Exercice dévaluation

PARTIE 1 : Génération d'une paire de clés

📌 Question 1 : Générer une paire de clés de 2048 bits dans e.priv

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl genrsa -out e.priv 2048
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$
```

📌 Question 2 : Voir le contenu au format PEM

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ cat e.priv
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDS/OnMa2yv1a1g
in+uIOq6/OeG5oe5yu0O1MepHpVxdZL0wUcQ7wrz6iIIcJdQfjp3XZWfL3l/n/J7
SSbT1RYOd6lK3mWOkBhq/42lkn9H/DnGtbRVWuepLfrNUsYkIWaq4WuBqZe6d85J
PrWGsCcbKu/1OgtKLloSqUydxZ6RUYKo0w/nAPjoYoACGXOq3QyoOkeJZSa7R00i
EiMcgEY0FExmc0qA3z74E5wE8quu/eMERdlQ0vQ8QNEgAMLbudI7BmJwMbGBNq5j
IuSUT+vKeyO87PNCFmCCqJaKGicWWrSwKhgFG5RhTV4vlN3k4UbvsKtX6WhRrVAH
ShZY8ViBAgMBAAECggEAGdcc4Urk2FIwtkqwIYjphrYzuvtf8xrN1surmmRjBC7U
rUJArGvCaSYrPeJgPdz/1x+wk/ryA/X4ci3+4eHOnj38j21WfdPbEbuzwuwCx7O1
9cbKS8vi5H77oQr6tnEazGHoU0yMhytmKeVSwSiw2Wx0UBHoUREFOhTfC9Govn5G
oeRZ5XPghSyXynt+lTaRsEYkfvm17PKGmLTF3pO0dWZv9JkGsGdi2Ip2ROj0PKRO
```

➡ Le fichier est au format PEM (Privacy Enhanced Mail) , **Le contenu n'est PAS lisible** directement par un humain .C'est comme un code secret qu'il faut décoder pour comprendre.

📌 Question 3 : Afficher d'une manière décodée la paire de clés

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -in e.priv -text -noout
Private-Key: (2048 bit, 2 primes)
modulus:
    00:d2:fc:e9:cc:6b:6c:af:d5:ad:60:8a:7f:ae:20:
    ea:ba:fc:e7:86:e6:87:b9:ca:ed:0e:d4:c7:a9:1e:
    95:71:75:92:f4:c1:47:10:ef:0a:f3:ea:22:08:70:
    97:50:7e:3a:77:5d:95:9f:2f:79:7f:9f:f2:7b:49:
    26:d3:d5:16:0e:77:a9:4a:de:65:8e:90:18:6a:ff:
    8d:a5:92:7f:47:fc:39:c6:b5:b4:55:5a:e7:a9:2d:
    fa:cd:52:c6:24:21:66:aa:e1:6b:81:a9:97:ba:77:
    ce:49:3e:b5:86:b0:27:1b:2a:ef:f5:3a:0b:4a:2e:
    5a:12:a9:4c:9d:c5:9e:91:51:82:a8:d3:0f:e7:00:
    f8:e8:62:80:02:19:73:aa:dd:0c:a8:3a:47:89:65:
    26:bb:47:4d:22:12:23:1c:80:46:34:14:4c:66:73:
```

📌 Question 4 : Quels sont les différents éléments affichés ?

Les différents éléments de la clé **RSA** sont :

1. **Private-Key (2048 bit)**: La taille de la clé
   - *Comme dire : "mon cadenas a 2048 verrous"*
2. **modulus (n)**: Un très grand nombre
   - *C'est le numéro de série unique de ta clé*
3. **publicExponent (e)**: Généralement 65537
   - *Partie utilisée pour chiffrer*
4. **privateExponent (d)**: Un grand nombre secret
   - *La partie vraiment secrète pour déchiffrer*
5. **prime1 (p)**: Un nombre premier secret
   - *Premier ingrédient secret de la recette*
6. **prime2 (q)**: Un autre nombre premier secret
   - *Deuxième ingrédient secret*
7. **exponent1, exponent2, coefficient**: Valeurs pour accélérer les calculs
   - *Des raccourcis mathématiques*

📌 Question 5 : Chiffrer le fichier e.priv

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -in e.priv -out e.priv.enc -aes256
writing RSA key
Enter pass phrase:
Verifying - Enter pass phrase:
```

⇨ *J'ai chiffré ma clé privée avec l'algorithme AES-256 et un mot de passe. Cela ajoute une **couche de protection supplémentaire** : même si quelqu'un vole mon fichier **e.priv.enc**, il ne pourra pas l'utiliser sans connaître mon mot de passe.*

📌 Question 6 : Afficher de nouveau la clé et conclure

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -in e.priv.enc -text -noout
Enter pass phrase for e.priv.enc:
Private-Key: (2048 bit, 2 primes)
modulus:
    00:d2:fc:e9:cc:6b:6c:af:d5:ad:60:8a:7f:ae:20:
    ea:ba:fc:e7:86:e6:87:b9:ca:ed:0e:d4:c7:a9:1e:
    95:71:75:92:f4:c1:47:10:ef:0a:f3:ea:22:08:70:
    97:50:7e:3a:77:5d:95:9f:2f:79:7f:9f:f2:7b:49:
    26:d3:d5:16:0e:77:a9:4a:de:65:8e:90:18:6a:ff:
    8d:a5:92:7f:47:fc:39:c6:b5:b4:55:5a:e7:a9:2d:
```

⇨ **Observation :** Maintenant, quand j'essaie de lire la clé, le système me demande un **mot de passe**.

Conclusion :

- Avant le chiffrement : la clé s'affichait directement
- Après le chiffrement : il faut entrer le mot de passe
- C'est une **double protection**:
  1. La clé **RSA** elle-même est secrète
  2. En plus, elle est protégée par un mot de passe
- C'est comme avoir un coffre (RSA) à l'intérieur d'un autre coffre (AES) !

📌 Question 7 : Exporter la clé publique

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -in e.priv.enc -pubout -out e.pub
Enter pass phrase for e.priv.enc:
writing RSA key
```

