Chiffrement Symétrique avec AES

1. Génération d'une clé symétrique

Vous pouvez générer une clé symétrique aléatoire en utilisant OpenSSL. Pour AES-256, nous avons besoin de 32 octets (256 bits).

```
openssl rand -hex 32 > key.txt
cat key.txt
```

2. Chiffrement d'un fichier avec AES

Utilisez la clé symétrique pour chiffrer un fichier.

```
openssl enc -aes-256-cbc -salt -in fichier.txt -out fichier.txt.enc -
pass file:./key.txt
```

Explications :

- `-aes-256-cbc` : Algorithme de chiffrement AES avec une clé de 256 bits en mode CBC.
- `-salt` : Utilise un sel pour renforcer la sécurité.
- `-in fichier.txt` : Fichier d'entrée à chiffrer.
- `-out fichier.txt.enc` : Fichier de sortie chiffré.
- `-pass file:./key.txt` : Utilise la clé contenue dans `key.txt`.

3. Déchiffrement du fichier avec AES

Pour déchiffrer le fichier chiffré, utilisez la même clé symétrique.

```
openssl enc -d -aes-256-cbc -in fichier.txt.enc -out
fichier_dechiffre.txt -pass file:./key.txt
```

Explications :

- `-d` : Mode déchiffrement.
- `-in fichier.txt.enc` : Fichier chiffré d'entrée.
- `-out fichier_dechiffre.txt` : Fichier de sortie déchiffré.

Chiffrement Asymétrique avec RSA

1. Génération d'une paire de clés RSA

Générez une paire de clés RSA (clé privée et clé publique).

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt
rsa_keygen_bits:2048
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

Explications :

- `genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048` : Génère une clé privée RSA de 2048 bits.
- `rsa -pubout -in private_key.pem -out public_key.pem` : Extrait la clé publique de la clé privée.

2. Chiffrement d'un message avec la clé publique

Utilisez la clé publique pour chiffrer un fichier.

```
openssl rsautl -encrypt -inkey public_key.pem -pubin -in message.txt -
out message.enc
```

Explications :

- `rsautl -encrypt -inkey public_key.pem -pubin` : Utilise la clé publique pour chiffrer.
- `-in message.txt` : Fichier d'entrée à chiffrer.
- `-out message.enc` : Fichier de sortie chiffré.

3. Déchiffrement du message avec la clé privée

Utilisez la clé privée pour déchiffrer le fichier chiffré.

```
openssl rsautl -decrypt -inkey private_key.pem -in message.enc -out
message_dechiffre.txt
```

Explications :

- `rsautl -decrypt -inkey private_key.pem` : Utilise la clé privée pour déchiffrer.
- `-in message.enc` : Fichier chiffré d'entrée.
- `-out message_dechiffre.txt` : Fichier de sortie déchiffré.

Utilisation Combinée (Hybrid Encryption)

Pour combiner les avantages des deux méthodes (symétrique pour la rapidité et asymétrique pour la sécurité de l'échange de clés), nous allons utiliser un chiffrement hybride.

1. Chiffrement de la clé symétrique avec la clé publique RSA

Chiffrez la clé symétrique générée avec la clé publique RSA.

```
openssl rsautl -encrypt -inkey public_key.pem -pubin -in key.txt -out
key.enc
```

2. Déchiffrement de la clé symétrique avec la clé privée RSA

Déchiffrez la clé symétrique chiffrée avec la clé privée RSA.

```
openssl rsautl -decrypt -inkey private_key.pem -in key.enc -out
key_dechiffree.txt
```

Workflow Complet de Chiffrement Hybride

1. **Génération de la clé symétrique** :

   ```
   openssl rand -hex 32 > key.txt
   ```

2. **Chiffrement du fichier avec la clé symétrique** :

   ```
   openssl enc -aes-256-cbc -salt -in fichier.txt -out fichier.txt.enc
   -pass file:./key.txt
   ```

3. **Chiffrement de la clé symétrique avec la clé publique RSA** :

   ```
   openssl rsautl -encrypt -inkey public_key.pem -pubin -in key.txt -
   out key.enc
   ```

4. **Transmission du fichier chiffré et de la clé chiffrée**.

5. **Réception et déchiffrement de la clé symétrique avec la clé privée RSA** :

   ```
   openssl rsautl -decrypt -inkey private_key.pem -in key.enc -out
   key_dechiffree.txt
   ```

6. **Déchiffrement du fichier avec la clé symétrique déchiffrée** :

   ```
   
   ```

```
openssl enc -d -aes-256-cbc -in fichier.txt.enc -out
fichier_dechiffre.txt -pass file:./key_dechiffree.txt
```

Conclusion

Cette démonstration montre comment utiliser le chiffrement symétrique pour protéger les données avec une clé unique et comment utiliser le chiffrement asymétrique pour sécuriser l'échange de cette clé. En combinant ces deux techniques, vous bénéficiez de la rapidité du chiffrement symétrique et de la sécurité du chiffrement asymétrique pour les échanges de clés, réalisant ainsi une solution de chiffrement hybride robuste.