CRYPTA
Chapitre 1 : Problème du logarithme discret

1 Exponentiation discrète et logarithme discret

1.1 Exponentiation discrète

Une opération particulièrement importante en cryptographie à clé publique est l'*exponentiation discrète* dans un groupe abélien fini $\mathbb{G}$ (généralement noté multiplicativement). Elle consiste, étant donnés un élément g de $\mathbb{G}$ et un entier $a \in \mathbb{N}$, à calculer l'élément g^a de $\mathbb{G}$. Calculer cet élément demande *a priori* a multiplications dans le groupe $\mathbb{G}$ mais on remarque facilement qu'il peut être obtenu en seulement $O(\log a)$ multiplications en décomposant l'entier a en base 2. En effet, si

$$a = \sum_{i=0}^{\ell-1} a_i 2^i \in \mathbb{N} \text{ avec } a_i \in \{0,1\} \text{ pour } i \in \{0,\ldots,\ell-1\},$$

où ℓ est la taille en bits de l'entier n, nous avons

$$\begin{aligned} g^a = \prod_{i=0}^{\ell-1} g^{a_i 2^i} &= g^{a_0} \left(g^{a_1}\right)^2 \left(g^{a_2}\right)^4 \left(g^{a_3}\right)^8 \ldots \left(g^{a_{\ell-1}}\right)^{2^{\ell-1}} \\ &= g^{a_0} \left(g^{a_1} \left(g^{a_2} \left(g^{a_3} \ldots \left(g^{a_{\ell-1}}\right)^2 \right)^2 \right)^2 \right)^2 . \end{aligned}$$

Cette dernière expression peut être calculée en $\ell - 1$ élévations au carré entrelacées avec des multiplications par g^{a_i} pour $i \in \{0,\ldots,\ell-2\}$. En moyenne, le nombre de a_i non nuls pour $i \in \{0,\ldots,\ell-2\}$ est égal à $(\ell-1)/2$. Le nombre de multiplications nécessaires pour calculer g^a est donc égal à $3(\ell-1)/2$ en moyenne et à $2(\ell-1)$ dans le pire des cas. L'algorithme qui effectue ce calcul (*cf.* Algorithme (1)) est appelé algorithme d'*exponentiation dichotomique* ou algorithme « élever au carré et multiplier » (*square and multiply* en anglais).

Algorithme 1 Exponentiation discrète dichotomique

Entrée: $g \in \mathbb{G}$, $a = \sum_{i=0}^{\ell-1} a_i 2^i \in \mathbb{N}$ avec $a_i \in \{0,1\}$ pour $i \in \{0,\ldots,\ell-1\}$
Sortie: $g^a \in \mathbb{G}$

```
h ← 1_G
pour i de ℓ − 1 à 0 faire                    ▷ l'indice i varie en décroissant
    h ← h^2
    si a_i = 1 alors
        h ← h · g
    fin si
fin pour
retourner h
```

Il existe de nombreuses variantes de cet algorithme en fonction du contexte et du groupe $\mathbb{G}$ considéré.

1.2 Logarithme discret

Étant donnés un groupe abélien $\mathbb{G}$ d'ordre connu q, un élément $g \in \mathbb{G}$ et un élément h appartenant au sous-groupe multiplicatif engendré par g (noté $\langle g \rangle = \{g^i, i \in \mathbb{Z}\}$), le problème qui consiste à retrouver le plus petit entier positif x tel que $h = g^x$ est appelé *problème du logarithme discret*. En faisant une recherche exhaustive de cette valeur, un algorithme peut retrouver ce logarithme discret $x = \log_g(h)$ en x multiplications dans le groupe $\mathbb{G}$. Nous allons améliorer cet algorithme trivial et montrer comment il est possible de retrouver x en $O(\sqrt{|\mathbb{G}|})$ multiplications dans $\mathbb{G}$.
Il suffit de remarquer que pour tout élément $h = g^x \in \mathbb{G}$, l'entier x s'écrit sous la forme $x = x_1 T + x_0$ avec $0 \leq x_1 < T$ et $0 \leq x_2 < T$ pour $T = \lceil \sqrt{q} \rceil + 1$. L'égalité $h = g^x$ avec $x = x_1 T + x_0$ s'écrit également

$$h \cdot \left(g^{-T}\right)^{x_1} = h g^{-x_1 T} = g^{x_0}.$$

En appliquant un compromis temps-mémoire, il suffit dans un premier temps de calculer les T valeurs prises par le terme de droite de cette égalité (*i.e.* les éléments g^i pour $i \in \{0, \ldots, T-1\}$ et de stocker les valeurs obtenues (g^i, i) dans une table de hachage indexée par les éléments de groupe.
Dans un second temps, l'algorithme va rechercher parmi les valeurs $h \cdot \left(g^{-T}\right)^j$ pour $j \in \{0, \ldots, T-1\}$ si l'une d'elles apparaît dans la table de hachage.
Le nombre de multiplications à effectuer est égal à :

