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

50 avenue Tony Garnier
69007 Lyon, France

Master 1 informatique
Université Claude Bernard Lyon 1

Cryptologie : TP noté

— *Ce TP noté se fait en binôme. NOM1 et NOM2 sont les noms de famille des membres du binôme.*

— *Les réponses aux questions théoriques (marquées* **t***) seront rédigées dans un fichier à part appelé « NOM1_NOM2.txt »(ou pdf si vous faites du LaTeX).*

— *Votre code (qui correspond aux questions marquées* **p***) est à nous rendre par email :* `gerald.gavin@univ-lyon1.fr` *et* `ida.tucker@ens-lyon.fr`*.*

Pour nous simplifier la correction, vous ne nous enverrez que 2 fichiers (un pour chaque exercice) qui devront compiler/s'interpréter.

Vous nous enverrez une archive portant le nom `NOM1_NOM2.tar.gz` *(ou zip) contenant les fichiers correspondants à chacun des deux exercices et le fichier de réponse aux questions.*

Le sujet *du mail sera « [TP noté MIF29 2020] ».*

— **La note tiendra compte du respect des consignes.**

Exercice (Signature RSA)**.** Dans cet exercice, vous allez programmer une signature RSA qui assure une sécurité élevée (plus qu'une signature « textbook »).

Le schéma de signature RSA est constitué d'un algorithme de génération de clés KeyGen, d'un algorithme de signature Sign et d'un algorithme de vérification des signatures Verify.

Q1 t Donnez les entrées et les sorties de chacun des algorithmes ci-dessus.

La génération des clés est la même que celle pour chiffrement, que vous avez déjà programmé.
La génération des signatures utilise une fonction de hachage h, et une signature σ est produite ainsi

$$\sigma = h(m\|r)^d \mod N,$$

où m est le message à signer, d l'exposant secret, N le module RSA et r est un aléa de 256 bits. Pour implémenter cela, vous utiliserez la fonction de hachage suivante pour étendre le haché à tout $\mathbb{Z}/N\mathbb{Z}$ (ce qui est nécessaire pour la sécurité). Vous partirez de la fonction SHA-256 (que vous trouverez dans une bibliothèque adéquate) et calculerez h ainsi :

$$h(m\|r) = \mathsf{SHA256}(m\|r\|1)\|\mathsf{SHA256}(m\|r\|2)\|\ldots\|\mathsf{SHA256}(m\|r\|\ell) \mod N,$$

ℓ étant l'entier le plus petit tel que $\ell \times 256$ est plus grand que $n + 100$ où n est la taille binaire de N. Le symbole $\|$ représente la concaténation. L'entier ℓ sera alors ajouté σ et r pour la vérification.

Q2 p Programmez les 3 algorithmes du schéma de signature RSA, et vérifiez qu'ils fonctionnent sur des exemples de taille cryptographique.

Exercice (Chiffrement de Merke et Hellman)**.** Cet algorithme de chiffrement à clé publique est dû à R. Merkle et M. Hellman et repose sur la complexité algorithmique du problème du *sac à dos*. Ce problème consiste, étant donné un n-uplet $(a_1, a_2, \ldots, a_n) \in \mathbb{N}^n$ et $S \in \mathbb{N}$, à trouver un n-uplet $(m_1, m_2, \ldots, m_n) \in \{0, 1\}^n$ tel que $S = \sum_{i=1}^n a_i m_i$.

Si les $(a_i)_i$ n'ont pas de propriétés particulières, la recherche des $(m_i)_i$ (le message codé en binaire) demande un travail exponentiel en n.

Pour pouvoir utiliser cette propriété en cryptographie, il faut trouver une classe de n-uplet $(a_1, \ldots, a_n)$ pour lesquels on puisse résoudre facilement le problème grâce à une trappe. Les *suites super-croissantes* vont permettre de servir de trappe. Un n-uplet $(b_1, \ldots, b_n)$ est dit *super-croissant* si et seulement si

$$\forall i \in [\![2, n]\!] \quad b_i > \sum_{j=1}^{i-1} b_j.$$

$\boxed{\text{Q5 } \mathbf{t}}$ Donnez un exemple de suite super-croissante constituée de 10 entiers.

$\boxed{\text{Q6 } \mathbf{p}}$ Programmez un algorithme qui produit une suite super-croissante.

$\boxed{\text{Q7 } \mathbf{t}}$ Montrez que dans le cas des suites super-croissantes, le problème du sac à dos est facile.

$\boxed{\text{Q8 } \mathbf{p}}$ Programmez l'algorithme pour le résoudre.

L'idée de Merkle et Hellman consiste maintenant à “tordre” les b_i de facon à obtenir un n-uplet qui n'est plus super-croissant.

Pour ce faire, on choisit un nombre $u > \sum_{i=1}^n b_i$, et un entier $v \in \mathbb{N}$ tel que $\text{pgcd}(u, v) = 1$. Alors v est inversible dans $\mathbb{Z}/u\mathbb{Z}$. On note $w = v^{-1} \in \mathbb{Z}/u\mathbb{Z}$. On calcule alors le n-uplet $a = (a_1, \ldots, a_n)$ tel que pour tout $i \in [\![1, n]\!]$,

$$a_i = v \cdot b_i \in \mathbb{Z}/u\mathbb{Z}.$$

Le n-uplet a est alors rendu public, tandis que v est gardé secret. Pour transmettre le message m à Bob, Alice va à partir de l'écriture de m sur n bits $m = m_n m_{n-1} \ldots m_1$, calculer la somme

$$c = \sum_{i=1}^{n} m_i a_i$$

qu'elle va envoyer à Bob. Si le message est intercepté par Ève, celle-ci ne pourra pas (*a priori*) retrouver la suite des m_i à partir des c et du n-uplet a car c'est un problème de type sac à dos.

$\boxed{\text{Q9 } \mathbf{t}}$ Donnez l'algorithme de déchiffrement.

$\boxed{\text{Q10 } \mathbf{p}}$ Implémentez les algorithmes de génération de clés, de chiffrement et de déchiffrement.