RSA ET OPENSSL

Le terme SSL est un acronyme pour ***Secure Socket Layer*** qui est un protocole (en fait un ensemble de protocoles) qui a été développé par la société Netscape Communication Corporation pour permettre de la communication sécurisée en mode client/serveur pour des applications réseaux utilisant TCP/IP. Le protocole TLS (Transport Layer Security) est une évolution de SSL réalisé par l'IETF et qui sert de base à HTTPS

openSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS qui offre une commande en ligne (openssl) permettant :

- **la création de clés RSA,** DSA (Signature Digitale) ;
- la création de certificats X509 ;
- le calcul d'empreintes (MD5, SHA, RIPEMD160, ...) ;
- **le chiffrement et déchiffrement (RSA**, DES, RC4, AES, ...) ;
- la réalisation de tests de clients et serveurs SSL/TLS ;
- la signature et le chiffrement de courriers (S/MIME) ;

La syntaxe générale de la commande OpenSSL est :

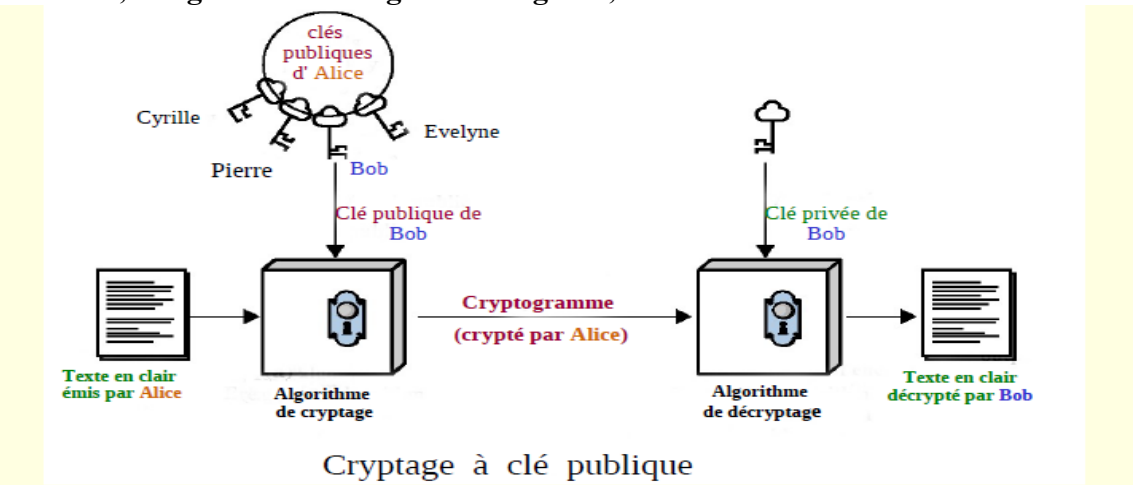
$ openssl <commande> <options>

I. Cryptographie Asymétrique

La cryptographie à clef publique repose sur un schéma asymétrique qui utilise une paire de clefs pour le chiffrement : une clef publique, qui chiffre les données, et une clef privée correspondante, aussi appelée clef secrète, qui sera utilisée pour le déchiffrement.

Parmi les crypto-systèmes à clef publique, on cite:

- **Elgamal (Taher Elgamal);**
- **RSA;**
- **Diffie Hellman;**
- **DSA, l'Algorithme de Signature Digitale;**



Cryptage à clé publique

I.1. Algorithme de création des clefs

- Il faut choisir deux grands nombres premiers distincts **p et q .**
- On calcule **n = pq et φ(n) = (p -1)(q -1)** .
- On choisit un entier **e ∈{2,...,n -1}** premier avec **φ(n).**
- Il faut déterminer **e^{-1} = d ∈ {2,...,n -1 } : de =1 mod φ(n) .**
- La clef privée d'Alice est **d** et sa clef publique est **(n, e) .**

I.2. Algorithme de chiffrement

Lorsque Alice veut envoyer un message confidentiel à Bob :

- Alice code le message en code ASCII de taille trois.
- Il représente le message par un nombre **m∈{1,...,n -1};**
- Il se procure la clef publique **(n, e)** de Bob; il doit s'assurer qu'il s'agit effectivement de la clef publique de Bob (Certificat numérique).
- Il calcule **$c \equiv m^e \bmod n$** qui est un bloc chiffré.
- Il transmit la concaténation des **c à Bob**.

