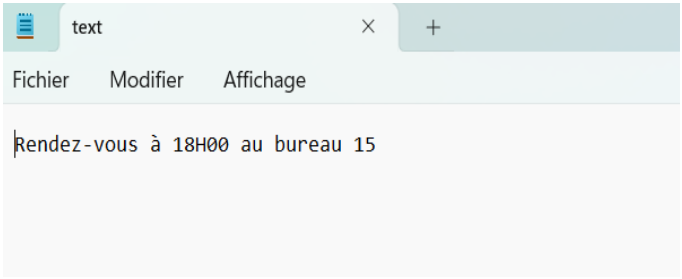
TD5 : Cybersécurité

Exercice 1) Chiffrement symétrique

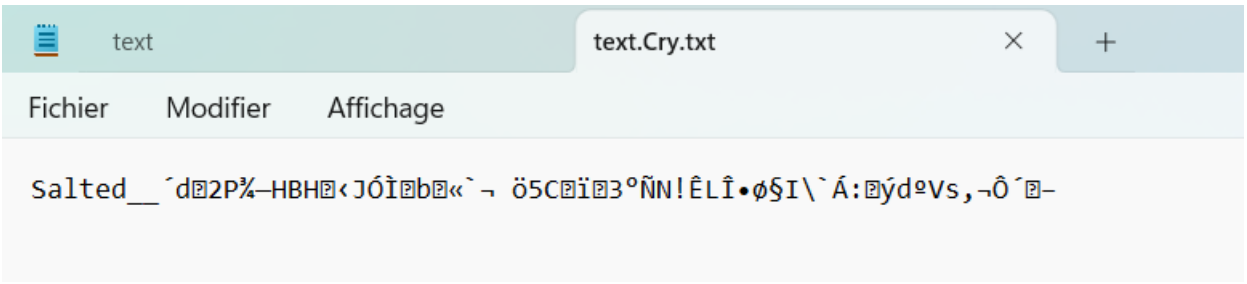
1) Le chiffrement symétrique est un méthode de chiffrement qui repose sur le principe que l'initiateur du message crypté possède la même clé que l'individu qui va déchiffrer le message.
2) Ce chiffrement permet de garantir l'authenticité du message clair et la non répudiation du message vis à vis du destinataire.
3) Création du fichier text.txt :



Cryptage du fichier text.txt se trouvant dans C://Openssl/bin

```
C:\OpenSSL\bin>openssl aes-256-cbc -in text.txt -out text.Cry.txt.enc
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.
```

Utilisation de l'« aes-256-cbc » et saisie d'une clé : « fctCryptage ».
4) Le contenu du fichier textCry.txt.des est incompréhensible.

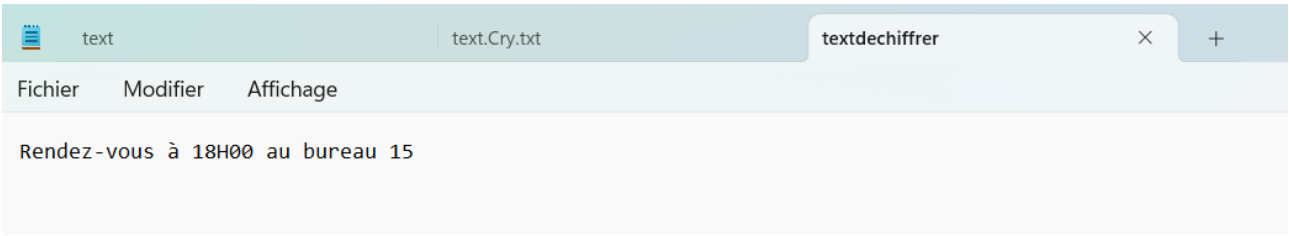


5) Décryptage du fichier textCry.txt.des

```
C:\OpenSSL\bin>openssl aes-256-cbc -a -d -in textCry.txt.des -out text.txt
enter AES-256-CBC decryption password:
```

