

# RÉFÉRENCE RAPIDE OPENSLL

Certificats, clés, chiffrement et débogage

## Certificats

### Afficher les détails d'un certificat

```
openssl x509 -in cert.pem -text -noout
openssl x509 -in cert.pem -subject -noout
openssl x509 -in cert.pem -dates -noout
openssl x509 -in cert.pem -issuer -noout
```

### Convertir les formats

```
# PEM vers DER
openssl x509 -in cert.pem -outform DER \
-out cert.der
# DER vers PEM
openssl x509 -in cert.der -inform DER \
-out cert.pem
```

### Formats courants

**PEM** Base64 encodé, `-----BEGIN CERTIFICATE-----`  
**DER** Format binaire, compact  
**PFX / P12** Bundle PKCS#12 (cert + clé + chaîne)  
**CRT / CER** Fichier de certificat (généralement PEM ou DER)

## Génération de clés

### Clés RSA

```
openssl genrsa -out key.pem 4096
openssl rsa -in key.pem -pubout \
-out pubkey.pem
openssl rsa -in key.pem -text -noout
```

### Clés EC

```
openssl ecparam -genkey -name prime256v1 \
-out ec_key.pem
openssl ec -in ec_key.pem -pubout \
-out ec_pub.pem
```

### Clés Ed25519

```
openssl genpkey -algorithm Ed25519 \
-out ed25519_key.pem
openssl pkey -in ed25519_key.pem -pubout \
-out ed25519_pub.pem
```

### Comparaison des algorithmes de clés

**RSA 2048/4096** Largement supporté, clés plus grandes  
**ECDSA (P-256)** Clés plus petites, plus rapide, TLS moderne  
**Ed25519** Le plus rapide, le plus petit, non supporté partout

## CSR

### Générer une CSR

```
openssl req -new -key key.pem \
-out request.csr
# Non interactif
openssl req -new -key key.pem -out req.csr \
-subj "/CN=example.com/O=MyOrg/C=US"
```

### Générer clé + CSR ensemble

```
openssl req -new -newkey rsa:4096 \
-nodes -keyout key.pem -out req.csr \
-subj "/CN=example.com"
```

### Inspecter une CSR

```
openssl req -in request.csr -text -noout
openssl req -in request.csr -verify -noout
```

### Champs CSR courants

**CN** Nom commun (domaine ou nom d'hôte)  
**O** Nom de l'organisation  
**OU** Unité organisationnelle  
**C** Pays (code à 2 lettres)  
**ST** Etat ou province  
**L** Localité / ville

## Auto-signé

### Certificat auto-signé rapide

```
openssl req -x509 -newkey rsa:4096 -nodes \
-keyout key.pem -out cert.pem -days 365 \
-subj "/CN=localhost"
```

### Avec SAN (Subject Alternative Name)

```
openssl req -x509 -newkey rsa:4096 -nodes \
-keyout key.pem -out cert.pem -days 365 \
-subj "/CN=myapp.local" \
-addext "subjectAltName=\
DNS:myapp.local,DNS:*.myapp.local,IP:127.0.0.1"
```

### Depuis une clé existante

```
openssl req -x509 -key key.pem \
-out cert.pem -days 365 \
-subj "/CN=example.com"
```

## Vérification

### Vérifier un certificat

```
openssl verify -CAfile ca.pem cert.pem
openssl verify -CAfile ca.pem \
-untrusted intermediate.pem cert.pem
```

### Vérifier la correspondance clé/cert

```
# Le module doit correspondre pour la clé et le cert
openssl x509 -in cert.pem -modulus -noout
openssl rsa -in key.pem -modulus -noout
openssl req -in req.csr -modulus -noout
```

### Vérifier l'expiration

```
openssl x509 -in cert.pem -checkend 86400
# Retourne 0 si valide pour 86400s (24h)
openssl x509 -in cert.pem -enddate -noout
```

### Certificat d'un serveur distant

```
openssl s_client -connect example.com:443 \
< /dev/null 2> /dev/null \
| openssl x509 -text -noout
```