Lorsque **Bob** reçoit le message **c,** il calcule le texte en clair en utilisant sa **clef privée d :**

$$m = c^d \bmod n.$$

Remarques :

- La compréhension de RSA nécessite quelques connaissances mathématiques plus précisément des connaissances sur l'arithmétique ou calculs modulaires.
- Le RSA est encore le système cryptographique à clef publique le plus utilisé dans le monde de nos jours.

I.3. Description du fonctionnement des composantes:

- **Calcul de n et ℓ(n) :** après avoir choisis les deux nombre **p et q** qui sont deux paramètres d'entrée de RSA, l'algorithme procède au calcul de :

  n = p*q et φ(n) = (p-1)(q-1).
- **Génération de e :** pour la génération de la clé publique **e** l'algorithme recherche l'ensemble des nombres qui seront premiers avec **ℓ(n)** et inférieurs à **n**.
- **Génération de d :** la génération de la clé privée **d** consiste à trouver l'inverse de **e** de tel sorte que : **e * d = 1 % n.**
- **Cryptage :** le chiffrement se fait comme suit : **$C = M^e$ %n.**
- **Décryptage : $M = C^d$%n.**

I.4. Etude de RSA

Si une clé **publique e** accepte son propre inversible **d, on aura e = d.** Dans ce cas, la clé est utilisée pour le chiffrement et le déchiffrement du message (chiffrement symétrique).

- On propose **p = 7 et q = 11;**
- On détermine toutes des clés pour que RSA soit symétrique & asymétrique;
- **n=p*q=7*11=77;**
- **φ(n) = (p-1)(q-1)=(7-1)*(11-1)=60;**
- **e∈{2,…,77}\PGCD(e,60)=1;**
- **e∈{7,11,13,17,19,23,29,31,37, 41,43, 47, 49, 51, 53, 57, 59, 61, 71, 73};**

Pour que RSA soit symétrique, on doit vérifier que chaque clé accepte son propre inversible: **x*x=1%60;**

- **RSA symétrique: {11, 19, 29, 31, 41,49, 59, 61, 71};**
- **RSA asymétrique:{7, 13, 17, 23, 37, 43, 47, 53, 67, 73}.**

Cas clé publique e = 7 :

Dans ce cas RSA est asymétrique puisque la clé 7 se trouve dans l'ensemble asymétrique démontré dans la question précédente.

- Calcul de la clé privée d :

$$d \in \{7, 13, 17, 23, 37, 43, 47, 54, 67, 73\} \setminus d \cdot 7 = 1 \,\%\, 60$$

- On trouve alors dans cet ensemble que d = 43.

I.5. Chiffrement avec RSA

Pour chiffrer un message avec RSA on commence tout d'abord par la conversion du message on code ascii de taille 3 le message choisis est « **misr** » :

M=misr= {109, 105, 115, 114} : code ASCII

- Ensuite il faut représenter le message en blocs de même taille que n qui est égale à 77 dans notre cas ; chaque bloc sera de taille 2 :

M = {10,91,05,11,51,14}

- Il faut aussi vérifier que les blocs contiennent des nombres inférieurs à n (77) ce qui nous pousse à réduire la taille de chaque bloc on obtient alors :

M={1,0,9,1,0,5,1,1,5 ,1,1,4}

- Maintenant on procède au calcul de C (message chiffré) :

$$C = \{1^7 \,\%\, 77, 0^7 \%\, 77, 9^7 \%\, 77, 1^7 \%\, 77, 0^7 \%\, 77, 5^7 \%\, 77, 1^7 \%\, 77, 1^7 \%\, 77, 5^7 \%\, 77, 1^7 \%\, 77, 1^7 \%\, 77, 4^7 \%\, 77\}$$

$$C = \{1, 0, 37, 1, 1, 0, 47, 1, 1, 47, 1, 1, 60\}$$

I.6. Déchiffrement du message

- Pour le déchiffrement du message on aura :

$$M = \{1^{43} \%\, 77, 0^{43} \%\, 77, 37^{43} \%\, 77, 1^{43} \%\, 77, 1^{43} \%\, 77, 0^{43} \%\, 77, 47^{43} \%\, 77, 1^{43} \%\, 77, 1^{43} \%\, 77, 47^{43} \%\, 77, 1^{43} \%\, 77, 1^{43} \%\, 77, 60^{43} \%\, 77\}$$

$$M = \{1, 0, 9, 1, 0, 5, 1, 1, 5, 1, 1, 4\}$$

- Ensuite on reprend la représentation en bloc de code ascii de taille 3 on obtient :

