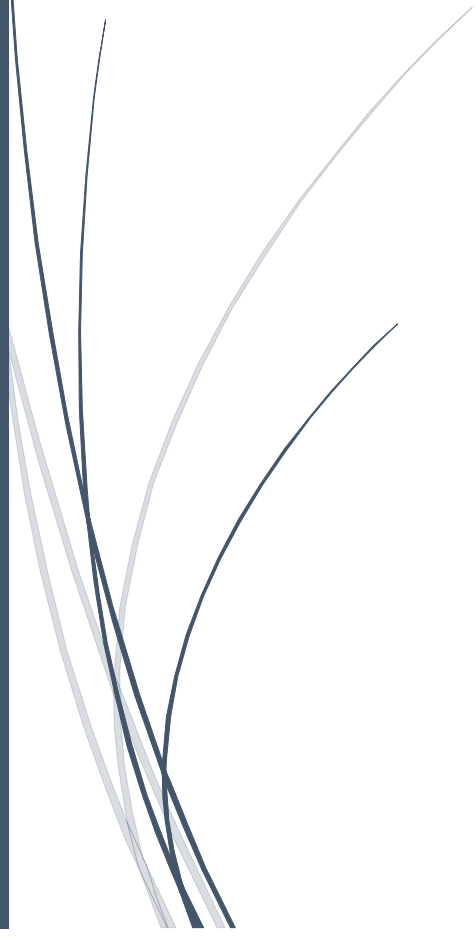
Année 2016-2017

Modélisation
Mathématiques

Cryptographie et RSA



Clément Lopez, Charles Thiebaud, Emmanuel Tocabens
Professeur : Rémi Boulle

Table des matières

I. Abstract

In the over-connected world where we live in today, secured communications are essentials. In order to guarantee this security, several cryptosystems are available. We looked at the most common methods, symmetric and asymmetric, and searched for the most secure and most convenient system. We tested all these cryptosystems' security and speed, especially the RSA. Although the One-time pad method cannot be cracked ,it requires a one-time pre-shared key, And therefore does not allow its use for many cases, such as securing Internet exchanges. That's why RSA is the most common system used to encrypt exchanges. It has only a few flaws and allows user to exchange symmetric encryption key.

II. Résumé

Dans le monde ultra-connecté dans lequel nous vivons aujourd'hui, des communications sécurisées sont essentielles. Dans le but de garantir cette sécurité, plusieurs cryptosystèmes sont disponibles.
Nous avons analysé les méthodes les plus courantes, symétriques et asymétriques, et cherché le système le plus sécurisé et pratique. Nous avons testé la sécurité et la rapidité de tous ces systèmes, plus particulièrement le RSA. Bien que la méthode du masque jetable ne peut pas être cassé, il requiert une clé à usage unique partagée à l'avance et donc ne permet pas son utilisation pour de nombreux cas, comme la sécurisation des échanges sur Internet. C'est pourquoi le RSA est le système le plus utilisé pour crypter les échanges. Il n'a seulement que peu de vulnérabilités et permet aux utilisateurs d'échanger des clés symétriques.

III. Introduction

La cryptographie est une discipline aussi ancienne que passionnante. Depuis de nombreuses décennies, les Hommes ont inventé une multitude de méthodes pour protéger leurs correspondances, leurs ordres ou leurs messages en les rendant incompréhensible à toute personne autre que son destinataire et au moins autant de techniques pour « casser » les messages codés de leurs ennemis. Le chiffrement RSA, apparu à la fin des années 1970, est l'une des méthodes de cryptage parmi les plus utilisées dans le monde actuel. Sans même le savoir et sans nous en rendre compte, nous utilisons le chiffrement RSA chaque jour, notamment dès que nous allons sur Internet. Partant de ce constat, nous sommes en droit de nous interroger sur sa sureté.

Quelle est la fiabilité du chiffrement RSA ? Quels sont ses avantages et ses inconvénients ?

Pour répondre à cette question, nous avons commencé par analyser plusieurs systèmes de chiffrement existants autres que le RSA pour ensuite nous demander en quoi le chiffrement RSA, qui est un chiffrement très récent, nous assurait-il une meilleure sécurité que ces autres chiffrements et s'il permettait à lui tout seul une sécurité suffisante.