- T pour la construction de la table des éléments g^i pour $i \in \{0, \ldots, T-1\}$;
- $2(\lfloor \log(q-T) \rfloor + 1)$ pour le calcul de g^{-T} (puisque $g^{q-T} \cdot g^T = g^q = 1_{\mathbb{G}}$) ;
- T pour la recherche des éléments $h \cdot \left(g^{-T}\right)^j$ pour $j \in \{0, \ldots, T-1\}$.

soit un total de $2(T + \lfloor \log(q-T) \rfloor + 1) = O(T) = O(\sqrt{q})$. La complexité en mémoire est donnée par la table de hachage qui utilise $O(T) = O(\sqrt{q})$ éléments du groupe $\mathbb{G}$. □

Algorithme 2 Algorithme de Shanks (ou « pas de bébé, pas de géant »)

Entrée: $g, h \in \mathbb{G}$, $q = |\mathbb{G}|$
Sortie: $x \in \{0, \ldots, q-1\}$ tel que $h = g^x$

```
T ← ⌈√q⌉ + 1
ϒ ← ∅
pour i de 0 à T − 1 faire
    x_i ← g^i                          ▷ 1 multiplication à chaque itération (x_{i+1} ← x_i · g)
    ϒ ← ϒ ∪ (x_i, i)                   ▷ table de hachage
fin pour
k ← g^{q−T}                            ▷ O(log(q − T)) multiplications
pour j de 0 à T − 1 faire
    y ← h · k^j                        ▷ 1 multiplication à chaque itération (y ← y · k)
    si y = x_i avec (x_i, i) ∈ ϒ alors
        retourner i + Tj mod q
    fin si
fin pour
```

L'algorithme obtenu (*cf.* Algorithme (2)) a été proposé par D. Shanks en 1970. Il est appelé l'algorithme « *pas de bébé, pas de géant* » (la construction des valeurs g^i pour $i \in \{0, \ldots, T-1\}$ étant les « pas de bébé », celle des $\left(g^{-T}\right)^j$ pour $j \in \{0, \ldots, T-1\}$ les « pas de géant »).

L'inconvénient majeur de l'algorithme précédent est sa complexité en mémoire. En 1978, J. M. POLLARD a proposé un algorithme probabiliste dont la complexité en multiplications dans $\mathbb{G}$ est similaire mais qui ne requiert le stockage que d'un nombre constant d'éléments de $\mathbb{G}$ (*cf.* TD).

1.3 Algorithme de Pohlig-Hellman

Nous allons voir que dans un groupe $\mathbb{G}$ d'ordre n dont la décomposition en facteurs premiers est connue, il est possible de réduire le problème du logarithme discret dans $\mathbb{G}$ au problème du logarithme discret dans chacun de ses sous-groupes d'ordre premier.
Soit $\mathbb{G}$ un groupe cyclique d'ordre fini n dont la décomposition en facteurs premiers est connue.

1. **Cas 1 :** si $n = pq$ est le produit de nombres premiers distincts p et q alors il existe un algorithme qui résout le problème du logarithme discret dans $\mathbb{G}$ en $O(\sqrt{p} + \sqrt{q})$ multiplications dans $\mathbb{G}$.

   Soient g et h deux éléments de $\mathbb{G}$ avec $g \neq 1_{\mathbb{G}}$. Le logarithme discret de h en base g est l'élément x de $\mathbb{Z}_n$ tel que $h = g^x$. Si la factorisation de $n = pq$ est connue et si les valeurs de $(x \bmod p)$ et $(x \bmod q)$ sont connues, il est facile de reconstruire x par le théorème des restes chinois.
   En posant $x = x_0 p + x_1$ avec $x_0, x_1 \in \mathbb{N}$ et $x_1 < p$, nous avons $x_1 \equiv x \bmod p$ et pour obtenir x_1 il suffit de remarquer que $h^q = (g^x)^q = (g^q)^x$ et que l'élément g^q est d'ordre p. Nous avons alors

   $$h^q = (g^q)^{x_0 p + x_1} = g^{x_0 pq + q x_1} = (g^{pq})^{x_0} \cdot (g^q)^{x_1} = (g^q)^{x_1}$$