M = {109, 105, 115, 114} = misr

I.7. Refaire les exercices I.4, I.5 et I.6 en proposant p = 5 et q = 11;

II. Chiffrement et déchiffrement sous OpenSSL

Le message qu'on veut crypter « asimi younes », pour ce faire, on va créer un fichier **nommé « plain.txt ».**

II.1. Création d'une clef privée

La première tache à faire, c'est de créer votre clé privée à partir de laquelle on va générer tous les autres paramètres de RSA.

```
root@asimi-HP-620:/home/asimi# openssl genrsa -out rsa_privkey.pem 1024
Generating RSA private key, 1024 bit long modulus
...................++++++
....++++++
e is 65537 (0x10001)
```

Cette commande permet de générer une clef privée d'une taille de 1024 en utilisant l'algorithme de chiffrement asymétrique RSA.

Pour générer une paire de clés de 1024 bits, stockée dans le fichier **maPrivCle.pem,** on tape la commande :

$ openssl genrsa -out maPrivCle . pem 1024

Elle génère une clé privée RSA de taille 1024. Les valeurs possibles pour size sont : 512 bits, 1024 bits, 2048 bits etc. Pour un usage sensible, il est recommandé de choisir une clé de 2048 bits, voire 4096 bits.

Pour afficher cette clef sous le terminal, on exécute la commande suivante :

```
root@asimi-HP-620:/home/asimi# cat rsa_privkey.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDWiODpHQLdqcC7OnH+JRdtO0JHf0He+KB6ppfXfbmvwCoQdD53
yM2s6/3fKpN5bbidxXhC7E2EKQR+ttnZFiMMfIiAUlhSVY5nseLf0XL+OVmvFuyt
VqdA18NiZMDfh8ewuDTN4oJpR/Gq4rOxXiGKDd8eVRpL6FQSBzJNH8PgZwIDAQAB
AoGAaLIuCLWmw2RncTepj9ysHRuRUfERxW3Sa2jHq1to9hMytAr2vWBat2wH3hQa
GmVodl57PTWy8q9NyUUU9XNvMHEDhFuSHhYLorc1WHsI84vNojbRDYHu6gOaP5w+
GE4m4NlwAWKjBpUKpikexn0mDnXbRUTQzWTL38Rct1ihJkECQQDunsHLu/Vi5IN9
kPsA+1KHpqMzyyhBp15a2I/GqQaAaYMB0gkTwCef3tFhWM/7eanqPEeCEoWBHbMV
QTeb4dMHAkEA5ikH6HmYaQI8fAmaFWWJUV3lpbZQ+XYn7cBJUx6jlRkDiRdGKs/7
ZQlCSGWygUy33tmF8NBJsS9/rC5JcalPoQJARG+rzS6/FfnL7HtkQcfSx6z3Kaee
iBXkbdGmiqd3geTA+givKcb2lldlXbIHFGy0WsWnDJsbRwipj/MAa+i8iQJBALbT
oweFB49qildRXUdf15t9xf1DfMEcnY0bcGOJdBHsw8UzarZ5bc4Nc8yElC5nnat1
CROIsPEBtR41yntyLeECQQCn3V3LL17gjeU6Sp821UMurRVQv+KjyRbVfv7tjJIG
qf0wK1368jREosKKpVa6QDNblRkXCdMTpnrtxWfWLo46
```

II.2. Génération d'une clef publique

A partir de cette clef privée qui nous venons de créer, on va générer la clé publique par la commande suivante :

```
root@asimi-HP-620:/home/asimi# openssl rsa -in rsa_privkey.pem -pubout -out rsa_
pubkey.pem
writing RSA key
root@asimi-HP-620:/home/asimi# cat rsa_pubkey.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDWiODpHQLdqcC7OnH+JRdtO0JH
f0He+KB6ppfXfbmvwCoQdD53yM2s6/3fKpN5bbidxXhC7E2EKQR+ttnZFiMMfIiA
UlhSVY5nseLf0XL+OVmvFuytVqdA18NiZMDfh8ewuDTN4oJpR/Gq4rOxXiGKDd8e
VRpL6FQSBzJNH8PgZwIDAQAB
-----END PUBLIC KEY-----
root@asimi-HP-620:/home/asimi#
```

