Université Cheikh Anta Diop

École supérieur Polytechnique
Département Génie Informatique
Année Universitaire 2024-2025
Licence Professionnelle
Théorie de la sécurité et cryptographie

Atelier 1 : OpenSSL – une introduction

Réalisé par :
Mouhamadou Moustapha LO
L3SRT

Dr Doudou FALL

I. Installation d'OpenSSL

- Ubuntu : Il peut être facilement installé avec

```
$ sudo apt-get install openssl
```

J'ai déjà installé openssl sur ma machine virtuelle. Donc j'ai utilisé la **commande sudo apt policy openssl** pour vérifier que openssl est bel et bien installe.

Mais il est fort probable qu'il soit déjà installé sur votre système. La ligne précédente pourrait installer une version assez dépassée (par exemple, dans Ubuntu 16.04, il installe OpenSSL 1.0.2g 01/03/2016, mais il est maintenant hors de support ! mais dans Ubuntu 18.04 la version installée d'OpenSSL est 1.1.1). Utilisez :

```
$ openssl version -a
```

Pour obtenir tous les détails de la version installée d'OpenSSL. Les versions plus récentes (supportées) peuvent être installées à partir du code source (la version actuelle est 3.0.5). Cependant, la version obsolète 1.0.2 est suffisante à des fins éducatives.

- Windows : Il existe plusieurs façons d'installer OpenSSL sur les systèmes Windows. L'une d'entre elles consiste à utiliser CygWin pour créer un environnement de type Linux et ensuite installer OpenSSL. Sur les systèmes Windows 10+, il est possible d'activer Windows Subsystem for Linux (WSL, Sous-système Windows pour Linux), puis d'installer Linux sur Windows. Un terminal Linux peut être lancé à partir du menu de démarrage de Windows.

II. Qu'est-ce que OpenSSL ?

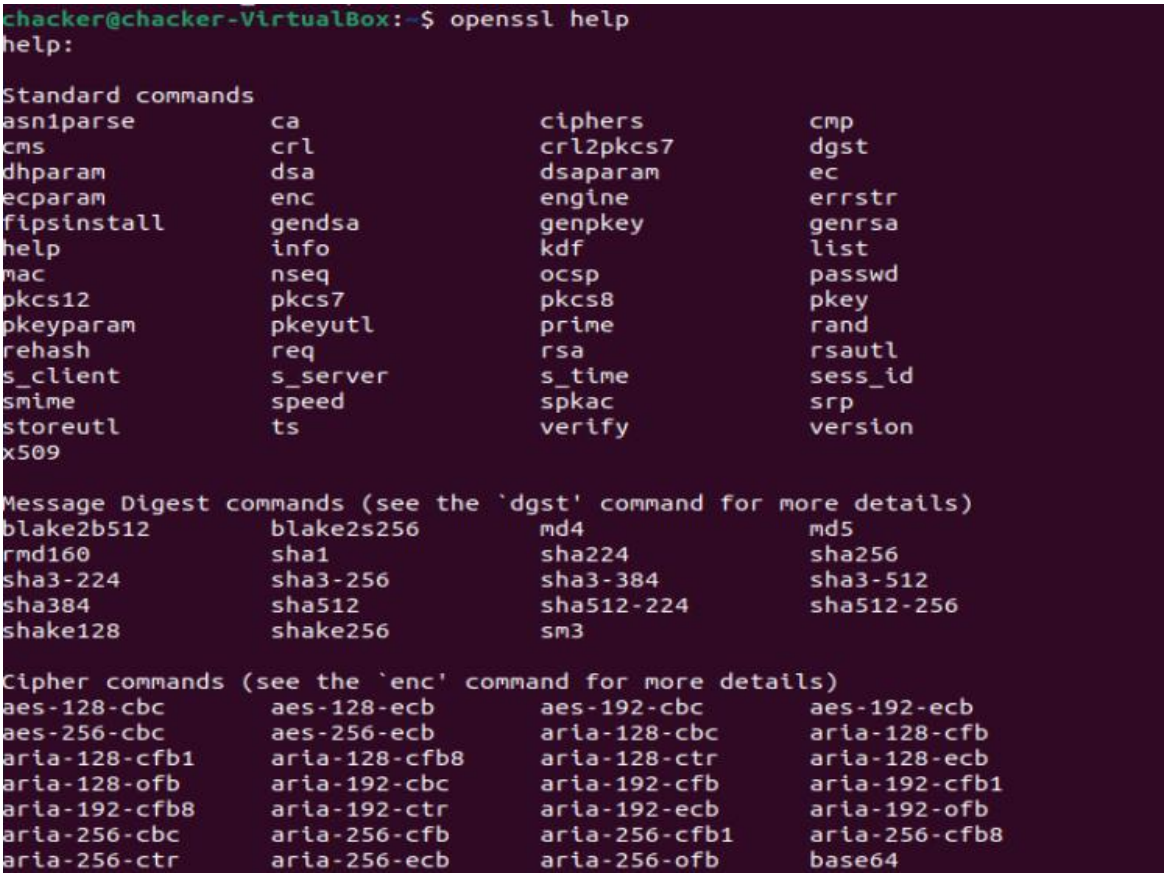
Selon la documentation, OpenSSL est une boîte à outils de cryptographie qui met en œuvre les protocoles réseau Secure Sockets Layer (SSL) et Transport Layer Security (TLS) ainsi que les normes de cryptographie connexes requises par ces protocoles.
Le programme openssl est un outil de ligne de commande permettant d'utiliser les diverses fonctions de cryptographie de la bibliothèque cryptographique d'OpenSSL à partir du shell. Il peut être utilisé pour :

