ING3 Informatique - Ecole Sup Galilée

TP2 - RSA

Sécurité et Cryptographie

OUEDRAOGO Wendlasida Tertius
PETITBOULANGER--MELEUX Erwan
RABEMANANTSOA Telina

1- Génération d’une paire de clefs RSA

Question 1

```
openssl genrsa -out key.pem 2048
```

```
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.................................+++++
e is 65537 (0x010001)
```

2 Visualisation des clés RSA

Question 2

```
cat key.pem
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAvD8Zg6jR00sP1KdzJ08k3odE0ttECxJQWd+sYkWYlhwb330X
skpYfOrCc1ypaAJss05jgVkHYvPmSLiQ4F7JRnPieENE+geaeGuzRmzXdBNi0AzD
8Ztypq1RYvSIZA+HWdcNA6meD9VqdrbaKgC8cxVhFzz4qq6tqIBH5Aj1Bde6+SqU
OPgpoMSUdH8eynhPiX1DpWSTIiFx4GB9NQXSTI0KmBLr3adehita2UOGmUU0WRlr
Wi1CIuIAogMvD0H/g5G7okybpaeRBCvDpTTcugH+Ft0f6qTkTDV0xPn42ZC+0s3K
olojblkwWVBejGR/bsoFYkXwniz0FiCK8Paj8wIDAQABAoIBAE+0k/QINhe175UF
9Z8ORrkmI0D4cD4T2gvDxjSx7SKZ8j9VswPdsMvrhrvb3yrX4/ij10P8U0FVrgra
gxVNbQG0rY5oRTtsnr4gM/88tJlEYBFhZlHVlNz4A77XdSyQ2qlygeMsHW3s2jA7
yfIA78n2om55dXyni4YOEuJ93GlFLam0Ej6uHWHxquy6lS1vlJQSnFHFu0rwaFRr
1C3UNBS1eH/tfHspZycTssiOXohDEPiilqnW4Dn/BBl/JKzkeMop8LbBhX/b3j9Q
MHxW8zJKnDsC6QJItNM3IoXLW2iqZBPqPTaQgXqIrjLfrFU9sWqDmh2rPypZNgXy
ROCNtYkCgYEA5MOBb/29cFD4DY34Al8VIdX/QcSKQr7B9m4w6sHf8phCIDsdAo9j
Ukw0TUxHwu6j7dIP/8q8lBQAn9BGHBkdpieINWl+WCWEcOmgEn/yCPXcDysFioC+
1uI1pdxJpmZBPRQG5pZHHbLxzwyCXlAZ9Mpn+CmfiVNuJXhllWair40CgYEA0qir
QTFTuXyIf7gnyx1N9ai5aDbCvftLAhqWczACkYcXb1iKq3FADNluIOT0+5Ope+2C
Vje39Loh1hcAdxYNRQed2pxiW/Q4EQJOXMEmhwj2oQiKohHARdHhIF8LZ3TyAhae
```

```
openssl rsa -in key.pem -text -noout
```

```
RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:bc:3f:19:83:a8:d1:d3:4b:0f:d4:a7:73:27:4f:
    24:de:87:44:d2:db:44:0b:12:50:59:df:ac:62:45:
    98:96:1c:1b:df:7d:17:b2:4a:58:7c:ea:c2:73:5c:
    a9:68:02:6c:b3:4e:63:81:59:07:62:f3:e6:48:b8:
    90:e0:5e:c9:46:73:e2:78:43:44:fa:07:9a:78:6b:
    b3:46:6c:d7:74:13:62:d0:0c:c3:f1:9b:72:a6:ad:
    51:62:f4:88:64:0f:87:59:d7:0d:03:a9:9e:0f:d5:
    6a:76:b6:da:2a:00:bc:73:15:61:17:3c:f8:aa:ae:
    ad:a8:80:47:e4:08:f5:05:d7:ba:f9:2a:94:38:f8:
    29:a0:c4:94:74:7f:1e:ca:78:4f:89:7d:43:a5:64:
    93:22:21:71:e0:60:7d:35:05:d2:4c:8d:0a:98:12:
    eb:dd:a7:5e:86:2b:5a:d9:43:86:99:45:34:59:19:
    6b:5a:2d:42:22:e2:00:a2:03:2f:0f:41:ff:83:91:
    bb:a2:4c:9b:a5:a7:91:04:2b:c3:a5:34:dc:ba:01:
    fe:16:dd:1f:ea:a4:e4:4c:35:74:c4:f9:f8:d9:90:
    be:d2:cd:ca:a2:5a:23:6e:59:30:59:50:5e:8c:64:
    7f:6e:ca:05:62:45:f0:9e:2c:f4:16:20:8a:f0:f6:
```

La commande cat affiche le fichier key.pem tel quel, c'est à dire la clé encodée. La commande rsa de son côté permet d'afficher des manière décodée la clé présente dans le fichier key.pem. A part l'exposant public, les différents éléments de la clé sont affichés en hexadécimal avec la commande rsa.