IV. Notion de clé

La clé est une notion essentielle dans la cryptographie. En effet, de la même manière que l'on peut utiliser une clef pour cacher (et récupérer) des informations dans un coffre-fort fermé à clef, une clé en cryptographie est un paramètre utilisé en entrée d'une opération cryptographique (on l'utilise pour crypter/décrypter un message). Pour ce faire, une clé peut se présenter sous plusieurs formes (exemples : un mot ou une phrase, un nombre, des données codées sous une forme binaire, etc…)

Partant de cette notion, nous pouvons découper les différentes méthodes de cryptographie en deux grandes familles :

- Les chiffrements symétriques : c'est-à-dire toutes les méthodes de chiffrement dont la clé pour chiffrer est également utilisé lors du déchiffrage.
- Les chiffrements asymétriques : c'est-à-dire toutes les méthodes dont les clés de chiffrement et de déchiffrement sont deux clés différentes.

Dans les parties qui vont suivre, nous allons explorer plus en détails ces deux types de cryptage ainsi que d'en voir quelques exemples.

V. Chiffrements symétriques

La cryptographie symétrique est la plus ancienne forme de chiffrement, les Egyptiens utilisaient déjà une méthode de ce type vers 2000 av. J.-C. tout comme Jules César qui a donné son nom au chiffre de César, une autre méthode de ce type de cryptographie.

Comme expliqué précédemment, ce type de méthodes repose sur le fait que la clé pour chiffrer et déchiffrer le message est la même. Il est donc impératif que la clé soit secrète et seulement connue par l'émetteur et le destinataire du message.

Nous avons retenu quelques exemples de cryptage symétrique :

A. Chiffre de César

1. Explications

Connu sous différentes dénominations (chiffre de César, substitution de César, code de César), cette méthode fut utilisée par le célèbre général Romain dans ses correspondances secrètes.

C'est une méthode de chiffrement par décalage qui consiste simplement à remplacer chaque lettre du message clair par une autre située à une distance N fixe.

Le résultat final est souvent incompréhensible pour qui n'a pas la clé permettant de déchiffrer le message. Cependant, si on connait la méthode employée pour chiffrer le message, il est très facile de le déchiffrer puisque la clé de chiffrement n'a que 26 valeur possible (limité à cause du nombre de lettres de l'alphabet latin). En les testant toutes, on retrouve rapidement le message initial.

De la même manière, on peut retrouver le message initial en utilisant l'analyse de fréquence : dans chaque langue, certaines lettres ou combinaisons de lettres apparaissent avec une certaine fréquence. En calculant la fréquence d'apparition de chaque lettre du message codée, on peut faire une hypothèse sur le message clair.

2. Exemple

a) Partie chiffrement

Prenons par exemple le message clair que l'on veut crypter suivant :

CECI EST UN MESSAGE CLAIR

Soit N = 3 (le paramètre de décalage)

Principe de décalage :

Avec le chiffre de César, le C (3ème lettre de l'alphabet) devient un F (3 + N = 3 + 3 = 6ème lettre de l'alphabet). De la même manière, le E devient H, le I devient L, etc…

Ce qui nous donne comme message crypté :

FHFL HVW XQ PHVVDJH FODLU

b) Partie déchiffrement

Le destinataire reçoit donc le message crypté « FHFL HVW XQ PHVVDJH FODLU » et va devoir le décrypter.

Il possède lui aussi N = 3 et de la même manière que l'émetteur a crypté le message en décalant dans un sens, le destinataire décrypte en décalant dans l'autre sens.

De ce fait, le F (6è lettre de l'alphabet) redevient un C (6-3 = 3è lettre de l'alphabet) et on retrouve le message d'origine (CECI EST UN MESSAGE CLAIR).

B. Chiffre de Vigenère

1. Explications

Le chiffre de Vigenère (du nom du cryptologue Blaise de Vigenère) peut en quelque sorte s'apparenter à une version améliorée du chiffre de César. En effet, cette méthode de cryptage utilise le même principe de substitution que le chiffre de César mais la clé utilisée pour cette méthode n'est plus mono alphabétique mais intervient sur une séquence.

