Rappels du cours précédent

El-Gamal, Problèmes calculatoires (LD, CDH, DDH), Attaque = Objectif (TB, OW, IND) + moyen (CCA,CPA), résultats de sécurité sur El-Gamal.

2.3 Les algorithmes de calcul de log discret

$G = \langle g \rangle$ cyclique d'ordre q

Problème du LD Étant donné $h \in G$, calculer x tq $g^x = h$.

Algo naïf :

- for $i = 0 \dots q$, tester $g^i == h$?
- Taille des entrées / sorties : $O(\log(q))$
- Complexité : $O(q)$, exponentiel en la taille des E/S

2.3.1 Pohlig-Hellman

$G = \langle g \rangle$ cyclique d'ordre q

Hypothèse : q n'est pas premier et on connait sa factorisation $q = \prod q_i$ avec q_i premiers entre eux.

Idées:

1. On va réduire le LD sur G à plusieurs LD sur des *petits* groupes d'ordre q_i.
2. Rappel : Définition : Ordre de $h \in G$, $ord(h) = min\{i > 0 | h^i = 1\}$.
3. g^{q/q_i} est d'ordre q_i (cf. racines de l'unité - Cours FFT):
   i. $(g^{q/q_i})^{q_i} = g^q = 1$
   ii. Si $0 < k < q_i$ alors $(g^{q/q_i})^k = g^{kq/q_i} \neq 1$ car $0 < kq/q_i < q_i$
4. Soit $x = LD(h, g)$, i.e. $h = g^x$. Alors
   i. Posons $g_i := g^{q/q_i}$ et $h_i := h^{q/q_i}$.
   ii. Alors $h_i = g_i^x$: en effet $g_i^x = (g^{q/q_i})^x = (g^x)^{q/q_i} = h^{q/q_i} = h_i$
   iii. Ce calcul de LD est plus simple car $ord(g_i) = q_i \ll ord(g) = q$.
   iv. Mais on n'apprend pas tout à fait x !
5. Quelle information apprend-on sur x ? On apprend $(x \bmod q_i)$.
   i. **Lemme:** Soit $q = ord(g)$ alors $g^x = g^y \Leftrightarrow x = y \bmod q$.
   ii. **Preuve:**
      - $\Leftarrow$ Comme $x = y \bmod q$ alors $\exists k \in \mathbb{Z}, x = y + kq$.
        Puis $g^q = 1$ donc $g^x = g^{y+kq} = g^x(g^q)^k = g^y$.
      - $\Rightarrow$ Supposons $y \geq x$.
        Écrivons la division euclidienne $(y - x) = kq + r$ avec $0 \leq r < q$.
        Notons que $g^x = g^y \Rightarrow 1 = (g^{-1})^x g^x = (g^{-1})^x g^y = g^{y-x}$
        Puis $1 = g^{y-x} = (g^q)^k g^r = g^r$.
        Or $r < q = ord(g) = min\{i > 0 | g^i = 1\}$ donc nécessairement $r = 0$ et $x = y \bmod q$.
6. Pour reconstruire $(x \bmod q)$ à partir des $(x \bmod q_i)$, nous allons utiliser le
   Théorème des restes chinois : L'application φ est une bijection

$$\varphi : \begin{array}{c} \mathbb{Z}/q\mathbb{Z} \to \mathbb{Z}/q_1\mathbb{Z} \times \cdots \times \mathbb{Z}/q_k\mathbb{Z} \\ x \bmod q \mapsto (x \bmod q_1, \dots, x \bmod q_k) \end{array}$$

   Preuve :

   i. φ est injective:
      Si $\varphi(x) = \varphi(y)$ alors $\varphi(x-y) = (0, \dots, 0)$ donc $q_i | (x-y)$.
      Comme les q_i sont premiers entre eux, alors $\prod q_i | (x-y)$ et $x = y \mod q$.
   ii. Les ensembles de départ et d'arrivée ont le même nombre d'éléments
   iii. Donc l'application est bijective.

