Travaux Dirigés

Algorithme "Baby-Step Giant-Step"

Soit G un groupe, noté multiplicativement, et $g \in G$ un élément d'ordre n connu. Le but de cet exerice est de présenter et étudier un algorithme "générique" de calcul de logarithme discret en base g : l'algorithme "Baby-Step Giant-Step", inventé par D. Shanks en 1971.

On note h l'élément dont on veut calculer le logarithme discret.

- Calculer $d = \lfloor \sqrt{n} \rfloor$ (partie entière).
- Étape "pas de bébé" : pour tout $i \in \{0, \dots, d-1\}$ calculer $g^i \in G$ et stocker le couple (g^i, i).
- Étape "pas de géant" : pour j partant de 0,
  - calculer $h.(g^{-d})^j$
  - rechercher dans les valeurs stockés s'il existe un couple (g^i, i) tel que $g^i = h.(g^{-d})^j$:
    si oui, renvoyer $i + dj$,
    si non, incrémenter j.

Terminaison.

1. On suppose qu'il existe un entier $s \in \{0, 1, \dots, n-1\}$ tel que $g^s = h$. Montrer qu'il existe $a, b \in \mathbb{N}$ tels que $s = ad + b$ avec $b \leq d - 1$ et $a < n/d$.
2. Expliquer pourquoi la deuxième boucle ("pas de géant") est effectuée au plus n/d fois. Que peut-on dire si l'algorithme n'a rien renvoyé après n/d étapes ?

Analyse de complexité.

3. Quelle est la complexité en mémoire de cet algorithme ?
4. Expliquer comment réaliser la première étape en $O(\sqrt{n})$ opérations dans G.

L'algorithme tel qu'il est présenté ne précise pas, pour la dernière étape, comment rechercher un couple (g^i, i) qui conviendrait.

5. Utilisation de listes triées.
   (a) Expliquer à quelle condition cette recherche peut se faire en $O(\ln(\sqrt{n}))$ opérations. Quelle étape faut-il alors rajouter à l'algorithme ?
   (b) Montrer que la complexité temporelle globale est alors en $O(\sqrt{n}\ln(\sqrt{n}))$.

(Remarque : on peut en fait descendre à $O(\sqrt{n})$ en utilisant des tables de hachage.)

Challenge.

6. Soit $p = 400\,001\,201$. Utiliser cet algorithme pour calculer dans $(\mathbb{Z}/p\mathbb{Z})^*$ le logarithme discret en base $g = 4444$ de $h = 349$, sachant que l'ordre de g est $n = 1\,000\,003$.

Une approche "diviser pour régner" : l'algorithme de Pohlig-Hellman

Si l'ordre d'un groupe cyclique G n'est pas premier, il est possible de ramener le calcul de logarithmes discrets dans G à plusieurs calculs dans des sous-groupes de G, ce qui simplifie significativement la complexité du problème.

Cas $\mathbf{n = p^\alpha}$.

On considère un élément g d'un groupe G (noté multiplicativement) dont l'ordre est de la forme $n = p^\alpha$, où p est un nombre premier et α est un entier supérieur ou égal à 2. Soit $h = g^x$ un élément dont on veut calculer le logarithme discret $x \in \{0; \dots; p^\alpha - 1\}$ en base g.

1. On note $g' = g^{p^{\alpha-1}}$ et $G' = \langle g' \rangle$. Montrer que le cardinal du sous-groupe G' est p.
2. Justifier qu'il existe $x_0, x_1, \dots, x_{\alpha-1} \in \{0; \dots; p-1\}$ tels que $x = \sum_{k=0}^{\alpha-1} x_k \, p^k$.
3. Montrer que pour tout $0 \leq j \leq \alpha$, l'élément $\left(h.g^{-(x_0+x_1 p+\cdots+x_{j-1}p^{j-1})}\right)^{p^{\alpha-j-1}}$ appartient à G' et que son logarithme discret en base g' est congru à x_j mod p.
4. En déduire une méthode ramenant le calcul de logarithmes discrets en base g à celui de α logarithme discret dans G'.
   Si le coût d'une telle résolution est en $O(\sqrt{p})$, qu'a-t-on gagné par rapport à l'algorithme "baby-step giant-step" directement appliqué dans $\langle g \rangle$?

Cas général.

