que $a = a_0 + a_1 m$ et $0 \leqslant a_0, a_1 < m$. Il s'en suit :

$$g^{a_0} = h(g^{-m})^{a_1},$$

et cette observation mène à l'algorithme 7.

Cependant, l'utilisation importante de l'espace mémoire pour les valeurs calculées par l'algorithme baby-step–giant-step devient rapidement prohibitif, car il requiert $\mathcal{O}(\sqrt{r}\log(r))$ bits de stockage (si les éléments du groupe sont représentables en mémoire en $\mathcal{O}(\log(r))$ bits).

**Algorithme 7** Algorithme baby-step–giant-step

**Entrée :** $g$ d'ordre $r$, $h \in \langle g \rangle$
**Sortie :** $a$ tel que $g^a = h$

1: $m \leftarrow \lceil \sqrt{r} \rceil$
2: $L$ est une liste vide
3: $x \leftarrow 1$
4: **pour** $i$ allant de 0 à $m$ **faire** {Calculs des *baby-steps*}
5: $\quad L \leftarrow L + [x, i]$
6: $\quad x \leftarrow xg$
7: **fin pour**
8: $u \leftarrow g^{-m}$
9: $y \leftarrow h$
10: $j \leftarrow 0$
11: **tant que** $[y, *] \notin L$ **faire** {Calculs des *giant-steps*}
12: $\quad y \leftarrow yu$
13: $\quad j \leftarrow j + 1$
14: **fin tant que**
15: trouver $[x, i] \in L$ tel que $x = y$
16: **retourner** $i + mj$

*Exemple* 7.7.

**Théorème 7.8.** *Soit $g \in G$ d'ordre $r$. Si les éléments du groupe sont représentables en mémoire en $\mathcal{O}(\log(r))$ bits et si les opérations dans le groupe se font en $\mathcal{O}(\log(r)^2)$ opérations binaires, alors l'algorithme baby-step–giant-step résout le problème du logarithme discret en $\mathcal{O}(\sqrt{r}\log(r)^2)$ opérations binaires.*

# 8 La méthode rho de Pollard

Après une introduction succincte, nous expliquons le fonctionnement de la méthode rho de Pollard, qui a été conçue en 1978 par John Michael Pollard [6]. Puis, nous étudierons les améliorations apportées par Edlyn Teske [7] en 2001 et Shi Bai et Richard Pierce Brent [1] en 2008.

Il est encouragé de consulter [3, section 14, pp. 264].

## 8.1 Discussion liminaire : partitionner, marcher, chercher

**Introduction :** La méthode rho permet, avec une forte probabilité, de résoudre le problème du logarithme discret – définition 7.1 – dans un groupe abélien. Elle se base sur des marches aléatoires : elle définit donc un algorithme non-déterministe, i.e. sur une même entrée, chaque exécution de l'algorithme peut avoir une issue distincte. Soit $G := \langle g \rangle$ et supposons $g$ d'ordre un nombre premier $r$ – le cas ardu par Pohlig-Hellman – et $h = g^a$ pour un certain $0 \leqslant a < r$.

**Notation :** Si $W$ est un ensemble et $w$ est un élément de $W$ choisi aléatoirement selon la distribution uniforme, alors nous notons $w \in_{\mathfrak{a}} W$.

**Idée :** Observons que si nous trouvons $a_i$, $b_i$, $a_j$ et $b_j$ dans $\mathbb{Z}/r\mathbb{Z}$ tels que :

$$(\star) \qquad g^{a_i} h^{b_i} = g^{a_j} h^{b_j}$$

et $b_i \not\equiv b_j \pmod{r}$, alors nous pouvons résoudre le problème du logarithme discret comme suit :

$$h = g^{(a_i - a_j)(b_j - b_i)^{-1} \bmod r}.$$

**Marche pseudo-aléatoire :** Pollard [6] s'est inspiré de la théorie des marches aléatoires et a donc pensé à construire une suite pseudo-aléatoire $x_i = g^{a_i} h^{b_i}$ d'éléments de $G$ par itération d'une fonction convenable $f : G \to G$, appelée *fonction d'itération*. En d'autres termes, il choisit une valeur de départ $x_1$ et définit le reste de la suite par $x_{i+1} = f(x_i)$, pour $i \geqslant 1$. Une telle suite $x_1, x_2, \ldots$ est appelée une *marche pseudo-aléatoire déterministe*.