-Saisie de la ligne de commande pour le déchiffrement en utilisant toujours l'« aes – 256 - cbc ».
Fichier déchiffré : textdechiffrer.txt
-Demande du mots de passe saisi lors du cryptage : cftCryptage

-Décryptage réussi et création du fichier décrypté



Lorsqu'on saisit un mots de passe erroné, le décryptage n'est pas achevé.

```
C:\OpenSSL\bin>openssl aes-256-cbc -a -d -in text.Cry.txt.enc -out textdechi
ffrer.txt
enter AES-256-CBC decryption password:
error reading input file
```

6) Le fichier text.txt et le fichier textdechhiffrer.txt ont le même contenu.
7) L'inconvénient majeur de ce type de cryptage est l'utilisation de la même clé pour le cryptage et le décryptage. Cette situation augmente les risques d'interception et du déchiffrement du message par un individu mal intentionné.
Exercice 2) Chiffrement asymétrique
1) Le chiffrement asymétrique repose sur l'idée que les utilisateurs doivent avoir des clés distincts pour éviter le problème quant à la transmission de cette clé.
Correction : Il s'agit d'une fonction de chiffrement à deux clés :
-une clé publique et une clé privée.
2) Pour deux utilisateurs, il faudrait générer 4 clés. Chaque utilisateur devra avoir deux clés pour pouvoir envoyer et recevoir des messages . De même pour trois utilisateurs, il faudrait six clés. Et pour n utilisateurs, il faudrait 2n clés.
3) Génération d'une clé privée dans un fichier k.priv avec le système de chiffrement RSA :
→ openssl genrsa(rsa) -in k.priv
4)Génération d'une clé publique dans un fichier k.pub
→ openssl rsa(genrsa) -in k.priv -pubout -out k.pub
5)Chiffrement du message « Bonjour » dans un fichier msg.rsa contenu dans le fichier Bonjou .txt à partir de la clé publique
→ openssl pkeyutl -encrypt -in Bonjour.txt -pubin -inkey k.pub -out msg.rsa
t
Notes : -pubout générer une clé publique et -pubin -inkey text.txt utiliser la clé publique se trouvant dans le fichier text.txt
6) Chiffrement à nouveau grâce à la clé publique dans un message msg1.rsa :
→ openssl pkeyutl -encrypt -in Bonjour.txt -pubin -inkey k.pub -out msg1.rsa
Le contenu entre du fichier msg.rsa n'est pas le même que celui du msg1.rsa
7) Commande pour décrypter le contenu du fichier crypté msg.rsa avec le fichier contenant la clé privée dans le fichier Bonjourclair.txt

→ openssl pkeyutl -decrypt -in msg.rsa -inkey k.priv -out Bonjourclair.txt
Notes : - encrypt (- e) pour crypter et - decrypt (- d) pour décrypter
On obtient le même message que celui contenu dans Bonjour.txt
8) Opération inverse :
Création du fichier crypté msg3.rsa à partir de la clé privée au lieu de la clé publique en insérant une signature (-sign).
→ openssl pkeyutl -sign -in Bonjour.txt -inkey k.priv -out msig3.rsa

Déchiffrement du fichier msg3.rsa avec la clé publique avec une vérification du signature (si le contenu de Bonjour1txt est le même qu'avec celui de Bonjour.txt alors l'émetteur est bien vérifié) :
→ openssl pkeyutl -verify -in msg3.rsa -inkey key.pub -out Bonjour1.txt
Exercice 3) Calcul du condensé : MD5 et SHA1
1) Le calcul du haché permet de renforcer la sécurité du message en générant une identité unique par rapport au contenu de celui ci.
2) Le calcul de hachage permet de prouver la propriété de l'intégrité grâce à une identité qui justifie que le contenue du message est intact ou non.
3) calcul du haché du fichier text.txt
→ openssl dgst -md5 -out text_h.txt texte.txt

```
C:\OpenSSL\bin>openssl dgst -md5 -out text_h.txt texte.txt
```

