Chiffrement RSA

Réalisé par : Aicha BOUICHOU

Plan

Introduction **2**

RSA **2**

Définition 2

Problèmes 2

Process 3

Calcul de la clef publique et privée **3**

Choix de deux nombres premiers 3

Choix d'un exposant et calcul de son inverse 3

Résumé 3

Chiffrement de message **3**

Message 3

Le chiffrement 4

Déchiffrement du message **4**

Déchiffrement 4

Lemme de déchiffrement 4

Implémentation Python **5**

Calcule des cléfs 5

Chiffrement de message 6

Déchiffrement 6

Euclide Étendu 6

Résumé **6**

Conclusion **8**

I. Introduction

La **cryptographie asymétrique** est un domaine de la cryptographie où il existe une distinction entre des données *publiques* et *privées*, en opposition à la cryptographie symétrique où la fonctionnalité est atteinte par la possession d'une donnée secrète commune entre les différents participants ,dont le but est de garantir la **confidentialité** d'une donnée.

Le terme asymétrique s'applique dans le fait qu'il y a deux clefs de chiffrement (que l'utilisateur qui souhaite recevoir des messages fabrique lui-même), telles que si l'utilisateur utilise une première clef dans un algorithme dit « de chiffrement », la donnée devient inintelligible à tous ceux qui ne possèdent pas la deuxième clef, qui peut retrouver le message initial lorsque cette deuxième clef est donnée en entrée d'un algorithme dit « de déchiffrement ».

Par convention, on appelle la clef de déchiffrement la **clé privée** et la clef de chiffrement la **clé publique**.

La clef qui est choisie privée n'est jamais transmise à personne alors que la clef qui est choisie publique est transmissible sans restrictions.

Ce système permet deux choses majeures :

- **Chiffrer le message à envoyer** : l'expéditeur utilise la clef publique du destinataire pour coder son message. Le destinataire utilise sa clef privée pour décoder le message de l'expéditeur, garantissant la **confidentialité** du contenu.

- S'assurer de l'**authenticité de l'expéditeur** : L'expéditeur utilise sa clef privée pour coder un message que le destinataire peut décoder avec la clef publique de l'expéditeur ; c'est le mécanisme utilisé par la **signature numérique** pour **authentifier** l'auteur d'un message.

II. RSA

1. Définition

Le **chiffrement RSA** (nommé par les initiales de ses trois inventeurs) est **un algorithme de cryptographie asymétrique**, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman. RSA a été breveté par le Massachusetts Institute of Technology (MIT) en 1983 aux États-Unis.

2. Problèmes

Le chiffrement RSA se base sur des principes relatives à des outils mathématiques :

- Factorisation d'un entier n, en produit de deux nombres premiers ; le cas des nombres à plusieurs chiffres ,rend ce protocole plus puissant.
- Les clefs seront calculées par l'Algorithme d'Euclide étendu et coefficients de Bézout .
- L'environnement sera dans la plage Z/n Z.
- Le déchiffrement sera fait on se basant sur le petit Théorème de Fermat .

3. Process

Il se base sur trois étapes :

- Alice prépare sa clef privée et sa clef publique .
- Bob utilise la clef publique d'Alice pour crypter son message .
- Alice reçoit le message crypté de Bob et le déchiffre grâce à sa clef privée.

III. Calcul de la clef publique et privée

1. Choix de deux nombres premiers

Alice préparera une clef publique qui sera diffusée à tout le monde et une clef privée qui sera la seule à connaître. Elle va effectuer les étapes suivantes :

1. Choisir p et q, deux nombres premiers distincts ;
2. Calculer leur produit $n = p*q$, appelé ***module de chiffrement ;***
3. Calculer $\varphi(n) = (p - 1)(q - 1)$ (c'est la valeur de l'indicatrice d'Euler en n) ;vue que p et q sont difficile à retrouver d'où $\varphi(n)$ sera encore plus difficile à connaître.

2. Choix d'un exposant et calcul de son inverse

1. Choisir un entier naturel e premier avec $\varphi(n)$ et strictement inférieur à $\varphi(n)$, appelé *exposant de chiffrement* ;
2. Calculer l'entier naturel d, inverse de e modulo $\varphi(n)$, est strictement inférieur à $\varphi(n)$, appelé *exposant de déchiffrement* ; d peut se calculer efficacement **par l'algorithme d'Euclide étendu.**

3. Résumé

La clef publique d'Alice est donc : (n , e)
La clef privée est bien évidement : d

IV. Chiffrement de message

1. Message

Bob veut envoyer un message à Alice , alors qu'il doit effectuer ces tâches :

1.Transformer son message en un ou plusieurs entier m.
2. m doit être 0<=m<n.

2. Le chiffrement

Bob récupère la clef publique d'Alice (n ,e) :

1.Il chiffre son message m : X=pow (m , e) (mod n)
2.Puis il transmet X. « message chiffré »

V. Déchiffrement du message

1. Déchiffrement

Alice va recevoir X , alors elle effectue le traitement suivant :

- Décrypter à l'aide de sa clef privée d : m=pow(X ,d)(mod n).

