Compte-Rendu Final

Dans ce projet on se propose de réaliser un système de vote électoral avec la possibilité pour chaque participant de déclarer sa candidature et/ou donner sa voix à un candidat.

Dans un premier temps le programme n'est pas encore partitionné (faute de temps),Nous avons utilisé tout au long de notre programme des structures décrites dans ce tableau :

Structure	Description

`typedef struct key{`
`  long val;`
`  long n;`
`}Key;`
	on crée la structure key avec un long val et un long n

`typedef struct signature{`
`  long *mess;`
`  int taille;`
`}Signature;`
	on crée la structure signature qui un tableau de long et dont on connaît sa taille

`typedef struct protected{`
`  Key* k;`
`  Signature* s;`
`  char*mess;`
`} Protected;`
	la structure protected qui contient la clé publique de l'émetteur (l'électeur), son message (sa déclaration de vote), et la signature associée

`typedef struct cellKey {`
`  Key * data;`
`  struct cellKey* next;`
`} CellKey;`
	La structure Cellkey nous permet de faire une liste chainé de Key

`typedef struct cellProtected{`
`  Protected* data;`
`  struct cellProtected* next;`
`} CellProtected;`
	La structure Cellprotected nous permet de faire une liste chainé de déclaration

typedef struct hashcell{
  Key* key;
  int val;
} HashCell;
	la structure hashcell nous permet de stocker une clé et une valeure associée afin de la placer dans une table de hachage

typedef struct hashtable{ HashCell** tab; int size; } HashTable;	Une table de hachage de HashCell, on gère les collisions par probing linéaire
typedef struct block { Key * author ; CellProtected *votes ; unsigned char *hash ; unsigned char *previous_hash; int nonce ; } Block ;	Block contenant dix déclarations de vote, la clé de l'assesseur, auteur de ce bloc, la valeur hachée associée au bloc, la valeur hachée du bloc précédent, et nonce la preuve de travail
typedef struct block_tree_cell { Block * block ; struct block_tree_cell * father ; struct block_tree_cell * firstChild ; struct block_tree_cell * nextBro ; int height ; } CellTree ;	Structure d'arbre binaire avec un nombre arbitraire de fils, les fils sont représentés par une liste chaînée (nextBro), on peut également accéder au père de chaque nœud et on connaît la hauteur de chaque nœud

Les fonctions principales de notre projet nous avons generate_random_data, qui nous permet de générer aléatoirement des électeurs, des candidats et des déclarations de votes, qui seront stockés dans des fichiers pour pouvoir les manipuler par la suite. nous avons également compute_winner qui à partir d'une liste chaînée de clés d'électeurs, de clés de candidats et de déclarations de votes, permet de déterminer le gagnant dans une complexité correcte à l'aide des tables de hachages qui nous permettent de faire des recherches en O(1). par la suite nous avons toutes nos fonctions de gestion de la blockchain et compute_winner_BT, qui nous permet d'obtenir un système de gestion décentralisé où tout le monde peut vérifier les résultats de l'élection

1.1

Implémentez la fonction `int_is_prime_naive(long p)` *qui, étant donné un entier impair p,renvoie 1 si p est premier et 0 sinon. Quelle est sa complexité en fonction de p ?*

La complexité de `int_is_prime_naive(long p)` en fonction de p est en O(p)

1.2

Quel est le plus grand nombre premier que vous arrivez à tester en moins de 2 millièmes de seconde avec cette fonction ?

Le plus grand nombre premier que l'on a testé en moins de 2 millième de seconde avec int_is_prime_naive est 188935