De ce fait, deux même lettres du message initial peuvent être transformées en deux lettres différentes dans le message crypté en fonction de leur position. Cette méthode résiste ainsi à l'analyse de fréquence tant que la personne qui tente de déchiffrer ne connait pas la longueur de la fréquence de la clé.

2. Exemple

a) Partie chiffrement

Prenons le message clair suivant :

MESSAGES

Soit N = INFO (la clé du chiffrement)

Principe de décalage :

La clé utilisée ici (INFO) a une longueur de 4 que l'on va répéter autant qu'il le faudra. C'est-à-dire que la première lettre du message, M (13è lettre), sera chiffrée avec la première lettre de la clé, I(9è lettre mais on décale de 8 car on considère A équivalent à un décalage de 0, B de 1, etc…), et donnera donc U (13 + 8 = 21è lettre de l'alphabet). De la même manière on chiffre E par N (R), S par F (X), S par O (G) puis on recommence avec la première lettre de la clé (A par I, G par N, etc…)

Ce qui nous donne au final le message crypté suivant : URXGITJG

b) Partie déchiffrement

Comme pour César, on applique la clé dans le sens inverse du cryptage. U (21è lettre) redevient M (21 – 8 = 13è lettre de l'alphabet). Et on retrouve :

MESSAGES

C. Chiffrements par permutation

1. Explications

Ce type de chiffrement a plusieurs dérivés, dans l'Antiquité les Spartiates utilisaient déjà une méthode de ce genre.

Le principe de base à toutes les méthodes de ce type est que, contrairement aux méthodes par substitution (dont on a vu deux exemples précédemment), on garde les mêmes lettres que le texte original mais on va les mélanger. De ce fait, plus le texte est long, plus il y a de possibilités de mélange et donc plus il est compliqué à décoder si on n'a pas la clé. Par exemple, un simple message de 3 lettres ne pourra prendre que 6 combinaisons possibles, ainsi « oui » ne peut être transformé qu'en « oui », « oiu », « uoi », « uio », « iou » ou « iuo ».

Plusieurs techniques existent pour rendre le texte codé (en spirale, en biais, méthode de Rail Fence ou méthode par zigzag, etc…).

2. Exemple de chiffrement par permutation : Transposition par colonne

Une méthode simple de permutation consiste à placer le texte voulu dans un tableau et à chiffrer le message en prenant toutes les lettres d'une colonne puis toutes les lettres d'une autre colonne, etc… en suivant l'ordre de la clé de transposition.

Par exemple, on veut chiffrer le message suivant :

JE VAIS VOUS PRESENTER UN EXEMPLE DE CODE EN SYSTEME DE PERMUTATION.
REMI BOULLE EST SUPER

Avec la clé de transposition suivante : 7 3 9 2 1 4 8 5 10 6

On va tout d'abord mettre le texte dans un tableau de 10 colonne (car la clé est de longueur 10 : 10 nombres présents dans la clé de transposition). On peut choisir de remplacer les espaces par des x et les points par xx, ce qui donne :

1	2	3	4	5	6	7	8	9	10
J	**E**	x	**V**	**A**	**I**	**S**	x	**V**	**O**
U	**S**	x	**P**	**R**	**E**	**S**	**E**	**N**	**T**
E	**R**	x	**U**	**N**	x	**E**	**X**	**E**	**M**
P	**L**	**E**	x	**D**	**E**	x	**C**	**O**	**D**
E	x	**E**	**N**	x	**S**	**Y**	**S**	**T**	**E**
M	**E**	x	**D**	**E**	x	**P**	**E**	**R**	**M**
U	**T**	**A**	**T**	**I**	**O**	**N**	x	x	**R**
E	**M**	**I**	x	**B**	**O**	**U**	**L**	**L**	**E**
x	**E**	**S**	**T**	x	**S**	**U**	**P**	**E**	**R**

On lit ensuite verticalement chaque colonne dans l'ordre prédéfini par la clé. C'est-à-dire qu'on va commencer par la colonne 7 (SSEXYPNUU), puis la 3 (XXXEEXAIS), la 9 (VNEOTRXLE), etc…

Ce qui nous donne le message suivant :

SSEXY PNUUX XXEEX AISVN EOTRX LEESR LXETM EJUEP EMUEX VPUXN DTXTX EXCSE XLPAR NDXEI BXOTM DEMRE RIEXE SXOOS

Pour le déchiffrer il suffit de re-remplir les colonnes verticalement, dans l'ordre défini par la clé.