2. Lemme de déchiffrement

Enoncé :

Soit d l'inverse de e modulo phi(n) avec n=p*q (p !=q) :

Si X=m^e (mod n) , alors m = X^d (mod n).

Preuve :

VI. Implémentation Python

1. Calcul des clefs

```
def cle_privee(p,q,e):
        n=p*q
        phi=(p-1)(q-1)
        c,d,dd=euclide_etendu(e,phi)  /* pgcd et coef de Bézout
        return (d%phi)
```

2. Chiffrement du message

```
def chiffrement-rsa(m,n,e) :
  return pow(m,e,n)
```

3. Déchiffrement

```
def dechiffrement-rsa(x,n,d) :
    Return pow(x,d,n)
```

4. Euclide Étendu

```
def euclide_etendu(a,b):
        x = 0 ; xx = 1
        y = 1 ; yy = 0
        while b !=0 :
              q = a // b
              a , b = b , a % b
              x , xx = xx - q*x , x
              y , yy = yy - q*y , y
        return (a,xx,yy)
```

VII. Résumé

```
def euclide_etendu(a,b):
    x = 0 ; xx = 1
    y = 1 ; yy = 0
    while b !=0 :
        q = a // b
        a , b = b , a % b
        x , xx = xx - q*x , x
        y , yy = yy - q*y , y
    return (a,xx,yy)

def generation_deN(p,q):
    print ("l'environnement est :n=", p*q)
    return p*q

def generationde_phi(p,q):
    print("Le phi est  : ", (p-1)*(q-1))
    return(p-1)*(q-1)

def init_rsa(p,q,e) :
    n= generation_deN(p,q)
```

```
    phi= generationde_phi(p,q)
    c,d,dd = euclide_etendu(e,phi)          # Pgcd et coeff de Bézout
#    print("Coeff. Bézout : d,dd = ", d,dd)
    d = d % phi
    if c != 1 :
        print("Mauvais exposant ! Essayez encore.")
        return None
    else :
        print("Clef publique : n = ", n ,", e = ", e)
        print("Clef privée   : d = ", d)
        print("Outils intermédiaires (à  oublier) : p = ", p,", q = ", q,", phi = ", phi)
        return (d)
def ver_premier(x):
    if (x > 1):
        divisor = 2
        for i in range(divisor,x):
            if (x % i) == 0:
                return False
    else:
        return False
    return True
def chiffrement(m,n,e):
    return pow(m,e,n)
def dechiffrement(x,d,n):
    return pow(x,d,n)



if __name__ == '__main__':
       p= int(input("choisissez un nombre premier :"))
       q = int(input("choisissez un nombre premier :"))
       if ver_premier(p) & ver_premier(q):
               n = generation_deN(p,q)
               phi =generationde_phi(p,q)
               e = int(input("choisissez un exposant premier avec phi :"))
               d=init_rsa(p,q,e)
               print("Le clé privée est : d = ",d,"n = ",n)
               m = int(input("Enter à crypter: "))
               x = chiffrement(m,n,e)
               print("Le message crypté : ",x)
               print("Décryptage ...")
               print("Votre message était :")
               print(dechiffrement(x,d,n))
       else :
           print ("le p ou q ou les deux et nn premier ")
```

la console affichera comme résultat de p=59 , q=61 et e=31 le résultat suivant :

```
runfile('C:/Users/aicha/Desktop/sanstitre14.py', wdir='C:/Users/aicha/Desktop')

choisissez un nombre premier :59

choisissez un nombre premier :61

l'environnement est :n= 3599
```

```
Le phi est  :  3480

choisissez un exposant premier avec phi :31

l'environnement est :n= 3599

Le phi est  :  3480

Clef publique : n =  3599 , e =  31

Clef privée   : d =  3031

Outils intermédiaires (à  oublier) : p =  59 , q =  61 , phi =  3480

Le clé privée est : d =  3031 n =  3599

Enter à crypter: 5

Le message crypté :  615

Décryptage ...

Votre message était : 5
```

VIII. Conclusion

Ce qui fait le succès du RSA est qu'il n'existe pas d'algorithme connu de la communauté scientifique pour réaliser une attaque force brute avec des ordinateurs classiques. Si une personne possède un moyen « rapide » de factoriser le nombre *n*, tous les algorithmes de chiffrement fondés sur ce principe seraient remis en cause ainsi que toutes les données chiffrées dans le passé à l'aide de ces algorithmes.

Les autres types d'attaques , beaucoup plus efficaces, visent la manière dont les nombres premiers *p* et *q* ont été générés, comment *e* a été choisi, si l'on dispose de messages codés ou de toute autre information qui peut être utilisée.

En 1998, Daniel Bleichenbacher décrit la première attaque pratique de type « chiffré choisi adaptable » contre des messages RSA En raison de défauts dans le schéma de remplissage PKCS #1 v1, il fut capable de récupérer des clefs de session SSL. À la suite de ce travail, les cryptographes recommandent maintenant l'utilisation de méthodes de remplissage formellement sûres, telles que OAEP, et les laboratoires RSA ont publié de nouvelles versions de PKCS #1 résistantes à ces attaques.