L3 2015-2016 **Projet Numérique** **Université de Cergy-Pontoise**

Logiciel : Python, Sage, Maple ou Xcas.
Difficulté : 5/5

Vote électronique

L'objet de ce projet est de travailler à l'aide d'un logiciel adapté sur des calculs sur les grands nombres, et de montrer une des applications du chiffrement à clé publique.
Il faut dans un premier temps réviser l'arithmétique de Terminale S, et notamment la notion de congruence.
Si $a \equiv b\ (n)$ et $c \equiv d\ (n)$ alors $ac \equiv bd\ (n)$.

1. Programmer l'algorithme du RSA dans votre logiciel. (voir par exemple le livre de Stinson). L'objet de ce projet n'est pas le chiffrement RSA, donc on peut « recopier » sans états d'âmes.
2. (a) Ecrire une fonction h_k ($k \in \mathbb{N}^*$) qui, étant donné deux nombres entiers renvoie un entier dont l'écriture binaire a exactement k chiffres non nuls communs avec les deux entiers de départ. Si ce n'est pas possible, la fonction doit renvoyer 0 (ou *False*).
   (b) Si les entiers précédents sont pris au hasard parmi les nombres inférieurs à 10^{600}, et que $k = 30$, quelle est la probabilité que la fonction ne réponde pas 0 ?
   (c) Améliorer la fonction précédente pour que les k chiffres soit pris au hasard parmi tous les chiffres communs.
   Cette fonction va nous permettre de fournir des certificats de vote conforme.
3. On appelle C l'autorité principale qui va délivrer les certificats. B sera la personne qui va récupérer les votes, et A_1, ... A_a seront les votants ($a \in \mathbb{N}^*$ est le nombre de votants).
   Dans un premier temps, C va se doter d'une clé publique n produit de deux nombres premiers secrets p et q. C fabrique également les nombres c et d servant à chiffrer et à déchiffrer : $cd \equiv 1 \bmod (p-1)(q-1)$. Seul C peut décrypter. On pose $f\colon x \mapsto x^c \bmod n$ la fonction de chiffrement de $[0; n[$ dans $[0; n[$. La fonction de déchiffrement est $f^{-1}\colon x \mapsto x^d \bmod n$. On suppose que le vote est « oui » ou « non ».
   Dans la suite on suppose connus n et c (donc f). On suppose également que k a été choisi.
   (a) Construire une fonction qui renvoie a nombres aléatoires v_1, v_2, $\dots v_a$ tels que $f(v_i)$ et $f(2v_i \bmod n)$ ont au moins k chiffres binaires communs pour tout i.
   (b) C envoie à chaque votant A_i le nombre v_i, et il envoie à B la liste des certificats $h_k(f(v_i), f(2v_i \bmod n))$. Les votants choisissent de voter oui ou non en envoyant respectivement à B, soit $f(v_i)$ (vote oui), soit $f(2v_i \bmod n)$. On appelle e_i ce nombre.
   B vérifie pour chaque votant que le vote est conforme, à l'aide du certificat (faire une fonction pour la vérification).
   Puis B calcule $e = \prod_{i=1}^{a} e_i \bmod n$ (on suppose que tout le monde a voté) et envoie e à A.
   (c) A calcule $f^{-1}(e) \times f^{-1}(\prod_{i=1}^{a} v_i)$ ce qui doit donner une puissance de 2 qui correspond au nombre de personnes ayant voté « non ». Expliquer pourquoi on doit avoir $2^a < n$ pour que ce vote soit valable.
4. Simuler un exemple de vote avec 5 votants.
5. Améliorer la procédure pour tenir compte des personnes qui s'abstiennent ou votent nul.
6. Améliorer la procédure pour des votes avec plus de choix (3, 4, 5 candidats ...).
   On pourra faire une fonction qui en fonction du nombre de votants et du nombre de choix donne la taille de la clé publique n.