   puisque $g^{pq} = g^n = 1_{\mathbb{G}}$.
   L'entier x_1 est donc le logarithme discret de h^q en base g^q. En appliquant l'un des algorithmes de résolution de logarithme discret vu précédemment, il est ensuite possible de retrouver x_1 en $O(\sqrt{p})$ multiplications dans $\mathbb{G}$. Le coût total pour obtenir $(x \bmod p)$ est donc $O(\sqrt{p} + \log(q))$ multiplications dans $\mathbb{G}$ (le terme $O(\log(q))$ provient des deux exponentiations discrètes nécessaires pour le calcul de h^q et g^q).
   De même, il est possible de retrouver la valeur de $(x \bmod q)$ en $O(\sqrt{q} + \log(p))$ multiplications dans $\mathbb{G}$ et après application du théorème des restes chinois, l'algorithme retourne x en $O(\sqrt{p} + \sqrt{q})$ multiplications dans $\mathbb{G}$.
2. **Cas 2** si $n = p^e$ où p est un nombre premier et $e \geq 2$ est un entier alors il existe un algorithme qui résout le problème du logarithme discret dans $\mathbb{G}$ en $O(e(\sqrt{p} + \log(n)))$ multiplications dans $\mathbb{G}$.

   Notons encore x le logarithme discret de h en base g avec $g, h \in \mathbb{G}$. En décomposant x en base p, nous avons

   $$x = x_0 + x_1 p + \cdots + x_{e-1} p^{e-1} \text{ avec } x_i \in \{0, \ldots, p-1\} \text{ pour } i \in \{0, \ldots, e-1\}.$$

   L'approche est similaire à celle de la question précédente mais en calculant cette fois les valeurs x_i successivement pour $i \in \{0, \ldots, e-1\}$. Pour obtenir la valeur de $x_0 = (x \bmod p)$, l'algorithme calcule le logarithme discret de $h^{p^{e-1}}$ en base $g^{p^{e-1}}$. Cet

élément engendre un sous-groupe d'ordre p et le calcul de x_0 demande $O(\sqrt{p}+\log(n))$ multiplications dans le groupe $\mathbb{G}$.

Pour déterminer la valeur de x_1, remarquons que

$$hg^{-x_0} = g^{x_1 p+\cdots+x_{e-1}p^{e-1}} = (g^p)^{x_1+\cdots+x_{e-1}p^{e-2}}.$$

En élevant cette égalité à la puissance p^{e-2}, nous obtenons $(hg^{-x_0})^{p^{e-2}} = (g^{p^{e-1}})^{x_1}$ et la valeur de x_1 s'obtient donc comme la valeur du logarithme discret de $(hg^{-x_0})^{p^{e-2}}$ en base $g^{p^{e-1}}$ (ce qui demande de nouveau $O(\sqrt{p}+\log(n))$ multiplications dans le groupe $\mathbb{G}$).

En itérant cette approche, nous voyons que le logarithme discret de l'élément

$$(hg^{-(x_0+px_1+\cdots+p^{\ell-1}x_{\ell-1})})^{p^{e-\ell-1}}$$

en base $g^{p^{e-1}}$ est égal à x_ℓ pour tout $\ell \in \{2,\ldots,e-1\}$. Nous pouvons donc construire un algorithme qui calcule la valeur de x en $O(e(\sqrt{p}+\log(n)))$ multiplications dans le groupe $\mathbb{G}$.

3. **Cas général :** si $n = q_1^{e_1}\cdots q_k^{e_k}$ où les q_i sont des nombres premiers deux à deux distincts), alors il existe un algorithme qui résout le problème du logarithme discret en $O\left(\sum_{i=1}^{k} e_i(\sqrt{p_i}+\log(n))\right)$ opérations dans le groupe $\mathbb{G}$.

   Il suffit de combiner les méthodes des deux questions précédentes pour calculer la valeur de x modulo les $q_i^{e_i}$ pour $i \in \{1,\ldots,k\}$ puis de reconstituer la valeur de x en appliquant le théorème des restes chinois.

Notons LogarithmeDiscret(h,g) le logarithme discret de h en base g (calculé par exemple par l'algorithme de Shanks (2)) et RestesChinois$((x_1,n_1),(x_2,n_2),\ldots,(x_t,n_t))$ la valeur $x \in \mathbb{Z}_n$ telle que $x \equiv x_i \bmod n_i$ pour $i \in \{1,\ldots,t\}$ et $n = n_1 n_2 \ldots n_t$. L'algorithme (3) résout le problème du logarithme discret en $O\left(\sum_{i=1}^{k} e_i(\sqrt{p_i}+\log(n))\right)$ opérations dans le groupe $\mathbb{G}$. □