3. Exemple de chiffrement par permutation : méthode de Rail Fence

La méthode de Rail Fence (barrière de rail ou palissade en français) est un chiffre de transposition consistant à écrire un texte en zigzag et à le lire en ligne droite.

Par exemple, prenons le message : HELLO WORLD

Que l'on veut coder avec une clé de 2, c'est-à-dire que l'on va écrire le message sur 2 lignes, une lettre sur une ligne et la suivante sur l'autre ligne. Ce qui nous donne :

H L O O L
E L W R D

Soit le message HLOOL ELWRD.

VI. Chiffrements asymétriques

L'autre grande famille de la cryptographie est la cryptographie asymétrique. Cette dernière est beaucoup plus récente que la cryptographie symétrique puisque le concept de cryptographie asymétrique (ou cryptographie à clé publique et privée) n'est apparu que vers la fin du XXe siècle, dans les années 70.

Le principe du chiffrement asymétrique est d'avoir 2 clés que l'utilisateur construit lui-même : une clé dite publique permettant d'encoder les messages et une autre dite privée permettant le décodage de messages cryptés.

La clé publique est transmissible sans restriction, cela permet à tout le monde d'envoyer des messages codés à l'utilisateur.

La clé privée en revanche n'est jamais transmise puisqu'elle permet de décoder les messages reçus.

Le principe est donc le suivant : l'expéditeur a recours à la clé publique du destinataire pour coder son message et le destinataire utilise sa clé privée pour décoder le message de l'expéditeur.

A. Chiffrement RSA

1. Explications

L'une des méthodes de chiffrement asymétrique les plus connues tenant son nom des initiales des noms de ses 3 inventeurs (Rivest, Shamir et Adleman). Le chiffrement RSA est basé sur l'utilisation des nombres premiers et d'un modulo, c'est une méthode très utilisée pour l'encodage de pages web. Cependant, les nombres utilisés pour le chiffrement étant très grand, il est très coûteux en temps et en mémoire d'encoder tous les messages avec cette méthode. Le chiffrement RSA est donc souvent utilisé pour transmettre une clé symétrique qui permet ensuite de poursuivre l'échange de manière confidentielle.

2. Fonctionnement

On appellera Alice la personne qui désire recevoir un message chiffré, et Bob la personne qui envoie le message.

a) Création des clés

Alice choisit deux entiers naturels premiers : P et Q (généralement, plus ces nombres sont grands, plus le système de cryptage/décryptage sera sécurisé).

Alice fait ensuite le produit de ces deux nombres : N = P*Q (module de chiffrement)

Alice calcule ensuite $\phi(N) = (P - 1)(Q - 1)$ (indicatrice d'Euler en N)

Ensuite, Alice choisit un entier E premier avec $\phi(N)$ (aussi appelé exposant de chiffrement)

A ce stade, on a la clé publique (N, E) qu'Alice peut communiquer à ses contacts pour qu'ils lui envoient des messages.

Pour obtenir la clé privée, il reste une dernière étape, Alice doit calculer l'entier naturel D, inverse de E modulo ϕ(N) (aussi appelé exposant de déchiffrement). Alice vient d'obtenir sa clé privée que l'on note (N, D)

b) Chiffrement et Déchiffrement

Prenons M le message clair que Bob veut envoyer à Alice et C le message chiffré.

