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.

Partie 2 – Code correcteurs

Q 2.1)

Les tables d’addition et de multiplication de F2 sont les suivantes :

|  |  |  |
| --- | --- | --- |
| F2 | F2 | F2 + F2 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| F2 | F2 | F2 x F2 |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(On remarque que la table d’addition est égale à la table de vérité du connecteur logique OR et celle de la multiplication à AND)

Q 2.2)

On a e1 = (1,0,0,0), e2 = (0,1,0,0), e3 = (0,0,1,0), e4 = (0,0,0,1) en additionnant de certains façons ces vecteurs nous devrons retrouver l’ensemble des vecteurs de F42 les voici tous :

(0, 0, 0, 0) = e1 + e1 (1, 0, 0, 0) = e1

(0, 0, 0, 1) = e4 (1, 0, 0, 1) = e1 + e4

(0, 0, 1, 0) = e3 (1, 0, 1, 0) = e1 + e3

(0, 0, 1, 1) = e3 + e4 (1, 0, 1, 1) = e1 + e3 +e4

(0, 1, 0, 0) = e2 (1, 1, 0, 0) = e1 + e2

(0, 1, 0, 1) = e2 + e4 (1, 1, 0, 1) = e1 + e2 + e4

(0, 1, 1, 0) = e2 + e3 (1, 1, 1, 0) = e1 + e2 + e3

(0, 1, 1, 1) = e2 + e3 + e4 (1, 1, 1, 1) = e1 + e2 + e3 + e4

On peut bien représenter tous l’ensemble des vecteurs qui font partie de l’ensemble F42 avec une somme de vecteurs ei .

Q 2.3)

On cherche à déterminer l’ensemble des vecteurs de l’image de l’application linéaire ø (définie dans le sujet). On va donc utiliser les matrices M et v et avec ø (v) = M · v

Soit M = et v = , on obtient M · v = =

# On en déduit donc que Im(ø) = { ((a+b+d), (a+c+d), (a), (b+c+d), (b), (c), (d) | (a,b,c,d) ∈ F42 }

Q 2.4)

Soit u, v ∈F42 et u ≠ v, on cherche à montrer que pour tous vecteurs de Im(ø), d(u, v) >= 3

Tout d’abord à cause des règles d’addition des matrices, on peut déduire que si u ≠ v, Il peut y avoir soit 4 soit 3, soit 2, soit 1 différences entre les éléments de u et v mais aussi qu’une différence équivaut à un élément égal à 1(voir tableau addition Q 2.1). Nous allons donc étudier ces quatre cas avec u + v = (a, b, c ,d) ∈F42:

* Cas avec 4 différences :

S’il y a 4 différences, alors u + v = (1, 1, 1, 1) et donc ø(u+v) = (1, 1, 1, 1, 1, 1, 1) et w(ø(u+v)) = 7

* Cas avec 3 différences :

S’il y a 3 différences, il y aura 3 valeurs parmi a, b, c, d qui seront égales à 1 or comme nous avons vu dans la question précédente l’image de l’application ø = { ((a+b+d), (a+c+d), (a), (b+c+d), (b), (c), (d) | **(**a,b,c,d**)** ∈F42}. On retrouve a, b, c, d seul et donc logiquement w(ø(u+v)) >= 3

* Cas avec 2 différences :

Preuve fait avec le programme python pour éviter les redondances

* Cas avec 1 différence :

Preuve fait avec le programme python pour éviter les redondances

On en conclut donc que d(u, v) >= 3 avec u, v de Im(ø) tel que u ≠ v.

Q 2.5)

On veut prouver que d(u, ũ) < d(v, ũ), on sait que u et ũ ont qu minimum six composantes égales.

Ce qui implique que d(u, ũ) est soit égale à 0 ou 1. Mais on sait aussi que u ≠ v et donc par la question précédente que :

d(u, v) >= 3 ⬄ d(u, v) = d(u, ũ) + d(ũ, v) >= 3

mais on sait que d(u, ũ) <= 1 on en déduit que d(ũ, v) >= 2.

On en conlut que d(u, ũ) <= 1 < d(v, ũ) <= 2 ⬄ d(u, ũ) < d(v, ũ)

Alice pourra corriger son message si elle est sur qu’il y est un bit maximum bruité car il suffit de retrouver le vecteur avec la distance égale à 1 en effet on vient de le prouver avec les 2 questions précédentes. Avec l’exemple précèdent u et ũ, On montre que tous les autres vecteurs on une distance supérieur à la distance du message bruitée.

Partie 3 – Communication sécurisée