Cryptographie à clé publique
Cours 9 et 10

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC

08-15/04/2026

La semaine dernière

Questions ?

Motivation

Soit $\mathbb{G}$ un groupe cyclique d'ordre n fini, et g un générateur de $\mathbb{G}$.

Définition. Le problème du **logarithme discret** dans le groupe $\mathbb{G}$ est défini comme suit.

- **Données** : $y \in \mathbb{G}$,
- **Objectif** : trouver $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$.

Application principale : en cryptographie. La sécurité de nombreux systèmes cryptographiques repose sur la **difficulté** de ce problème : ElGamal, DSA.

Quels groupes ?

- groupe multiplicatif $\mathbb{F}_q^\times$,
- groupe de points d'une courbe elliptique.

Sauf indication contraire, on va noter le groupe $\mathbb{G}$ multiplicativement.

Plan

Plan

Méthode « pas de bébé - pas de géant »

Objectif. Pour $y \in \mathbb{G}$, on cherche $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$.

Méthode exhaustive. Tester tous les g^i coûte $O(n)$.

Méthode « pas de bébé - pas de géant » (*baby-step-giant-step*), initiée par Shanks dans :

Class number, a theory of factorization and genera. D. Shanks. Proc. Symp. Pure Math.. **1971**.

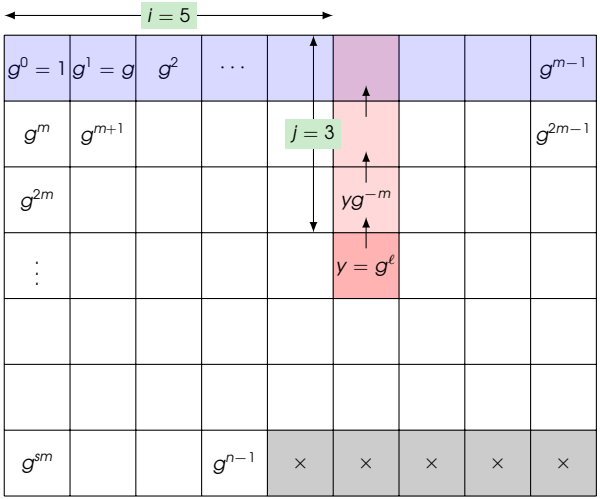
Idée. Soit $m \geq \sqrt{n}$.

1. On calcule la liste
$$L = [g^0, \ldots, g^{m-1}]$$
2. On cherche un j tel que yg^{-jm} est dans la liste L.

Si c'est le cas, on a alors :
$$yg^{-jm} = g^i \quad \Longleftrightarrow \quad y = g^{i+jm}$$

Remarque. Un tel couple (i, j) existe toujours. On peut s'en apercevoir en effectuant la division euclidienne de ℓ par m.

Méthode « pas de bébé - pas de géant »



Sur l'exemple, on obtient $\ell = i + mj = 5 + 9 \times 3 = 32$

Méthode « pas de bébé - pas de géant » : algorithme

Dans l'algorithme qui suit, on va utiliser comme structure de données le **dictionnaire**.

$$D = \left\{ \begin{array}{ccc} \texttt{key}_1 & \to & \texttt{value}_1 \\ \vdots & & \vdots \\ \texttt{key}_s & \to & \texttt{value}_s \end{array} \right.$$

Grâce à une table de hachage : ajout d'un élément, et test d'appartenance en $O(1)$.

MÉTHODE « PAS DE BÉBÉ – PAS DE GÉANT » (SHANKS)

Entrée : un élément $y \in \mathbb{G}$, un générateur g de $\mathbb{G}$ et l'ordre n de $\mathbb{G}$
Sortie : $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$

1. Calculer $m = \lceil \sqrt{n} \rceil$.
2. Calculer un dictionnaire $D = \{(g^0 \to 0), \ldots, (g^{m-1} \to m-1)\}$.
3. Initialiser $x \leftarrow y$ et $j \leftarrow 0$
4. **Tant que** x n'est pas une clé de D, **faire :**
   - $x \leftarrow x \times g^{-m}$
   - $j \leftarrow j + 1$
5. **Retourner** $jm + D[x]$.

Méthode « pas de bébé - pas de géant » : exemple 1

Dans le **groupe multiplicatif** $\mathbb{F}_q^\times$ avec $q = 97$.
L'élément $g = 5$ engendre $\mathbb{F}_{97}^\times$.

1. On a $m = \lceil \sqrt{97} \rceil = 10$.
2. On construit le dictionnaire $D = \{g^i \to i\}_{i \leq m-1}$. Un dictionnaire n'est pas nécessairement « ordonné ». Ici, `sage` donne :

$$D = \{1 \to 0,\ 5 \to 1,\ 6 \to 8,\ 8 \to 6,\ 43 \to 4,\ 40 \to 7,\ 21 \to 5,\ 25 \to 2,\ 28 \to 3,\ 30 \to 9\}$$

3. Pour $y = 18$, on obtient ensuite

j	yg^{-jm}	Test($yg^{-jm} \in D$)
0	18	$\times$
1	4	$\times$
2	44	$\times$
3	96	$\times$
4	86	$\times$
5	73	$\times$
6	27	$\times$
7	6	✓ ($D[6] = 8$)

4. Donc $\log_g(y) = i + jm = 8 + 7 \times 10 = 78$.

Méthode « pas de bébé - pas de géant » : exemple 2

Dans le **groupe de points de la courbe elliptique** (noté additivement)

$$E : \quad y^2 = x^3 + x + 1,$$

définie sur $\mathbb{F}_q$ avec $q = 101$. Ce groupe est cyclique, isomorphe à $\mathbb{Z}/105\mathbb{Z}$, et admet comme générateur le point $P = [41 : 92 : 1] \in E(\mathbb{F}_q)$.