4) Après répétition de la commande plusieurs fois , le condensé du contenu ne diffère pas du premier contenu.
5) En modifiant l'heure du rendez-vous de 18H 30 à 18H 56, on obtient un condensé différent du précédent :
→ openssl dgst -md5 -out text_hh.txt texte.txt

```
C:\OpenSSL\bin>openssl dgst -md5 -out text_hh.txt texte.txt
```

6) La propriété qu'on peut en déduire pour cette catégorie de fonction est la propriété à sens unique. C'est à dire que le sens du hachage est
7) Calcul du haché de text.txt (avec le contenu initial) avec la fonction SHA1
→ openssl dgst -sha1 text_hsha.txt texte.txt

```
C:\OpenSSL\bin>openssl dgst -sha1 -out text_hsha.txt texte.txt
```

8) La différence selon les résultats obtenus, est surtout au niveau des identités générées. L'identité générée avec la fonction sha1 est différente de celle générée avec la fonction md5. La taille de sortie des identités générées est également différente. Pour le sha1, la taille est de 25bytes tandis que pour le md5 ; la taille de sortie est de 128 bytes.

TD6)
Exercice 1) Chiffrement asymétrique
1)
a) La formule utilisée pour le chiffrement est : $c=m^e \bmod(n)$
avec (e, n) la clé de chiffrement (la clé publique)
b) la formule utilisée pour le déchiffrement est : $m = c^d \bmod(n)$
avec (d,n) la clé de déchiffrement (la clé privée)
c)Les valeurs qui doivent rester secrètes parmi celles qui sont proposées sont : d, $\phi(n)$ et n.
2)
a) On génère une clé publique (n, e)
p = 47 et q = 71
n = p*= 47*71 = 3337
$\phi(n) = (p - 1)(q - 1) = (47 - 1)(71 - 1) = 46*70 = 3220$
Soit e tel que : $1<e<\phi(n)$ et pgcd $(\phi(n), e)= 1$
$1<e = 3< \phi(n) =3220$
3220 = 3*1073 + 1 donc pgcd (3220,3) =1
Donc (3219,3337) la clé publique.
b) On génère la clé privée (d, e)
$d*e \bmod \phi(n) = 1$
On a

$d*e + \phi(n)*k = 1$
avec $k = - t \Rightarrow d*e = 1 + \phi(n)*t \Rightarrow d = (1+\phi(n)*t) / e$
A continuer
Exercice 2)
1) Le nombre minimal de clefs symétriques nécessaires est :

$n = n(n-1)/2$

2) Noms d' algorithmes de chiffrement symétrique reconnus : des et aes
3) Le nombre minimal de couples de clefs asymétriques nécessaires est : 2n
4) Pour envoyer des informations chiffrées, Alice doit utiliser la clé privée de Jean (destinataire) et pour recevoir et déchiffrer les même informations, Jean doit utiliser sa clé privée.
Pour signer les informations chiffrées, Alice doit utiliser sa clé privée et Jean pour vérifier les informations ; il doit utiliser la clé publique de Alice.
TD7)

Exercice 1)

1) Jean peut encore envoyer des courriers électroniques avec la clé publique du destinataire mais il ne peut pas en recevoir car il ne peut pas déchiffrer le message reçu car il a perdu sa clé privée.
2)Jean ne peut pas signer les courriers électroniques qu'il envoie car la signature est réalisée avec la clé privée. Par contre, il peut bel et bien vérifier les signatures des courriers électroniques qu'il reçoit en utilisant la clé publique de l'expéditeur du courrier.
3) Pour être à nouveau d'effectuer à nouveau toutes les opérations, Jean doit générer un nouveau couple (clé privée, clé publique).