Exercice I: Algorithme de Cipolla

Soit p un nombre premier impair. On considère le corps fini $\mathbb{F}_p$ et l'on note $\mathbb{F}_p^2$ l'ensemble de ses carrés. Soit $a \in \mathbb{F}_p$, on souhaite trouver une racine carrée de a en s'aidant de l'extension $\mathbb{F}_{p^2}$.

1) Soit u un élément de $\mathbb{F}_p$ tel que $u^2 - a$ n'est pas un carré modulo p. On note v une racine carrée de $u^2 - a$ dans $\mathbb{F}_{p^2}$. Montrer que l'on a:

$$(u+v)^{p+1} = (u+v)(u^p + v^p) = (u+v)(u-v) = u^2 - v^2$$

En déduire que l'élémént $w = (u+v)^{\frac{p+1}{2}}$ est une racine carrée de a dans $\mathbb{F}_{p^2}$. (w est dans $\mathbb{F}_p$ si et seulement si a est un carré dans $\mathbb{F}_p$.) On en déduit l'algorithme de Cipolla:

- Trouver un u dans $\mathbb{F}_p$ tel que $u^2 - a$ ne soit pas dans $\mathbb{F}_p^2$.
- Calculer par puissance rapide $(u+v)^{\frac{p+1}{2}}$ dans $\mathbb{F}_{p^2} \simeq \mathbb{F}_p[v]/(v^2 - (u^2 - a)))$

2) On considère le premier nombre premier p supérieur à 10^{50}.

a) Montrer que 6 est un carré dans $\mathbb{F}_p$ par un calcul de puissance rapide.

b) Trouvez une racine carrée de 6 en utilisant `powmod` (sous sage: `power_mod`) et vérifiez votre réponse.

3) Documentez vous sur la syntaxe d'un corps fini non premier dans votre logiciel. (Création, choix des noms par défaut ou imposé, choix du polynôme par défaut ou imposé ....

a) (Il est souvent plus prudent d'essayer de libérer les noms de variables que l'on souhaite utiliser) Créez le corps $K = \mathbb{F}_{3^4}$. Si vous le créez avec un polynôme irréductible de $\mathbb{F}_3[x]$ quelconque et que l'on note t la classe de x, que peut t'il arriver à l'ordre multipicatif de t? Créez maintenant K en utilisant la variable t et un choix automatique du polynôme.

4) $t+1$ est t'il un carré dans K? Si oui, trouvez en une racine carrée en remarquant que la méthode reste valable dans un corps non premier. (On fera tout de même des essais pour voir si powmod a implémenté le cas des corps non premier)

5) Quel est l'inverse de $t+1$? Vérifiez/expliquez ce résultat avec des fonctions arithmétiques de $\mathbb{F}_3[x]$.

Exercice II: Méthode de Berlekamp (cf. Pb de rang, corps finis, polynômes)

1) Créer une fonction `randP(n)` qui donne un polynôme unitaire au hasard de degré n à coefficients inferieurs à 20.

2) On considère le polynôme P obtenu en développant un produit de 5 éléments de type `randP(rand(7))`

3) Vérifier que P n'a que des racines simples dans $\mathbb{C}$, sinon recommencer.

4) On considère un nombre premier p.

a) Créer une fonction `berl(p,P)` qui calcule la matrice (de Berlekamp) `F` de l'application $\mathbb{F}_p/(P) \to \mathbb{F}_p[x]/(P)$ $f \mapsto f^p - f$ dans la base $(x^i)_{0,\ldots,\deg P-1}$. On utilisera une fonction explicite d'xcas/sage pour les puissances rapides.

b) Tester pour différentes valeurs de p si votre polynôme P est dans facteurs multiples dans $\mathbb{F}_p[x]$, et si c'est le cas, calculez le rang de F. Illustrer/Conjecturer un lien entre la dimension du noyau de F, et la factorisation de P sans $\mathbb{F}_p[x]$.

5) Corps finis non premier.

a) Dans xcas ou sage, quel nom faut il chercher pour construire un corps fini non premier? Par exemple comment construire et travailler avec des éléments de $\mathbb{F}_{27}$? On notera a la classe de x dans ce corps de rupture. Quel est l'ordre multiplicatif du a lorsque le logiciel choisit lui même le polynome irréductible? (On pourra lire dans l'aide: `Menu Aide>Algorithmes` la partie sur la représentation des corps non premiers.

b) Trouver (on peut piocher au hasard) un polynôme irreductible de degré 6 dans $\mathbb{F}_3[x]$. Faites le factoriser dans $\mathbb{F}_{27}[x]$ (dans certaines version d'xcas il vaut mieux remonter le polynôme dans $\mathbb{Z}$ $[x]$ avant de l'envoyer dans $\mathbb{F}_{27}[x]$). A quelle condition a t'il une racine dans $\mathbb{F}_{3^n}$?

