Université Claude Bernard Lyon 1
Institut de Science financière et d'Assurances (ISFA)

50 avenue Tony Garnier
69007 Lyon, France

Master 1 informatique
Université Claude Bernard Lyon 1

Cryptologie : TP n° 1

1 GPG

1. Installez GnuPG sur votre machine. Il existe une version pour chaque système d'exploitation.
   Ce logiciel libre est disponible ici : `https://www.gnupg.org/index.html`.
2. Explorez la documentation.
3. Explorez les commandes de `gpg` avec l'option `-h`.
4. Générez une paire de clés de taille raisonnable. Vérifiez la création des clés *via* les commandes `list` de `gpg`.
5. Extrayez votre clé publique et échangez-les vous par mail.
6. Importez alors les clés.
7. Vérifiez les clés et signez-les avec votre propre clé secréte (retestez les commandes `list`).
8. Chiffrez des fichiers. Déchiffrez-les. (Vous pouvez préciser plusieurs destinataires)
9. Expérimentez les différents commandes de signatures.
10. Examinez et éditez les valeurs de confiances des clés de votre trousseau.

2 $\mathbb{Z}/n\mathbb{Z}$

Pour étudier l'anneau $(\mathbb{Z}/n\mathbb{Z}, +, \times)$ (qu'on utilisera de façon intensive par la suite), vous allez programmer en Python et utiliser la librairie SymPy (`https://www.sympy.org/`).

En attendant le cours sur le sujet, nous pouvons voir $\mathbb{Z}/n\mathbb{Z}$ comme l'ensemble des entiers de 0 à $n-1$, avec comme opérations l'addition et la multplication "modulo n".

1. Affichez la table de multiplication (sous forme d'un tableau à 2 dimensions) de $\mathbb{Z}/12\mathbb{Z}$ et de $\mathbb{Z}/17\mathbb{Z}$. Voyez-vous des différences ?
2. La fonction $\varphi(n)$ représente le nombre d'entiers entre 1 et n qui sont premiers à n, autrement dit,

   $$\varphi(n) = \#\{x \in [\![1, n]\!] : \text{pgcd}(x, n) = 1\}.$$

   Nous verrons en cours que si $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

   $$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right)$$

   Grâce à la méthode factorint de SymPy, écrivez une méthode phi_n qui prend en entrée un entier n et calcule $\varphi(n)$. Appliquez-la sur des entiers n de tailles différentes.

3. Écrivez une méthode inversible qui prend en entrée un entier n et renvoie les éléments premiers à n.
4. Vérifiez que le nombre d'éléments inversibles modulo n est $\varphi(n)$.
5. Calculez $x^{\varphi(n)} \mod n$ pour plusieurs n et pour des x inversibles modulo n.
6. L'*ordre* d'un élément $g \in (\mathbb{Z}/n\mathbb{Z})^\star$ est le plus petit entier r tel que $g^r = 1 \mod n$. Écrivez une méthode ordre qui prend en entier deux éléments g et n, teste si g est inversible modulo n et calcule son ordre.
7. Tracer une courbe du temps de calcul de l'ordre d'un élément en fonction de la taille binaire. Qu'observez-vous ?
8. Écrivez une fonction ordres qui prend en entrée un entier n et affiche les ordres de tous les entiers inversibles modulo n. Qu'observez-vous ?