Algorithme obtenu:

```
Algo LD-PH(h,g,[q_1,...,q_k])
# Où q := ord(g) = (\prod q_i) avec q_i premiers entre eux
1. Pour i=1..k
2.   g_i = g^{q/q_i}
3.   h_i = h^{q/q_i}
4.   x_i = LD(h_i,g_i)
5. x = ResteChinois(x_1,...,x_k)
```

Complexité:

- Ligne 2, 3 : exponentiation rapide en $O(\log q)$
- On boucle k fois :
  - $k \leq \log_2 q$ puisque $q = \prod_{i=1}^{k} q_i \geq \prod_{i=1}^{k} 2 = 2^k$.
  - donc coût cumulé $O(\log^2 q)$
- Ligne 4 : On va noter $\sum LD(q_i)$ les coûts à d'autres implémentations du LD.
- Ligne 5 : Polynomial en $\log q$ (admis).
- **Total:** $\sum LD(q_i) + Polynomial(\log q)$.

Exemples :

1. Si $q_1 \simeq q_2$ et algo LD naïf en ligne 4 $\Rightarrow$ $O(\sqrt{q})$
2. Si $q_1 \simeq q_2 \simeq q_3$ et algo LD naïf en ligne 4 $\Rightarrow$ $O(\sqrt[3]{q})$

Conséquence importante: Prendre q premier !

En pratique,

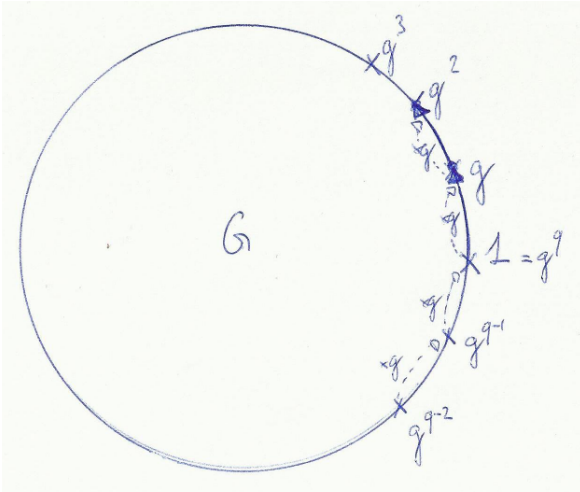
- si $G = (\mathbb{Z}/p\mathbb{Z})^*$ alors $q = p - 1$ est pair et peut avoir beaucoup de facteurs.
- Prendre plutôt p de la forme $p = 2p' + 1$ avec p et p' premiers (ici $q = q_1 q_2$ avec $q_1 = 2, q_2 = p'$).
- Puis travailler dans un sous-groupe de G d'ordre p' premier :
  - Si g générateur de G alors $ord(g) = p - 1 = 2p'$ et $ord(g^2) = p'$ (autrement dit $ord(g^{q/q_2}) = q_2$).
  - Donc on travaille dans $G' \subset G$ avec G' engendré par g^2.

2.3.1 Algo pas de bébé, pas de géant ou Baby Step Giant Step (BSGS) (Shanks 1971)

Valable pour tout groupe G, quelque soit son ordre q.

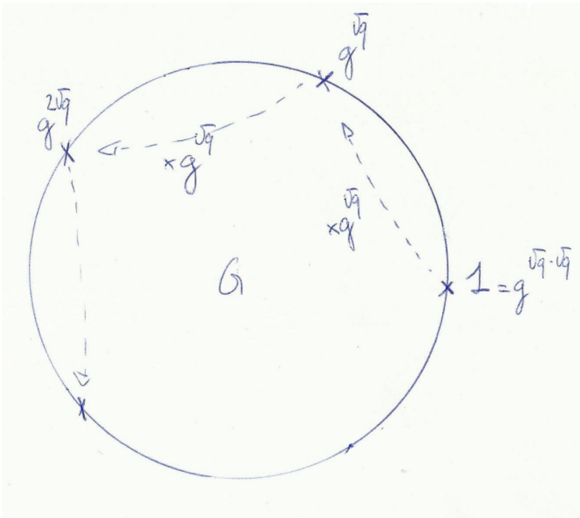
Idée:

1. Les puissances de g forment un cycle ($= G$) et h est dans le cycle.
   Mais calculer tous les points du cycle est trop coûteux : $\Omega(q)$ opérations dans G (cf. Algo LD naïf).



