Blockchain appliquée à un processus électoral — compte-rendu (rendu intermédiaire)

Z. Aloui, Z. Hua — lu2in006

1 Introduction

L'objectif de ce projet est de mettre au point un système de vote basé sur une blockchain.

Le code comporte à ce stade trois fichiers principaux : `primalite.(h|c)`, `rsa.(h|c)` et `voting.(h|c)`. `primalite` et `rsa` comportent les réponses à la plupart des questions jusqu'au début de l'exercice 3. `voting` comporte les fonctions liées spécifiquement à l'élection. Ce fichier sera probablement scindé en plusieurs parties d'ici à la soumission du projet final...

À ces fichiers s'ajoutent un fichier de test (`tests.c`) et un fichier définissant une table de hachage (`hashset.(h|c)`) utilisée dans l'exercice 4. (La table de hachage était d'abord censée n'être qu'un ensemble, ce qui epxlique que le fichier et peut-être encore certaines variables et fonctions soient nommées `hashset`. Sera également corrigé d'ici au rendu final...) Le fichier de test comporte plusieurs fonctions de tests, dont certaines sont appelées depuis le `main`.

Nous avons créé un constructeur par copie pour chaque structure. Tout dans notre programme se fait avec des copies : nous avons essayé de minimiser les partages de pointeurs non-constants d'une fonction à une autre, pour simplifier la gestion de la mémoire.

2 Exercice 1

2.1 Question 2

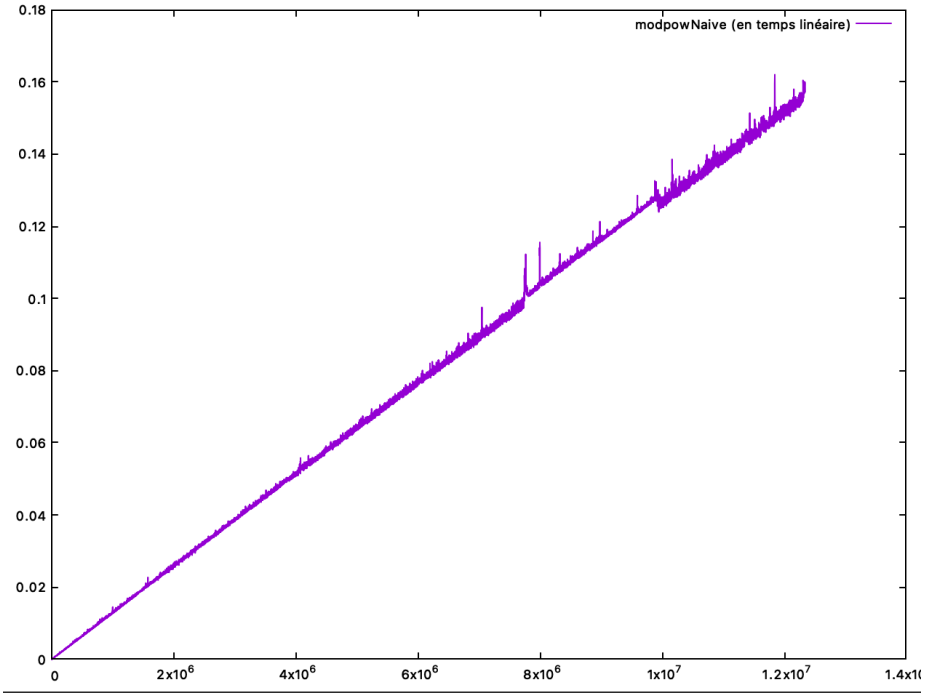
Le plus grand nombre premier que nous avons pu calculer en moins de 2 secondes est 239.317.339, calculé en 1.964 secondes.

