Table des matières

1) Générateur aléatoire

a) Un générateur aléatoire permet de générer des nombres, des lettres ou n'importe quoi d'autre en fonction de ce que l'on veut sans qu'on puisse prédire à l'avance ce qui va en sortir.

b) Je génère un flot de 16 bits :

```
C:\Users\iut>openssl rand 16
#-—ôó¨ #:#î¨Mf#
```

Puis un flot de 256 bits :

```
C:\Users\iut>openssl rand 256
µKXo@ "##º╔)å×î#■#V—"A#■fèÌµ▄■"ÍòBjèÑß0s┐?
┬#Ä¹x]¨ÏØÞfÏ¨Û#»ªÓ4#~╗+#▒┬#î<├×§vY#ëôZlRQÈSø~Âö╟OÑâÐZ┼ÿMp
bKL##£Hß/ÕTOÆ#ÉÂt
        À ]┬¹ =N#i1hA7Eä"¦Yx@ýªÂÓ+ÜrÄ[@qß¥[©°+NäñGûµ
#D9&.ejcOÅ±¾+NñH╣éêò8#¥÷±!1¼¡¤^Å└iW#§╝È│Þ#÷5§
¢#ê#Á×Â²ëï#øF—;¶#©uPï#sÂ|ÜwA¢=çMDÄL╔8#u    ##ð¶
```

c) Je génère une clé AES de 256 bits et je regarde avec mes camarades si avec notre même mot de passe ça nous redonne la même clé :

```
C:\Users\iut>openssl enc -aes256 -nosalt -p
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
key=9F86D081884C7D659A2FEAA0C55AD015A3BF4F1B2B0B822CD15D6C15B0F00A
08
```

```
iv =206DFC4E0335FA0AD986B9C1942DD653
```

d) Nous n'avons pas mis l'option rand dans la commande et on a tous retrouvé la même clé ce qui signifie que le générateur est bien contrôlé par le mot de passe si on ne lui dit pas d'être random

2) Chiffrement à clé secrète :

e) Je crée un fichier test.txt où je met n'importe quoi dedans. Ensuite je le chiffre graâce à la commande suivante :

```
C:\Users\iut\Desktop>openssl enc -k val -nosalt -p -in test.txt
oui mais non
C:\Users\iut\Desktop>openssl enc -aes256 -nosalt -p -in test.txt -
out test1.txt
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
key=110812F67FA1E1F0117F6F3D70241C1A42A7B07711A93C2477CC516D9042F9
DB
iv =6540301BFC821B61EB208BB5F6D5B036
```

f) Maintenant je déchiffre le texte en renseignant mon mot de passe et je vois bien qu'il me redonne la même clé et le même iv

```
C:\Users\iut\Desktop>openssl enc -d -aes256 -nosalt -p -in
test1.txt -out final.txt
enter aes-256-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
key=110812F67FA1E1F0117F6F3D70241C1A42A7B07711A93C2477CC516D9042F9
DB
iv =6540301BFC821B61EB208BB5F6D5B036
```

g) Je génère une clé à partir d'un mot de passe en utilisant le salt. Quand je le refais une deuxième fois avec le même mot de passe la clé et l'iv changent : (ici mon mot de passe est test)

```
C:\Users\iut\Desktop>openssl enc -aes256 -salt -p
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
Salted__ú#U#┼ýãø*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=A314550EC5ECC69B
key=D0A8AC92B7E469824FE6AB82F8239E0C689701B3C81078072A92DAFAFF0EC9
71
```

```
iv =9C33CD225F85F76F642531F0A577C17D
```

```
C:\Users\iut\Desktop>openssl enc -aes256 -salt -p
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
Salted__hQì▒×Aðá*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=68518DB19E41D0A0
key=8273AF74893B99E70044E13F53D1F54C236680E4DAB2CFB54A3C0F04B3F83C
4E
iv =DE28744DA01756F99AB789D14E860EAD
```

h) On vient de voir au dessus que le vecteur d'initialisation dépendait du salt

i) Le générateur aléatoire est maintenant contrôlé par le salt

j) Quand je chiffre mon fichier vers un autre fichier de sortie (ici opensslchiffre) je peux voir le salt qui m'a été donné :

```
test@202-19:~$ openssl enc -aes256 -salt -p -in openssl -out
opensslchiffre
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=A5815BBB1510221B
key=A839EF4A62BFDCD55C7D436A15202202097B3275C1CF72F41DDE9FDC48E3AA
24
iv =7526D4716B367C0549CDD9F5B0DC3DCF
```