Comme $G$ est fini, cette suite est *ultimement périodique* : il existe deux entiers $\lambda \geqslant 1$ et $\mu \geqslant 0$ tels que les éléments $x_1, x_2, \ldots, x_{\lambda+\mu-1}$ soient distincts deux à deux et $x_k = x_{\lambda+k}$ pour tout $k \geqslant \mu$. Nous appelons $\mu$ la *pré-période de la suite* et $\lambda$ la *période de la suite*. Sous l'hypothèse que $x_1 \in_{\mathfrak{a}} G$ et que $f$ est une fonction aléatoire, la valeur attendue de $\lambda$ et $\mu$ est proche de

$$\sqrt{\frac{\pi r}{8}} \approx 0.627\sqrt{r}.$$

**Collision :** Une pair $(x_i, x_j)$ est appelée *collision* si $x_i = x_j$ et $1 \leqslant i < j$, comme dans $(\star)$. Si les éléments dans la marche sont choisis uniformément et indépendants dans $G$, alors par le paradoxe des anniversaires (théorème E.1), la valeur attendue pour $j$ est

$$\sqrt{\frac{\pi r}{2}} \approx 1.253\sqrt{r}.$$

**Rho :** Comme l'image obtenue en "dessinant" les termes de $(x_i)_{i\geqslant 1}$, en commençant en bas et finissant par le cycle en haut, forme la lettre grecque $\rho$, une méthode utilisant une telle suite s'appelle une *méthode rho*.

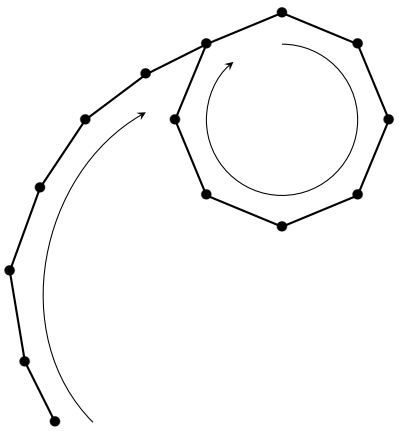

FIGURE 1 – Illustration de la forme "rho"

**Partition :** Pour simuler une fonction aléatoire – la fonction d'itération – de $G$ dans lui-même, il est pratique de commencer par partitionner $G$ en $n$ ensembles [1] (d'à peu près la même taille), de sorte que $G = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{n-1}$. Les ensembles $\mathcal{S}_i$ sont définis en utilisant une *fonction de sélection*

$$s : G \to \{0, \ldots, n-1\},$$

par $\mathcal{S}_i = \{g \in G : s(g) = i\}$.

**Résumé :** Les trois ingrédients principaux d'une bonne méthode rho sont : une partition de $G$ grâce à une fonction de sélection, une suite $(x_i)_{i\geqslant 1}$ à l'aspect aléatoire construite grâce à une fonction d'itération et à la partition précédente – la fonction d'itération a autant de règles que le cardinal de la partition – et un algorithme pour chercher la collision dans la suite $(x_i)_{i\geqslant 1}$.

## 8.2 Méthode rho de Pollard originale dans $(\mathbb{Z}/r\mathbb{Z})^\times$

Dans [6], Pollard décrit sa méthode pour $G = (\mathbb{Z}/r\mathbb{Z})^\times$, où $r$ est un nombre premier, et $g$ un élément primitif du corps $\mathbb{Z}/r\mathbb{Z}$.

*Remarque* 8.1. Ici, $r$ est premier mais le groupe $G$ est d'ordre $r-1$.

Soit $h \in (\mathbb{Z}/r\mathbb{Z})^\times$ dont nous cherchons le logarithme en base $g$. La fonction d'itération $f : (\mathbb{Z}/r\mathbb{Z})^\times \to (\mathbb{Z}/r\mathbb{Z})^\times$ utilisée par Pollard est :

$$f(x) = \begin{cases} gx & \text{si } 0 < x \leqslant p/3, \\ x^2 & \text{si } p/3 < x \leqslant 2p/3, \\ hx & \text{si } 2p/3 < x < p. \end{cases}$$