1. On a $m = \lceil \sqrt{105} \rceil = 11$.
2. On construit le dictionnaire $D = \{iP \to i\}_{i \leq m-1}$

$$D = \{[0 : 1 : 0] \to 0, [41 : 92 : 1] \to 1, [64 : 35 : 1] \to 2, [38 : 88 : 1] \to 8,$$
$$[43 : 93 : 1] \to 3, [35 : 17 : 1] \to 9, [93 : 84 : 1] \to 4, [55 : 36 : 1] \to 10,$$
$$[74 : 84 : 1] \to 5, [29 : 52 : 1] \to 6, [87 : 24 : 1] \to 7\}$$

3. Pour un point $Q = [76 : 39 : 1] \in E(\mathbb{F}_q)$, on obtient ensuite

j	$Q - jmP$	Test($Q - jmP \in D$)
0	$[76 : 39 : 1]$	$\times$
1	$[82 : 71 : 1]$	$\times$
2	$[99 : 30 : 1]$	$\times$
3	$[30 : 93 : 1]$	$\times$
4	$[46 : 25 : 1]$	$\times$
5	$[55 : 36 : 1]$	$\checkmark$ ($D[[55 : 36 : 1]] = 10$)

4. Donc $\log_P(Q) = i + jm = 10 + 5 \times 11 = 65$.

Méthode « pas de bébé - pas de géant » : complexité

MÉTHODE « PAS DE BÉBÉ – PAS DE GÉANT » (SHANKS)

Entrée : un élément $y \in \mathbb{G}$, un générateur g de $\mathbb{G}$ et l'ordre n de $\mathbb{G}$
Sortie : $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$

1. Calculer $m = \lceil \sqrt{n} \rceil$.
2. Calculer un dictionnaire $D = \{(g^0 \to 0), \ldots, (g^{m-1} \to m-1)\}$.
3. Initialiser $x \leftarrow y$ et $j \leftarrow 0$
4. **Tant que** x n'est pas une clé de D, **faire :**
   - $x \leftarrow x \times g^{-m}$
   - $j \leftarrow j + 1$
5. **Retourner** $jm + D[x]$.

Complexité pour les étapes :

- étapes 1, 3, et 5 : $O(1)$
- étape 2 : $O(\sqrt{n})$
- étape 4 : $O(\sqrt{n})$

$$\Longrightarrow \quad \text{Complexité totale en } O(\sqrt{n})$$

Plan

Algorithme de Pohlig-Hellman

Fait. L'algorithme *baby-step-giant-step* permet de calculer le logarithme discret d'un élément $y \in \mathbb{G}$ en temps $O(\sqrt{n})$ où $n = |\mathbb{G}|$.

Lorsque $n = \prod p_i^{e_i}$ est composé, $\mathbb{G}$ admet des sous-groupes cycliques propres.

$$\mathbb{G} \simeq \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_k^{e_k}\mathbb{Z}$$

Question. Peut-on mettre à profit cette structure pour améliorer la complexité ?

Deux cas à étudier :

1. $n = st$, où s et t sont premiers entre eux,
2. $n = p^e$ où p est premier et $e \geq 2$.

Pohlig-Hellman : module $n = st$

Premier cas. Si $n = st$ avec s et t premiers entre eux.

Soit $y \in \mathbb{G}$ dont on cherche $\ell = \log_g(y)$.

On va utiliser le théorème des restes chinois. L'application

$$\begin{array}{rccc} \phi : & \mathbb{Z}/n\mathbb{Z} & \to & \mathbb{Z}/s\mathbb{Z} \times \mathbb{Z}/t\mathbb{Z} \\ & x & \mapsto & (x \mod s,\ x \mod t) \end{array}$$

est un isomorphisme d'anneaux que l'on sait évaluer et inverser efficacement.

Pour calculer ℓ, il reste donc à trouver $(\ell \mod s)$ et $(\ell \mod t)$.

Question intermédiaire. À partir de l'instance (y, g), construire un couple (y', g') dans un groupe $\mathbb{G}'$ tel que $\log_{g'}(y') = \ell \mod s$.

Idée. On construit $g' = g^t$ et $y' = y^t$. Alors, g' engendre un sous-groupe $\mathbb{G}'$ de $\mathbb{G}$ d'ordre s, et on a

$$y' = y^t = g^{\ell t} = (g')^{\ell}\,.$$

Donc, y' appartient à ce sous-groupe $\mathbb{G}'$, et $\log_{g'}(y') = \ell \mod s$.

Pohlig–Hellman : module $n = st$

On suppose que l'**on dispose d'un algorithme** L **qui calcule le logarithme discret** dans un groupe fini quelconque. L'algorithme L peut être :

- l'algorithme de Pohlig-Hellman lui-même (appel récursif),
- un autre algorithme, comme *baby-step-giant-step* par exemple.