La commande suivante permet la création d'une clé publique associée à la clef privée créée :

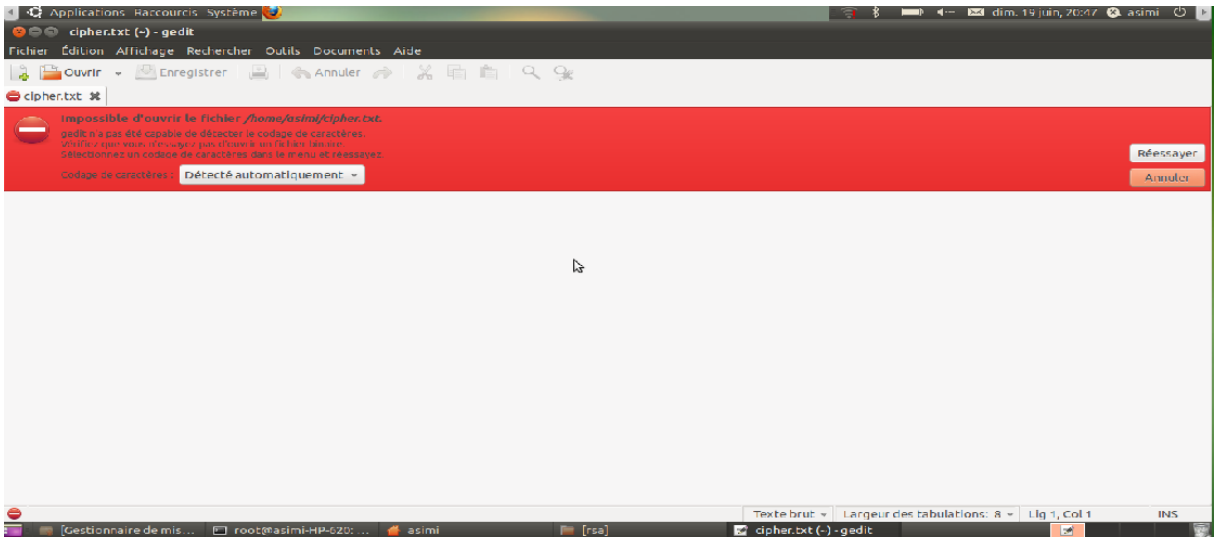
$ openssl rsa -in <fichier_priv_rsa.pem> -pubout -out <fichier_pub_rsa.pem>

II.3. Chiffrement du message

Pour chiffrer le message on va utiliser la clef publique. On va mettre le résultat de cette commande dans un autre fichier nommé « ciphir.txt ».

```
root@asimi-HP-620:/home/asimi# openssl rsautl -encrypt -pubin -inkey rsa_pubkey.
pem -in plain.txt -out cipher.txt
root@asimi-HP-620:/home/asimi#
```

Le résultat :



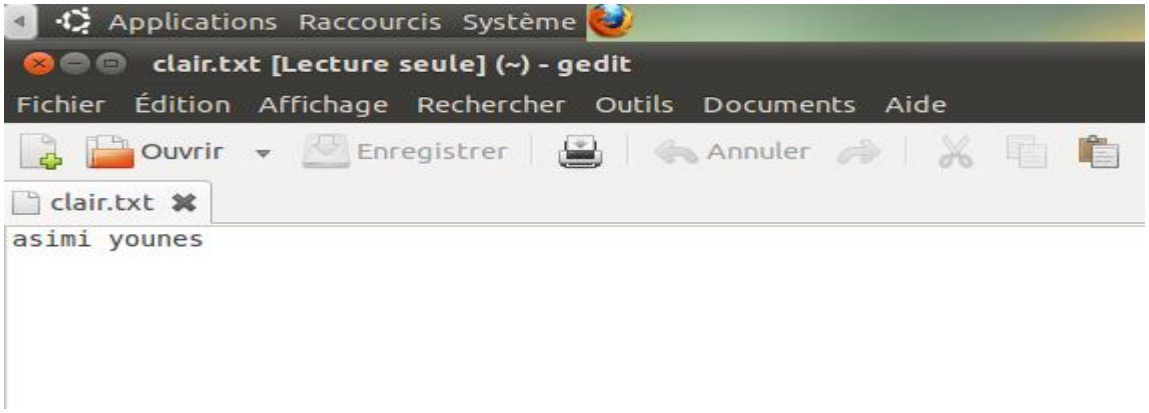
On ne peut pas accéder au message car le fichier chiffré.