Algorithme 3 Algorithme de Pohlig-Hellman

Entrée: $g, h \in \mathbb{G}$ et $(q_1, e_1, \ldots, q_k, e_k)$ tel que $|\mathbb{G}| = n = q_1^{e_1}\ldots q_k^{e_k}$
Sortie: $x \in \{0,\ldots,n-1\}$ tel que $h = g^x$

```
pour i de 1 à k faire
    x_i ← 0
    g_i ← g^{n/q_i}
    pour j de 1 à e_i faire
        α ← (h · g^{-x_i})^{n/q_i^j}
        t ← LogarithmeDiscret(α, g_i)
        x_i ← x_i + t · p^{j-1}
    fin pour
fin pour
x ← RestesChinois((x_1, q_1^{e_1}), (x_2, q_1^{e_2}), ..., (x_k, q_1^{e_k}))
retourner x
```

L'algorithme précédent a été proposé en 1978 par S. Pohlig et M. Hellman.

2 Problème du logarithme discret dans $\mathbb{Z}_p^*$

Dans l'article fondateur de la cryptographie à clé publique, W. Diffie et M. Hellman ont suggéré d'utiliser la difficulté supposée du problème du logarithme discret dans certains groupes pour créer différentes primitives cryptographiques. Ils ont proposé d'utiliser le groupe multiplicatif d'un corps fini $\mathbb{Z}_p^*$ où p est un nombre premier. Dans cette section, nous allons voir qu'il existe des algorithmes sous-exponentiels pour résoudre le problème du logarithme discret dans ces groupes. Ces algorithmes appartiennent à la famille des algorithmes dits de *calcul d'indice* qui reposent sur l'existence d'éléments *friables* (*smooth*, en anglais) dans le groupe considéré.

2.1 Entiers friables

Définition 1 (Entiers friables)**.** *Soit $M \geq 0$ un nombre réel positif. Un entier $n \in \mathbb{N}$ est dit M-*friable *si tous les diviseurs premiers de n sont inférieurs à M.*

La complexité des algorithmes de logarithme discret dans $\mathbb{Z}_p^*$ repose de façon essentielle sur le nombre d'entiers M-friables inférieurs à p pour une valeur de M bien choisie.

Définition 2 (Fonction de Dickman-De Bruijn)**.** *Soient x, y deux nombres réels positifs. Notons*

$$\Psi(x, y) = \#\{n \in \mathbb{N}, n \leq x \text{ et } n \text{ est } y\text{-friable}\}.$$

La fonction $\Psi : \mathbb{R}_+ \times \mathbb{R}_+ \longrightarrow \mathbb{N}$ est appelée fonction de Dickman-De Bruijn.

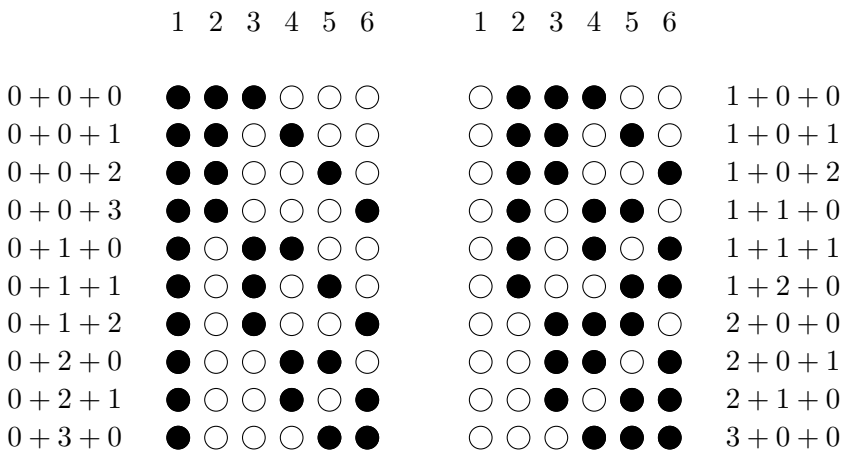
Il existe des estimations très fines de cette fonction suivant les valeurs de x et y. Nous allons en démontrer une par des méthodes élémentaires en admettant le théorème des nombres premiers (où pour tout nombre réel positif x, $\pi(x)$ désigne le nombre d'entiers premiers inférieurs à x) :

Théorème 1. *Lorsque x tend vers $+\infty$, nous avons $\pi(x) \sim \dfrac{x}{\ln(x)}$*
*(*i.e. *$\pi(x)\ln(x)/x$ tend vers 1 lorsque x tend vers $+\infty$).*

Notons $(p_n)_{n\geq 1}$ la suite des nombres premiers (avec $p_1 = 2$, $p_2 = 3$, ...). Le théorème des nombres premiers s'énonce aussi sous la forme $p_n \sim n \ln n$ (*i.e.* $p_n/(n \ln n)$ tend vers 1 lorsque n tend vers $+\infty$).