Maintenant si j'affiche le fichier chiffré en hexadécimal je peux voir que mon salt y est affiché mais pas dans l'ordre :

```
test@202-19:~$ hexdump opensslchiffre
0000000 6153 746c 6465 5f5f 81a5 bb5b 1015 1b22
0000010 2993 b88a 7173 b0d9 6aec 8e16 7d9e 8d0b
0000020 e36a 68e2 319f 6e44 623e b0f0 4547 7903
0000030 622b d44b fd4c a091 05dc be3f 3f8c 8742
0000040
```

3) Clés asymétriques

a) Je génère une clé rsa non chiffrée de 2048 bits :

```
test@202-19:~$ openssl genrsa 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.............................................+++++
...................+++++
e is 65537 (0x010001)
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAzVLPQZXMLX5u/bVPJC7VYzaE95k5pEg9ednsE64izi8Gc10B
RmuFficb7OSgD4sMAKlDAufBPi9yXswE3zx02hJq7U87/VQT+eA43vzdIPBvnxCv
hp3Lh+0P4bFaCgsnPyhOykivGGrUYD4WdPWW42U5IqKQmY2aft78SLufMSXZ+iZl
aP9j4nYc4x2zUYD0Z8OYHi/KvI30YxUd80qMCja2YKX8R7l2e2zg0QxXTbqOxdl4
kZ+YO8iNB1zU8BVT6lRMG3jOOOMJ9QuKx3hc8gKCE376bhKMW4h/wwSbO0yfXyPy
67/PeDx3iB5TxrveX0593e/yvUmlULt5BaBqXwIDAQABAoIBAQDEeqvuPOSn3Edl
7SyQmWiFilRvg2cPeDmr8wfgRVD1MTbhCmz/8EBMWix04srwVMTllROVIiVf1pOB
d4ZqATZowBsmgB4N63k6MoD5kcwhizOlaZguH632oroB5W8eEVI7M91B4Y/8s64L
cx6WSbVcYS7tn9bzuffP1J5jE3fg1HouDUmxEB2/D7WgDaa0XZTMYJ4vM0BHpZ5i
EYgoEYkkxQbs8Tmy1aok1UiEhJCEC1NzOTMLDyQ8yt6S3/+xoLd3xJ0WivOKxwVB
aesJZvQvgenHmKn3c1fQdETxnvPNSH1dY4PhIrNrt3fvvi0rnXWgSdS3Zy3Ggd+J
61YRIrKBAoGBAPSoxJw6piFFyWsAafDyQCFvpCOT4YLkX22Tcg/IJW5hV2GY0FPK
ihg1WmQ2RD4pIika9WP5uj7ZZ/339FzS28RspqFxFf/qzZZKYwppoYnOqY1KJxFx
Yc3NlaJLzQ3LlkndIP5VZZpQtw2BngOouhO9Xec3FNgvoPpgmdY0a2rhAoGBANbX
RC8fwqNuy3yh6Dfkud4fGbSYmnQMkiLRsdK8ZI3AMkVd9FgdHPZmiOisQ70MzlCH
UUowctGRRm2lUnb7GwnxBQr/IOvzJGZqJCCz+mGANo5J9T3A/2qF1VJmV6O5J6HL
D1GbFcx/bY9rgZ+m9ude/RmQvWoY0QOmPkJbab0/AoGBAO2p0CvvcRhbQXkXEtLD
kPaqC6mf6u4KyISaObAI/guGtaqSFrV+x70SfvdgG9TCgnOpw3zVx01sRe/Y4WEW
V/Qsk4W6Y+ORhRqcsi8YQ5qpzdz5jpAMMzUHo3bf6ZagESM5kuZdsHBCPj/5jLUD
P5cTWlfUEwSa5+aw1l/V1g8hAoGBAI/fONoV+tJXsA2Wn1xO3+nQCfbW+xgcEwEe
TGdDolMLEp9BfBBF7s9TevvV4McJfGCRPe7g4wNUtkVXIRJstn7B2Q60K+fWu9mZ
2vfLjzEcG2zOOl1QkbtURLcZsjcCF3hR88BCiyaXb1dMN1IjDdWD1xr3wuD6HX7E
JY1qijUhAoGAS4/vM0J+HxY54mqL6BMMRoKhvSTy2zRaELu2I+TNifb6WPP472Cs
8LbbdVw3mKKZsWtJDqKDByfRwm8RZ6dYgIlrNQhMGfdUURWrX3zlwSg10st5JqhP
c0RcWR7HKquifOMIiFKEiVReBScYL2MTDokNEBpwNxJNWT8MyzzY+WM=
-----END RSA PRIVATE KEY-----
```