ALGORITHME DE POHLIG-HELLMAN (ÉTAPE « PREMIERS ENTRE EUX »)

Entrée : un élément $y \in \mathbb{G}$ où $|\mathbb{G}| = n = st$ et $\mathrm{pgcd}(s, t) = 1$, et un générateur g de $\mathbb{G}$
Sortie : $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$

1. Calculer y^s, y^t, g^s et g^t.
2. À l'aide de l'algorithme L, calculer $\ell_s = \log_{g^t}(y^t)$ et $\ell_t = \log_{g^s}(y^s)$.
3. Calculer les coefficients de Bezout a, b tels que $as + bt = 1$
4. Effectuer la reconstruction par restes chinois :

$$\ell = as\ell_t + bt\ell_s$$

5. **Retourner** $\ell \bmod n$

Remarque : il faut connaître la factorisation de n...

Pohlig–Hellman : module $n = st$ (exemple)

Exemple. On prend la courbe elliptique E définie sur $\mathbb{F}_q$, $q = 97$, par l'équation

$$y^2 = x^3 - 9x + 33$$

Le groupe $E(\mathbb{F}_q)$ est cyclique, de cardinal $n = 115$, et $P = [91 : 29 : 1]$ en est un générateur.

1. On a choisi $Q = [9 : 14 : 1] \in E(\mathbb{F}_q)$
2. On a $n = s \times t$ où $s = 5$ et $t = 23$.
3. Puis, on obtient

sQ	sP	ℓ_t	tQ	tP	ℓ_s
$(54 : 68 : 1)$	$(71 : 32 : 1)$	13	$(48 : 96 : 1)$	$(48 : 96 : 1)$	1

4. Les coefficients de Bezout de s et t sont $-9s + 2t = 1$
5. Donc on peut reconstruire (théorème des restes chinois) :

$$\ell = -9 \times 5 \times 13 + 2 \times 23 \times 1 \equiv 36 \mod 115$$

Pohlig–Hellman : module $n = p^e$

Second cas. Si $n = p^e$ avec p premier et $e \geq 2$.

On n'a plus l'isomorphisme des restes chinois. Néanmoins, $\mathbb{G} \simeq \mathbb{Z}/p^e\mathbb{Z}$ admet des sous-groupes cycliques propres $\mathbb{G}_i$ de la forme $\mathbb{Z}/p^i\mathbb{Z}$ où $1 \leq i \leq e-1$.

Si g engendre $\mathbb{G}$, un de ces sous-groupes $\mathbb{G}_i$ est engendré par $g^{p^{e-i}}$.

Question. Comment utiliser ces sous-groupes ?

Écrivons $\ell = \log_g(y)$ en base p :

$$\ell = \ell_0 + \ell_1 p + \cdots + \ell_{e-1} p^{e-1}$$

On calcule $y^{p^{e-1}}$:

$$y^{p^{e-1}} = g^{\ell_0 p^{e-1} + \ell_1 p^e + \cdots + \ell_{e-1} p^{2e-2}} = g^{\ell_0 p^{e-1} + p^e(\ell_1 + \cdots + \ell_{e-1} p^{e-2})} = g^{\ell_0 p^{e-1}} = \left(g^{p^{e-1}}\right)^{\ell_0}$$

Donc, l'entier ℓ_0 est le logarithme de $y^{p^{e-1}}$ en base $g^{p^{e-1}}$ dans $\mathbb{G}_1$.

Question. Comment poursuivre pour obtenir ℓ_1 ?

Question. Comment poursuivre pour obtenir ℓ_1 ?

On itère le procédé avec :

$$\frac{y}{g^{\ell_0}} = (g^p)^{\ell_1 + \ell_2 p + \cdots + \ell_{p-1} p^{e-2}}.$$

Puis

$$\left(\frac{y}{g^{\ell_0}}\right)^{p^{e-2}} = g^{\ell_1 p^{e-1}}$$

Donc, ℓ_1 est le logarithme de $(yg^{-\ell_0})^{p^{e-2}}$ en base $g^{p^{e-1}}$ dans $\mathbb{G}_1$.
Par **induction**, on va calculer tous les ℓ_i grâce à :

$$\frac{y}{g^{\ell_0 + \ell_1 p + \cdots + \ell_{i-1} p^{i-1}}} = (g^{p^{i-1}})^{\ell_i + \ell_{i+1} p + \cdots + \ell_{p-1} p^{e-i-1}}$$

Au passage, on a démontré

Lemme. Soit $\ell = \log_g(y)$, $y_j = y^{p^{e-j}}$ et $g_j = g^{p^{e-j}}$. Alors, le logarithme de y_j en base g_j est $\ell_j = \ell \mod p^j$.

Pohlig-Hellman : module $n = p^e$ (algorithme)

Une nouvelle fois, on suppose que l'**on dispose d'un algorithme** L **qui calcule le logarithme discret** dans un groupe fini quelconque.

ALGORITHME DE POHLIG-HELLMAN (ÉTAPE « p^e »)

Entrée : un élément $y \in \mathbb{G}$ où $|\mathbb{G}| = n = p^e$, et un générateur g de $\mathbb{G}$
Sortie : $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$

1. Calculer $g_1 \leftarrow g^{p^{e-1}}$.
2. Initialiser $\ell \leftarrow 0$
3. **Pour** i allant de 1 à e, **faire** :
   - 3.1 Calculer $z \leftarrow (y/g^\ell)^{p^{e-i}}$
   - 3.2 À l'aide de l'algorithme L, calculer le logarithme ℓ_{i-1} de z en base g_1
   - 3.3 Mettre à jour $\ell \leftarrow \ell + \ell_{i-1} p^{i-1}$
4. **Retourner** ℓ

Remarque : on peut s'épargner quelques exponentiations en calculant $(y/g^\ell)^{p^{e-i}}$ intelligemment.