Lemme 1. *Soient $n \geq 1$ et $N \geq 1$ deux entiers. Le nombre de suites d'entiers $e = (e_1, \ldots, e_n)$ telles que $e_1 + \cdots + e_n \leq N$ est égal au coefficient binomial $\binom{N+n}{n}$.*

Démonstration. Il suffit représenter une telle suite en coloriant les entiers de 1 à $N + n$ en deux couleurs : les e_1 premiers en blanc, suivi d'un noir, suivi de e_2 blancs et ainsi de suite. Avec $N = 3$ et $n = 3$, nous voyons que le nombre de partitions en trois parts des entiers 0, 1, 2 et 3 est bien égal aux nombres de suites obtenues en suivant l'indication de l'énoncé :



Plus généralement, considérons l'ensemble $E_{N,n}$ constitués des sous-ensembles de $\{1,\dots,N+n\}$ à n éléments (correspondant aux entiers coloriés en noir). Posons f l'application injective

$$f:\left\{\begin{array}{ccc} E_{N,n} & \longrightarrow & \mathbb{N}^n \\ \{\alpha_1,\dots,\alpha_n\} & \longmapsto & (\alpha_1-1,\alpha_2-\alpha_1-1,\dots,\alpha_n-\alpha_{n-1}-1)\end{array}\right.$$

où nous supposons $\alpha_1<\cdots<\alpha_n$ dans la définition de f. Pour tout élément de $E_{N,n}$, la somme des éléments de $f(\{\alpha_1,\dots,\alpha_n\})$ est égal à α_n-n où $\alpha_n=\max\{\alpha_1,\dots,\alpha_n\}$. La somme des éléments de $f(\{\alpha_1,\dots,\alpha_n\})$ est donc bien majorée par N.
Réciproquement, toute suite d'entiers $e=(e_1,\dots,e_n)$ telle que $e_1+\cdots+e_n\leq N$ est bien obtenue comme image par f en posant

$$\begin{aligned}\alpha_1 &= e_1+1\\ \alpha_2 &= e_2+\alpha_1+1\\ &\vdots\\ \alpha_n &= e_n+\alpha_{n-1}+1\end{aligned}$$

de sorte que $\alpha_n=e_1+\cdots+e_n+n\leq N+n$ et $f(\{\alpha_1,\dots,\alpha_n\})=(e_1,\dots,e_n)$. Le cardinal recherché est donc égal au cardinal de $E_{N,n}$ c'est-à-dire au coefficient binomial $\binom{N+n}{n}$. □

Lemme 2. *Soient $n\geq 1$ un entier et $\lambda>2p_n$ un entier impair ; posons $N=\lfloor \ln\lambda/\ln p_n\rfloor$. La probabilité qu'un entier choisi uniformément aléatoirement dans $[1,\lambda]$ soit p_n-friable est supérieure à $(N^n/\lambda n!)$.*

Démonstration. Un entier t est p_n-friable s'il est de la forme

$$t=p_1^{e_1}p_2^{e_2}\cdots p_n^{e_n}\quad\text{avec}\quad e_1,\dots,e_n\in\mathbb{N}.$$

En notant $N=\lfloor \ln\lambda/\ln p_n\rfloor$, une condition suffisante pour que t appartienne à l'intervalle $[1,\lambda]$ est $e_1+\cdots+e_n\leq N$. En effet, par la croissance de la suite (p_i), nous avons

$$p_1^{e_1}p_2^{e_2}\cdots p_n^{e_n}\leq p_n^{e_1+\cdots+e_n}\leq p_n^N\leq\lambda.$$

Le nombre d'entiers p_n-friables inférieur à λ est donc minoré par le nombre de suite $(e_1,\dots,e_n)$ tels que $e_1+\cdots+e_n\leq N$, c'est-à-dire le coefficient binomial $\binom{N+n}{n}$ d'après la question précédente. Cet entier vérifie

$$\binom{N+n}{n}=\frac{(N+n)!}{N!n!}\geq\frac{N^n}{n!}$$

et la probabilité qu'un entier de $[1, \lambda]$ soit p_n-friable est donc supérieure à $(N^n/\lambda n!)$. □

