OpenSSL 快速参考

证书、密钥、加密与调试

证书

查看证书详情

```
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
```

格式转换

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

常见格式

PEM	Base64 编码，**-----BEGIN CERTIFICATE-----**
DER	二进制格式，体积紧凑
PFX / P12	PKCS#12 包（cert + key + chain）
CRT / CER	证书文件（通常为 PEM 或 DER）

密钥生成

RSA 密钥

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

EC 密钥

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

Ed25519 密钥

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

密钥算法对比

RSA 2048/4096	兼容性最广，密钥较大
ECDSA (P-256)	密钥更小，速度更快，现代 TLS 首选
Ed25519	最快最小，并非所有系统均支持

CSR

生成 CSR

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

同时生成密钥和 CSR

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

查看 CSR

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

常见 CSR 字段

CN	通用名称（域名或主机名）
O	组织名称
OU	组织单位
C	国家（两字母代码）
ST	省份或州
L	城市

自签名证书

快速生成自签名证书

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

带 SAN 的自签名证书

```
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"
```

使用已有密钥

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

验证

验证证书

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

检查密钥与证书是否匹配

```
# 模数必须一致
openssl x509 -in cert.pem -modulus -noout
openssl rsa -in key.pem -modulus -noout
openssl req -in req.csr -modulus -noout
```

检查过期时间

```
openssl x509 -in cert.pem -checkend 86400
# 若在 86400s (24h) 内有效则返回 0
openssl x509 -in cert.pem -enddate -noout
```

远程服务器证书

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

加密

对称加密

```
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
```

非对称加密

```
# 用公钥加密
openssl pkeyutl -encrypt \
    -pubin -inkey pub.pem \
    -in secret.txt -out secret.enc
# 用私钥解密
openssl pkeyutl -decrypt \
    -inkey key.pem \
    -in secret.enc -out secret.txt
```

常见加密算法

aes-256-cbc	AES 256 位 CBC 模式（常用默认）
aes-256-gcm	AES 256 位 GCM 模式（带认证）
chacha20-poly1305	现代流密码（ARM 上速度快）

列出所有算法：`openssl enc -list`

哈希

文件哈希

```
openssl dgst -sha256 file.txt
openssl dgst -sha512 file.txt
openssl dgst -md5 file.txt  # 仅用于遗留场景
```

HMAC

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

哈希算法

SHA-256	完整性校验的标准选择
SHA-384 / SHA-512	更强的 SHA-2 变体
SHA3-256	最新标准（基于 Keccak）
MD5	已不安全，仅用于遗留场景
BLAKE2	快速、安全的替代方案（如支持）

S/MIME

签名邮件

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

验证签名邮件

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

加密/解密邮件

```
# 为收件人加密
openssl smime -encrypt -aes256 \
    -in msg.txt -out encrypted.msg \
    recipient_cert.pem
# 解密
openssl smime -decrypt -in encrypted.msg \
    -recip cert.pem -inkey key.pem
```

调试

测试 TLS 连接

```
openssl s_client -connect host:443
openssl s_client -connect host:443 \
    -servername example.com  # SNI
openssl s_client -connect host:443 \
    -tls1_3  # 强制 TLS 1.3
```

OpenSSL 快速参考

查看证书链

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

检查 TLS 密码套件

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

PKCS#12 操作

```
# 创建 PFX 包
openssl pkcs12 -export -out bundle.pfx \
    -inkey key.pem -in cert.pem -certfile ca.pem
# 从 PFX 提取
openssl pkcs12 -in bundle.pfx -nodes \
    -out all.pem
```

常用模式

生成安全随机数

```
openssl rand -hex 32    # 32 字节随机数，hex 格式
openssl rand -base64 24 # 24 字节随机数，base64 格式
```

Base64 编解码

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

密码哈希

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

快速速查：密钥 + 证书 + 验证

```
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
```