Tran Florian Groupe 4-2

Freret Tom

**BUT1 - SAE S1-02**

**COMMUNIQUER, EN TOUTE SÉCURITÉ**

**RAPPORT**

Partie 1 – Cryptographie : le chiffrement RSA

Pour commencer nous nous concentrons sur le chiffrage et le déchiffrage d’information avec un système de clé publique et clé privé. Nous avons tous d’abord créer notre fonction **key\_creation()**. Dans celle-ci nous avons créé une liste qui contient des nombres premiers grâce à une fonction extérieur **list\_prime(n)** qui renvoie une liste de l’ensemble des nombres premiers inférieurs ou égaux à ***n,*** dans celle-ci nous avons pris ***n = 1000***. Donc grâce à cette liste de nombres premiers nous créer deux variables ***p*** et ***q*** qui prennent chacun un entier dans la liste de nombres premiers aléatoirement. Ensuite nous créons deux autres variables ***n*** à qui nous attribuons la valeur de ***n = p \* q***, et ***Ⴔ*** à qui nous attribuons la valeur de ***Ⴔ(n) = (p - 1)\*(q - 1)*** ( à faire: démonstration :

<https://www.quora.com/What-is-p-1-q-1-in-cryptography-maths>

)

Après cela nous avons créé la variable ***pub*** qui prends un valeur un nombre premier parmi la liste des nombres premiers tant que ***pgcd(pub, Ⴔ(n)) == 1*** (***pgcd()*** est une fonction qui prends en entrée deux entiers ***a*** et ***b*** et renvoie ***d*** le plus grand diviseur commun et ***r*** le reste). Nous avons ensuite créé la fonction ***extended\_gcd()*** qui est l’Algorithme d’Euclide étendue prend en entrée 2 entiers ***a*** et ***b*** et qui retourne trois entiers ***d*** est le plus grand diviseur commun de ***a*** et ***b***, et ***u*** *et* ***v*** le coefficient de Bézout. Et grâce à cette fonction nous créons la variable ***priv*** qui prends en valeur ***u*** de notre fonction ***extended\_gcd()*** avec en entré ***pub*** et ***Ⴔ(n) .*** Enfin La fonction **key\_creation()** nous renvoie ***n, pub et priv***.

Avec cela nous avons créé nos clés, alors nous passons à la prochaine fonction ***encryption()*** qui prends en entrée notre clé publique***(n, pub)*** et un message texte ***msg***qui est convertit en une liste de nombres avec une nouvelle fonction ***convert\_msg()*** qui prends un message textuel et est convertie grâce à la table ASCII, chaque caractères est convertit en un nombre à trois chiffres. Et cette fonction renvoie une liste de nombres groupé de 4 en 4 pour éviter les attaques fréquentielles et ne pas avoir besoin d’un ***n*** trop grand. Ensuite la fonction ***encryption()*** qui prends en entrée ***n,*** notre clé public ***pub ,msg*** , et qui va pouvoir manipuler notre message convertit. Le but de cette fonction est de chiffré notre message, alors nous avons créé une liste ***crypted\_msg*** et avec la formule du chiffrement de message Me ≡C[n], avec M notre message convertit et C notre clé publique > n, ensuite grâce à une boucle cela nous renvoie la liste ***crypted\_msg*** qui est le message chiffré.

Après cela, nous devons créé la fonction ***decryption\_msg()*** qui prends en entrée ***n,*** notre clé privé ***priv*** et ***msg*** notre message chiffré. Grâce à la formule M≡Cd[n] avec ***d*** l’inverse de ***e*** modulo ***Ⴔ(n)*** on déchiffre C. Ensuite si la clé privée est négatif on utilise l’opposé de notre clé privé et l’opposé du message chiffré grâce au ***extended\_gcd() ,*** pour notre formule car un nombre à la puissance d’une clé privé négatif serait trop petit. Une fois fini la fonction nous renvoie notre message déchiffré.

Pour finir nous avons créé une dernier fonction ***reconvert\_msg()*** qui prends en entré notre message déchiffré, [ expliqué l’histoire des zéro (pas encore viable) ] et cette fonction nous renvoie le message déchiffré en caractères.