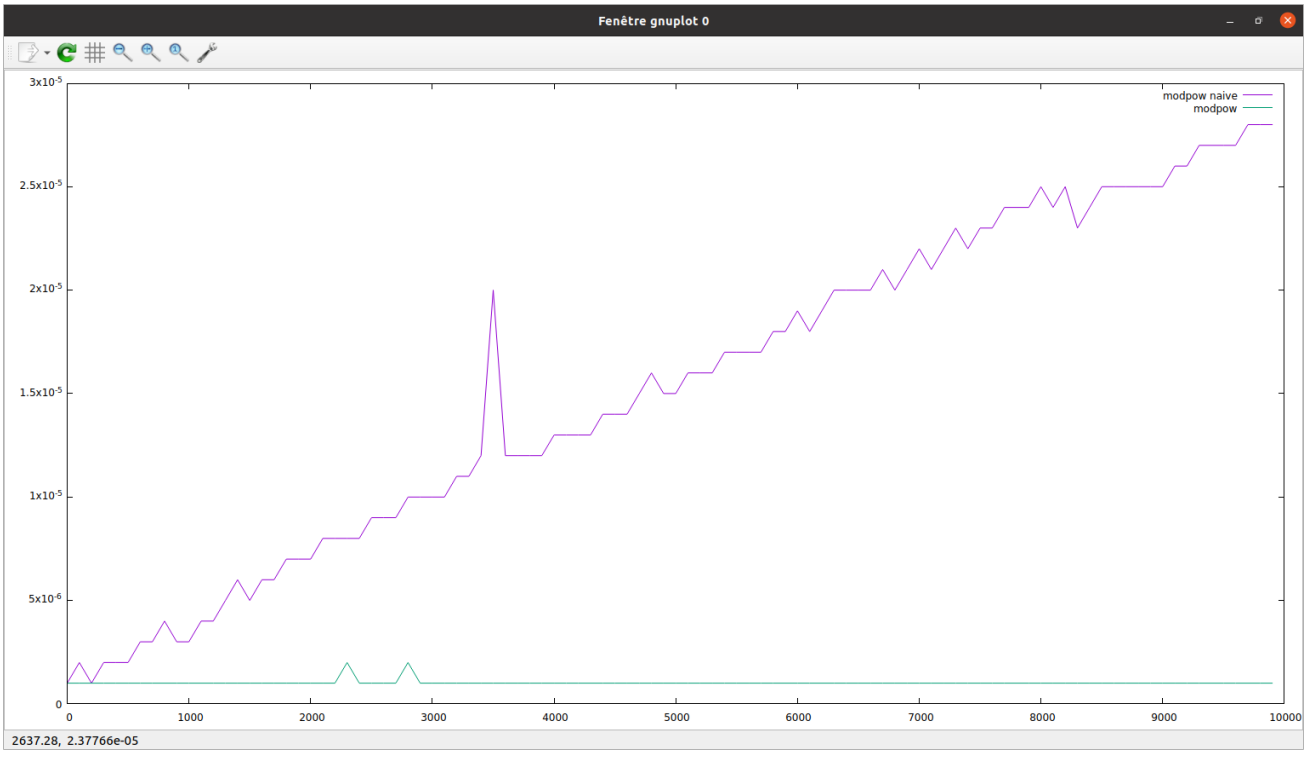
1.3

Implémenter la fonction `long modpow_naive(long a,long m, long n)` *qui prend en entrée trois entiers a, m et n, et qui retourne la valeur ab mod n par la méthode naïve. Quelle est sa complexité?*

La complexité de modpow_naive est en O(m)

1.5

comparez les performances des deux méthodes d'exponentiation modulaire en traçant des courbes de temps en fonction de m. Qu'observez-vous ?