- La création et la gestion des clés privées, des clés publiques et des paramètres ;
- Les opérations cryptographiques des clés publiques ;
- Création de certificats X.509, CSRs et CRLs ;
- Le calcul de digits de messages ;
- Le cryptage et le décryptage avec des Ciphers ;
- Tests des clients et serveurs SSL/TLS ;
- Traitement du courrier signé ou crypté S/MIME ;
- Demandes, génération et vérification d'horodatage ;
- etc.

III. Comment utiliser l'outil en ligne de commande

L'outil de ligne de commande openssl permet d'accéder aux différents outils implémentés dans les bibliothèques OpenSSL. Vous pouvez obtenir la liste des commandes d'openssl avec

```
$ openssl help
```



Des informations sur une commande openssl individuelle (par exemple `enc`) et ses arguments peuvent être récupérées avec :

```
$ openssl enc -help
```

```
chacker@chacker-VirtualBox:~$ openssl enc -help
Usage: enc [options]

General options:
 -help               Display this summary
 -list               List ciphers
 -ciphers            Alias for -list
 -e                  Encrypt
 -d                  Decrypt
 -p                  Print the iv/key
 -P                  Print the iv/key and exit
 -engine val         Use engine, possibly a hardware device

Input options:
 -in infile          Input file
 -k val              Passphrase
 -kfile infile       Read passphrase from file

Output options:
 -out outfile        Output file
 -pass val           Passphrase source
 -v                  Verbose output
 -a                  Base64 encode/decode, depending on encryption flag
 -base64             Same as option -a
 -A                  Used with -[base64|a] to specify base64 buffer as a single lir

Encryption options:
 -nopad              Disable standard block padding
 -salt               Use salt in the KDF (default)
 -nosalt             Do not use salt in the KDF
 -debug              Print debug info
 -bufsize val        Buffer size
 -K val              Raw key, in hex
 -S val              Salt, in hex
 -iv val             IV in hex
```