Nous avons donc

$$\Psi(x^k, x) \geq \binom{\pi(x)+k}{k} \geq \frac{\pi(x)^k}{k!} \geq \left(\frac{\pi(x)}{k}\right)^k$$

et en appliquant le théorème des nombres premiers, nous obtenons pour x suffisamment grand

$$\Psi(x^k, x) \geq \left(\frac{x}{k \ln x}\right)^k.$$

Il existe des estimations plus fines de $\Psi(x, y)$ comme le développement asymptotique suivant :

$$\Psi(x^k, x) = x\rho(k) + (1-\gamma)\frac{x\rho(k-1)}{\log x} + O\left(x\rho(k)\frac{\log^2 k}{\log^2 y}\right),$$

où γ est la constante d'Euler, $\rho(k) = 1$ pour $0 \leq k \leq 1$, $\rho(k) = 1 - \log(k)$ pour $1 \leq k \leq 2$ et

$$\rho(u) = \frac{1}{u}\int_{u-1}^{u} \rho(t)dt, \text{ pour } u > 1.$$

L'estimation précédente sera cependant suffisante pour nos besoins. En posant

$$L_N(\alpha, c) = \exp(c(\ln N)^\alpha (\ln \ln N)^{1-\alpha}),$$

pour $\alpha \in [0, 1]$ et $c > 0$, nous obtenons en particulier que

$$\Psi(N, L_N(1/2, c))/N \simeq (L_N(1/2, 1/2c))^{-1}.$$

pour N suffisamment grand.

2.2 Méthode de Kraitchik - Calcul d'indice

Dans cette section, nous présentons et analysons un algorithme sous-exponentiel de résolution de logarithme discret dans un sous-groupe $\mathbb{G}$ de $\mathbb{Z}_p^*$. Soit γ un générateur de $\mathbb{G}$ et $\alpha \in \mathbb{G}$. L'algorithme utilise l'ensemble $\mathcal{B} = \{2, 3, 5, \ldots, p_k\}$ constitués des k premiers nombres premiers (*i.e.* $p_1 = 2$, $p_2 = 3$, ...) appelé *base de facteur* et l'idée générale (due à M. Kraitchik) est la suivante :

— dans une première étape, l'algorithme recherche des relations de la forme

$$\alpha^{r_i} \cdot \gamma^{s_i} \cdot h_i \equiv p_1^{e_{i,1}} \ldots p_k^{e_{i,k}} \bmod p \tag{1}$$

avec $r_i, s_i \in \mathbb{Z}_q$ et $h_i \notin \mathbb{G}$ pour $i \in \{1, \ldots, k\}$;

— dans une seconde étape, l'algorithme cherche à combiner ces relations (en appliquant des techniques d'algèbre linéaire) pour obtenir une relation de la forme $\alpha^r \gamma^s = 1$ et ainsi retrouver le logarithme discret de α en base γ.

Cet algorithme est appelé algorithme par *calcul d'indice* (*index calculus*, en anglais).

Notations. Soient p un nombre premier et q un diviseur premier de $p-1$. Notons $\mathbb{G}$ le sous-groupe de $\mathbb{Z}_p^*$ d'ordre q et $\mathbb{H}$ le sous-groupe de $\mathbb{Z}_p^*$ d'ordre $(p-1)/q$. Notons γ un générateur de $\mathbb{G}$ et supposons que $q^2 \nmid (p-1)$.

Lemme 3. $\mathbb{G} \cap \mathbb{H} = \{1\}$.

Démonstration. Soit $\theta \in \mathbb{G} \cap \mathbb{H}$. Nous avons $\theta^q \equiv 1 \bmod p$ et $\theta^{(p-1)/q} \equiv 1 \bmod p$, donc l'ordre de θ divise q et $(p-1)/q$. Comme q est premier, l'ordre de θ est égal à 1 ou q et comme par hypothèse q^2 ne divise pas $(p-1)$, nous obtenons que $\theta = 1$. □

Lemme 4. *Si r est tiré uniformément aléatoirement dans $\mathbb{Z}_q$ et h est tiré uniformément aléatoirement dans $\mathbb{H}$, alors l'élément $\gamma^r \cdot h$ est uniformément distribué dans $\mathbb{Z}_p^*$.*

Démonstration. Le morphisme de groupe défini par

$$\begin{array}{rcl} \mathbb{G} \times \mathbb{H} & \longrightarrow & \mathbb{Z}_p^* \\ (g, h) & \longmapsto & g \cdot h \end{array}$$

est injectif d'après la question précédente. Puisque $|\mathbb{G}| \times |\mathbb{H}| = (p-1)$, il est également surjectif. Pour une valeur de $r \in \mathbb{Z}_q$ tirée uniformément aléatoirement, γ^r est uniformément distribué dans $\mathbb{G}$. Si h est tiré uniformément aléatoirement dans $\mathbb{H}$, l'image du couple (γ^r, h) par ce morphisme est donc uniformément distribuée dans $\mathbb{Z}_p^*$. □

Nous avons besoin d'un algorithme probabiliste qui génère un élément uniformément distribué dans $\mathbb{H}$. En effet, le sous-groupe $\mathbb{H}$ est formé des éléments d'ordre $(p-1)/q$; il s'agit donc de l'image du morphisme

$$\begin{array}{rcl} \mathbb{Z}_p^* & \longrightarrow & \mathbb{Z}_p^* \\ x & \longmapsto & x^q \end{array}$$

et tout élément de $\mathbb{H}$ a q antécédents par ce morphisme. Pour générer un élément aléatoire de $\mathbb{H}$, il suffit donc de tirer uniformément aléatoirement un élément de $\mathbb{Z}_p^*$ et de calculer son image par ce morphisme (*i.e.* sa puissance q-ème).

