

OpenSSL

OpenSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS qui offre

1. Une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS.
2. Une commande en ligne (OpenSSL) permettant :
 - la création de clés RSA, DSA (signature)
 - la création de certificats X509
 - le calcul d'empreintes (MD5, SHA, RIPEMD160, ...)
 - le chiffrement et déchiffrement (RSA, DES, IDEA, RC2, RC4, Blowfish, ...)
 - la réalisation de tests de clients et serveurs SSL/TLS
 - la signature et le chiffrement de courriers (S/MIME)

EXERCICE

1. Générez-vous votre clé privée avec une longueur de 1024 bits.

```
(kali㉿kali)-[~/Desktop]
└─$ openssl genrsa -out rsa.private 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
```

2. Dérivez la clé publique associée à la clé privée.

```
(kali㉿kali)-[~/Desktop]
└─$ openssl rsa -in rsa.private -out rsa.public -pubout -outform PEM
writing RSA key
```

3. Générer une clé de 256 bits comme une clé de cryptage symétrique pour une utilisation ultérieure.

```
(kali㉿kali)-[~/Desktop]
└─$ openssl rand 32 > aes.key
```

4. Créer un fichier cigale.txt contenant un message secret.

```
(kali㉿kali)-[~/Desktop]
└─$ echo Mot de passe Github/ASR2-DORANCO ASR2Stagiaire > cigale.txt
```

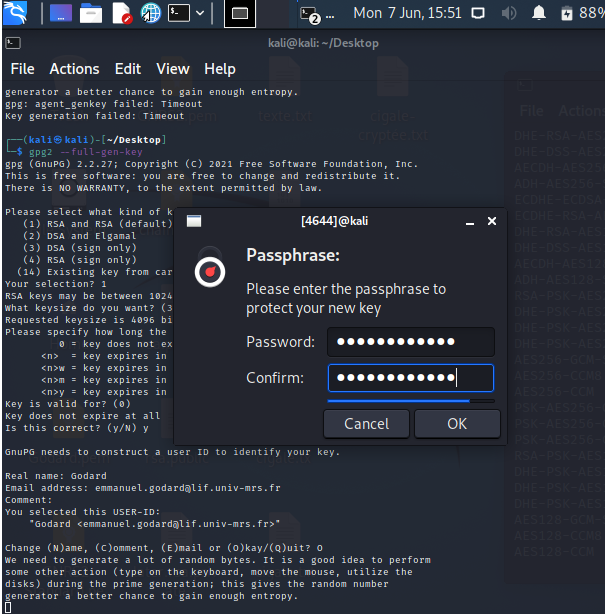
```
(kali㉿kali)-[~/Desktop]
└─$ cat cigale.txt
Mot de passe Github/ASR2-DORANCO ASR2Stagiaire
```

5. Chiffrer le fichier cigale.txt avec l'algorithme AES 256 bits.

```
(kali㉿kali)-[~/Desktop]
└─$ openssl enc -e -aes-256-cbc -in cigale.txt -out cigale-cryptée.txt -pbkdf2
enter aes-256-cbc encryption password: 1234
Verifying - enter aes-256-cbc encryption password: 1234
```

6. Chiffrer la clé de la session (étape 3) avec l'algorithme RSA, en utilisant la clé publique de votre binôme.

```
(kali@kali)~[/Desktop]
$ gpg2 --full-gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
gpg: keybox '/home/kali/.gnupg/pubring.kbx' created
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysizes do you want ? (3072) 4096
Requested keysizes is 4096 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) Enter
Key does not expire at all
Is this correct? (y/N) y
GnuPG needs to construct a user ID to identify your key.
Real name: Godard
Email address: emmanuel.godard@lif.univ-mrs.fr
Comment: Enter
You selected this USER-ID:
  "Godard <emmanuel.godard@lif.univ-mrs.fr>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```



```
gpg: /home/kali/.gnupg/trustdb.gpg: trustdb created
gpg: key AF95E51F4AFFB279 marked as ultimately trusted
gpg: directory '/home/kali/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revocs.d/D985C1EA7662E87DA8B5FC28AF95E51F4AFFB279.rev'
public and secret key created and signed.
pub  rsa4096 2021-06-07 [SC]
    D985C1EA7662E87DA8B5FC28AF95E51F4AFFB279
uid          Godard emmanuel.godard@lif.univ-mrs.fr
sub  rsa4096 2021-06-07 [E]
```

7. Générer un digest (hashage) pour le fichier cigale.txt avec md5 et sha1.

```
(kali) [~/Desktop]
└─$ openssl dgst -md5 cigale.txt
MD5(cigale.txt)= 016b34946097eb1b4f4da4a09748a0d6

(kali) [~/Desktop]
└─$ openssl dgst -sha1 cigale.txt
SHA1(cigale.txt)= f2ccdb991d770b49adb8ce3488303396590b1910
```

8. A quoi sert le digest en général ?

Afficher le hash d'un fichier selon un certain algorithme

Peut également être utilisé pour générer et afficher des signatures numériques.

9. Envoyer la clé chiffrer à votre binôme (via : FTP, Telnet, scp, email, clé usb, etc.).

10. Envoyer le fichier chiffré à votre binôme, ainsi que sa signature.

11. Traiter les fichiers reçus par votre binôme (déchiffrer la clé et le fichier), après la vérification de la signature. Extraire le message original.