## Chiffrement

### Chiffrement symétrique

```
openssl enc -aes-256-cbc -salt -pbkdf2 \
-in plain.txt -out encrypted.bin
openssl enc -aes-256-cbc -d -pbkdf2 \
-in encrypted.bin -out plain.txt
```

### Chiffrement asymétrique

```
# Chiffrer avec la clé publique
openssl pkeyutl -encrypt \
-pubin -inkey pub.pem \
-in secret.txt -out secret.enc
# Déchiffrer avec la clé privée
openssl pkeyutl -decrypt \
-inkey key.pem \
-in secret.enc -out secret.txt
```

### Chiffrements courants

**aes-256-cbc** AES 256 bits, mode CBC (défaut courant)  
**aes-256-gcm** AES 256 bits, mode GCM (authentifié)  
**chacha20-poly1305** Chiffrement de flux moderne (rapide sur ARM)

Lister tous : `openssl enc -list`

## Hachage

### Hachage de fichiers

```
openssl dgst -sha256 file.txt
openssl dgst -sha512 file.txt
openssl dgst -md5 file.txt # héritage uniquement
```

### HMAC

```
openssl dgst -sha256 -hmac "secret" file.txt
echo -n "message" | openssl dgst \
-sha256 -hmac "mykey"
```

### Algorithmes de hachage

**SHA-256** Choix standard pour les vérifications d'intégrité  
**SHA-384 / SHA-512** Variantes SHA-2 plus robustes  
**SHA3-256** Dernière norme (basée sur Keccak)  
**MD5** Casé, héritage uniquement — ne pas utiliser pour la sécurité  
**BLAKE2** Alternative rapide et sécurisée (si supporté)

## S/MIME

### Signer un e-mail

```
openssl smime -sign -in msg.txt \
-signer cert.pem -inkey key.pem \
-out signed.msg
```

### Vérifier un e-mail signé

```
openssl smime -verify -in signed.msg \
-CAfile ca.pem -out original.txt
```

### Chiffrer / déchiffrer un e-mail

```
# Chiffrer pour le destinataire
openssl smime -encrypt -aes256 \
-in msg.txt -out encrypted.msg \
-recipient cert.pem
# Déchiffrer
openssl smime -decrypt -in encrypted.msg \
-recv cert.pem -inkey key.pem
```

## Débogage

### Tester une connexion TLS

```
openssl s_client -connect host:443
openssl s_client -connect host:443 \
-servername example.com # SNI
openssl s_client -connect host:443 \
-tls1_2 # forcer TLS 1.2
```

### Afficher la chaîne de certificats

```
openssl s_client -connect host:443 \
-showcerts < /dev/null
```

### Vérifier les chiffrements TLS

```
openssl ciphers -v 'HIGH:!aNULL'
openssl s_client -connect host:443 \
-cipher 'ECDHE-RSA-AES256-GCM-SHA384'
```

### Opérations PKCS#12

```
# Créer un bundle PFX
openssl pkcs12 -export -out bundle.pfx \
-inkey key.pem -in cert.pem -certfile ca.pem
# Extraire depuis PFX
openssl pkcs12 -in bundle.pfx -nodes \
-out all.pem
```

## Motifs courants

### Générer des données aléatoires sécurisées

```
openssl rand -hex 32 # 32 octets aléatoires, hex
openssl rand -base64 24 # 24 octets aléatoires, b64
```

### Encoder / décoder en Base64

```
openssl base64 -in file.bin -out file.b64
openssl base64 -d -in file.b64 -out file.bin
```

### Hachage de mot de passe

```
openssl passwd -6 -salt xyz "password"
# -6 = SHA-512, -5 = SHA-256, -1 = MD5
```

### Aide-mémoire : clé + cert + vérification

```
openssl req -x509 -newkey rsa:4096 -nodes \
-keyout k.pem -out c.pem -days 365 \
-subj "/CN=test"
openssl x509 -in c.pem -text -noout
```