MiniRSA : travail à rendre

La ressource R3.09 de cryptographie sera, pour partie, évaluée par le travail réalisé sur le projet `MiniRSA` développé au cours du module (l'autre partie de l'évaluation étant une interrogation portant sur les principaux concepts de la cryptographie et de la sécurité informatique). **On vous demande de travailler en équipe (2-3 étudiants).**

Les critères d'évaluation sont :

- l'aboutissement des différents objectifs,
- la progression dans les développements,
- la qualité du code des extensions,
- et les tests et la gestion de votre projet sous `Git`.

A partir des spécifications proposées au cours du module R3.09 (en particulier, `exponentiation modulaire`, `pgcd`, `Bezout`, ...), on vous demande de poursuivre le développement du `MiniRSA` en utilisant les concepts de cryptographie asymétrique. Voici une liste de contributions attendues (mais vous pouvez ajouter des éléments à cette liste) :

- On retourne le chiffrement asymétrique afin de **signer** une information de manière numérique. On vous demande d'implémenter cette méthode en signant un message donné (ici un nombre de grande taille pour simplifier). On utilisera une fonction simple pour calculer une **empreinte** du message à envoyer [**requis**].
- Etendre cette implémentation pour générer un **certificat** pour la clé publique d'Alice signée par une autorité de certification (CA) disposant également d'une paire de clés asymétriques avec la séquence suivante [**requis**] :

```
// Alice : tire 2 nombres premiers pA et qA, nA=pA*qA
// Alice : clé publique (eA,nA)
// Alice : clé privée (dA,nA)

// CA : tire 2 nombres premiers pCA et qCA, nCA=pCA*qCA
// CA : clé publique (eCA,nCA)
// CA : clé privée (dCA,nCA)

// Alice : construire un message contenant sa clé publique et son empreinte
// (signe en chiffrant l'empreinte avec sa clé privée)
// (communique de manière confidentielle avec CA,
    en chiffrant avec la clé publique de CA)

// CA : récupère la clé publique d'Alice et vérifie l'empreinte
// CA : génère le certificat de la clé publique d'Alice
// (chiffre de la clé publique d'Alice avec la clé privée de CA)

// Bob : vérifie le certificat d'Alice
// (la clé publique d'Alice doit correspondre au déchiffrement du certificat
    avec la clé publique de CA)
```

- Ecrire une mini-application proposant une **interface utilisateur** (en mode texte ou graphique) [**optionnel**] :
  - pour initialiser les clés des 3 participants (Alice, Bob et CA),
  - pour saisir un message et générer le message chiffré et/ou signé,
  - pour vérifier/décodé le message reçu.

  On ne demande pas ici d'effectuer un réel échange de message sur un réseau ....

Pour aller plus loin, vous pouvez faire évoluer le `MiniRSA` afin de répondre aux fonctionnalités suivantes :

- Le test de `Fermat`, implémenté pour vérifier la pseudo-primalité d'un entier, peut être remplacé par le test de `Miller-Rabin` réputé plus robuste (voir `https://fr.wikipedia.org/wiki/Test_de_primalit%C3%A9_de_Miller-Rabin`) [**optionnel**].
- Jusqu'ici les outils mis en place ci-dessus fonctionnent pour des messages représentés par des nombres de grande taille. On vous demande d'étendre les opérations de chiffrement/déchiffrement et de signature pour manipuler des **messages textes** [**optionnel**].

  On pourra par exemple s'appuyer sur les fonctions suivantes (en `Python`) :

```
# Helper function
def str_to_int(m):
    s = 0
    b = 1
    for i in range(len(m)):
        s = s + ord(m[i])*b
        b = b * 256
    return s

# Helper function
def int_to_str(c):
    s = ""
    q,r = divmod(c,256)
    s = s+str(chr(r))
    while q != 0:
        q,r = divmod(q,256)
        s = s+str(chr(r))
    return s
```

Ce travail est à rendre, au plus tard, le **vendredi 18 novembre 2022**, dans un projet hébergé sur le `Gitlab` de l'IUT. Pensez à inviter (avec un rôle $\geq$ `Reporter`) votre enseignant responsable du module R3.09.