Pohlig-Hellman : module $n = p^e$ (exemple)

Exemple. On se place dans le groupe $\mathbb{F}_q^\times$ où $q = 257 = 2^8 + 1$ (troisième nombre de Fermat).
Un générateur de $\mathbb{F}_q^\times$ est $g = 3$.

1. Pour $y = 101$, on a

i	y/g^ℓ	ℓ_{i-1}
1	101	1
2	205	1
3	137	0
4	137	1
5	241	0
6	241	0
7	241	1
8	1	0

2. On en déduit que $\ell = \sum_{i=0} \ell_i 2^i = 75$

Algorithme de Pohlig-Hellman : résumé

Question. Comment regrouper ces étapes ?

Idée : si $\mathbb{G} \simeq \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_k^{e_k}\mathbb{Z}$, alors on calcule itérativement les logarithmes dans les sous-groupes $\mathbb{G}_i = \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_i^{e_i}\mathbb{Z}$, pour i allant de 1 à k.

ALGORITHME DE POHLIG-HELLMAN

Entrée : un élément $y \in \mathbb{G}$ où $|\mathbb{G}| = n$, et un générateur g de $\mathbb{G}$ et une factorisation $F = [(p_i, e_i)]_i$ de $n = \prod_i p_i^{e_i}$
Sortie : $\ell \in \mathbb{Z}/n\mathbb{Z}$ tel que $g^\ell = y$

1. Initialiser $\ell \leftarrow 0$ et $m \leftarrow 1$.
2. **Pour tout** (p_i, e_i) dans F, **faire** :
   - 2.1 Calculer $\ell_i \leftarrow$ `pohlig_hellman_pow`(y, g, p_i, e_i)
   - 2.2 **Si** $m \neq 1$, **alors** calculer $\ell \leftarrow$ `pohlig_hellman_coprime`$(y, g, s = p_i^{e_i}, t = m)$
   - 2.3 **Sinon**, assigner $\ell \leftarrow \ell_i$
   - 2.4 $m \leftarrow m \times p_i^{e_i}$
3. **Retourner** ℓ

Algorithme de Pohlig-Hellman : complexité

Complexité. On note $C(p_i)$ le coût d'un algorithme L de résolution de logarithme discret dans un sous-groupe d'ordre p_i.

Pour chaque facteur $p_i^{e_i}$ de n :

- `pohlig_hellman_pow`(y, g, p_i, e_i) effectue deux appels à L, un calcul de $\gcd$, quelques calculs modulaires et exponentiations dans le groupe

$$\implies \text{complexité en } O\Big(C(p_i)\,\mathrm{polylog}(p_i^{e_i})\Big)$$

- `pohlig_hellman_coprime`$(y, g, s = p_i^{e_i}, t = m)$ effectue e_i appels à L et quelques exponentiations dans le groupe

$$\implies \text{complexité en } O\Big(e_i C(p_i)\,\mathrm{polylog}(p_i^{e_i})\Big)$$

- les autres opérations sont un nombre constant d'exponentiations

Complexité. Si la factorisation de n est connue et si L a complexité en $O(\sqrt{|\mathbb{G}_i|})$, alors l'algorithme de Pohlig-Hellman fonctionne en temps

$$O\Big(\sqrt{p}\,\mathrm{polylog}(n)\Big)$$

où p est le plus gros facteur premier de n.

Plan

Algorithme λ de Pollard

Présentation rapide de l'**algorithme** λ de Pollard, aussi appelé « méthode des kangourous ».

Idée. On définit une suite récurrente de la forme

$$x_{i+1} = x_i g^{f(x_i)}$$

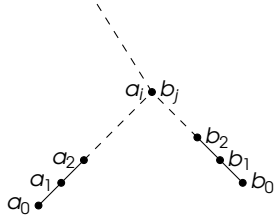
où $f : \mathbb{G} \to S \subset \mathbb{Z}$ est surjective (S pas trop grand).

Puis, on considère deux instances (a_i) et (b_i) de cette suite, avec deux conditions initiales $a_0 = y^\alpha$ et $b_0 = y^\beta$ telles que $\mathrm{pgcd}(\alpha - \beta, n) = 1$. **S'il advient que** $a_i = b_j$, alors on aura :

$$y^\alpha g^{f(a_0)+\cdots+f(a_{i-1})} = y^\beta g^{f(b_0)+\cdots+f(b_{j-1})}$$

Puis, ℓ est solution de :

$$(\beta - \alpha)\ell = \Big(f(a_0) + \cdots + f(a_{i-1})\Big) - \Big(f(b_0) + \cdots + f(b_{j-1})\Big)$$



Complexité. Avec le paradoxe des anniversaires, on obtient une collision après avoir calculé $O(\sqrt{n})$ termes au total.

Algorithme λ de Pollard : exemple

Exemple. On prend $\mathbb{G} = \mathbb{F}_{11}^{\times}$, engendré par $g = 2$.