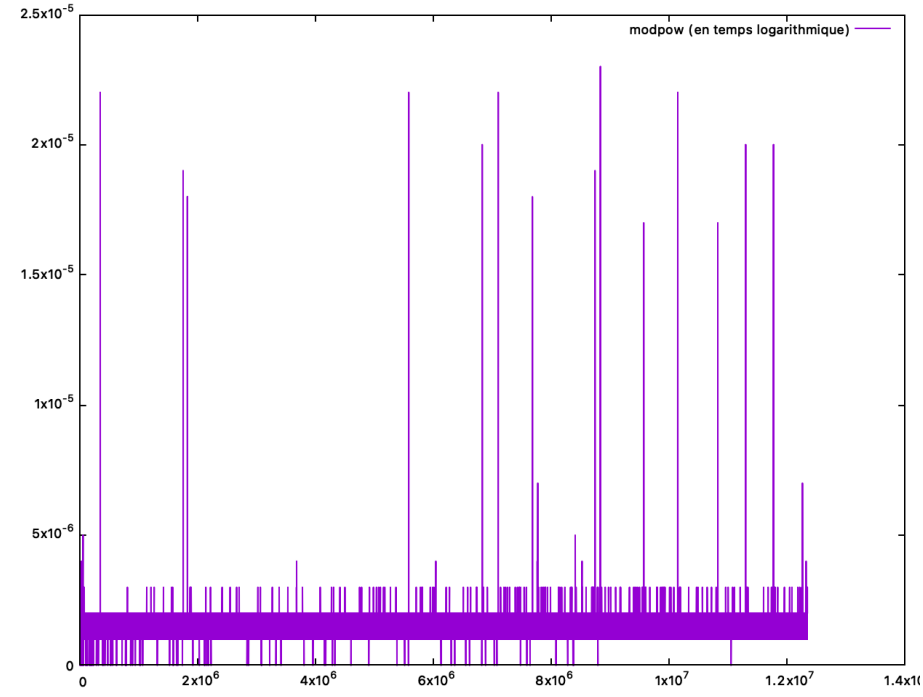
2.2 Question 3

L'algorithme naïf tourne en temps linéaire par rapport à l'exposant.

2.3 Question 5

Nous avons effectué 10.000 tests avec comme base 12, comme résidu 51 et un exposant de $1234i$, pour i variant de 1 à 10.000. Les graphes ci-dessous montrent l'esposant en abscisse et le temps d'exécution en ordonnée. La différence était trop grande pour pouvoir afficher les deux courbes sur le même graphe.





2.4 Question 7

On modélise les k tirages faits par un appel de l'algorithme par une suite de k éléments à valeurs dans $\{T, T^c\}$, où T indique que le nombre tiré est un témoin et T^c indique qu'il ne l'est pas. Si p n'est pas premier, la probabilité des témoins est de $\frac{3}{4}$. On cherche à calculer la probabilité de ne trouver aucun témoin après k expériences.

Les évènements "le $i-$ème tirage donne T^c" sont indépendants, donc $\mathbb{P}\left(\bigcap_{i=1}^{k} \textit{ le i-ème tirage donne } T^c\right) = \frac{1}{4^k}$.

La probabilité d'avoir un faux positif est donc inférieure $\left(\frac{1}{4}\right)^k$.

3 Exercice 3

La structure `Key` est définie dans le fichier `rsa.h`. `protected` et `signature` sont définies dans `$voting.h`.

4 Exercice 4

Pour le tirage aléatoire des nc candidats, on commence par tirer nc entiers distincts entre 1 et nv. On génère alors les clés des candidats, que l'on stocke dans une table de hachage. Cela permet égalemement de se souvenir des indices qui ont déjà été tirés et donc de garantir qu'il n'y a pas d'électeur tiré deux fois pour être candidat.

Une fois les nc candidats choisis et leurs clés générées, on itère sur $[\![0, nv-1]\!]$, on crée les clés de chaque électeur non-candidat, on choisit la personne pour qui il va voter, on génère la déclaration de vote et on remplit les fichiers qui sont à remplir.