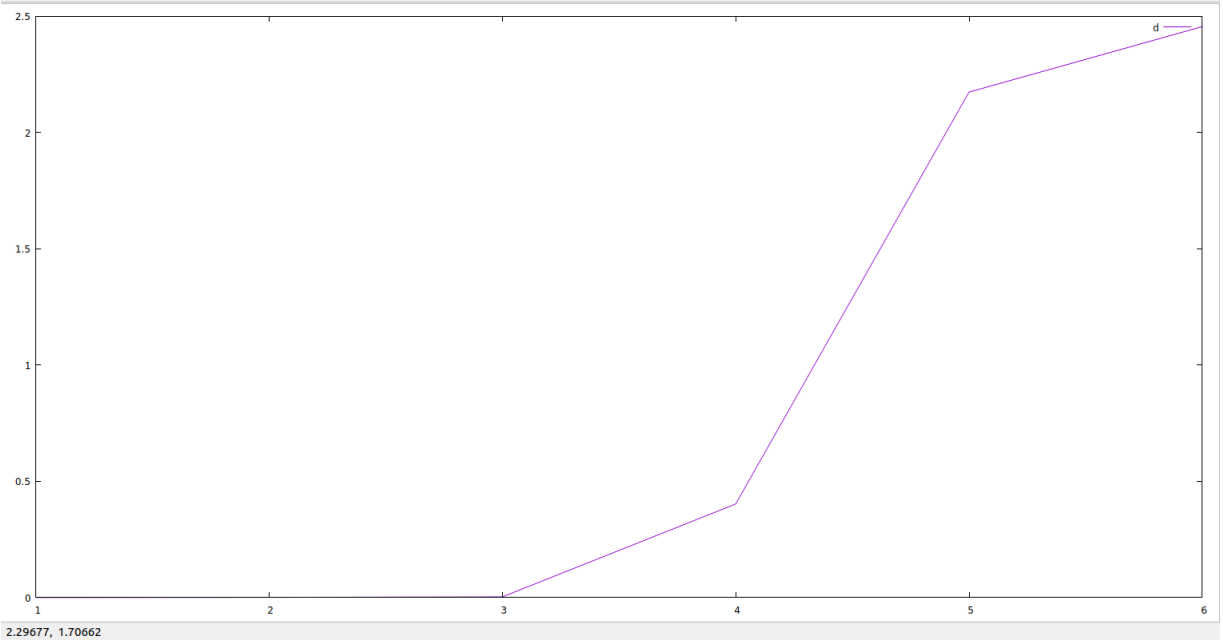
Titre :schéma des courbes traçant les performances des deux m´ethodes d'exponentiation modulaire en fonction de m

Dans ce schéma on constate que le temps de calcul de `modpow_naive` augmente quand m augmente on passe de a² à un temps de calcul proche de 0 seconde alors que a^{10000} on plus vers $2,7 \times 10^{-5}$ secondes.
Tandis que pour modpow on a quelque chose de très régulier a²= a^{10000} = proche de 0 sec .

1.7
En utilisant le fait que, pour tout entier p non premier quelconque, au moins ¾ des valeurs entre 2 et p − 1 sont des témoins de Miller pour p, donner une borne supérieure sur la probabilité d'erreur de l'algorithme.

R: 0.25^k car 0 succès sur k tentatives indépendantes et aléatoires avec p =¾

7.8
Étudiez le temps moyen de la fonction compute proof of work selon la valeur de d. Tracez une courbe du temps moyen en fonction de d, et en déduire la valeur de d à partir de laquelle ce temps dépasse une seconde.



la valeur de d à partir de laquelle ce temps dépasse une seconde est 3

8.8

Écrivez une fonction permettant de fusionner deux listes chaînées de déclarations signées. Quelle est la complexité de votre fonction ? Quelle modification dans la structure faudrait-il faire pour avoir une fusion en O(1) ? On ne vous demande pas d'implémenter cette modification.

pour une chaine debut de longueur L, on a une complexité de O(L), afin d'obtenir une complexité en O(1) nous devrions avoir une liste chaînée circulaire doublement chaînée afin d'arriver directement au dernier élément de la liste

9.7

Conclusion du projet : que pensez-vous de l'utilisation d'une blockchain dans le cadre d'un processus de vote ? Est-ce que le consensus consistant à faire confiance à la plus longue chaîne permet d'éviter toutes les fraudes ?

Les avantages d'un système comme tel semblent nombreux mais malheureusement dans le cas où la chaîne n'est pas très longue nous pouvons avoir des soucis de fiabilité, mais au delà des soucis liés directement au code nous devons se poser des questions sur les différentes attaques de social engineering que des personnes malveillantes pourraient exploiter

Instructions pour tester notre code

- cd projet
- make
- ./exec

Notre fonction main contient des affichages pour toutes les parties du projet, des fonctions de protocole RSA, aux fonctions de stockages de structures et tables de hachages et finalement la blockchain