1. On cherche le logarithme en base g de $y = 3$.
2. On prend la fonction $f : \mathbb{G} \to \mathbb{Z}$ définie par $f(x) = x$ (on ne pourra pas toujours).
3. On choisit $\alpha = 2$ et $\beta = 3$.
4. Alors, les suites (a_i) et (b_i) sont :

i	a_i	b_i
0	$3^2 = 9$	$3^3 = 5$
1	$9 \times 2^9 = 10$	$5 \times 2^5 = 6$
2	$10 \times 2^{10} = 10$	$6 \times 2^6 = 10$

5. On a $a_1 = b_2$. Donc ℓ est solution de

$$(\beta - \alpha)\ell = f(a_0) - f(b_0) - f(b_1) \mod 10$$

6. On obtient $\ell = 9 - 5 - 6 \equiv 8 \mod 10$.

Une variation de l'idée de la méthode ρ de Pollard

Idée : on réécrit $yg^{-\ell} = 1$ et on cherche encore une collision sur les $\{y^a g^b \mid (a,b) \in \mathbb{Z}\}$.

S'il advient que $y^a g^b = y^c g^d$, alors on a $\ell \equiv (d-b) \times (a-c)^{-1} \mod n$, pourvu que $a - c$ soit inversible dans $\mathbb{Z}/n\mathbb{Z}$.

Il faut maintenant construire une suite récurrente aléatoire $(x_i)_{i\in\mathbb{N}}$ où

$$x_{i+1} = f(x_i) \quad \text{et} \quad x_i = y^{a_i} g^{b_i}.$$

En **pratique**, on obtient de bons résultats en choisissant f de la sorte. On se fixe trois sous-ensembles **quelconques** S_0, S_1, S_2 de $\mathbb{G}$.

$$f(x) = \begin{cases} yx & \text{si } x \in S_0 \quad \text{(on incrémente } a\text{)} \\ x^2 & \text{si } x \in S_1 \quad \text{(on double } a \text{ et } b\text{)} \\ gx & \text{si } x \in S_2 \quad \text{(on incrémente } b\text{)} \end{cases}$$

Autrement dit, les séquences (a_n) et (b_n) sont régies par :

$$a_{n+1} = \begin{cases} n+1 & \text{si } a_n \in S_0 \\ 2n & \text{si } a_n \in S_1 \\ n & \text{si } a_n \in S_2 \end{cases} \quad \text{et} \quad b_{n+1} = \begin{cases} n & \text{si } b_n \in S_0 \\ 2n & \text{si } b_n \in S_1 \\ n+1 & \text{si } b_n \in S_2 \end{cases}$$

Ensuite, on peut utiliser la méthode de recherche de collision de Floyd ($x_i \stackrel{?}{=} x_{2i}$) pour chercher les collisions. **Complexité** : $O(\sqrt{n})$ par le paradoxe des anniversaires.

Méthode ρ de Pollard : exemple

Exemple sans la recherche de collision de Floyd (pour clarifier). On prend $\mathbb{G} = \mathbb{F}_{11}^{\times}$, engendré par $g = 2$. On cherche le logarithme en base g de $y = 3$ (même exemple que pour λ).

1. On partitionne $\mathbb{G} = \{1, \dots, 10\}$ en trois ensembles de taille semblable :

$$S_0 = \{1, 2, 3\}, \quad S_1 = \{4, 5, 6\}, \quad S_2 = \{7, 8, 9, 10\}.$$

2. On initialise avec $a_0 = 0$, $b_0 = 0$ et $x_0 = g^{b_0} = 1$. On obtient alors

i	x_i	a_i	b_i
0	1	0	0
1	$yx_0 = 3$	$a_0 + 1 = 1$	$b_0 = 0$
2	$yx_1 = 9$	$a_1 + 1 = 2$	$b_1 = 0$
3	$gx_2 = 18 \equiv 7$	$a_2 = 2$	$b_2 + 1 = 1$
4	$gx_3 = 14 \equiv 3$	$a_3 = 2$	$b_3 + 1 = 2$

3. On obtient alors

$$(a_4 - a_1)\ell = b_1 - b_4 \mod 10 \equiv \ell = -2 \equiv 8 \mod 10.$$

En pratique, avec la recherche de collision de Floyd on aurait plutôt calculé les $(x_i, a_i, b_i, x_{2i}, a_{2i}, b_{2i})$ et simplement testé si $x_i = x_{2i}$.

Plan

Motivation

L'**algorithme de Pohlig-Hellman** couplé à la méthode « pas de bébé – pas de géant » (par exemple) permet de résoudre le problème du logarithme discret dans un groupe $\mathbb{G}$ d'ordre n.

Ces algorithmes utilisent **uniquement la structure du groupe** $\mathbb{G}$. On dit qu'on résout le logarithme discret dans un groupe générique.

La **complexité** de Pohlig-Hellman est approximativement de $O(\sqrt{p})$ où p est le plus grand facteur premier de n.

Question. Est-il possible de faire sensiblement mieux ?

Théorème de Shoup sur la difficulté du log discret

Définition. Un groupe générique d'ordre n est la donnée :

- d'un groupe $\mathbb{G}$ d'ordre n,
- d'un isomorphisme de groupe aléatoire $\sigma : \mathbb{Z}/n\mathbb{Z} \to \mathbb{G}$.

Idée : on fait un codage quelconque (aléatoire) de $\mathbb{G}$ pour masquer toute éventuelle structure supplémentaire.