Bob est en possession de la clé publique d'Alice (N, E). Il va donc coder le message comme suit :

C ≡ M^E [N]

Alice reçoit donc le message C qu'elle va déchiffrer en utilisant sa clé privée (N, D) :

M ≡ C^D [N]

3. Exemple concret de chiffrement RSA

a) Création des clés

Prenons deux nombres premiers P et Q tel que :

P = 3 et Q = 11

Dans la continuité, on calcule leur produit N :

N = 33

Alice calcule ensuite l'indicatrice d'Euler en N, ϕ(N) :

ϕ(N) = (3-1)*(11-1) = 20

On choisit E, l'exposant de chiffrement, premier avec ϕ(N) :

E = 3 (premier avec 20)

Enfin, on calcule D, l'exposant de déchiffrement, inverse de E modulo ϕ(N) :

D =7 (E*D = 3*7 ≡ 1 [20])

On vient donc de créer notre clé publique (33, 3) et une clé privée (33, 7).

b) Chiffrement et Déchiffrement

Prenons un message à coder :

M = 4

En utilisant la clé publique, on obtient le message codé suivant :

C = 4^3 ≡ 31 [33]

Donc C = 31

Le destinataire du message, possédant la clé privée, va pouvoir décoder le message crypté pour retrouver le message d'origine :

M = 31^7 ≡ 4 [33]

Donc M = 4

Le message a bien été transmis.

4. Limites

Le chiffrement RSA est un des plus utilisé à l'heure actuelle, on le retrouve notamment sur Internet. Cependant, malgré un niveau de sécurité parmi les meilleurs de la longue liste des méthodes de chiffrements connues et hautement plus supérieur aux chiffrements symétriques que nous avons vus précédemment, RSA comporte plusieurs limites à la fois sur un plan pratique (temps et mémoire utilisé) comme sur un plan purement technique (il peut être cassé).

a) Temporelles/Pratiques

On peut noter qu'il est très long de générer de très grands nombres premiers et du fait de ces grands nombres utilisés, l'encodage et le décodage de messages peut lui aussi être très long.

De plus, pour que la sécurité de cette méthode soit optimale et que le message chiffré par RSA soit difficilement décodable sans la clé privée, les clés doivent être très grande. Très souvent, les clés sont plus grandes que le message en lui-même.

b) Techniques

Le système RSA est un des systèmes les plus sécurisés. La principale sécurité du chiffrement RSA repose sur la difficulté à factoriser N. En effet, à l'heure actuelle il est pratiquement impossible de retrouver dans un temps raisonnable les nombres P et Q à partir de N si ce dernier est très grand.

Cependant, il existe plusieurs attaques possibles pour « casser » un message codé avec le chiffrement RSA :

C'est le cas notamment pour l'attaque de Wiener qui tient son nom au cryptologue Michael J. Wiener qui est particulièrement performante si D est petit (si D < N^(1/4)). On peut retrouver dans ce cas l'exposant secret de la clé privée en développant en fractions continues E/N.

Une autre attaque possible est l'attaque de Håstad qui ne peut être utilisé que si le message doit être diffusé à plusieurs destinataires en utilisant donc plusieurs clés publiques. En interceptant tous ces messages, il est possible de retrouver le message d'origine à l'aide du théorème des restes chinois (résolution de systèmes de congruences). Cette attaque fonctionne surtout si l'exposant public E est petit.

De plus, le RSA est un chiffrement déterministe (si l'on chiffre plusieurs fois le même message, on obtient le même message chiffré) et donc pas sémantiquement sûr.

VII. Conclusion

Pour conclure, les chiffrements symétriques utilisent différentes méthodes comme la substitution ou la permutation. Cependant, même si ce chiffrement est simple et rapide, ce type de chiffrement n'est pas assez sécurisé, le RSA se base donc sur l'utilisation d'un chiffrement asymétrique, à l'aide d'une clé privée et d'une clé publique. Ces clés sont générées aléatoirement à l'aide de nombres premiers sur lesquels on applique des calculs pour créer des couples. Nous avons réalisé un programme de chiffrement RSA dit naïf : le message est découpé afin de faciliter son traitement, cependant cette méthode ne permet que de présenter et comprendre l'utilisation du RSA, elle ne représente pas son utilisation réelle : un RSA naïf est très facilement attaquable par de nombreuses méthodes.
En pratique, la génération de nombres premiers demande beaucoup de temps, de même que les calculs qui sont faits par la suite : le RSA n'est pas adapté pour chiffrer tous les échanges. Sur Internet, il est couplé avec d'autres techniques pour crypter les clés de chiffrement symétrique, qui serviront ensuite pour les échanges d'information.
Le chiffrement RSA est un donc un des systèmes les plus sécurisés à l'heure actuelle, mais il est cependant attaquable par certaines méthodes :
Notamment, l'attaque de Wiener est assez performante si D est petit (si $D < N^{1/4}$). On peut retrouver dans ce cas l'exposant secret de la clé privée en développant en fractions continues E/N.