Comme d'habitude, les informations détaillées peuvent être trouvées dans la page de manuel correspondante :

```
$ man openssl
```

```
OPENSSL(1SSL)                                OpenSSL                                OPENSSL(1SSL)

NAME
       openssl - OpenSSL command line program

SYNOPSIS
       openssl command [ options ... ] [ parameters ... ]

       openssl list standard-commands | digest-commands | cipher-commands | cipher-algorithms | digest-algorithms | mac-
       algorithms | public-key-algorithms

       openssl no-XXX [ options ]

DESCRIPTION
       OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1)
       network protocols and related cryptography standards required by them.

       The openssl program is a command line program for using the various cryptography functions of OpenSSL's crypto library
       from the shell.  It can be used for

        o  Creation and management of private keys, public keys and parameters
        o  Public key cryptographic operations
        o  Creation of X.509 certificates, CSRs and CRLs
        o  Calculation of Message Digests and Message Authentication Codes
        o  Encryption and Decryption with Ciphers
        o  SSL/TLS Client and Server Tests
        o  Handling of S/MIME signed or encrypted mail
        o  Timestamp requests, generation and verification

COMMAND SUMMARY
       The openssl program provides a rich variety of commands (command in the "SYNOPSIS" above).  Each command can have many
       options and argument parameters, shown above as options and parameters.

       Detailed documentation and use cases for most standard subcommands are available (e.g., openssl-x509(1)).

 Manual page openssl(1ssl) line 1 (press h for help or q to quit)
```

Les différents algorithmes de cryptage implémentés dans votre version d'OpenSSL peuvent être listés avec :

```
$ openssl list-cipher-algorithms
```

ou avec :

```
$ openssl enc -ciphers
```

```
chacker@chacker-VirtualBox:~$ openssl enc -ciphers
Supported ciphers:
-aes-128-cbc               -aes-128-cfb               -aes-128-cfb1
-aes-128-cfb8              -aes-128-ctr               -aes-128-ecb
-aes-128-ofb               -aes-192-cbc               -aes-192-cfb
-aes-192-cfb1              -aes-192-cfb8              -aes-192-ctr
-aes-192-ecb               -aes-192-ofb               -aes-256-cbc
-aes-256-cfb               -aes-256-cfb1              -aes-256-cfb8
-aes-256-ctr               -aes-256-ecb               -aes-256-ofb
-aes128                    -aes128-wrap               -aes192
-aes192-wrap               -aes256                    -aes256-wrap
-aria-128-cbc              -aria-128-cfb              -aria-128-cfb1
-aria-128-cfb8             -aria-128-ctr              -aria-128-ecb
-aria-128-ofb              -aria-192-cbc              -aria-192-cfb
-aria-192-cfb1             -aria-192-cfb8             -aria-192-ctr
-aria-192-ecb              -aria-192-ofb              -aria-256-cbc
-aria-256-cfb              -aria-256-cfb1             -aria-256-cfb8
-aria-256-ctr              -aria-256-ecb              -aria-256-ofb
-aria128                   -aria192                   -aria256
-bf                        -bf-cbc                    -bf-cfb
-bf-ecb                    -bf-ofb                    -blowfish
-camellia-128-cbc          -camellia-128-cfb          -camellia-128-cfb1
-camellia-128-cfb8         -camellia-128-ctr          -camellia-128-ecb
-camellia-128-ofb          -camellia-192-cbc          -camellia-192-cfb
-camellia-192-cfb1         -camellia-192-cfb8         -camellia-192-ctr
-camellia-192-ecb          -camellia-192-ofb          -camellia-256-cbc
-camellia-256-cfb          -camellia-256-cfb1         -camellia-256-cfb8
-camellia-256-ctr          -camellia-256-ecb          -camellia-256-ofb
-camellia128               -camellia192               -camellia256
-cast                      -cast-cbc                  -cast5-cbc
-cast5-cfb                 -cast5-ecb                 -cast5-ofb
-chacha20                  -des                       -des-cbc
-des-cfb                   -des-cfb1                  -des-cfb8
-des-ecb                   -des-ede                   -des-ede-cbc
-des-ede-cfb               -des-ede-ecb               -des-ede-ofb
-des-ede3                  -des-ede3-cbc              -des-ede3-cfb
```

En fonction de la version d'OpenSSL.