On a alors le résultat suivant :

Théorème. (Shoup 1997) Soit $(\mathbb{G}, \sigma)$ un groupe générique d'ordre $n = p^t s$ où p est premier, et p et s sont premiers entre eux. Soit g un générateur de $\mathbb{G}$ et $y \in \mathbb{G}$. Alors, pour tout algorithme $L(y, g)$ faisant au plus m requêtes à σ, on a

$$\mathbb{P}\left(g^{L(y,g)} = y\right) \leq \frac{m^2}{p}.$$

Autrement dit, pour obtenir une probabilité constante de réussite, il faut que m soit au moins de l'ordre de $\sqrt{p}$.

« La complexité du log discret dans un groupe générique est $\Theta(\sqrt{p})$. »

Plan

Plan

Rappels.

- Un entier x est B-**lisse** / B-**friable** si tous ses facteurs premiers sont plus petits que B.
- **Théorème de Canfield, Erdös, Pomerance, très simplifié** : la proportion d'entiers B-lisses inférieurs à N est d'environ
$$\varepsilon^{-\varepsilon} \quad \text{avec} \quad \varepsilon = \frac{\log N}{\log B}.$$
- Quand B est de taille « moyenne », on peut trouver efficacement tous les facteurs $\leq B$ d'un entier N en temps moyen
$$O\big(L_{\log B}(\tfrac{1}{2}, \sqrt{2})^{1+o(1)} \log^d N\big), \quad \text{où } d = 2 \text{ en pratique.}$$
Méthode utilisée : **ECM de Lenstra**.

Objectif

Dans ce cours, on se place dans le cas plus simple où $q = p$ est **premier**.

Problème. Étant donné g un générateur de $\mathbb{F}_p^\times$ et $y \in \mathbb{F}_p^\times$, on cherche $x \in \{0, \dots, p-2\}$ tel que $y = g^x$.

Idée. Exprimer l'entier $x = \log_g y$ recherché comme une combinaison linéaire de

- certains entiers u_i d'une forme particulière ;
- certains $\log_g b_j$, où les b_j sont « petits » ;

puis résoudre un système linéaire.

Idée

Comment faire ? Voici une stratégie (à raffiner) :

1. **Base de facteurs.** On se fixe une base de facteurs $\mathcal{B} = \{b_1, \dots, b_m\}$ où les entiers $b_j \in \mathbb{F}_p^\times$ sont « petits ».
2. **Collecte de relations.** On cherche des entiers u_i tels que g^{u_i} se décompose dans la base $\mathcal{B}$:
$$g^{u_i} = b_1^{e_1} \dots b_m^{e_m} \mod p$$
On a alors la *relation* linéaire :
$$u_i = e_1 \log_g(b_1) + \dots + e_m \log_g(b_m) \mod (p-1)$$
3. **Algèbre linéaire.** Une fois qu'on a obtenu un certain nombre de relations, on résout le système linéaire pour trouver tous les $\log_g(b_j)$.
4. **Résolution individuelle.** On cherche $v \in \mathbb{N}$ tel que yg^v s'écrit dans $\mathcal{B}$ comme
$$yg^v = b_1^{f_1} \dots b_m^{f_m} \mod p\,.$$
Puis on obtient
$$\log_g y = -v + f_1 \log_g(b_1) + \dots + f_m \log_g(b_m)\,.$$

Exemple : effritement

Présentation de l'algorithme par l'exemple.

Soit $p = 160499$.

L'élément $g = 19$ est un générateur de $\mathbb{F}_p^\times$.

On fixe la base de facteurs

$$\mathcal{B} = \{2, 3, 5, 7, 11\}.$$

On calcule alors

i	g^i	factorisation	décomp. $\in \mathcal{B}$?
1	19	19	×
2	361	19^2	×
3	6859	19^3	×
4	130321	19^4	×
5	68614	$2 \times 7 \times 13^2 \times 29$	×
6	19674	$2 \times 3^2 \times 1093$	×
7	52808	$2^3 \times 7 \times 23 \times 41$	×
	...		
30	100800	$2^6 \times 3^2 \times 5^2 \times 7$	✓
	...		
174	108900	$2^2 \times 3^2 \times 5^2 \times 11^2$	✓
	...		
239	39366	2×3^9	✓
	...		
297	704	$2^6 \times 11$	✓
	...		
357	17820	$2^2 \times 3^4 \times 5 \times 11$	✓

Remarque : pas besoin de factoriser complètement en pratique (cf crible quadratique).

Exemple : mise en place du système

On peut retranscrire les relations obtenues de la manière suivante.

i	g^i	factorisation
30	100800	$2^6 \times 3^2 \times 5^2 \times 7$
174	108900	$2^2 \times 3^2 \times 5^2 \times 11^2$
239	39366	2×3^9
297	704	$2^6 \times 11$
357	17820	$2^2 \times 3^4 \times 5 \times 11$

$$\Longrightarrow \begin{cases} 30 = & 6\log_g 2 + 2\log_g 3 + 2\log_g 5 + \log_g 7 \\ 174 = & 2\log_g 2 + 2\log_g 3 + 2\log_g 5 + 2\log_g 11 \\ 239 = & \log_g 2 + 9\log_g 3 \\ 297 = & 6\log_g 2 + \log_g 11 \\ 357 = & 2\log_g 2 + 4\log_g 3 + \log_g 5 + \log_g 11 \end{cases}$$

On obtient donc une équation matricielle :

$$\begin{pmatrix} 6 & 2 & 2 & 1 & 0 \\ 2 & 2 & 2 & 0 & 2 \\ 1 & 9 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 \\ 2 & 4 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \log_g 2 \\ \log_g 3 \\ \log_g 5 \\ \log_g 7 \\ \log_g 11 \end{pmatrix} = \begin{pmatrix} 30 \\ 174 \\ 239 \\ 297 \\ 357 \end{pmatrix}$$

et on veut maintenant retrouver $\log_g 2$, $\log_g 3$, $\ldots$, $\log_g 11$. On peut résoudre le système linéaire...