Une autre attaque possible est l'attaque de Håstad qui ne peut être utilisé que si le message doit être diffusé à plusieurs destinataires en utilisant donc plusieurs clés publiques. En interceptant tous ces messages, il est possible de retrouver le message d'origine à l'aide du théorème des restes chinois (résolution de systèmes de congruences). Cette attaque fonctionne surtout si l'exposant public E est petit.

VIII. Ouverture

IX. Existence d'un chiffrement parfait ?

On a vu plusieurs types de chiffrements, certains plus efficaces que d'autres mais aucun n'est parfaitement sûr. Alors on peut se demander si un tel chiffrement existe ? Est-ce qu'un chiffrement parfait existe ?

A. Chiffre de Vernam ou masque jetable

Un chiffrement si parfait que toute personne interceptant le message, même en ayant à sa disposition une puissance de calcul immense, soit dans l'impossibilité de retrouver la moindre information du message d'origine sans la bonne clé ?

Ce chiffrement a été inventé par Gilbert Vernam au début du XXe siècle. Cette méthode reprend les principes du chiffrement de Vigenère mais en rajoutant 3 conditions impératives à la clé de chiffrement :

- Elle doit être aussi longue que le texte à chiffrer
- Elle doit être parfaitement aléatoire
- Chaque clé ne doit être utilisée qu'une seule et unique fois, pour chiffrer un seul message, puis est immédiatement détruite (d'où le nom de « masque jetable »).

Ces trois conditions réunies on obtient une méthode de chiffrement parfaitement sûre de telle sorte que si on intercepte le message codé, la seule information dont on dispose est la longueur du message clair.

B. Inconvénients du chiffre de Vernam

Cependant, un tel chiffrement comporte plusieurs inconvénients majeurs qui ne permettent un champ de mise en œuvre que très limité de cette méthode.

Tout d'abord, cette méthode de chiffrement réclame une clé unique à chaque message ce qui entraine rapidement un grand nombre de clés créées. Si on utilise deux fois la même clé, on peut extraire beaucoup d'informations des deux messages chiffrés en les comparant.

Ensuite, les clés utilisées pour un tel chiffrement sont très longues et donc couteuses en temps de calcul.

De plus, comment peut-on être sûr du caractère aléatoire de la création de la clé ? Aujourd'hui, des clés parfaitement aléatoires ne peuvent pas être créées par une machine par un simple calcul. Il existe des algorithme pseudo-aléatoires qui peuvent créer l'illusion d'une clé aléatoire mais ne peuvent pas, en réalité, répondre à la condition du parfait aléa. Si l'on connait l'algorithme et les données initiales du calcul de la machine devant créer la clé aléatoire, il est possible de prévoir la clé.

Enfin, la transmission de la clé est très compliquée. L'échange de clés entre les deux correspondants nécessite un canal de transmission parfaitement sécurisé. Or la clé ayant la

même longueur que le message en lui-même, cette méthode ne semble pas très utile sauf si le canal n'existe qu'au moment de la transmission des clés et plus lors de la transmission du message.

Pour toutes ces raisons, le chiffre de Vernam n'est que très peu utilisé dans l'encodage de messages à l'heure actuelle.