IV. Cryptage et décryptage d'un fichier

Utilisons le chiffrement CTR pour chiffrer un fichier. Commencez par ouvrir un terminal sur la machine virtuelle Kali Linux. Créez un fichier texte contenant le message "Code Top Secret" en exécutant la commande suivante :

```
$ echo "Code Top Secret" > clair.txt
```

```
chacker@chacker-VirtualBox:~$ echo "Code Top Secret"> clair.txt
```

Pour afficher le contenu du fichier, exécutez la commande `cat` :

```
$ cat clair.txt
```

```
chacker@chacker-VirtualBox:~$ cat clair.txt
Code Top Secret
```

Nous utiliserons la bibliothèque openssl, qui comprend plusieurs algorithmes de chiffrement et qui est préinstallée sur Kali Linux. Cryptez le fichier en exécutant la commande suivante et en entrant un mot de passe à l'invite :

```
chacker@chacker-VirtualBox:~$ openssl enc -aes-256-ctr -pbkdf2 -e -a -in clair.txt -out crypter.txt
enter AES-256-CTR encryption password:
Verifying - enter AES-256-CTR encryption password:
```

L'indicateur enc -aes-256-ctr spécifie que vous souhaitez utiliser le chiffrement par blocs aes256ctr. Le nom du chiffrement par blocs est divisé en trois parties. La première section (aes) représente la fonction de mappage utilisée dans chaque bloc, dans ce cas le chiffrement AES mentionné précédemment. La section suivante (256) représente la taille du bloc, qui est de 256 bits dans ce cas. La dernière section (ctr) représente un mode de chiffrement par bloc CTR. L'option suivante, -pbkdf2, représente la fonction de dérivation de clé, et l'option -e indique à openssl de chiffrer le fichier. L'option -a met un fichier crypté en encodage Base64 au lieu de binaire, ce qui nous facilitera l'impression du fichier crypté dans le terminal. Enfin, nous utilisons les options -in et -out pour spécifier le fichier que nous voulons crypter et le nom du fichier de sortie, respectivement.

Pour visualiser le contenu de votre fichier crypté, utilisez la commande cat :

```
$ cat crypter.txt
```

```
chacker@chacker-VirtualBox:~$ cat crypter.txt
U2FsdGVkX1/HEDEJpSjV7X4AssiaVaFFEwcSoNZGnH0=
```

Pour décrypter le fichier, exécutez la commande suivante :

```
chacker@chacker-VirtualBox:~$ openssl enc -aes-256-ctr -pbkdf2 -d -a -in crypter.txt -out decrypter.txt
enter AES-256-CTR decryption password:
chacker@chacker-VirtualBox:~$
```

L'option -d indique à openssl de décrypter le fichier. Entrez le mot de passe que vous avez utilisé précédemment. Comme l'algorithme onetime pad, le CTR déchiffre le texte chiffré en le soumettant à un XOR avec la clé produite par le bloc, inversant ainsi le processus de chiffrement.

Notez qu'un hacker qui volerait ce fichier crypté ne serait peut-être pas en mesure de le décrypter, mais il pourrait quand même le corrompre en modifiant les bits cryptés.