1. En pratique, les valeurs typiques de $n$ sont 32, 256 ou 2048.

La suite est alors définie par $x_1 = 1$ et $x_{i+1} = f(x_i)$ pour $i \geqslant 1$. Pour garder trace de la décomposition $x_i = g^{a_i} h^{b_i}$, nous calculons les suites $(a_i)_{i \geqslant 1}$ et $(b_i)_{i \geqslant 1}$ comme suit :

$$a_1 = 0 \quad \text{et} \quad a_{i+1} = \begin{cases} a_i + 1 \pmod{r-1} & \text{si } 0 < x \leqslant p/3, \\ 2a_i \pmod{r-1} & \text{si } p/3 < x \leqslant 2p/3, \\ a_i \pmod{r-1} & \text{si } 2p/3 < x < p. \end{cases}$$

$$b_1 = 0 \quad \text{et} \quad b_{i+1} = \begin{cases} b_i \pmod{r-1} & \text{si } 0 < x \leqslant p/3, \\ 2b_i \pmod{r-1} & \text{si } p/3 < x \leqslant 2p/3, \\ b_i + 1 \pmod{r-1} & \text{si } 2p/3 < x < p. \end{cases}$$

Nous rassemblons cela sous la notation :

$$(x_{i+1}, a_{i+1}, b_{b+1}) = \texttt{walk}(x_i, a_i, b_i).$$

*Remarque* 8.2. Il faut garder à l'esprit que les valeurs $a_{i+1}$ et $b_{i+1}$ sont uniquement déterminées par la valeur $x_i$, et que la fonction `walk` dépend seulement de la fonction d'itération $f$.

Maintenant, il faut rechercher une collision dans la suite $(x_i)_{i \geqslant 1}$. Pollard utilise l'*algorithme de recherche de cycle Floyd*, aussi appelé l'*algorithme du lièvre et de la tortue*. Cet algorithme compare $x_i$ et $x_{2i}$, et donne une collision $(x_i, x_{2i})$, pour $i \geqslant 1$. Le plus petit $i$ tel que $x_i = x_{2i}$ est appelé l'*épacte*.

**Algorithme 8** Algorithme rho de Pollard avec recherche de cycle de Floyd

**Entrée :** $r$ un nombre premier, $g \in \mathbb{Z}/r\mathbb{Z}$ primitif et $h \in (\mathbb{Z}/r\mathbb{Z})^\times$
**Sortie :** $a$ tel que $g^a = h$ avec $1 \leqslant a < r$, ou "échec"

$x_1 \leftarrow 1$
$a_1 \leftarrow 0$
$b_1 \leftarrow 0$
$(x_2, a_2, b_2) \leftarrow \texttt{walk}(\texttt{walk}(x_1, a_1, b_1))$
$(x_1, a_1, b_1) \leftarrow \texttt{walk}(x_1, a_1, b_1)$
**tant que** $x_1 \neq x_2$ **faire**
    $(x_1, a_1, b_1) \leftarrow \texttt{walk}(x_1, a_1, b_1)$
    $(x_2, a_2, b_2) \leftarrow \texttt{walk}(\texttt{walk}(x_2, a_2, b_2))$
**fin tant que**
**si** $\text{pgcd}(b_1 - b_2, r - 1) = 1$ **alors**
    **retourner** $(a_2 - a_1)(b_1 - b_2)^{-1} \pmod{r-1}$
**sinon**
    **retourner** "échec"
**fin si**

Notons $\mu$ la pré-période de la suite et $\lambda$ la période de la suite.

**Proposition 8.3.** *En gardant les notations précédentes, $x_{2i} = x_i$ si, et seulement si, $\lambda$ divise $i$ et $i \geqslant \mu$. En outre, il existe un $\mu \leqslant i < \mu + \lambda$ tel que $x_{2i} = x_i$.*

*Démonstration.* Si $x_i = x_j$, avec $1 \leqslant i < j$, alors $\lambda$ divise $j - i$, ce qui prouve la première assertion du lemme. La seconde vient du fait qu'il existe un multiple de $\lambda$ entre $\mu$ et $\mu + \lambda$. □

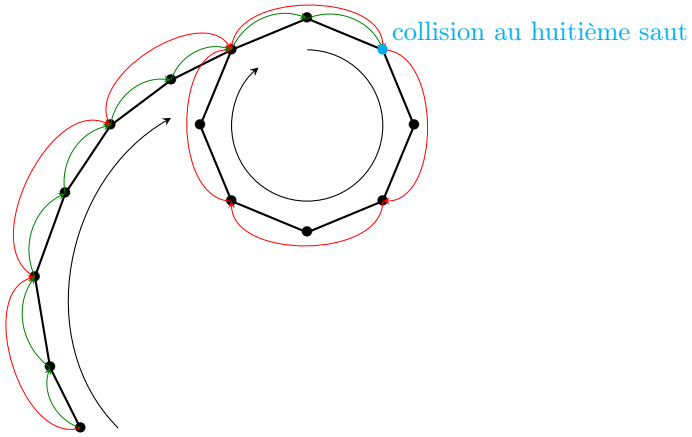


FIGURE 2 – Illustration de l'algorithme du lièvre (rouge) et de la tortue (verte)