Question. Mais dans quoi (quel anneau / corps) vit ce système ?

$\Longrightarrow$ Ici, on est dans $\mathbb{Z}/(p-1)\mathbb{Z}$.

Exemple : algèbre linéaire

Question : comment résoudre un système linéaire dans $\mathbb{Z}/(p-1)\mathbb{Z}$?

Comme d'habitude : restes chinois + relèvement de Hensel.

Par exemple, ici $\ell = 13$ divise $p-1$. On résout

$$\begin{pmatrix} 6 & 2 & 2 & 1 & 0 \\ 2 & 2 & 2 & 0 & 2 \\ 1 & 9 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 \\ 2 & 4 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \log_g 2 \\ \log_g 3 \\ \log_g 5 \\ \log_g 7 \\ \log_g 11 \end{pmatrix} = \begin{pmatrix} 30 \\ 174 \\ 239 \\ 297 \\ 357 \end{pmatrix} \equiv \begin{pmatrix} 4 \\ 5 \\ 5 \\ 11 \\ 6 \end{pmatrix} \mod 13$$

Cela donne

$$\begin{pmatrix} \log_g 2 \\ \log_g 3 \\ \log_g 5 \\ \log_g 7 \\ \log_g 11 \end{pmatrix} \equiv \begin{pmatrix} 6 \\ 10 \\ 5 \\ 3 \\ 1 \end{pmatrix} \mod 13$$

Après résolution modulo les autres facteurs de $p-1$, puis reconstruction, on obtient la solution

$$\begin{pmatrix} \log_g 2 \\ \log_g 3 \\ \log_g 5 \\ \log_g 7 \\ \log_g 11 \end{pmatrix} = \begin{pmatrix} 120659 \\ 147118 \\ 54722 \\ 156380 \\ 78833 \end{pmatrix}$$

Exemple : résolution individuelle

Résolution individuelle. Supposons maintenant que l'on souhaite retrouver le logarithme de $y = 29750$ en base $g = 19$, toujours dans $\mathbb{F}^{\times}_{160499}$.

Idée : un peu comme dans « baby-step-giant-step », en partant de y, on cherche à retomber dans la base de facteurs en multipliant par des g^v.

v	$y \times g^v$	factorisation	décomp. $\in \mathcal{B}$?
0	29570	$2 \times 5 \times 2957$	$\times$
1	80333	$11 \times 67 \times 109$	$\times$
2	81836	$2^2 \times 41 \times 499$	$\times$
3	110393	101×1093	$\times$
4	10980	$2^2 \times 3^2 \times 5 \times 61$	$\times$
5	48121	48121	$\times$
6	111804	$2^2 \times 3 \times 7 \times 11^3$	✓

Puis, on reconstruit

$$\begin{aligned}\log_g(y) &= -v + 2\log_g(2) + \log_g(3) + \log_g(7) + 3\log_g(11) \\ &= -6 + 2 \times 120659 + 147118 + 156380 + 3 \times 78833 = 139317\end{aligned}$$

Remarque. En pratique, on aurait dû utiliser l'algorithme de Pohlig–Hellman, et calculer le logarithme discret dans les sous-groupes de $\mathbb{Z}/(p-1)\mathbb{Z}$. C'est (grossièrement) équivalent à nos calculs modulo ℓ précédents.

Complexité

Si on note $\pi(x, B)$ la probabilité qu'un entier $\leq x$ soit B-lisse. Alors :

1. **(base de facteurs)** Complexité en $O(B)$
2. **(collection des relations)** Les éléments g^i peuvent être considérés comme aléatoires dans $\mathbb{F}_p^\times$, donc complexité en $O(B/\pi(p, B))$.
3. **(algèbre linéaire)** Complexité en $O(B^{2+o(1)})$ avec Wiedemann par blocs (les matrices sont creuses).
4. **(logarithme individuel)** Les $g^v y$ sont aléatoires dans $\mathbb{F}_p^\times$. Donc, complexité en $O(1/\pi(p, B))$.

Donc la complexité du calcul d'indice est

$$C = \frac{B}{\pi(p,B)} + B^{2+o(1)} + \frac{1}{\pi(p,B)} \leq 2B^{1+o(1)} \max\left\{\frac{1}{\pi(p,B)}, B\right\}$$

Cette complexité est minimale pour $B = \frac{1}{\pi(p,B)}$. Pour $\pi(p, B) = \left(\frac{\log p}{\log B}\right)^{-\frac{\log p}{\log B}}$, on obtient après résolution

$$B = 2^{\frac{1}{\sqrt{2}}\sqrt{\log p \log\log p}}$$

Finalement, la **complexité de la résolution par calcul d'indice** est en

$$O^{\sim}\left(L_{\log p}(\tfrac{1}{2}, \sqrt{2})\right).$$

Plan

Algorithmes plus évolués

Comme pour la factorisation, on peut améliorer la complexité grâce à

- un crible dans un corps de nombres *(number field sieve, NFS)*
- un crible dans un corps de fonctions *(function field sieve, FFS)*

Dans $\mathbb{F}_q$, où q n'est pas nécessairement premier, on atteint alors une complexité en

$$L_{\log q}\left(\frac{1}{3}, \left(\frac{128}{9}\right)^{1/3}\right)$$

Améliorations en petite caractéristique

Il existe des améliorations suivant la **caractéristique du corps** $\mathbb{F}_q = \mathbb{F}_{p^k}$.