V. Cryptage d'un fichier avec RSA

Maintenant que vous connaissez la théorie derrière RSA, utilisons la bibliothèque openssl pour générer un email crypté. Pour commencer, générez une paire de clés publique et privée en exécutant la commande suivante :

```
$ openssl genrsa -out pub_priv_paire.key 1024
```

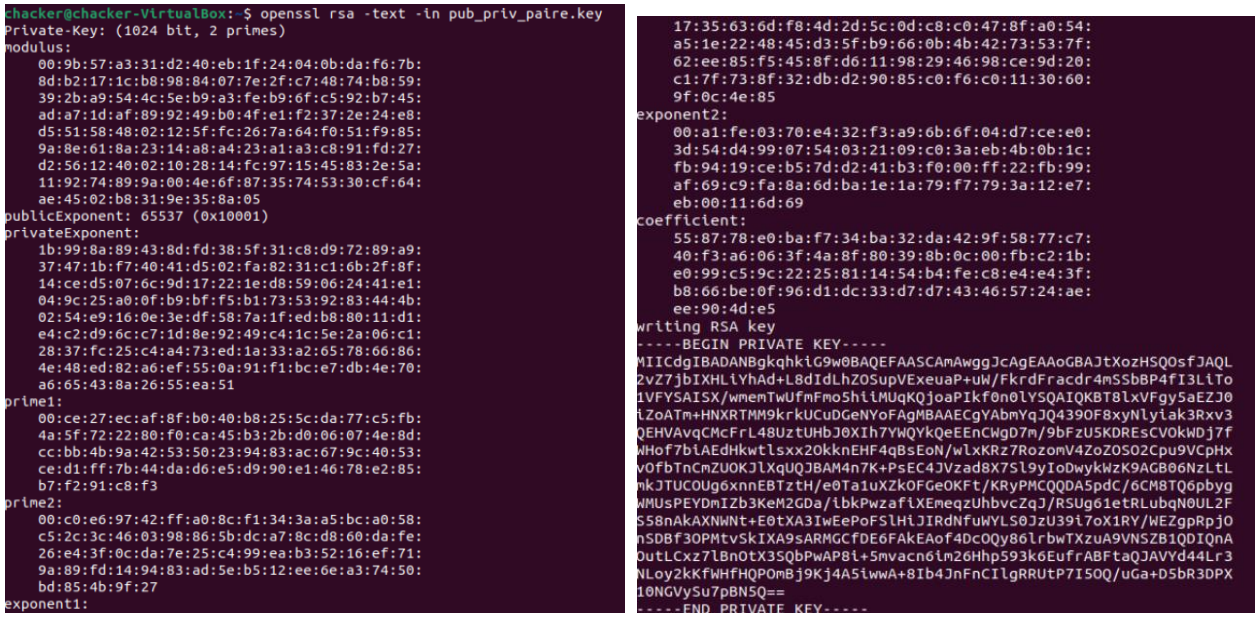
```
chacker@chacker-VirtualBox:~$ openssl genrsa -out pub_priv_paire.key 1024
```

L'option `genrsa` permet à openssl de savoir que vous voulez générer une clé RSA, l'option `-out` spécifie le nom du fichier de sortie, et la valeur `1024` représente la longueur de la clé. Les clés plus longues sont plus sûres. La NSA recommande des clés RSA d'une longueur de 3 072 bits ou plus.

Rappel : ne partagez pas votre clé privée avec qui que ce soit.

Vous pouvez afficher la paire de clés que vous avez générée en exécutant la commande suivante :

```
$ openssl rsa -text -in pub_priv_paire.key
```



L'option **rsa** indique à openssl d'interpréter la clé comme une clé RSA et l'option **-text** affiche la clé dans un format visible par l'humain.

Comme vous pouvez le constater ci-dessous après avoir saisi la commande **openssl rsa -text -in pub_priv_paire.key**

La section en bas représente la version codée en Base64 de la paire de clés publique-privée, avec tous ses composants.