## 8.3 Méthode rho de Pollard originale généralisée

**Définition 8.4** (rho-marche)**.** Une *rho-marche* est définie comme suit. Précalculer $g_j = g^{u_j} h^{v_j}$ pour $0 \leqslant j \leqslant n_s - 1$, où $0 \leqslant u_j, v_j < r$ sont choisit de façon uniformément aléatoire. Poser $x_1 = g$. La *rho-marche originale* est

$$x_{i+1} = f(x_i) = \begin{cases} x_i^2 & \text{si } S(x_i) = 0, \\ x_i g_j & \text{si } S(x_i) = j, \quad 1 \leqslant j \leqslant n_s - 1. \end{cases}$$

La *rho-marche additive* est

$$x_{i+1} = f(x_i) = x_i g_{S(x_i)}.$$

*Remarque* 8.5. Une fois la fonction de sélection $S$ et les valeurs de $u_j$ et $v_j$ choisies, la marche est déterministe.

Il est nécessaire de garder trace de la décomposition

$$x_i = g^{a_i} h^{b_i}.$$

Pour la rho-marche originale, les valeurs $a_i, b_i \in \mathbb{Z}/r\mathbb{Z}$ sont obtenues en posant $a_1 = 1$, $b_1 = 0$ et mettant à jour :

$$a_{i+1} = \begin{cases} 2a_i \pmod r & \text{si } S(x_i) = 0, \\ a_i + u_{S(x_i)} \pmod r & \text{si } S(x_i) > 0, \end{cases}$$

$$b_{i+1} = \begin{cases} 2b_i \pmod r & \text{si } S(x_i) = 0, \\ b_i + v_{S(x_i)} \pmod r & \text{si } S(x_i) > 0. \end{cases}$$

Pour la rho-marche additive, les valeurs $a_i, b_i \in \mathbb{Z}/r\mathbb{Z}$ sont obtenues par :

$$a_1 = 1 \text{ et } b_1 = 0,$$

$$a_{i+1} = a_i + u_{S(x_i)} \pmod r,$$
$$b_{i+1} = b_i + v_{S(x_i)} \pmod r.$$

Nous rassemblons cela sous la notation :

$$(x_{i+1}, a_{i+1}, b_{b+1}) = \texttt{walk}(x_i, a_i, b_i).$$

# 9 La réduction de réseau Lenstra-Lenstra-Lovász

# 10 L'attaque Lagarias-Odlyzko sur les sacs à dos à densité faible

# A Le b.a.-ba de la théorie des groupes

**Définition A.1** (Groupe)**.**

# B L'anatomie des corps

Nous souhaitons nous rafraîchir la mémoire sur la théorie des corps. Commençons par donner quelques définitions :

**Définition B.1** (Corps)**.** Soit $(K, +_K, \cdot_K)$, où $K$ est un ensemble, $+_K : K \times K \to K$ et $\cdot_K : K \times K \to K$ sont deux lois de composition interne. Pour faire de $(K, +, \cdot)$ un *corps*, ces deux lois doivent satisfaire aux conditions suivantes :

1. $(K, +_K)$ est un groupe abélien, dont l'élément neutre est noté $0_K$.
2. $(K\backslash\{0_K\}, \cdot_K)$ est un groupe abélien, dont l'élément neutre est noté $1_K$.
3. La loi $\cdot_K$ se distribue sur la loi $+_K$.

Pour alléger les notations, les "$K$" en indice seront omis – mais cette écriture sera utilisée pour ôter le doute en cas d'ambigüité. De même, nous dirons "Soit $K$ un corps", au lieu de "Soit $(K, +, \cdot)$ un corps" – les lois seront sous-entendues. La loi $+$ est appelée *addition*, et la loi $\cdot$ est appelée *multiplication*, et comme à l'accoutumée, le produit $a \cdot b$ sera noté $ab$.

*Exemple* B.2. Les complexes $\mathbb{C}$, les réels $\mathbb{R}$ et les rationnels $\mathbb{Q}$, munis des opérations usuels, sont des corps. Pour tout $p$ nombre premier, l'ensemble des classes d'équivalence $\mathbb{Z}/p\mathbb{Z}$, munis des opérations induites par celles de $\mathbb{Z}$, est un corps.

