Cryptographie

Exercice 1. ÉCHAUFFEMENT

1. En utilisant `primes`, faire la liste de tous les nombres premiers inférieurs à 10000 et congrus à 9 modulo 13. Combien y en a-t-il ? Donner la proportion de ces nombres premiers dans l'ensemble des nombres premiers inférieurs à 10000, et comparer avec la fraction $\frac{1}{12}$.
2. À l'aide des fonctions `euler_phi` et `point`, représenter la valeur de l'indicatrice d'Euler pour n compris entre 2 et 1000. Comment reconnaît-on qu'un nombre est premier en regardant le graphique ?

Exercice 2. RESTES CHINOIS ET POLYNÔMES

On considère les polynômes $P = X^2 - 1$ et $Q = X^2 + 2X$ de $k[X]$, avec k un corps.

1. Dans cette question, on suppose que $k = \mathbb{Q}$.
   (a) Définir $k[X]$ ainsi que P et Q dans `Sage`, et calculer le pgcd de P et Q.
   (b) Soit
   $$\begin{array}{rccc} \psi: & k[X]/(PQ) & \longrightarrow & k[X]/(P) \times k[X]/(Q) \\ & R \bmod PQ & \longmapsto & (R \bmod P,\ R \bmod Q)\,. \end{array}$$
   Justifier à l'aide du cours que ψ est un isomorphisme et expliciter son inverse.
   (c) À l'aide de la fonction de `Sage` dédiée, vérifier la formule de l'inverse obtenue à la question précédente pour $(1, 0)$ puis pour $(X + 2, X - 1)$.
2. On suppose maintenant que $k = \mathbb{Z}/3\mathbb{Z}$.
   (a) Calculer le pgcd de P et Q. Commenter le résultat par rapport à celui de la question 1 (a).
   (b) Soit
   $$\begin{array}{rccc} \phi: & k[X] & \longrightarrow & k[X]/(P) \times k[X](Q) \\ & R & \longmapsto & (R \bmod P,\ R \bmod Q)\,. \end{array}$$
   Montrer que le noyau de ϕ est l'idéal $(X^3 - X)$.
   (c) On note $\pi : k[X] \longrightarrow k[X]/(X^3 - X)$ la projection canonique, et $k_2[X]$ l'ensemble des polynômes à coefficients dans k de degré au plus 2. Montrer que la restriction de π à $k_2[X]$ est bijective. Est-ce un isomorphisme d'anneaux ?
   (d) Définir dans `Sage` une liste L de représentants des éléments de $k[X]/(X^3 - X)$, et calculer leurs images par le morphisme induit par ϕ.
   (e) Déterminer un élément qui n'est pas dans l'image de ϕ. Justifier (sans utiliser la question précédente).

Exercice 3. RSA

1. Écrire une fonction `rsa(p, q)` prenant en entrée deux nombres premiers $p \neq q$ et renvoyant un couple de clés RSA, c'est-à-dire un couple

$$(\text{clé publique, clé privée}) = ((n, e), (n, d)),$$

où $n = pq$. On pourra utiliser `randint` pour choisir un entier au hasard. Pour avoir de grands nombres premiers p et q, on pourra combiner cette fonction et `next_prime` (`Sage` trouve immédiatement un nombre premier avec 128 bits ; il lui faut quelques secondes pour 1024 bits, et quelques minutes pour 2048 bits).

2. Écrire une fonction `chiffre(m, cle_pub)` prenant en entrée un entier m et une clé publique RSA (n, e) et renvoyant le message chiffré associé à m par le cryptosystème RSA relatif à (n, e).
3. Écrire une fonction `dechiffre(l, cle_priv)` prenant en entrée un entier l et une clé privée RSA (n, d) et renvoyant en sortie le message clair correspondant.
4. Écrire, en utilisant `factor`, une fonction `casse(cle_pub)` pour casser une clé RSA, c'est-à-dire passer d'une clé publique (n, e) à la clé privée (n, d) correspondante.

Exercice 4. RSA AMÉLIORÉ

La théorie s'applique à des messages composés d'entiers. Pour pouvoir aussi transmettre des chaînes de caractères, il faut commencer par traduire ces chaînes de caractères en une liste d'entiers. C'est le rôle des fonctions `encode` et `decode` que vous trouverez sur ecampus. Voici quelques explications sur ces fonctions. Les caractères les plus courants sont codés sur 8 bits soit, en base dix, par un nombre entre 0 et 255. La fonction `ord` permet d'obtenir ce nombre. Par exemple, `ord('A')` renvoie `65` et `ord('!')` renvoie `33`. La fonction réciproque est `chr`. Par exemple, `chr(84)` renvoie `'T'`. Les chaînes de caractères peuvent ainsi être représentées en base $a = 2^8$. Plus précisément, on fait correspondre à une chaîne $c_0c_1 \cdots c_{k-1}$ de longueur k l'élément $\sum_{i=0}^{k-1} w_i a^i$ où w_i est l'entier associé au caractère c_i. Comme on veut travailler dans un $\mathbb{Z}/n\mathbb{Z}$ plutôt que dans $\mathbb{Z}$, il faut donc choisir n assez grand pour avoir $a^k < n$. Pour assouplir un peu cette contrainte, on va s'autoriser ici à découper un long message en suite de sous-messages de longueur k avec $a^k < n$. La fonction `encode` prend en entrée une chaîne de caractères m et un entier k et renvoie une liste de nombres ayant chacun au plus k chiffres en base a. La fonction réciproque `decode` prend en entrée une liste d'entiers N (en base dix) et renvoie en sortie la signification de N (en toutes lettres). Voici un exemple d'utilisation de ces fonctions :

```
sage: m = encode('Bonjour !', k=6); m
[129121387441986, 2170994]
sage: decode(m)
'Bonjour !'
```

1. En utilisant la fonction `encode`, écrire une fonction `chiffre_bis(m, cle_pub)` prenant en entrée une chaîne de caractères m et une clé publique RSA (n, e) et renvoyant le message chiffré associé à m par le cryptosystème RSA relatif à (n, e). On pourra choisir le nombre de lettres utilisées par `encode` de façon optimale.
2. En utilisant la fonction `decode`, écrire une fonction `dechiffre_bis(L, cle_priv)` prenant en entrée une liste L de nombres et une clé privée RSA (n, d) et renvoyant en sortie le message clair (en toutes lettres) correspondant.
3. Tester les fonctions précédentes sur la clé publique RSA (n, e) et les listes de mots chiffrés trouvées sur ecampus.