RS40 : Travaux pratiques RSA

Avertissement :

L'objectif du TP consiste à consolider les notions vues en cours concernant RSA. Il ne s'agit nullement, d'un TP destiné à l'industrialisation. Il existe des outils conçus pour ce fait qui seront abordés le prochain TP.
Il y a ainsi quelques faiblesses dans le TP. Voici les plus importants :

- Taille des clés très faible
- Il n'y a ni le bourrage ni le réseau de Feistel asymétrique (Optimal asymmetric encryption padding)
- Le HMAC s'appuie seulement sur une fonction de hachage pour la signature et cette fonction n'est pas sécurisée (md5).
- Le TP utilise la conversion en décimal (mauvaise manipulation de l'encodage) pour un objectif pédagogique.

L'ensemble des fonctions commencent par home pour rappeler qu'il s'agit d'une approche simplifiée faite maison.

Sujet :

Le TP considère un message envoyé de Bob vers Alice, où chacun dispose d'une paire de clés (publique, privée) pour le chiffrement RSA. Bob chiffre le message avec la clé publique d'Alice. Il procède aussi à la signature de l'empreinte numérique du message avec sa clé privée. Alice reçoit le message le déchiffre et vérifie la signature de Bob (Ce n'est pas sécurisé).
Pour ce faire, le TP est accompagné d'un fichier RSA_B_A.py contenant les éléments donnés dans le tableau suivant :

Variables	Alice	Bob
Les deux nombres premiers	x1a et x2a	x1b et x2b
La fonction d'Euler et n	phia, na	phib , nb
Exposants publique, privé	ea, da	eb, db
Le message en clair	dechif (en string)	secret (en string), num_sec (en nombre décimal)
Le message chiffré	chif (message chiffré en nombre décimal)	chif (message chiffré en nombre décimal)
Le hash	Ahachis3 (en nombre décimal)	Bhachis3 (en nombre décimal)
Vérification de la signature	designe (en nombre décimal, déchiffré avec la publique de Bob)	signe (en nombre décimal, chiffré avec la clé privée de Bob)

Il lui manque des fonctions pour fonctionner correctement. Vous devez ainsi compléter les fonctions suivantes :

- home_mod_expnoent(x,y,n): La fonction qui permet de réaliser l'exponentiation modulaire x^y%n. Vous utilisez la fonction donnée en cours. Pour vérifier le résultat, vous pouvez utiliser la fonction pow(x,y,z) de python.
- home_ext_euclide(y,b) : La fonction permettant d'obtenir la clé secrète. Pour ce faire, il faut utiliser l'algorithme d'Euclide généralisé
- Améliorer le processus de vérification de la signature de Bob (fonction de hachage)
- Augmenter la taille des messages échangés
- Utiliser CBC pour le chiffrement des blocs (Bonus).

Bon courage