TP3: RSA

Exercice 1: Chiffrement et déchiffrement suivant l'algorithme RSA

1. Programmez deux fonctions en Python (E et D) l'une permettant de chiffrer un message avec la méthode RSA l'autre permettant de déchiffrer. Pour le chiffrement E la fonction aura pour entrée un message “en clair” m ($m \in \mathbb{Z}/N\mathbb{Z}$) et une clé publique $k = (N, e)$ et la sortie sera le message chiffré c ($c \in \mathbb{Z}/N\mathbb{Z}$). Pour le déchiffrement D les entrées seront un message chiffré c et une clé privée $k' = (N, d)$ et la sortie sera bien sûr le message déchiffré. Pour cette question vous pouvez utiliser les algorithmes des TP précédents.

2. Tests des algorithmes de chiffrement et de déchiffrement RSA:

   (a) on considère la clé publique $k = (8633, 17)$ et le message “en clair” $m = 1111$.

      i. Quelle est la longueur en nombre de bits des mots que l'on peut chiffrer avec cette clé publique?

      ii. Utilisez votre fonction E pour trouver le message chiffré c .

      iii. Vérifiez que la clé k est bien construite et déduisez-en la clé privée k' correspondant à k .

      iv. Vérifiez que votre fonction $D(c, k')$ permet de retrouver m .

   (b) on considère la clé privée $k' = (6557, 67)$ et le message chiffré $c = 1234$. Retrouvez m .

Exercice 2:

1. Programmez une fonction permettant de savoir si un nombre entier n est un nombre premier ou un nombre pseudo-premier de base a (cette fonction renvoie 1 si n est premier ou pseudo-premier de base a et renvoie 0 sinon).

2. On considère qu'avec la fonction précédente pour $a = 2$ et pour les grandes valeurs aléatoires de n ($\beta \geq 500$) on obtient presque sûrement un nombre premier n lorsque l'algorithme renvoie 1. En utilisant la fonction **randint(min,max)** (il faut d'abord charger la bibliothèque **random: from random import ***) qui renvoie un nombre entier aléatoire entre **min** et **max**, et en utilisant aussi la fonction de la question 1 pour $a = 2$, programmez une fonction qui génère un nombre presque sûrement premier avec un nombre de bits fixés (le nombre de bits β sera donc la variable d'entrée de la fonction).

3. Pour $\beta = 1000$ estimez le nombre d'entiers que doit tester votre fonction avant de trouver un nombre “premier” de 1000 bits.

4. Comment modifier votre algorithme pour augmenter sa fiabilité (c'est à dire augmenter la probabilité que l'entier obtenu soit premier) ?

Exercice 3: création de clés pour RSA

1. Dans l'exercice précédent, nous avons créé une fonction permettant de trouver des nombres premiers ayant un nombre β de bits fixés. En utilisant cette fonction, programmez une autre fonction permettant de générer une clé publique $k = (N, e)$ et une clé privée de la forme $k' = (N, d)$. Ces deux clés devront permettre ensuite de chiffer et de déchiffrer avec l'algorithme RSA. Cette fonction aura pour entrée le nombre β de bits pour l'encodage de N.

2. Testez les programmes (création de clés et chiffrement) pour $\beta = 1000$ et pour $m = 111222333444555666777888999$. Conclusion?