OpenSSL クイックリファレンス

証明書、鍵、暗号化、デバッグ

証明書

証明書の詳細を表示

```
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
```

フォーマット変換

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

一般的なフォーマット

PEM	Base64 エンコード、**-----BEGIN CERTIFICATE-----**
DER	バイナリフォーマット、コンパクト
PFX / P12	PKCS#12 バンドル（証明書＋鍵＋チェーン）
CRT / CER	証明書ファイル（通常 PEM または DER）

鍵の生成

RSA 鍵

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

EC 鍵

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

Ed25519 鍵

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

鍵アルゴリズムの比較

RSA 2048/4096	広くサポートされている、大きな鍵
ECDSA（P-256）	小さな鍵、高速、モダン TLS
Ed25519	最速、最小、全システムでサポートされていない

CSR

CSR の生成

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

鍵と CSR をまとめて生成

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

CSR の確認

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

CSR の一般的なフィールド

CN	コモンネーム（ドメインまたはホスト名）
O	組織名
OU	組織単位
C	国（2 文字コード）
ST	都道府県
L	地域/市区町村

自己署名証明書

クイック自己署名証明書

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

SAN あり（サブジェクト代替名）

```
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"
```

既存の鍵から

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

検証

証明書の検証

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

鍵と証明書の一致確認

```
# モジュラスが一致することを確認
openssl x509 -in cert.pem -modulus -noout
openssl rsa -in key.pem -modulus -noout
openssl req -in req.csr -modulus -noout
```

有効期限の確認

```
openssl x509 -in cert.pem -checkend 86400
# 86400秒（24時間）有効な場合は0を返す
openssl x509 -in cert.pem -enddate -noout
```

リモートサーバーの証明書

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

暗号化

対称暗号化

```
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
```

非対称暗号化

```
# 公開鍵で暗号化
openssl pkeyutl -encrypt \
    -pubin -inkey pub.pem \
    -in secret.txt -out secret.enc
# 秘密鍵で復号
openssl pkeyutl -decrypt \
    -inkey key.pem \
    -in secret.enc -out secret.txt
```

一般的な暗号スイート

aes-256-cbc	AES 256 ビット、CBC モード（一般的なデフォルト）
aes-256-gcm	AES 256 ビット、GCM モード（認証付き）
chacha20-poly1305	モダンなストリーム暗号（ARM で高速）

全一覧：`openssl enc -list`

ハッシュ

ファイルハッシュ

```
openssl dgst -sha256 file.txt
openssl dgst -sha512 file.txt
openssl dgst -md5 file.txt  # レガシーのみ
```

HMAC

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

ハッシュアルゴリズム

SHA-256	完全性チェックの標準的な選択
SHA-384 / SHA-512	より強力な SHA-2 バリアント
SHA3-256	最新の標準（Keccak ベース）
MD5	破綻済み、レガシーのみ — セキュリティ用途は不可
BLAKE2	高速で安全な代替（サポートされている場合）

S/MIME

メールへの署名

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

署名済みメールの検証

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

メールの暗号化/復号

```
# 受信者のために暗号化
openssl smime -encrypt -aes256 \
    -in msg.txt -out encrypted.msg \
    recipient_cert.pem
# 復号
openssl smime -decrypt -in encrypted.msg \
    -recip cert.pem -inkey key.pem
```

デバッグ

TLS 接続のテスト

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

OpenSSL クイックリファレンス

証明書チェーンの表示

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

TLS 暗号スイートの確認

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

PKCS#12 操作

```
# PFXバンドルの作成
openssl pkcs12 -export -out bundle.pfx \
    -inkey key.pem -in cert.pem -certfile ca.pem
# PFXからの展開
openssl pkcs12 -in bundle.pfx -nodes \
    -out all.pem
```

よくあるパターン

セキュアな乱数を生成

```
openssl rand -hex 32    # 32バイトのランダム値（16進数）
openssl rand -base64 24 # 24バイトのランダム値（Base64）
```

Base64 エンコード/デコード

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

パスワードハッシュ

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

クイックチート：鍵＋証明書＋検証

```
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
```