*Remarque* B.3. La définition implique que, quel que soit $a \in K$, $0 \cdot a = 0$ et $-a = (-1) \cdot a$. De plus, pour tous $a$ et $b$ dans $K$, si $ab = 0$ alors $a$ ou $b$ est nul.

*Exemple* B.4. $\mathbb{Z}/n\mathbb{Z}$ n'est pas un corps, dès que $n$ est composé. En effet, soit $d$ un diviseur de $n$ distinct de 1 et $n$. Alors la classe d'équivalence de $d$ dans $\mathbb{Z}/n\mathbb{Z}$ est un diviseur de zéro, donc $\mathbb{Z}/n\mathbb{Z}$ n'est pas intègre.

**Définition B.5** (Groupe multiplicatif d'un corps)**.** Soit $K$ un corps. $(K\backslash\{0\}, \cdot)$ est un groupe appelé *groupe multiplicatif du corps* $K$, et il est noté $K^\times$.

## B.1 Anneaux : homomorphismes et caractéristique

**Définition B.6** (Homomorphisme d'anneaux)**.** Soient $A$ et $B$ deux anneaux. L'application $f : A \to B$ est un homomorphismes d'anneaux si :

1. $\forall (a, b) \in A^2 : \quad f(a + b) = f(a) + f(b)$,
2. $\forall (a, b) \in A^2 : \quad f(ab) = f(a)f(b)$,
3. $f(1_A) = 1_B$.

**Notation :** Soit $A$ un anneau. Pour tous $a \in A$ et $n \in \mathbb{Z}$, notons

$$na := \begin{cases} a + a + \cdots + a \ (n \text{ termes}) \text{ si } n > 0, \\ 0 \text{ si } n = 0, \\ (-a) + (-a) + \cdots + (-a) \ (-n \text{ termes}) \text{ si } n < 0. \end{cases}$$

**Proposition B.7.** *Soit $A$ un anneau. L'application $\phi_A : \mathbb{Z} \to A$ définie par $\phi_A(n) := n1_A$, est un homomorphisme d'anneaux dont le noyau est $c\mathbb{Z}$ avec $c \in \mathbb{N}$.*

*Démonstration.* $\phi_A$ est bien un homomorphisme d'anneaux et le noyau $\phi_A$ est un idéal de $\mathbb{Z}$, donc de la forme $c\mathbb{Z}$ avec $c \in \mathbb{N}$. □

**Définition B.8** (Caractéristique)**.** Soit $A$ un anneau. La *caractéristique* de l'anneau $A$, noté $\mathrm{car}(A)$, est l'entier $c \in \mathbb{N}$ tel que $\ker(\phi_A) = c\mathbb{Z}$.

*Exemple* B.9. $\mathrm{car}(\mathbb{Z}) = \mathrm{car}(\mathbb{Q}) = \mathrm{car}(\mathbb{R}) = \mathrm{car}(\mathbb{C}) = 0$.
Pour $n$ entier, $\mathrm{car}(\mathbb{Z}/n\mathbb{Z}) = n$.

**Proposition B.10.** *Si $A$ est un anneau intègre, alors sa caractéristique nulle ou est un nombre premier.*

*Démonstration.* Si $\mathrm{car}(A) = pq$, avec $p$ et $q$ des entiers non-nuls distincts de 1, alors les éléments $p1_A$ et $q1_A$ sont non-nuls de produit nul. Ainsi $A$ n'est pas intègre. □

*Exemple* B.11. La réciproque est fausse : $\mathbb{R} \times \mathbb{R}$ est un anneau de caractéristique nulle alors qu'il n'est pas intègre.

## B.2 Endomorphisme de Frobenius

**Rappel :** Soit $A$ un anneau. Si $a$ et $b$ sont deux éléments de $A$ qui commutent l'un à l'autre, i.e. $ab = ba$, alors pour $n \in \mathbb{N}$, la *formule du binôme de Newton* donne :

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k},$$

où $\binom{n}{k}$ est le nombre de combinaison de $k$ objets dans un ensembles à $n$ objets :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

**Proposition B.12.** *Soit $A$ un anneau commutatif de caractéristique un nombre premier $p$. L'application*

$$\begin{array}{rccc} f: & A & \longrightarrow & A \\ & x & \longmapsto & x^p \end{array}$$

