Kadri Yanis

Implémentation du chiffrement RSA

Cahier des charges

Implémenter le chiffrement RSA (mini)

Chiffrer et déchiffrer un message

Signer un message

Calcul de l'empreinte d'un message

Génération de clés publique et privée pour une personne

Générer un certificat pour un utilisateur donné à l'aide de sa clé publique

Vérification de l'empreinte d'un message reçu

Vérification d'un certificat de l'émetteur du message reçu

Interface textuelle

Effectuer des tests

Résultat de la mission

```
2 nombres probablement premier (p, q) :  27887459035448644865479   152671162227644313980 99
n = p*q =  4257610782517765283735680462965710497513244 21
phi(n) =  4257610782517765283735248917213128366750608 44
clé public d'Alice e =  25071025165130843897208292382077094024757 4685
clé privé d'Alice d =  1292644659982409228396926053195746289455441 33

2 nombres probablement premier (p, q) :  549218338554602489030113   476031561832841498712191
n = p*q =  261445263489385730789524716908301779294119207583
phi(n) =  261445263489385730789523691658401391850131465280
clé public CA e =  14956260608669695179330189625257066575458947 0287
clé privé CA d =  51754972053625470271420971339751436138736575663

Calcul empreinte 'empreinte(X) = X%13' :  9
Clé publique Alice et empreinte m (signature) =  25071025165130843897208292382077094024757 4685 9
Empreinte chiffré par la clé privé d'Alice :  40774469547657089435173067348364706232789253
message chiffré (par clé public CA) et empreinte chiffré (par clé privé d'Alice) :  181797636740027

message déchiffré :  25071025165130843897208292382077094024757 4685 9
Empreinte correcte ! Il y a pas eu de changement lors de l'envoie (intégrité respecté)

Génération certificat :
certificat pour la clé publique d'Alice :  15709743726438176817877063283399973658654734 6721  = cle
51754972053625470271420971339751436138736575663 )

Clé publique Alice :  25071025165130843897208292382077094024757 4685  = certificat ( 157097437264381
14956260608669695179330189625257066575458947 0287 )
Certificat correct ! Le message est bien authentique
```

```python
def creationCle(nbDepart, nbFin):
    #On prend deux nombres p et q
    pq = calculPetQ(nbDepart, nbFin)
    p = pq[0]
    q = pq[1]

    #On calcul n pour les clés
    n = calculN(p, q)

    #On calcul la fonction d'Euler pour avec phi
    phi = euler(p,q)
    print("phi(n) = ", phi)

    #On prend 'e' pour la clé public
    e = calculE(phi)

    #test si e est bien compris entre 1 et phi(n)
    assert(1 < e < phi)
    #Et que est bien premier avec phi(n)
    assert(pgcd(e, phi) == 1)

    #Pour le 'd' de la clé privé, on calcul l'inverse de e%n tel que (e*d) % phi(n) = 1
    d = bezout(e,phi)
    #test si e*d modulo n est bien égale à 1
    assert((e*d)%phi == 1 )

    #On à donc la clé public
    clePublic = e,n
    clePrive = d % euler(p,q), n # permet d'avoir toujours un nb positif car on prend l'invers
    return (clePublic),(clePrive)
```

Méthodes

Ecrire les étapes à faire sur papier

Utilisation des calculs mathématique liées au chiffrement RSA : nombre premier entre eux (PGCD), méthode d'Euler, inverse modulaire et modulo.

Implémentation faites sur python

Compétences acquises

Amélioration de la maîtrise du langage python

Une meilleure compréhension du fonctionnement de la cryptographie asymétrique RSA

Savoir appliquer les connaissances/notions RSA sous forme de code

Technologie (langages et application)

Visual Studio Code

Python