Vous pouvez extraire la clé publique de ce fichier en exécutant la commande suivante :

```
$ openssl rsa -in pub_priv_paire.key -pubout -out cle_publique.key
```

```
chacker@chacker-VirtualBox:~$ openssl rsa -in pub_priv_paire.key -pubout -out cle_publique.key
writing RSA key
```

L'option **-pubout** indique a openssl d'extraire la clé publique du fichier. Vous pouvez voir la clé publique du fichier en exécutant la commande suivante, dans laquelle l'option **-pubin** indique a openssl de traiter l'entrée comme une clé publique :

```
$ openssl rsa -text -pubin -in cle_publique.key
```

```
chacker@chacker-VirtualBox:~$ openssl rsa -text -pubin -in cle_publique.key
Public-Key: (1024 bit)
Modulus:
    00:9b:57:a3:31:d2:40:eb:1f:24:04:0b:da:f6:7b:
    8d:b2:17:1c:b8:98:84:07:7e:2f:c7:48:74:b8:59:
    39:2b:a9:54:4c:5e:b9:a3:fe:b9:6f:c5:92:b7:45:
    ad:a7:1d:af:89:92:49:b0:4f:e1:f2:37:2e:24:e8:
    d5:51:58:48:02:12:5f:fc:26:7a:64:f0:51:f9:85:
    9a:8e:61:8a:23:14:a8:a4:23:a1:a3:c8:91:fd:27:
    d2:56:12:40:02:10:28:14:fc:97:15:45:83:2e:5a:
    11:92:74:89:9a:00:4e:6f:87:35:74:53:30:cf:64:
    ae:45:02:b8:31:9e:35:8a:05
Exponent: 65537 (0x10001)
writing RSA key
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCbV6Mx0kDrHyQEC9r2e42yFxy4
mIQHfi/HSHS4WTkrqVRMXrmj/rlvxZK3Ra2nHa+JkkmwT+HyNy4k6NVRWEgCEl/8
Jnpk8FH5hZqOYYojFKikI6GjyJH9J9JWEkACECgU/JcVRYMuWhGSdImaAE5vhzV0
UzDPZK5FArgxnjWKBQIDAQAB
-----END PUBLIC KEY-----
```

Il est temps de faire usage de votre nouvelle paire de clés publique et privée.

Créez un fichier texte à crypter :

```
$ echo "Mais qu'est-ce qu'il raconte ce japonais" > clair.txt
```

```
chacker@chacker-VirtualBox:~$ echo "Mais qu'est ce qu'il raconte ce japonais"> clair.txt
chacker@chacker-VirtualBox:~$ cat clair.txt
Mais qu'est ce qu'il raconte ce japonais
```

Utilisez l'utilitaire RSA (rsautl), qui fait partie d'openssl, pour créer un fichier binaire crypté (cipher.bin) :

```
$ openssl rsautl -encrypt -pubin -inkey cle_publique.key -in clair.txt -out chiffrer.bin -oaep
```

Le message indique que la commande **rsautl** utilisée pour le chiffrement RSA a été **dépréciée** (c'est-à-dire qu'elle n'est plus recommandée) depuis OpenSSL version 3.0. Elle est remplacée par la commande **pkeyutl**, qui est plus moderne et flexible pour gérer les opérations liées aux clés publiques et privées.

```
chacker@chacker-VirtualBox:~$ openssl rsautl -encrypt -pubin -inkey cle_publique.key -in clair.txt -out chiffrer.bin -oaep
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
```

J'ai utilisé ci-dessous la commande **openssl pkeyutl -encrypt -pubin -inkey cle_publique.key -in clair.txt -out chiffrer.bin -pkeyopt rsa_padding_mode:oaep**

```
chacker@chacker-VirtualBox:~$ openssl pkeyutl -encrypt -pubin -inkey cle_publique.key -in clair.txt -out chiffrer.bin -pkeyopt rsa_p
adding_mode:oaep
```

Explication des options :

- **pkeyutl :** Commande moderne pour les opérations liées aux clés.
- **-encrypt :** Indique qu'on souhaite chiffrer.
- **-pubin :** Spécifie qu'on utilise une clé publique.
- **-inkey cle_publique.key :** Chemin de la clé publique.
- **-in clair.txt :** Fichier texte clair à chiffrer.
- **-out chiffrer.bin :** Fichier de sortie pour les données chiffrées.
- **-pkeyopt rsa_padding_mode:oaep :** Définit l'utilisation de l'algorithme de remplissage OAEP, qui est sécurisé pour RSA.