6) Application pour Factoriser.

a) Choisir un nombre premier impair de préférence[2] plus petit que 20 où le nombre de facteurs de P dans $\mathbb{Z}$ $[x]$ est assez petit, et tel que P ait peu de facteurs dans $\mathbb{F}_p[x]$, et soit encore sans facteurs

[1] `https://webusers.imj-prg.fr/~frederic.han/agreg/OptionC/`

[2] On le prend petit pour éventuellement illustrer la remontée Henselienne ensuite.

multiples dans $\mathbb{F}_p[x]$ pour cette valeur de p. Le nombre de facteurs est il forcément le nombre de facteurs dans $\mathbb{Z}\,[x]$? (illustrez)

b) Choisissez au hasard un élément Q du noyau de F pour la valeur de p choisie. Vérifier qu'il convient, et trouver un facteur non trivial de P en remaquant que l'un des 3 pgcd suivant est non trivial dans $\mathbb{F}_p[x]$: $Q \wedge P$, $Q^{\frac{p-1}{2}} - 1 \wedge P$, $Q^{\frac{p-1}{2}} + 1 \wedge P$.

c) En déduire une méthode pour factoriser un polynôme réduit sur un corps fini.

Exercice III: (Mini texte[3]). Mots clefs: Puissances rapide, Cryptographie

Si p est un nombre premier, il peut être parfois facile ou parfois difficile de factoriser $p-1$ bien que les nombre premier considérés aient des tailles similaires.

Si p est un nombre premier, on sait qu'il existe un élément g tel que: l'application suivante soit bijective:

$$\begin{array}{rccc} \phi: & (\mathbb{Z}\,/(p-1)\mathbb{Z}\,,+) & \to & (\mathbb{Z}\,/p\mathbb{Z}\,,.)^\times \\ & a & \mapsto & g^a \end{array}$$

On appelle alors log discret la bijection réciproque. Grâce aux puissances rapides, on dispose de bons algorithmes pour calculer l'image d'un élément par ϕ. En revanche, selon les valeur de p, calculer un log discret peut être très difficile. Par exemple voici un extrait de la documentation un peu ancienne (2008?) du logiciel `pari` qui est spécialisé en arithmétique.

```
znlog(x,g):
g must be a primitive root mod a prime p, and the result is the discrete log of
x in the multiplicative group (Z/pZ)^*. This function uses a simple-minded
combination of Pohlig-Hellman algorithm and Shanks baby-step/giant-step which
requires O(sqrt{q}) storage, where q is the largest prime factor of p-1. Hence
it cannot be used when the largest prime divisor of p-1 is greater than about
10^{13}.
The library syntax is znlog(x,g).
```

(Indication hors énoncé, utilisation de pari) *Xcas et Sage peuvent utiliser les instructions de pari. Dans xcas la documentation de pari est disponible hors ligne via le menu: Aide/Manuels/Pari-GP. Comparez la documentation actuelle à celle ci dessus*

Grâce a des tests de primalité efficaces, et aux puissances rapides, on peut considérer maintenant que l'on dispose d'un nombre premier p, tel que le problème du log discret soit difficile. Nous allons en déduire un cryptosystème à clef publique.

Pour recevoir un message secret, on choisit donc un entier b et un secret e et l'on rend publique le couple $(\alpha, \beta) = (b, b^e[p])$ ainsi que p. On s'arrangera tout de même pour que e ne soit pas évidemment déductible de $b^e[p]$ (comment?). Puisque les lettres ont des codes Ascii, on peut les représenter par leur code.

La personne qui souhaite vous envoyer un message va donc coder chaque lettre de la manière suivante: Soit x le code ascii de la lettre à crypter, il vous envoie: $(\gamma, \delta) = (\alpha^k[p], x.\beta^k[p])$ où k est un entier qu'il pioche au hasard et garde secret. Puisque vous connaissez e, vous pourrez retrouver x à partir de (γ, δ). En revanche sans e il faudrait résoudre un problème de log discret. De plus, une même lettre ne sera pas forcément codée de la même manière, un observateur ne peut donc pas faire de statistiques sur les lettres afin de les deviner.

1) On pourra démontrer certaines affirmations du texte.

2) Si p est premier, quelle est la probabilité de trouver un générateur de $((\mathbb{Z}\,/p\mathbb{Z}\,)^\times, \cdot\,)$ en piochant au hasard? On pourra la calculer sur quelques exemples.

3) On pourrait regarder si c'est facile ou pas de trouver un nombre premier qui soit difficile pour le log discret. Expliquer/illustrer aussi quand c'est facile.

4) On pourrait expliquer comment décoder le message envoyé, et pourquoi c'est effectivement réalisable.

5) On pourrait mettre en oeuvre ce cryptosystème ou certaines étapes.

6) Dans xcas sous linux on peut acceder aux commandes de pari, par exemple: `pari();` puis `g:=pari_znprimroot(101);` La documentation est aussi accessible dans le menu d'aide Manuels. Dans sage on fait `pari.znprimroot` (attention dans sage pari n'est pas pareil que gp. pari(1) est bien plus rapide que gp(1))

[3] préparez un exposé de 15-20min