Notations. Soit T un paramètre dont la valeur sera choisie pour optimiser l'efficacité de l'algorithme. Notons $p_1 = 2$, $p_2 = 3$, …, p_k les nombres premiers inférieurs à T.

Lemme 5. *La probabilité que $\alpha^r \cdot \gamma^s \cdot h \bmod p$ se factorise sous la forme (1) si r et s sont tirés uniformément aléatoirement dans $\mathbb{Z}_q$ et h est tiré uniformément aléatoirement dans $\mathbb{H}$ est égale à $\Psi(p-1, T)/(p-1)$.*

Démonstration. L'élément construit est un élément uniformément distribué de $\{1, \ldots, p-1\}$. De plus, il s'écrit sous la forme (1) s'il est T-friable. La probabilité recherchée est donc par définition $\Psi(p-1, T)/(p-1)$. □

Étant donné un triplet $(r, s, h) \in \mathbb{Z}_q^2 \times \mathbb{H}$, un algorithme pour vérifier si $\alpha^r \cdot \gamma^s \cdot h \bmod p$ est de la forme (1) consiste simplement à diviser cette valeur par les k nombres premiers $p_1, \ldots, p_k$ tant que c'est possible. À chaque division, la taille de l'entier divisé diminue et la complexité totale est bien de l'ordre de $O(k \log(p))$ opérations élémentaires.
L'algorithme complet pour générer les $(k+1)$ relations de la forme (1) requiert de construire des éléments aléatoires $\alpha^r \cdot \gamma^s \cdot h$ puis de tester qu'ils vérifient bien la relation (1) en les factorisant par divisions successives. Un couple fournit une relation avec probabilité $\Psi(p-1, T)/(p-1)$ et le coût moyen de la génération de $(k+1)$ relations est :

$$O\left(\frac{p-1}{\Psi(p-1, T)} \left(k(\log^3(p) + k \log(p))\right)\right).$$

Lemme 6. *Supposons connues* $(k+1)$ *relations de la forme* (1). *Il existe un vecteur* $(v_1, \cdots, v_{k+1})$ *tel que*

$$\begin{bmatrix} e_{1,1} & e_{2,1} & \dots & e_{k,1} & e_{k+1,1} \\ e_{1,2} & e_{2,2} & \dots & e_{k,2} & e_{k+1,2} \\ \vdots & & & \vdots & \vdots \\ e_{1,k} & e_{2,k} & \dots & e_{k,k} & e_{k+1,k} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \\ v_{k+1} \end{bmatrix} \in (q \cdot \mathbb{Z}_p)^k.$$

et tel que $\sum_{i=1}^{k+1} v_i s_i \equiv 0 \bmod q$ *avec probabilité* $1/q$.

Démonstration. L'existence du vecteur $\vec{v}$ est immédiate car la matrice a k lignes et $k+1$ colonnes.
Pour un ensemble de $k+1$ relations obtenues à partir des vecteurs $\vec{s^*}$, $\vec{r^*}$ dans $\mathbb{Z}_q^{k+1}$ et $\vec{h^*} \in \mathbb{H}^{k+1}$, le même ensemble de relations est obtenu pour un choix arbitraire de vecteur $\vec{s}$ dans $\mathbb{Z}_q^{k+1}$ en adaptant le vecteur $\vec{r} = \vec{r^*} - x\vec{s} + \vec{s^*}$ et en conservant $\vec{h^*} \in \mathbb{H}^{k+1}$.
La variable aléatoire $\vec{s}$ est donc indépendante de l'ensemble de relations obtenues et donc du vecteur $\vec{v}$. La probabilité que le produit scalaire $\vec{s} \cdot \vec{v}$ soit nul est donc la probabilité uniforme $\Pr[\vec{s} \cdot \vec{v} \equiv 0 \bmod q] = 1/q$. □