📌 Question 8 : Visualiser la clé publique

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ cat e.pub
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0vzpzGtsr9WtYIp/riDq
uvznhuaHucrtDtTHqR6VcXWS9MFHEO8K8+oiCHCXUH46d12Vny95f5/ye0km09UW
DnepSt5ljpAYav+NpZJ/R/w5xrW0VVrnqS36zVLGJCFmquFrgamXunfOST61hrAn
Gyrv9ToLSi5aEqlMncWekVGCqNMP5wD46GKAAhlzqt0MqDpHiWUmu0dNIhIjHIBG
NBRMZnNKgN8++BOcBPKrrv3jBEXZUNL0PEDRIADC27nSOwZicDGxgTauYyLklE/r
ynsjvOzzQhZggqiWihonFlq0sCoYBRuUYU1eL5Td5OFG77CrV+loUa1QB0oWWPFY
gQIDAQAB
-----END PUBLIC KEY-----
```

⇨ La clé publique est aussi en format **PEM,** mais avec les marqueurs `BEGIN PUBLIC KEY` et `END PUBLIC KEY`. Elle est **plus courte** que la clé privée car elle ne contient que la partie publique.

📌 Question 9 : Afficher la clé publique décodée

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -pubin -in e.pub -text -noout
Public-Key: (2048 bit)
Modulus:
    00:d2:fc:e9:cc:6b:6c:af:d5:ad:60:8a:7f:ae:20:
    ea:ba:fc:e7:86:e6:87:b9:ca:ed:0e:d4:c7:a9:1e:
    95:71:75:92:f4:c1:47:10:ef:0a:f3:ea:22:08:70:
    97:50:7e:3a:77:5d:95:9f:2f:79:7f:9f:f2:7b:49:
    26:d3:d5:16:0e:77:a9:4a:de:65:8e:90:18:6a:ff:
    8d:a5:92:7f:47:fc:39:c6:b5:b4:55:5a:e7:a9:2d:
    fa:cd:52:c6:24:21:66:aa:e1:6b:81:a9:97:ba:77:
```

📌 Question 10 : Créer une paire de clés pour le récepteur

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl genrsa -out r.priv 2048
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -in r.priv -out r.priv.enc -aes256
writing RSA key
Enter pass phrase:
Verifying - Enter pass phrase:
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl rsa -in r.priv.enc -pubout -out r.pub
Enter pass phrase for r.priv.enc:
writing RSA key
```

PARTIE 2 : Chiffrement/Déchiffrement avec RSA

📌 Question 1 : Créer un fichier message

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ echo "Bonjour, ceci est mon message secret pour le TP de cryptogra
phie !" > message
```

📌 Question 2 : Crypter le message avec r.pub

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl pkeyutl -encrypt -pubin -inkey r.pub -in message -out mess
agecrypt
```

📌 Question 3 : Décrypter le message avec r.priv

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl pkeyutl -decrypt -inkey r.priv.enc -in messagecrypt -out m
essagedecrypt
Enter pass phrase for r.priv.enc:
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$
```

Vérifier :

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ cat messagedecrypt
Bonjour, ceci est mon message secret pour le TP de cryptographie !
```

📌 Question 4 : Comparer message et messagedecrypt

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ diff message messagedecrypt
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$
```

⇨ Les deux fichiers sont **parfaitement identiques**. Cela prouve que :

- Le chiffrement avec la clé publique a fonctionné
- Le déchiffrement avec la clé privée correspondante a réussi
- **Aucune donnée n'a été perdue** dans le processus

📌 Question 5 : Signer le message avec e.priv

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl dgst -sha256 -sign e.priv.enc -out messagesigne message
Enter pass phrase for e.priv.enc:
```

📌 Question 6 : Vérifier la signature avec e.pub

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ openssl dgst -sha256 -verify e.pub -signature messagesigne message
Verified OK
```

📌 Question 7 : Comparer messagedecrypt et le message vérifié

```
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ cat message
Bonjour, ceci est mon message secret pour le TP de cryptographie !
hanen@hanen-VMware-Virtual-Platform:~/exCrypto$ cat messagedecrypt
Bonjour, ceci est mon message secret pour le TP de cryptographie !
```

⇨ Ce TP démontre les trois piliers de la sécurité informatique : la **confidentialité** par chiffrement RSA (seul le destinataire peut lire), **l'authenticité** par signature numérique (prouve l'identité de l'expéditeur), et **l'intégrité** (détecte toute modification). Ces mécanismes sont exactement ceux utilisés dans les emails sécurisés, les transactions bancaires, les certificats SSL/TLS et les applications de messagerie chiffrée comme WhatsApp.