♦◊♦♦◊♦♦◊♦◊♦

UFR Sciences et Technologies

Semestre 2 **Licence 3 en Ingénierie Informatique** Cours de Sécurité des réseaux

TP1 : Chiffrement Asymétrique

But du TP

- Utiliser openssl pour chiffrer/déchiffrer
- Utiliser openssl pour générer des signatures
-

Exercice 1: Chiffrement asymétrique avec openssl

Lab1: Génération de clefs RSA

Q 1.Trouvez comment générer une paire de clefs RSA.

Commande `genrsa` :

```
openssl genrsa –out <fichier> <taille>
```

où fichier est un nom de fichier de sauvegarde de la clef, et taille et la taille souhaitée (exprimée en bits) du modulus de la clef.

```
$ openssl genrsa –out rsakey  128
```

Q 2.Générez une paire de clefs de 1024 bits et stockez-la dans un fichier nommé maCle.pem. Par exemple, pour générer une paire de clefs de 1024 bits, stockée dans le fichier maCle.pem :

```
$ openssl genrsa –out maCle.pem 1024
```

Q 3. Affichez le fichier obtenu lors de la question précédente.

Le fichier obtenu est un fichier au format PEM (Privacy Enhanced Mail, format en base 64), dont voici un exemple :

```
$ cat maCle.pem
```

-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC13goltz++HKd/mO/8KfT6MstT6zBw4ArZlT3eoS1FkFL6VUmj
jf9hqtV0yhKxhUT1Uba5/5wz97DmtdKSAp7Z1V0gWNtFYn1gRaNLKKaN/KnofNPa
+bD7xxl0bT3OD2rFnk9uLyuSDsEV6WkoqkIbuFtq8uSugOSv9KG4Y0e3QwIDAQAB
AoGAOn6zNtRM3UHsId83+RwpnjuuKEyyJOv6sWcOIkgm/oPgHIJIECIA7RH8W/cO
sihSOLFSm1p6jEBzlpEOrEXJf24RGIZJSz35HvEzi8odZronFt/eFtfnlwJyN5yP
rLQwAKoLQlP3LPoheQ39aInPYdKSfvaE9TQs29MKtnR+l+ECQQDZrAuux5t7nnOr
K1PI9/oXtiP01Z3LrwnFUpG0I8athL0G2g8WvU1CTrk2UUkGyskALhptlllIQC5I
xVgzXJ8TAkEA1eQId2L8rW8F7vpwGNnpiFzR3eGqckrgNuIgLan0GSFsYoIVIYVk
o3b5kXVuPXBO0k8Iu5HitNi6p+hCavQdEQJBAIr3O74p0Snqzw3Ia4UyS4g30FzO
xB71cNd5D2xiKsU8qTFmpZtu35QljjkPS//6Frj2tgigo/9seiAbXWPtDeECQQDG
ywTvaSntS8XZAhyTjF0X5yULMwVG6r6PZAWCS3ZasQzaVYqsOB18LETyY5FNOweX
Qeuc4Y8yhEifbCfm3fHxAkEAm+t9F2wiAViVqu6PcKGejSyaObGkCmqHORcmKhx4
xFTBT1RzaooMk6wvUg0sg3p6VWbJauKuxXosFUC/ttHavA==
-----END RSA PRIVATE KEY-----

Q 4. Trouvez la commande et les options permettant de visualiser correctement le contenu d'un fichier au format PEM contenant une paire de clefs RSA.

La commande rsa permet de visualiser le contenu d'un fichier au format PEM contenant une paire de clefs RSA.

```
$openssl rsa –in <fichier> –text –noout
```

L'option -text demande l'affichage décodé de la paire de clefs. L'option -noout supprime la sortie normalement produite par la commande rsa.

Par exemple

Private-Key: (1024 bit) modulus:
00:de:84:ad:1f:6d:2f:bb:66:44:25:17:08:68:1c: ea:ec:1f:8f:05:c9:41:ef:bd:2d:e5:82:50:9e:c9: 74:08:3d:1d:f9:bd:f7:0c:68:37:e0:6f:d5:8b:ad:
98:c8:f7:f2:9a:2f:fb:98:22:8c:27:e3:6f:59:d7: 75:0e:81:9a:23:34:40:55:8d:04:2d:90:fe:d3:4b: a3:d1:e5:22:2c:26:b1:0a:07:43:10:f1:6a:93:f6:
57:21:18:97:d1:33:cd:c1:fd:f2:3c:ec:21:a0:56:e5:16:bc:8b:57:1e:f9:62:0e:b1:24:74:5b:12:0c:
82:96:9e:6f:da:86:eb:e6:2b
publicExponent: 65537 (0x10001)
privateExponent:
7d:8e:f7:f1:27:b7:2b:53:45:58:78:6a:b1:f4:bf: 4a:85:74:3a:9b:99:ad:21:5e:68:86:e3:de:f5:65: 97:d0:84:bd:8b:47:7f:a4:bf:02:d1:97:f9:bc:f6:
dd:4f:ab:80:fd:5a:46:3a:18:43:d9:bd:d7:40:10: 39:23:40:49:57:46:a5:5c:5f:81:dc:5a:10:b7:b2: 8d:0e:cc:c0:b0:fa:ac:3b:a5:5c:61:7f:ce:a8:d8:
54:9e:a2:45:2b:5d:0d:5f:fd:33:69:a7:30:8d:cc: 70:0f:6c:29:a5:cf:c3:b0:19:8e:dc:48:f2:76:b1: 01:e7:02:8b:f0:30:bf:21
prime1:
00:fc:87:81:3d:34:a7:df:43:b0:c7:61:97:fd:96: cf:2f:bc:14:00:b6:86:8d:2f:18:c6:ad:b3:a1:54: af:e2:63:26:a5:70:fa:e6:43:a6:42:be:2a:8f:69:
c1:56:b1:d4:68:80:6f:73:d3:8f:38:71:84:01:b8: 58:03:a4:8f:07
prime2:

00:e1:93:94:bd:a7:45:b3:c0:bf:a6:1a:0a:b5:d3: c0:00:8a:d0:7d:1b:fa:5a:4a:44:53:95:65:aa:0d: f7:29:5c:42:09:43:9f:67:59:0b:e7:97:47:40:75:
2c:fc:be:55:f6:d8:2f:ef:8e:6a:28:95:0a:cd:a4: a1:cf:dd:c2:bd
exponent1:
54:8a:38:a5:f8:de:ca:4b:aa:fe:d4:99:41:78:1f: 5c:67:a6:7a:a6:a0:5c:db:8b:7b:d7:e2:ee:fb:9e: f6:37:23:54:f7:81:c7:5c:96:68:79:a9:5d:e1:95:
ac:24:54:6b:b7:b3:98:1f:17:2f:5a:31:4b:32:1f: a4:f8:8e:39
exponent2:
00:a9:3a:5c:66:03:6d:69:32:fb:14:13:89:61:6b: 60:29:87:fa:6d:41:66:0a:02:99:4b:d3:52:97:c7:
2e:5b:5b:19:37:76:01:ca:38:a5:93:b2:8c:03:b1: 64:74:a0:1e:41:b7:62:0e:e7:da:80:63:7f:dd:52: db:09:e4:a6:49
coefficient:
00:e7:66:54:54:87:d9:9b:b9:5b:fe:cd:6e:f9:6e: 9f:b0:cd:1d:47:dc:f3:9e:48:c4:d7:78:7e:d1:21: 4b:ce:49:63:5d:0b:8e:0c:b3:d4:47:fb:8d:4e:21:
2e:59:ae:16:32:ae:1d:5b:24:55:7c:ea:64:e4:6d: 31:a6:30:b3:df

Q 5. Détaillez les différents éléments de l'affichage obtenu.
Les différents éléments de la clef sont affichés en hexadécimal (hormis l'exposant public). On peut distinguer le modulus, l'exposant public (qui par défaut est toujours 65537), l'exposant privé, les nombres premiers facteurs du modulus, plus trois autres nombres qui servent à optimiser l'algorithme de déchiffrement.

Exercice 2: Chiffrement d'un fichier avec des clefs RSA

Il n'est bien évidemment pas prudent de laisser une paire de clef en clair (surtout la partie privée).

Lab2

Q 1. Comment utiliser **openssl** pour chiffrer une paire de clefs en utilisant un algorithme de chiffrement symétrique ?
Avec la commande rsa, il est possible de chiffrer une paire de clefs. Pour cela trois options sont possibles qui précisent l'algorithme de chiffrement symétrique à utiliser : `-des, -des3 et -idea.`

```
$ openssl rsa -in maCle.pem -des3 -out maCleChiffre.pem
writing RSA key Enter PEM pass phrase : Verifying - Enter PEM pass phrase :
```

Un mot de passe est demandé deux fois pour générer une clef symétrique protégeant l'accès à la clef RSA.

Q 2. Avec la commande `cat` observez le contenu du fichier maCle.pem. Affichez ensuite le contenu de la clef
Utilisez à nouveau la commande rsa pour visualiser le contenu de la clef.

```
$ openssl rsa -in maCle.pem -text -noout
```

Exercice 3 : Diffusion des clefs RSA

La partie publique d'une paire de clefs RSA est publique, et à ce titre peut être communiquée à n'importe qui. Même s'il est chiffré, le fichier maCle.pem contient la partie privée de la clef, et ne peut donc pas être communiqué tel quel.

Lab3

Q 1. Comment extraire la partie publique d'une clef RSA? Stockez-la dans un fichier nommé maClePublique.pem.
Avec l'option -pubout on peut exporter la partie publique d'une clef.

```
$ openssl rsa -in maCle.pem -pubout -out maClePublique.pem
```

Notez le contenu du fichier maClePublique.pem et particulièrement les marqueurs de début et de fin.