Remarquez que nous avons inclus l'indicateur -oaep. Les implémentations sécurisées de RSA doivent également utiliser l'algorithme OAEP. Chaque fois que vous cryptez et décryptez des fichiers à l'aide d'OpenSSL, veillez à appliquer cette option pour sécuriser les opérations.

Convertissez le fichier binaire en Base64 en exécutant la commande suivante :

```
$ openssl base64 -in chiffrer.bin -out chiffrerBase64.txt
```

```
chacker@chacker-VirtualBox:~$ openssl base64 -in chiffrer.bin -out chiffrerBase64.txt
```

La conversion du fichier binaire en encodage Base64 vous permet de le coller dans un courriel sous forme de texte. Vous pouvez visualiser le texte codé en Base64 l'aide de la commande `cat` :

```
$ cat chiffrerBase64.txt
```

```
chacker@chacker-VirtualBox:~$ cat chiffrerBase64.txt
iVb4wNvi5/Ti+DibFamoEyqrOj2MbgyHibB/x1+qqBThrtdBEFpwI3OfOIKWDGvs
9+PkEYFSPws9HueX/mWpV8Pz2s/rrr/6EuVsug95t9sE1LNebSO8FzImoNYbKjMY
+DSuyo3HRRvul9xAqFuOevDNEdvnukbRsXi5zPL2CC0=
```

L'encodage Base64 du fichier ne chiffre pas vraiment le fichier, il le formate simplement. Chiffrez toujours le fichier avant de le coder en base64. Décryptez le message en reconvertissant le texte Base64 en binaire :

```
chacker@chacker-VirtualBox:~$ openssl base64 -d -in chiffrerBase64.txt -out chiffrerBase64.bin
```

```
$ openssl base64 -d -in chiffrerBase64.txt -out chiffrerBase64.bin
```

Ensuite, décryptez le binaire en utilisant la commande suivante :
Comme la commande **rsautl** n'est plus recommandée donc j'utilise **pkeyutl** aussi pour décrypter le binaire avec la commande **openssl pkeyutl -decrypt -inkey pub_priv_paire.key -in chiffrerBase64.bin -out clairDechiffrer.txt -pkeyopt rsa_padding_mode:oaep**

Explication des options :

- **pkeyutl :** La commande moderne pour les opérations de clé publique/privée.
- **-decrypt :** Indique que l'on veut décrypter.
- **-inkey pub_priv_paire.key :** Spécifie la clé privée pour déchiffrer (ici, pub_priv_paire.key contient la paire clé publique/privée).
- **-in chiffrerBase64.bin :** Fichier chiffré en entrée.
- **-out clairDechiffrer.txt :** Fichier texte en clair après déchiffrement.
- **-pkeyopt rsa_padding_mode:oaep :** Définit que le mode de remplissage utilisé est OAEP (Optimal Asymmetric Encryption Padding).

```
$ openssl rsautl -decrypt -inkey pub_priv_paire.key -in chiffrerBase64.bin -out clairDechiffrer.txt
-oaep
```

```
chacker@chacker-VirtualBox:~$ openssl pkeyutl -decrypt -inkey pub_priv_paire.key -in chiffrerBase64.bin -out clairDechiffrer.txt -pk
eyopt rsa_padding_mode:oaep
```

Enfin, vous pouvez visualiser le message décrypté en utilisant la commande `cat` :

```
$ cat clairDechiffrer.txt
```

```
chacker@chacker-VirtualBox:~$ cat clairDechiffrer.txt
Mais qu'est ce qu'il raconte ce japonais
chacker@chacker-VirtualBox:~$
```