Le fichier nous a généré une clé privée

b) Je génère maintenant une clé rsa chiffrée par l'algo aes256 avec une pass phrase :

```
test@202-19:~$ openssl genrsa -out rsa -aes256
Generating RSA private key, 2048 bit long modulus (2 primes)
......+++++
.................................+++++
e is 65537 (0x010001)
Enter pass phrase for rsa:
Verifying - Enter pass phrase for rsa:
```

En regardant dans le fichier de sortie on peut voir qu'il est crypté et aussi voir le type d'algo de chiffrement :

```
test@202-19:~$ cat rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,9EB52B5CDEB28253CF3A647558D87DB3

XOXG29GiAsogT2cchDInR60/6nhvWfXDa5DCR/1eP1kptFp5FE+aAPuCH3UtMOLJ
Hnz9j2eBBEnoOkeQz0EbS1cuIdT+zLpjBlC07ih+1JpoQa+BBvDgjQHTuljjomFE
6/MCBqYkLAb+1OIvfiiXTuYWWutn079UzWFeVirZkgvHSM0YNPIofderiLpUiub0
2RvabSaD+gQxIZE5GyJTTLBUsYFtgMZhqCaPMdg2FVs1JeVZIx4+c2N4HNh8aj/L
P8IxcSu6pazsrkn6xq01iioBK2n/r4cJMJpgdXaQvgdDxmrZrhP6oo2oU3T8F2c8
jMsnE1F0nt91RyHWygyDeKn0bhQuiuH0nv+TTxSxNlI52i321G5Jjvq0LQlvlG+u
FrGIwYC4XFrWbytJn1xQsu3hu2bjTmTqBp+9UyUciXlsvytnxS3P3Db5HBKjV5m4
fcSJfpHgVlssabhfCg18Pj+l4k5f/iM0VI3PFAHQzQI/MQ9mh/JOeHjO6tsscrhJ
Rtg90NDRwHNb0Oo+brKc9aKAcXMB1S7RQ9nlIBtDCi+3vZ780vQ5ADypdJfhHZhs
tZoIAeZsD0D4EDW2NQkVqmKg7mrb8SR/6cj+havzqryD5Sscf7kgs1ppB8uhhbtA
J4HmwyPLeqr1ZhqUsu6s+DOkFOJQygKtZNYRSQpoQu5Up2sb9egdLuZFt3lhbAIR
qIPI3WBPYuX1MR3J4UMlfECTkD0j3TB8IKbRHy5wpbFVBmqw4ZcMCkncl6HBKQbK
r9CLqzb6+XZ0AMILltEcNIQ1/ZSk5xQvGiKpzUynmAdxacKI0PFIuSqY89fc2NmU
LicAlmC1nrT1xVbLz7afwevU9dnpLiQzD7tcbJHGOcznmDtZeUhIy7virAqD30gT
iLUHukEiIBP5jFChfhDvBzxmhKwWhfJevAeHX1/Eqa8GoUQggtj/cFWSXZ065Nmx
qpy/N6to7R5+hAf97WXPVdpwLAu6VCBSE38bp+qUzVz0leyiatzYRvtkO/cGo7I+
QsGv9GBgW3qCCS8fKA5k1YM6stR+4jcNqRxUPttoYvXQgZOp1J0NTo5YSy4IXdHx
OPPY70HUAstCi1zKG38wqzKJkPR9PZvmetjwU74KTvCDuG+zJw6AMwSqjx/bDNis
krpJyrYD/YL0wLTCCx+bQ2GoiPsqKbDjFizgwAB9Cnj7m03GBXlAAQBOFvbOGAjB
wyGtf7nmFlLxz5fmRBaCi5czUDDs1RevXL5hj/7vzzL4/CxonNDqi38Ty6m/TccE
7tDKMmYHH/r4P4kwErny8m7ilNbxAP9eUESwzNv0/GpTDKLaAtRdpziG29z0+fRV
XiVWaKCYWBjrqDEjIwmt7PP/ZUD+yWD+Cb0aXCuXonWeHyn5scwZDOocmE275/Mm
hGczMucm+N38vOpRwg633dzT9+ccMqBZoP6VT3xhoMUy1l7boqFP9FyiTbHXFPwN
7jSsz3GM4vHvTMG4kG5TGb6xa7ZMas+ikaCjVs7DtzfSuTfDIo+3Srf5ACmQafij
CnRBxIJ8RmUX05F91cy76/NpI+MFFPRVwLN7HlhAe0fNpX+3xtNZSxIRln0CLRQt
-----END RSA PRIVATE KEY-----
```