Q 2. Visualisez la clef publique. Que remarquez-vous ?
Attention vous devez préciser l'option –pubin, puisque seule la partie publique figure dans le fichier maClePublique.pem.

```
$openssl rsa -in pubkey.pem -pubin -text
$openssl rsa -in pubkey.pem -pubin -text -noout //visualiser en mode décodé
```

Exercice 4: Chiffrement/déchiffrement de données avec RSA

Nous pouvons maintenant chiffrer des données avec une clef RSA. RSA étant un système imposant des calculs lourds nécessitant beaucoup de ressources, il est généralement utilisé pour chiffrer la clef utilisée lors d'un chiffrement symétrique, beaucoup rapide à exécuter.

Lab4

Q 1.Chiffrez le fichier de votre choix avec le système symétrique de votre choix. Chiffrez la clef ou le mot de passe utilisé(e) avec la clef publique de votre destinataire (demandez-lui sa clef publique si vous ne l'avez pas). Envoyez-lui le mot de passe chiffré ainsi que le fichier chiffré.
Pour cela on utilise la commande `rsautl` :

```
$ openssl rsautl – encrypt – in fichierentree –inkey cle –out fichiersortie
   – fichierentree est le fichier à chiffrer. Sa taille ne doit pas dépasser 116
      octets pour une clef de 1024 bits).
   – cle est le fichier contenant la clef RSA. Si ce fichier ne contient que la
      parte publique de la clef, il faut rajouter l'option –pubin.
   – fichier_sortie est le fichier de données chiffré.
$ echo "contenu du  fichier à chiffrer avec rsa" > fichier1
$ openssl enc -bf-cbc -in fichier1 -out fichier1chiffré // le mot de passé
ayant servi à générer la clé de chiffrement de fichier1 est responsable
$ echo "responsable" > clésymétrique
```

Génération de la paire de clé RSA stockée dans cléPubPriv.pem

```
$ openssl genrsa -out cléPubPriv.pem 1024
```

Nous pouvons visualiser le contenu du fichier (cléPubPriv.pem) au format PEM contenant la paire de clefs RSA

```
$ openssl rsa -in cléPubPriv.pem -text -noout
```

Nous allons extraire la partie publique de la paire de clefs RSA et la sotocker dans le fichier *cléPub.pem*

```
$ openssl rsa -in cléPubPriv.pem -out cléPub.pem -pubout
```

Visualisation de la clé publique

```
$ openssl rsa -in cléPub.pem -pubin –text
```

Visualisation de la clé publique avec affichage décodé

```
$ openssl rsa -in cléPub.pem -pubin –text -noout
```

Chiffrement du mot de passe "responsable" avec la commande openssl `rsautl`

```
$ openssl rsautl -encrypt -in  clésymétrique  -pubin -inkey cléPub.pem -out
clésymétriquechiffré
```

ou

```
$ openssl rsautl -encrypt -inkey -pubin PubKey.pem -in  SymtricKey  -out
SymtricKeyChiffre1
```

Déchiffrement du mot de passe "responsable" avec la clé privée

```
$ openssl rsautl -decrypt -in clésymétriquechiffré -inkey cléPubPriv.pem -
out clésymétriquedechiffré
```

Déchiffrement du fichier1chiffré avec cbc contenant le text clair.

```
$ openssl enc -bf-cbc -d -in fichier1chiffré -out fichier1dechiffré// vous
devez taper le mot de passé "reponsable"
Visualiser le texte Claire
$cat fichier1dechiffre
```

Exercice 7 : Signatures et empreintes

Signature d'un fichier: il n'est possible de signer que de petits documents. Pour signer un gros document on calcule d'abord une empreinte de ce document. Signer un document revient alors à signer son empreinte.

Q 1. Quelle commande permet de calculer une empreinte ?

La commande dgst permet de le faire.

```
$ openssl dgst < hachage > –out < empreinte >< fichierentree > où hachage est
une fonction de hachage. Avec openssl, plusieurs fonctions de hachage sont
proposées dont : – MD5 (option -md5), qui calcule des empreintes de 128 bits,
   – SHA1 (option -sha1), qui calcule des empreintes de 160 bits,
   – RIPEMD160 (option -ripemd160), qui calcule des empreintes de 160 bits.
```

Exemple

```
$openssl dgst -md5  -out  emp1.dgst fichier1
$ openssl dgst -sha1  -out  emp1.dgst fichier1
```

Q 2. Comment signer cette empreinte ?

Pour cela, on utilise l'option –`sign` de la commande `rsautl` :

```
$ openssl rsautl -sign -in <empreinte> -inkey <cle> -out <signature>
exemple:
$ openssl rsautl -sign -in emp1.dgst -inkey cléPubPriv.pem -out signature1
```

et pour vérifier la signature

```
$ openssl rsautl -verify -in <signature> -pubin -inkey <cle> -out
<empreinte>
```

exemple

```
$ openssl rsautl -verify -in signature1 -pubin -inkey  cléPub.pem -out
emp1.dgst
```