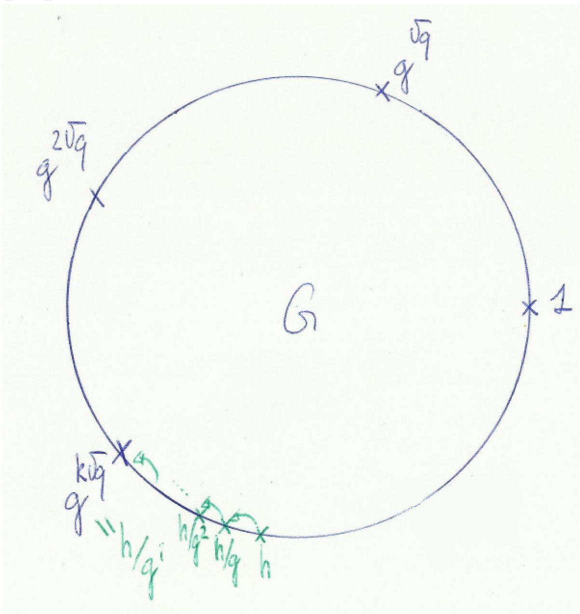
2. Au lieu de cela, on ne calcule que certains points

   i. **Pas de géant**: $t = \lfloor \sqrt{q} \rfloor$ et on stocke $S = \{g^0, g^t, g^{2t}, \ldots, g^{\lfloor q/t \rfloor t}\}$ les pas de géant.
   Ainsi h se trouve entre deux pas de géants g^{kt} et $g^{(k+1)t}$.



   ii. **Pas de bébé**: On teste si h est dans S, puis si h/g est dans S, puis h/g^2 ...

   - Si on a une collision alors $h/g^i = g^{kt}$ et donc $h = g^{kt+i}$.
   - On trouvera toujours un collision :
     Écrivons la division euclidienne $x = x_1 t + x_0$ avec $0 \leq x_0 < t$.
     Alors $h = g^x = g^{x_1 t + x_0}$ donc $h/g^{x_0} = g^{x_1 t}$.
     Notons que $x_1 = \lfloor x/t \rfloor \leq \lfloor q/t \rfloor$.



Algorithme obtenu:

```
Algo LD-BSGS(h,g,q)
# Où q := ord(g)
S = new Dictionnaire();
t = floor(sqrt(q))
gg = g^t
gs = 1
Pour k=0..floor(q/t)
  S.add(gs,k)
  gs = gs * gg
gs = h
Pour i=0..t
 if(S.contains(gs))
   return S[gs] * t + i
 gs = gs / g
```

Complexité:

1. g^t en $O(\log\sqrt{q}) = O(\log q)$ multiplications dans G
2. $g^{2t}, \ldots, g^{\lfloor q/t \rfloor t}$ en $O(\sqrt{q})$ multiplications dans G
3. Chaque nouveau h/g^i coûte $O(1)$ et test d'appartenance $h/g^i \in S$? en $O(1)$ avec des tables de hachage. Donc $O(\sqrt{q})$ opérations pour tous les tests
4. Au final, $O(\sqrt{q})$ opérations arithmétiques dans G et mémoire $O(\sqrt{q})$.

Bonus

$\mathbb{Z}/17\mathbb{Z}^*$ d'ordre 16 avec le générateur $\xi = 3$.

Quel est l'ordre de $\xi^2, \xi^3, \xi^4, \xi^5, \xi^{-1}$?