On considère maintenant, dans un groupe G, un élément g d'ordre $n = \prod_{i=1}^{r} p_i^{\alpha_i}$, où les p_i sont des nombres premiers distincts et $\alpha_i \geq 1$ pour tout $1 \leq i \leq r$. Soit $h = g^x$ un élément dont on veut calculer le logarithme discret x en base g.
Pour tout $1 \leq i \leq r$, on note $n_i = n/p_i^{\alpha_i}$.

5. Soit $i \in [\![1, r]\!]$; on note $g_i = g^{n_i}$ et $G_i = \langle g_i \rangle$. Montrer que le cardinal du sous-groupe G_i est $p_i^{\alpha_i}$.
6. Montrer que h^{n_i} appartient à G_i et que son logarithme discret en base g_i est congru à x modulo $p_i^{\alpha_i}$.
7. En déduire que le calcul du logarithme discret en base g se ramène au calcul d'un logarithme discret dans chacun des sous-groupes $\langle g_i \rangle$.
8. En appliquant la méthode précédente pour le calcul dans chacun des G_i, montrer que le coût total est en $O\left(\sum_{i=1}^{r} \alpha_i \sqrt{p_i}\right)$ opérations dans G.
9. Soit $p = \max_i p_i$ le plus grand facteur premier de n. Montrer que la complexité précédente est un $O(\sqrt{p}\ln(n))$.

Challenges.

10. Soit $g = 9 \in \mathbb{Z}/39\,367\mathbb{Z}$. Vérifier que g est d'ordre 3^9, et calculer le logarithme discret de $h = 9571$ en base g.
11. Soit $g = 29 \in \mathbb{Z}/120\,121\mathbb{Z}$. Vérifier que g est d'ordre 120120, et calculer le logarithme discret de $h = 48\,144$ en base g.

Le protocole Zero-Knowledge d'identification de Schnorr

Un protocole à divulgation nulle de connaissance (*zero-knowledge*) permet à un participant de démontrer qu'il connaît un certain secret, *sans dévoiler aucune information sur ce secret* (ce qui peut paraître paradoxal au premier abord !). Un tel protocole peut servir à des fins d'indentification ou d'authentification.

Soient g et h deux éléments d'un groupe G dans lequel le DLP est difficile. Le protocole de Schnorr permet à Pascal (le prouveur) de démontrer à Véronique (la vérificatrice) qu'il connaît le logarithme discret a de h en base g, sans le divulguer. On suppose connu l'ordre n, premier, de g.

— Pascal choisit en entier k au hasard entre 2 et $n-1$ de façon équiprobable, calcule un *engagement* (commitment) $c = g^k$, et l'envoie à Véronique.
— Véronique transmet en retour un *challenge* de son choix s_1 entre 0 et $n-1$.
— Pascal répond en envoyant $s_2 = k + as_1 \bmod n$, où a est le logarithme de h en base g.
— Véronique teste si $g^{s_2} = c.h^{s_1}$ ou pas.

Correction du protocole.

1. Montrer qu'on a bien $g^{s_2} = c.h^{s_1}$ si Pascal a suivi les consignes et connaît le logarithme discret a.

Véronique n'apprend rien.
On suppose que Pascal suit les consignes (et connaît a, évidemment).

2. Montrer que Véronique n'a aucune information sur k au moment où elle choisit s_1
3. En déduire que la valeur de s_2 ne lui apporte aucune information sur a.
4. Pourquoi est-il important que k soit bien aléatoire ?

Pascal ne peut pas tricher.
On suppose ici que Pascal *ne connaît pas* a, mais tente quand même de convaincre Véronique. Il n'est bien sûr pas tenu de respecter les consignes ; il doit cependant envoyer un engagement $c \in G$, qu'il choisit comme il veut.
On va montrer que si s_1 a été choisi équiprobablement, alors la probabilité que la réponse s_2 de Pascal passe la vérification de Véronique est inférieure ou égale à $1/n$.

5. Soient $s_1 \neq s_1'$ deux challenges potentiels de Véronique. Démontrer par l'absurde que Pascal n'est pas capable de fournir des réponses corrects s_2 et s_2' à ces deux challenges (sinon il connaîtrait a).
6. En déduire que Pascal ne peut fournir une réponse correcte qu'à au plus un des challenges possibles de Véronique.
7. Conclure.

Applications.

8. Expliquer comment ce protocole peut servir à identifier Pascal, en présence d'une autorité de confiance.