Nous obtenons un algorithme de résolution du problème du logarithme discret dans $\mathbb{G}$ de complexité

$$O\left(\frac{p-1}{\Psi(p-1,T)} k(\log^3(p) + k\log(p)) + k^3(\log(p))\right)$$

opérations élémentaires. L'algorithme consiste simplement à construire les $k+1$ relations comme indiqué dans les questions précédentes, puis à construire le vecteur $\vec{v}$ en résolvant un système linéaire. Lorsque ce vecteur $\vec{v}$ existe, nous obtenons le logarithme discret de α en base γ.
En effet, nous avons

$$\alpha^{r_i} \cdot \gamma^{s_i} \cdot h_i \equiv p_1^{e_{i,1}} \dots p_k^{e_{i,k}} \bmod p$$

pour tout $i \in \{1, \dots, k+1\}$. En élevant chacune de ces équations à la puissance v_i et en les multipliant, nous obtenons une relation de la forme

$$\alpha^r \cdot \gamma^s \cdot h \equiv p_1^{e_1 q} \dots p_k^{e_k q} \bmod p$$

avec $r = \vec{r} \cdot \vec{v}$, $s = \vec{s} \cdot \vec{v}$, $h \in \mathbb{H}$ et $e_1, \dots, e_k$ des entiers. Puisque $\mathbb{H}$ est l'image dans $\mathbb{Z}_p^*$ de l'élévation à la puissance q, nous obtenons l'existence d'un élément h' tel que $\alpha^r \gamma^s = h'$ et d'après la question 1, nous obtenons $\alpha^r \gamma^s = 1$. Si $s \neq 0$ (ce qui arrive avec probabilité $1 - 1/q$ d'après la question précédente), nous obtenons le logarithme discret $x \equiv -r/s \bmod q$.
Le coût total de l'algorithme est la somme du coût de la génération des relations (vu à la question 5) et de l'algèbre linéaire. En supposant que nous utilisons un algorithme cubique (comme une réduction de Gauss), nous obtenons le coût total suivant :

$$O\left(\frac{p-1}{\Psi(p-1,T)} k(\log^3(p) + k\log(p)) + k^3(\log(p))\right).$$

D'après l'estimation de la section précédente, si $T = L_p(1/2, \sqrt{2}/2 + o(1))$, nous avons

$$\Psi(p-1, T)/p \simeq L_p(1/2, \sqrt{2}/2 + o(1))^{-1}.$$

Par le théorème des nombres premiers, nous avons $k \simeq T \ln T$ et nous obtenons une complexité égale à

$$L_p(1/2, \sqrt{2}/2 + o(1)) \cdot L_p(1/2, \sqrt{2}/2 + o(1))^2 + L_p(1/2, \sqrt{2}/2 + o(1))^3$$

soit la complexité sous-exponentielle annoncée (les termes polynomiaux étant absorbés dans le $o(1)$) :

$$L_p(1/2, 3\sqrt{2}/2 + o(1)).$$

Algorithme 4 Méthode de Kraitchik - Calcul d'indice

Entrée: $\gamma, \alpha \in \mathbb{Z}_p^*$ et une base de facteur $\mathcal{B} = \{2, 3, 5, \ldots, p_k\}$
Sortie: $x \in \mathbb{Z}_q$ tel que $\alpha = \gamma^x \bmod p$ ou Échec.

```
i ← 0
L ← ∅
répéter
  i ← i + 1
  répéter
    r_i ←R Z_q ; s_i ←R Z_q
    g_i ←R Z_p^* ; h_i ←R g_i^q                    ▷ h_i uniforme dans H
    m_i ← α^{r_i} · γ^{s_i} · h_i mod p
  jusqu'à m_i est p_k-friable                      ▷ m_i = p_1^{e_{i,1}} ... p_k^{e_{i,k}} par divisions successives
  e_i ← (e_{i,1}, ..., e_{i,k})
  L ← L ∪ {(r_i, s_i, e_i)}
jusqu'à i = k + 1
Trouver v ∈ Z_q^{k+1} tel que v_1 e_1 + v_2 e_2 + ··· + v_{k+1} e_{k+1} ∈ (q · Z_p)^k.
                                                   ▷ par élimination gaussienne
r ← Σ_{i=1}^{k+1} v_i r_i mod q ; s ← Σ_{i=1}^{k+1} v_i s_i mod q
si s ≢ 0 mod q alors
  retourner −r/s mod q
sinon
  retourner Échec
fin si
```

La complexité de l'algorithme précédent peut être améliorée en utilisant un algorithme plus efficace pour tester le caractère friable des nombres générés (par exemple avec l'algorithme ρ de Pollard que nous verrons au chapitre suivant) et en utilisant des techniques d'algèbre linéaire spécifique (puisque la matrice construite à partir des relations obtenues est très creuse). Nous obtenons avec ces modifications, un algorithme de complexité $L_p(1/2, \sqrt{2} + o(1))$ tel qu'il a été formalisé en 1979 par L. M. Adleman. Il existe des variantes et des améliorations (heuristiques) de cet algorithme dans tous les corps finis.