II.4. Déchiffrement du message

Comment RSA est asymétrique, on va utiliser la clé privée pour déchiffrer le message crypté. On va mettre le résultat de déchiffrement dans un fichier nommé« clair.txt ».

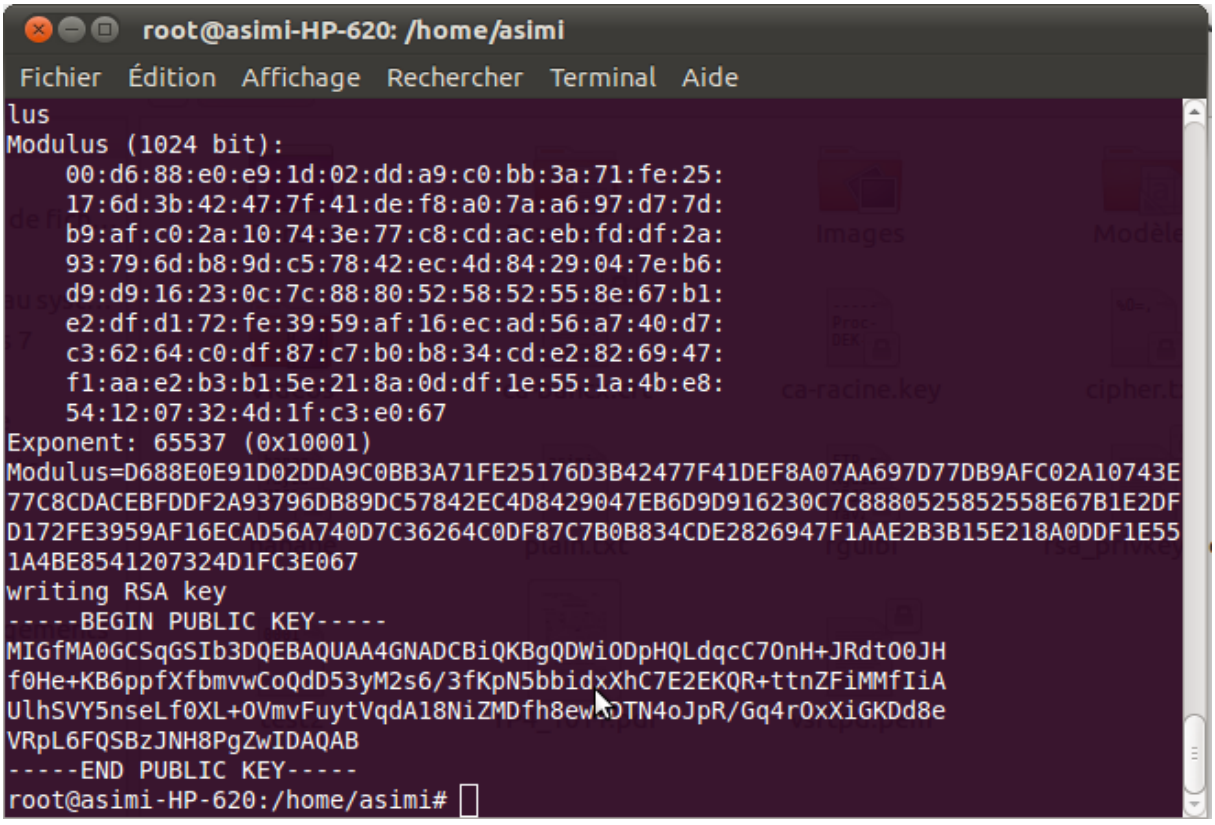
```
root@asimi-HP-620:/home/asimi# openssl rsautl -decrypt -inkey rsa_privkey.pem -i
n cipher.txt -out clair.txt
root@asimi-HP-620:/home/asimi#
```

Le résultat :



Le modulus

On sait que le modulo n'autre que le produit scalaire de deux nombre premier p et q. Donc, pour le voir, on doit exécuter la commande suivante :



Bonne chance