X. Annexe : RSA.py Codage d'un RSA naïf

```
import math
import random
import time

def toNum(msg):#transforme une string en tableau d'entiers et le
retourne
    tabret = []
    for i in range(len(msg)):
        char = msg[i]
        ordc = ord(char)
        tabret.append(ordc)
    return tabret

def toString(num):
    return chr(num)

def rsakeygen(): #génère et retourne les nombres premiers entre
0 et une valeur saisi au clavier
    maxPrime = int(input("Input the max value of  the prime
numbers (Definits the strengh of the encodage) \n"))
    start_time = time.time()
    tab = getNbPremiers(maxPrime)
    (e, d, n) = keyGen(tab)
    print("Clé publique : " + str(n)+ "(n)" + "," + str(e)+"(e)"
+ "\nClé privée : " + str(d)+"(d)")
    elapsed_time = time.time() - start_time
    print("Elapsed time : " + str(elapsed_time))
    return (e, d, n)

def keyGen(tab): #génère et retourne des clés à partir de nombres
premiers choisis aléatoirement dans tab
    i = random.randint(0,len(tab)-1)
    p = tab[i]
    i = random.randint(0,len(tab)-1)
    q = tab[i]
    while(p==q):
        i = random.randint(0,len(tab)-1)
        q = tab[i]
    n = p * q
    phin = (p - 1) * (q - 1)
    e = random.randint(1, len(tab))
    while (pgcd(e, phin) != 1 or e > phin):
        e = random.randint(1, len(tab))
    d = invmod(e, phin)
    return (e, d, n)

def invmod(a, q): #calcule et retourne de l'inverse modulaire
grâce à l'algorithme de Bezout
    g, x, y = bezout(a, q)
```

```
    return x % q

def bezout (a,b): #algorithme de Bezout
    r0=a
    r1=b
    u0=0
    u1=1
    v0=1
    v1=-r0//r1
    s=[r1,u0,v0]
    while r0%r1:
        r=r0
        r0=r1
        r1=r%r1
        u=u0
        v=v0
        u0=u1
        v0=v1
        q=r0//r1
        u1= u-q*u1
        v1=v-q*v1
        s=[r1,u0,v0]
    return s

def pgcd(a, b): #calcul de pgcd
    if a > b:
        dividende = a
        diviseur = b
    else:
        dividende = b
        diviseur = a
    reste = -1
    resteFinal = -1
    while (reste != 0):
        reste = dividende % diviseur
        dividende = diviseur
        diviseur = reste
        if reste != 0:
            resteFinal = reste
    return resteFinal

def getNbPremiers(tailleMax): #retourne une liste des nombres
premiers compris entre 0 et tailleMax
    listNbPremiers = [2]
    nombre = 3
    while nombre < tailleMax:
        premier = True
        i = 0
        while premier == True and i < len(listNbPremiers) and
listNbPremiers[i] <= math.sqrt(nombre):
            diviseur = listNbPremiers[i]
            if nombre % diviseur == 0:
```

```
                premier = False
            i += 1
        if premier:
            listNbPremiers.append(nombre)
        nombre += 1

    return listNbPremiers


maxPrime = int(input("Input the max value of  the prime numbers (Definits the strengh of the encodage) \n"))
start_time = time.time() #début du chrono
tab = getNbPremiers(maxPrime)# génération d'un tableau contenant les nombres premiers de 0 à maxPrime
(e, d, n) = keyGen(tab) # génération de la clé publique et privée à partir de ces nbres premiers
print("Clé publique : " + str(n) + "," + str(e) + "\nClé privée : " + str(d))
elapsed_time = time.time() - start_time #temps passé à générer les clés
print("Elapsed time : " + str(elapsed_time)) # temps passé à générer les clés
msg = input("Enter the message to encode\n")
nombre = toNum(msg) # transforme le message en tableau d'entiers
tabret = []
for i in range(len(nombre)):
        mc = pow(nombre[i], e) % n # cryptage
        tabret.append(mc)
        tab = tabret
        print(tab)
        strret=""
for i in range(len(tab)):
        msgint = pow(tab[i], d, n) # décryptage
        msgs = toString(msgint) #retour sous forme de message
        strret+=msgs
print (strret)
```

XI. Sources

http://www.cryptage.org

https://fr.wikipedia.org

http://bibmath.net

https://cryptobourrin.wordpress.com

http://www.nymphomath.ch

https://openclassrooms.com

« Initiation à la cryptographie » de Gilles Dubertret