On exprime la caractéristique p en fonction de q, par $p =: L_{\log q}(\ell_p, c_p)$.

- Pour $\ell_p < 1/3$, on dit que la caractéristique est **petite**.
- Pour $\ell_p > 2/3$, on dit que la caractéristique est **grande**.
- Sinon, elle est **moyenne**.

En **petite caractéristique**, les améliorations sont drastiques :

- à p constant, Joux (2013) donne un algorithme en $L_{\log q}(\frac{1}{4}, c)$ où $c > 0$
- pour $\ell_p = \varepsilon < 1/3$ petit, il existe un algorithme heuristique en temps quasi-polynomial, i.e. en temps $L_{\log q}(\varepsilon + o(1), 1)$.

Cette amélioration a eu une conséquence pratique pour les **couplages** (*pairings*), utilisés récemment en cryptographie (Diffie-Hellman à trois, IBE, ...), qui étaient originellement implantés en petite caractéristique.

En **grande caractéristique**, on a « seulement » une amélioration sur l'exposant secondaire : on passe de $(\frac{128}{9})^{1/3}$ à $(\frac{32}{9})^{1/3}$ grâce un algorithme de crible par corps de nombres *spécial* (SNFS).

En **moyenne caractéristique**, les améliorations sont encore moins notables (SNFS également).

Résolution en pratique

Évolution des résolutions du logarithme discret en **petite** caractéristique :

année	q	auteurs	heures CPU
1992	2^{401}	Gordon, McCurley	114 000
2005	2^{613}	Joux, Lercier	26 000
2013	2^{1778}	Joux	220
2013	2^{6168}	Joux	550
2014	2^{9234}	Granger, Kleinjung, Zumbragel	398 000

En **moyenne** caractéristique :

- $\mathbb{F}_q$ avec $q = 33341353^{57}$ (1425 bits)
- auteur : Antoine Joux
- 32 000 heures CPU

En **grande** caractéristique :

- $\mathbb{F}_{p^2}$ avec p premier de 160 chiffres (532 bits)
- auteurs : Barbulescu, Gaudry, Guillevic, Morain (2014)
- 68 jours CPU + 30 heures GPU

Et dans d'autres groupes ?

On pourrait penser **adapter** l'idée du calcul d'indice pour n'importe quel groupe.

- Cependant, il faut une notion de **friabilité**, autrement dit de « norme » pour les éléments du groupe.
- C'est difficile à définir pour les courbes elliptiques, par exemple.

Pour les courbes elliptiques, les meilleurs méthodes restent des optimisations du ρ de Pollard + Pohlig–Hellman.

année	courbe	auteurs	temps de calcul
2004	Koblitz, $q = 2^{109}$	Monico *et al.*	17 mois de calcul, 2600 machines
2016	binaire, 117.35 bits	Bernstein *et. al*	6 mois, jusque 576 FPGA en parallèle
2017	couplage, 114 bits	Kusaka *et al.*	6 mois sur 2000 CPUs

Et dans d'autres groupes ?

Un autre groupe de nature géométrique ?

Le groupe des points d'une **Jacobienne de courbe hyperelliptique**. Notons g le genre de la courbe.

- La taille du groupe $\mathrm{Jac}(\mathcal{C})(\mathbb{F}_q)$ est en $O(q^g)$.
- Alors, il existe un algorithme (Gaudry, Thomé, Thériault, Die) résolvant le problème du logarithme discret en temps $O^{\sim}(\min\{q^{g/2}, q^{2-2/g}\})$.

$\Longrightarrow$ on ne peut prendre que $g = 2$ ou $g = 3$, mais les opérations sont bien plus coûteuses que pour les courbes elliptiques.

Certaines courbes elliptiques sont « fragiles ».

Grâce à une **descente de Weil**, on transporte le problème du logarithme discret dans le groupe de points $E(\mathbb{F}_{q^k})$ avec $k > 1$, dans un groupe associé à une **courbe hyperelliptique** $\mathcal{X}$ définie sur $\mathbb{F}_q$ avec $q \ll q^k$.

Conseil pour de la cryptographie : courbes elliptiques avec p (presque) premier ou courbes hyperelliptiques de genre 2 (pas plus). En évitant les sous-familles fragiles.

Références

Calcul d'indice :


A subexponential algorithm for the discrete logarithm problem with applications to cryptography. Adleman. FOCS. **1979**.

Fast, rigorous factorization and discrete logarithm algorithms. Pomerance. Discrete algorithms and complexity. **1987**.

NFS et FFS :


Using number fields to compute logarithms in finite fields. Schirokauer. Math. Comp. **2000**.


Function field sieve method for discrete logarithmsover finite fields.. Adleman, Huang. Inf. Comput.. **1999**.

Algorithme quasi-polynomial en petite caractéristique :

A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. Barbuslecu, Gaudry, Joux, Thomé. EUROCRYPT. **2014**.

Un **état de l'art** assez récent :


The Past, evolving Present and Future of Discrete Logarithm. Joux, Odlyzko, Pierrot.
`https://members.loria.fr/CPierrot/papers/DlogSurvey.pdf`. **2014**.

Fin du cours

Questions ?