*est un endomorphisme de l'anneau $A$.*

*Démonstration.* Soit $0 < k < p$. La formule

$$k\binom{p}{k} = p\binom{p-1}{k-1},$$

et le lemme d'Euclide impliquent que $p$ divise $\binom{p}{k}$. Nous pouvons maintenant démontrer la proposition. Soit $(a, b) \in A^2$. Comme $A$ est commutatif, nous pouvons appliquer la formule du binôme de Newton :

$$f(a+b) = (a+b)^p = \sum_{k=0}^{p} \binom{p}{k} a^k b^{p-k} = a^p + b^p = f(a) + f(b).$$

De plus, $f(ab) = (ab)^p = a^p b^p = f(a)f(b)$, car $A$ est commutatif, et $f(1) = 1$. Donc $f$ est bien un endomorphisme de l'anneau $A$. □

*Exemple* B.13. La proposition n'est plus vraie dès que la caractéristique de l'anneau est un nombre composé : plaçons nous dans $\mathbb{Z}/4\mathbb{Z}$, nous avons d'une part $(1+1)^4 = 2^4 = 0$ et d'autre part $1^4 + 1^4 = 1 + 1 = 2$. Alors que clairement $0 \neq 2$ dans $\mathbb{Z}/4\mathbb{Z}$.

**Définition B.14** (Endomorphisme de Frobenius)**.** Soit $A$ un anneau commutatif de caractéristique un nombre premier $p$. L'application $x \mapsto x^p$ est appelé l'*endomorphisme de Frobenius* de l'anneau $A$. Il est noté $\mathrm{Frob}_A$.

## B.3 Sous-corps premiers

**Définition B.15** (Sous-corps)**.** Soit $L$ un corps. Une partie $K \subseteq L$ est un *sous-corps* de $L$ si $K$ est un corps pour les opérations héritées de $L$.

*Exemple* B.16. $\mathbb{Q}$ est un sous-corps de $\mathbb{R}$ et $\mathbb{R}$ est un sous-corps de $\mathbb{C}$.

**Définition B.17** (Corps premier)**.** $K$ est un *corps premier* si $K$ est un corps ne contenant aucun sous-corps autre que lui-même.

**Proposition B.18.** *Le corps $\mathbb{Q}$ est un corps premier et, pour tout nombre premier $p$, le corps $\mathbb{Z}/p\mathbb{Z}$ est un corps premier.*

**Définition B.19** (Sous-corps premier)**.** Soit $K$ un corps. Le sous-corps premier de $K$ est l'intersection de tous ses sous-corps.

**Proposition B.20.** *Soient $K$ un corps et $P$ le sous-corps premier de $K$. Si $\mathrm{car}(K) = 0$ alors $P$ est isomorphe à $\mathbb{Q}$, et si $\mathrm{car}(K) \neq 0$ alors $p = \mathrm{car}(K)$ est un nombre premier et $P$ est isomorphe à $\mathbb{Z}/p\mathbb{Z}$.*

## B.4 Extensions de corps

**Définition B.21** (Homomorphisme de corps)**.** Un *homomorphisme de corps* est un homomorphisme d'anneaux (voir définition B.6).

**Proposition B.22.** *Un homomorphisme de corps est injectif.*

*Démonstration.* Soient $K_A$ et $K_2$ deux corps et $f : K_1 \to K_2$ un homomorphismes de corps. Rappelons que les seuls idéaux d'un corps sont l'idéal nul et le corps tout entier. Le noyau $\ker(f)$ est un idéal de $K_1$, et par la remarque précédente, de deux chose l'une : soit $\ker(K_1) = K_1$, soit $\ker(K_1) = \{0_{K_1}\}$. Or comme $f$ est un homomorphisme d'anneaux, $f(1_{K_1}) = 1_{K_2} \neq 0_{K_2}$, cela implique donc $\ker(f) \neq K_1$. D'où $\ker(f) = \{0_{K_1}\}$ et $f$ est injective. □

**Définition B.23** (Extension de corps)**.** Soit $K$ un corps. Une *extension* du corps $K$ est un couple $(L, \iota)$, où $L$ est un corps et $\iota$ est un homomorphisme de corps.

Soit $(L, \iota)$ un extension du corps $K$. Comme $\iota$ est injective, nous pouvons identifier $K$ à son image $\iota(K)$ dans $L$. Dans la pratique, sauf risque de confusion, nous faisons systématiquement cette identification : nous disons alors que $L$ est une extension de $K$ si $K$ est un sous-corps de $L$, et nous le notons alors $K \subseteq L$. Il est alors évident que $K$ et $L$ ont la même caractéristique.