Question 3

```
publicExponent: 65537 (0x10001)
```

Tout le monde à le même exposant de chiffrement (65537), qui est l'exposant public par défaut. Ce choix à été fait pour des raisons de compatibilité et sécurité : il est premier sans être trop élevé et est assez large pour éviter des attaques par rapport aux plus petits exposants (ex : 3). De plus, openssl n'autorise que deux exposants publics : 65537 (valeur par défaut) ou 3 obtenu avec l'option -3.

Question 4

```
openssl rsa -in key.pem -pubout -out publicKey.pem
```

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvD8Zg6jR00sP1KdzJ08k
3odE0ttECxJQWd+sYkWYlhwb330XskpYfOrCc1ypaAJss05jgVkHYvPmSLiQ4F7J
RnPieENE+geaeGuzRmzXdBNi0AzD8Ztypq1RYvSIZA+HWdcNA6meD9VqdrbaKgC8
cxVhFzz4qq6tqIBH5Aj1Bde6+SqUOPgpoMSUdH8eynhPiX1DpWSTIiFx4GB9NQXS
TI0KmBLr3adehita2UOGmUU0WRlrWi1CIuIAogMvD0H/g5G7okybpaeRBCvDpTTc
ugH+Ft0f6qTkTDV0xPn42ZC+0s3KolojblkwWVBejGR/bsoFYkXwniz0FiCK8Paj
8wIDAQAB
-----END PUBLIC KEY-----
```

3- Chiffrement d'un fichier de clés RSA

Question 5

On crée la bi-clé à partir de la commande :

```
openssl genrsa -out fichier.pem 2048
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAu/jsWEDeYTZLBWGKmULGoMZSb5PDUH47zTpx40KMR6KHkPZE
jeSz7uo0YUgZcTFG18mVn9COT2lG8epR/skgOmGROXtjVxBFawo5apqHO8qY81g2
aoSdKq4NblN+uE8mlMt+m5asdRpWkvQ94elCeMAZx8hCdGSdxN0RvA9teE075fvi
```

On la chiffre en utilisant l'algorithme des3 :

```
openssl rsa -in fichier.pem -des3 -out fichierEncrypt.pem
```

Cela affiche :

```
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

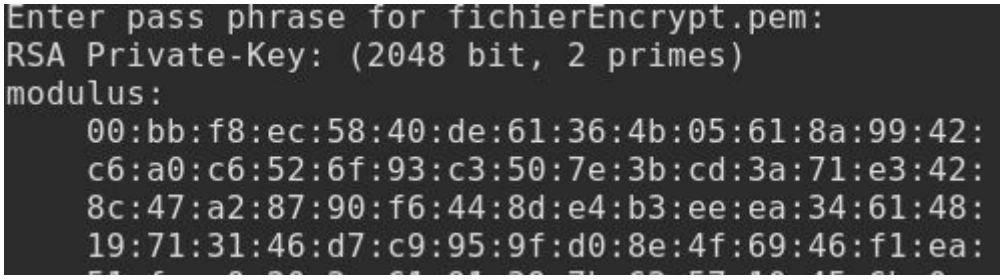
On a mis la pass phrase : pass. Puis on affiche le contenu du fichier fichierEncrypt.pem à l'aide de la commande cat :

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,67EC8454D27B3FC4

rGmAF7CJOAwMvAYWvBUCvSmLxhX8HO3NR0Znv6nl4wOSigfGKuRPehBT4aZXVGen
```

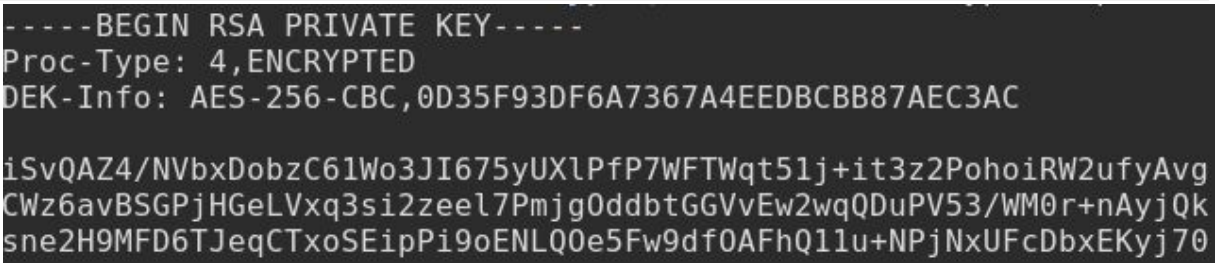
En utilisant :