c) J'extrais la clé publique de mon fichier contenant les deux clés :

```
test@202-19:~$ openssl rsa -in rsapriv -pubout -out rsapub
Enter pass phrase for rsa:
writing RSA key
```

J'ai maintenant deux fichiers (rsapriv et rsapub)

d) Je me crée un fichier texte appelé "o" où j'écris ce que je veux dedans. Ensuite je crypte le fichier avec ma clé rsa publique que je viens de faire :

```
test@202-19:~$ openssl rsautl -in o -out ocrypt -inkey rsapub -
pubin -encrypt
```

Je vérifie dans le fichier de sortie que mon texte soit bien crypté :

```
test@202-19:~$ cat ocrypt
??cc?fB?2?s?pá?)?c
'??w??+~?;p?P??b"'b?Of'?"??????s.X?aL?cS?8?,??<??G??-??|
??F?{F?fυ&`?h
       ???&?yL???e???i.P?0=?.
⁒S???gG2???#hn??j?W??K??f7svQ=<7????????z?8b??Z?#??"2mC?bQG$U?d??Q5
C8?Sp?,Nyγ?/?=:#Y??~??F????Bg??H
```

Pour finir je décrypte le fichier avec le fichier contenant la clé privée :

```
test@202-19:~$ openssl rsautl -in ocrypt -out odecrypt -inkey
rsapriv -decrypt
```

Une fois décrypté je peux lire mon fichier de sortie et je dois retrouver mon texte de base :

```
test@202-19:~$ cat odecrypt
test de chiffrement avec clé publique
```

4) Message Digest

a) Je me crée un fichier texte (ici test) et voici son contenu :

```
Test de décryptage du fichier
```

Je calcul l'empreinte de ce fichier texte

```
samuel@samuel-MS-7B51:~$ openssl dgst test
SHA256(test)=
f36c6e8795422d7bde62129890133a8098f73a3082607c7622f35fe4ffcba599
```

b) Je modifie maintenant un bit dans mon fichier (n'importe quelle lettre) et je re calcul son empreinte (j'enlève juste la majuscule au T) :

```
test de décryptage du fichier
```

Je peux voir que son empreinte a changé :

```
SHA256(test)=
793c20dbb5f22673f7864ce6cc7a29ff75191c850e7e8644c6b43a13dc58710b
```

c) Si un pirate peut accéder à la fois au fichier et à son empreinte quels sont les risques encourus ?

Si un pirate peut accéder au fichier et à son empreinte il peut récupérer la clé privée et donc se placer entre deux communications.

On peut y remédier en signant les fichiers

d)