**Proposition B.24.** *Soit $K \subseteq L$ une extension de corps. $L$ est un $K$-espace vectoriel. Une base du $K$-espace vectoriel $L$ est appelée une base de $L$ sur $K$.*

**Définition B.25** (Degré d'une extension)**.** Soit $K \subseteq L$ une extension de corps. Le *degré de l'extension* $K \subseteq L$ est le cardinal, fini ou non, d'une base de $L$ sur $K$. Le degré est noté $[L : K] := \dim_K(L)$. Un extension de degré fini (resp. 2, 3) est dite finie (resp. quadratique, cubique).

*Exemple* B.26. L'extension de corps $\mathbb{R} \subseteq \mathbb{C}$ est une extension quadratique.

**Définition B.27.** Soient $K \subseteq L$ un extension de corps et $S$ une partie de $L$. Notons $K[S]$ le sous-anneau de $L$ engendré par $K$ et $S$, et $K(S)$ le sous-corps de $L$ engendré par $K$ et $S$.

**Définition B.28** (Extension de type fini)**.** L'extension $K \subseteq L$ est *type fini* s'il existe une partie finie $S$ de $L$ telle que $L = K(S)$.

**Définition B.29** (Extension simple et élément primitif)**.** L'extension $K \subseteq L$ est *simple* s'il existe une partie $S$ de $L$ réduite à un élément telle que $L = K(S)$. Tout élément $x$ de $L$ vérifiant $L = K(x)$ est appelé un *élément primitif* de l'extension.

**Proposition B.30.** *Soient $K \subseteq L$ un extension de corps et $S$, $T$ deux parties de $L$, et $\mathcal{F}$ l'ensemble des parties finies de $S$. Alors :*

$$K(S \cup T) = K(S)(T) = K(T)(S), \quad K(S) = \bigcup_{U \in \mathcal{F}} K(U).$$

**Définition B.31** (Sous-extension de corps)**.** Une *sous-extension* d'une extension de corps $K \subseteq M$ est un corps $L$ tel que $K \subseteq L \subseteq M$. En d'autres termes, c'est un sous-corps de $M$ contenant $K$.

**Théorème B.32** (Multiplicativité du degré)**.** *Soient $K \subseteq L \subseteq M$ des extensions de corps.*

1. *Soient $(e_i)_{i \in I}$ une base de $L$ sur $K$ et $(f_j)_{j \in J}$ une base de $M$ sur $L$. Alors la famille $(e_i f_j)_{(i,j) \in I \times J}$ est une base de $M$ sur $K$.*
2. *L'extension $K \subseteq M$ est finie si, et seulement si, les extensions $K \subseteq L$ et $L \subseteq M$ le sont. Si tel est le cas :*

$$[M : K] = [M : L][L : K].$$

*Démonstration.* Il suffit de montrer la première assertion. Soient $(\lambda_{ij})_{(i,j) \in I \times J}$ une famille presque nulles d'élément de $K$. Alors

$$\sum_{i,j} \lambda_{ij} e_i f_j = 0 \Rightarrow \sum_j \left( \sum_i \lambda_{ij} e_i \right) f_j = 0.$$

Pour tout $(i, j) \in I \times J$, les éléments $\lambda_{ij} e_i$ sont dans $L$. Du fait que la famille $(f_j)_{j \in J}$ est libre sur $L$, nous déduisons :

$$\sum_i \lambda_{ij} e_i = 0,$$

quel que soit $j \in J$. Comme $(e_i)_{i \in I}$ est une base de $L$ sur $K$, les $\lambda_{ij}$ sont nuls pour tout $(i, j) \in I \times J$. Le reste se démontre aisément. □

## B.5 Corps finis

**Proposition B.33.** *Soit $K$ un corps de caractéristique $p > 0$ et notons* Frob *l'endomorphisme de Frobenius de $K$. Si $K$ est fini alors* Frob *est un automorphisme de $K$, et si $K = \mathbb{F}_p$ alors* $\text{Frob} = \text{id}_{\mathbb{F}_p}$.

*Démonstration.* Par la proposition B.10, $p$ est un nombre premier et la proposition B.12 dit que Frob est un endomorphisme de $K$. Comme tout morphisme de corps est injectif, si $K$ est fini alors Frob est un automorphisme de $K$. Si $K = \mathbb{F}_p$, le petit théorème de Fermat donne $x^{p-1} = 1$ pour tout $x \in \mathbb{F}_p^\times$, donc $x^p = x$ quel que soit $x \in \mathbb{F}_p$. □

# C La théorie de Kummer

# D La théorie d'Artin-Schreier

# E Le paradoxe des anniversaires

Le "paradoxe des anniversaires", comme expliqué dans [3, sous-section 14.1] vient de l'application suivante : dans un ensemble aléatoire de 23 personnes ou plus, la probabilité que deux individus partagent le même anniversaire est supérieure à $0,5$.