```
openssl rsa -in fichierEncrypt.pem -text -noout
```

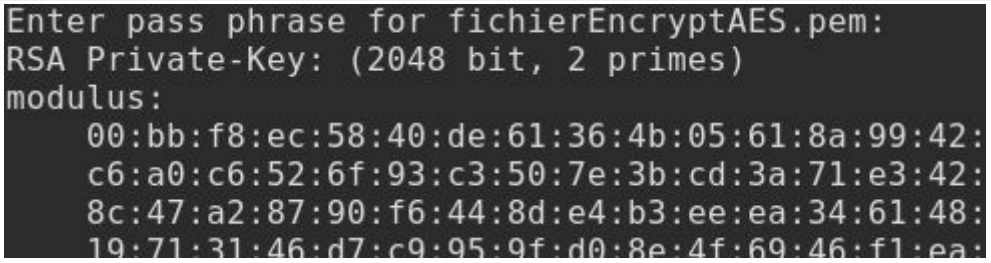


Si on utilise aes-256-cbc comme algorithme symétrique :

```
openssl rsa -in fichier.pem -aes-256-cbc -out fichierEncryptAES.pem
```




```
openssl rsa -in fichierEncryptAES.pem -text -noout
```



On constate que les fichiers contenant la clé privée encodée ont un contenu différent puisque un encryptage à été appliqué, mais une fois décodé on retrouve le même contenu quel que soit l'encryptage.

4- Chiffrement, déchiffrement avec RSA

Question 6

Dans cette partie, on essaie de crypter un fichier en entrée à l'aide du fichier qui contient la clé, puis de générer un fichier de sortie.

On écrit un petit texte dans le fichier “ficEntree.txt” comme suit :

```
11607017@g212-7:~/Documents/crypto$ cat ficEntree.txt
Ici le texte a crypter
11607017@g212-7:~/Documents/crypto$
```

On encrypte :

```
openssl rsautl -encrypt -in ficEntree.txt -inkey fichier.pem -out
ficSortie.txt
```

Pour encrypter avec uniquement la clé publique :

```
openssl rsa -in fichier.pem -pubout -out publicKey.pem
openssl rsautl -encrypt -in ficEntree.txt -inkey publicKey.pem -out
ficSortie.txt -pubin
```

Pour l’opération inverse, on va décrypter le fichier “ficSortie.txt” à l’aide du même fichier clé pour ensuite voir si le texte est bien celui que nous avions au départ et le mettre dans un autre fichier de sortie “sortie.txt”.

```
openssl rsautl -decrypt -in ficSortie.txt -inkey fichier.pem -out
sortie.txt
```

On peut constater à l’aide de la commande cat que le texte dans le fichier “sortie.txt” est le même que celui dans le fichier de départ “ficEntree.txt”

```
11607017@g212-7:~/Documents/crypto$ cat sortie2.txt
Ici le texte a crypter
11607017@g212-7:~/Documents/crypto$
```

5- Signature avec RSA

Question 7

Dans cette partie, on voudrait signer un fichier en entrée à l’aide de la commande suivante :

```
openssl rsautl -sign -in ficEntree.txt -inkey fichier.pem -out signature
```

Dans le fichier “ficEntree.txt”, on ajoute le texte suivant:

```
11607017@f207-2:~/Documents/crypto$ cat ficEntree.txt
Ici le texte a signer
```

Dans le fichier signature, on obtient fichier signé :

```
11607017@g212-7:~/Documents/crypto$ cat signature
}
```

Pour vérifier que le texte est bien signé, on utilise la commande suivante :

```
openssl rsautl -verify -in signature -pubin -inkey fichier.pem -out signe
```

Dans le fichier “signe”, on peut voir qu’on a réussi la signature car le texte contenu dans ce fichier est le même que celui dans le fichier “ficEntree”.

6- Empreinte d’un document

Question 8

On génère l'empreinte du fichier :

```
openssl dgst -md5 -out empreinte grosFichier.zip
```

On regarde son contenu :

```
cat empreinte
```

```
MD5(grosFichier.zip)= 3e7391d9e6133e463e25164a77c7dc96
```

On signe le fichier en signant son empreinte :

```
openssl rsautl -sign -in empreinte -inkey fichier.pem -out signature
```

On vérifie la signature :

```
openssl rsautl -verify -in signature -pubin -inkey publicKey.pem  -out
empreinteOut
```

```
cat empreinteOut
```

```
MD5(grosFichier.zip)= 3e7391d9e6133e463e25164a77c7dc96
```

On retrouve bien l'empreinte originale.