**Théorème E.1.** *Soit $\mathcal{S}$ un ensemble à $n$ éléments. Si les éléments sont tirés de façon uniformément aléatoire de $\mathcal{S}$, alors le nombre attendu d'éléments à tirer avant d'avoir un tirage identique à un précédent est moins de $\sqrt{\pi n/2} + 2$.*

*Démonstration.* Soit $X$ la variable aléatoire du nombre d'éléments choisis dans $\mathcal{S}$ (uniformément aléatoire) avant qu'un élément soit sélectionné deux fois. Après que $\ell$ éléments distincts aient été choisis, la probabilité que le suivant soit aussi distinct des précédents est $(1 - \ell/n)$. D'où, la probabilité $\mathbb{P}(X > \ell)$ est donnée par :

$$\mathbb{P}_{n,\ell} = 1\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\cdots\left(1 - \frac{\ell-1}{n}\right).$$

Remarquons que $\mathbb{P}_{n,\ell} = 0$ quand $\ell \geqslant n$. Maintenant, servons nous du fait que $1 - x \leqslant e^{-x}$ pour $x \geqslant 0$ :

$$\mathbb{P}_{n,\ell} \leqslant e^{-1/n}e^{-2/n}\cdots e^{-(\ell-1)/n} = e^{-\sum_{j=0}^{\ell-1} j/n} = e^{-(1/2)(\ell-1)\ell/n} \leqslant e^{-(\ell-1)^2/(2n)}$$

Par définition, l'espérance mathématique de $X$ est :

$$\begin{aligned}\mathbb{E}(X) &= \sum_{\ell=1}^{\infty} \ell\, \mathbb{P}(X = \ell) \\ &= \sum_{\ell=1}^{\infty} \ell\, (\mathbb{P}(X > \ell - 1) - \mathbb{P}(X > \ell)) \\ &= \sum_{\ell=0}^{\infty} (\ell + 1 - \ell) \mathbb{P}(X > \ell) \\ &= \sum_{\ell=0}^{\infty} \mathbb{P}(X > \ell) \\ &\leqslant 1 + \sum_{\ell=0}^{\infty} e^{-(\ell-1)^2/(2n)}\end{aligned}$$

Nous estimons cette dernière série par l'intégrale :

$$1 + \int_0^{\infty} e^{-x^2/(2n)} dx.$$

Comme $x \mapsto e^{-x^2/(2n)}$ est décroissante et prend ses valeurs dans $[0, 1]$, la différence entre la valeur de la série et la valeur de l'intégrale est au plus 1. Le changement de variable $u := x/\sqrt{2n}$ donne :

$$1 + \int_0^{\infty} e^{-x^2/(2n)} dx = 1 + \sqrt{2n} \int_0^{\infty} e^{-u^2} du = 1 + \sqrt{2n} \frac{\sqrt{\pi}}{2}.$$

D'où la valeur :

$$\mathbb{E}(X) \leqslant \sqrt{\frac{\pi n}{2}} + 2.$$

□

# Références


[1] S. Bai and R. P. Brent. **On the Efficiency of Pollard's Rho Method for Discrete Logarithms**. In *Conference in Research and Practice in Information Technology*. Departement of Computer Science & Center for Mathematics and its Applications, Australian National University, 2008.

[2] B. Chor and R. L. Rivest. **A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields**. *IEEE Transactions on Information Theory*, 1988.

[3] S. D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[4] K. Huber. **Specialised Attack on Chor-Rivest Public Key Cryptosystem**. *Journal of Cryptology*, 2000.

[5] S. C. Pohlig and M. E. Hellman. **An Improved Algorithm for Computing Logarithms over** GF($p$) **and Its Cryptographic Signifiance**. *IEEE Transactions on Information Theory*, 1978.

[6] J. M. Pollard. **Monte Carlo Methods for Index Computation** (mod $p$). *Mathematics of Computation*, 1978.

[7] E. Teske. **On Random Walks for Pollard's Rho Method**. *Mathematics of Computation*, 2001.

[8] S. Vaudenay. **Cryptanalysis of the Chor-Rivest Cryptosystem**. *Journal of Cryptology*, 2000.