OpenSSL 빠른 참조

인증서, 키, 암호화, 디버깅

인증서

인증서 상세 보기

```
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
```

형식 변환

```
# 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
```

일반 형식

PEM	Base64 인코딩, **-----BEGIN CERTIFICATE-----**
DER	바이너리 형식, 컴팩트
PFX / P12	PKCS#12 번들 (인증서 + 키 + 체인)
CRT / CER	인증서 파일 (보통 PEM 또는 DER)

키 생성

RSA 키

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

EC 키

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

Ed25519 키

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

키 알고리즘 비교

RSA 2048/4096	광범위하게 지원, 키 크기 큼
ECDSA (P-256)	키 작음, 빠름, 최신 TLS
Ed25519	가장 빠르고 작음, 일부 시스템 미지원

CSR

CSR 생성

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

키 + CSR 동시 생성

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

CSR 검사

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

일반 CSR 필드

CN	공통 이름 (도메인 또는 호스트명)
O	조직명
OU	조직 단위
C	국가 (2자리 코드)
ST	주 또는 도
L	지역 / 도시

자체 서명 인증서

빠른 자체 서명 인증서

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

SAN (주체 대체 이름) 포함

```
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"
```

기존 키에서 생성

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

검증

인증서 검증

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

키 / 인증서 일치 확인

```
# Modulus must match for key and cert
openssl x509 -in cert.pem -modulus -noout
openssl rsa -in key.pem -modulus -noout
openssl req -in req.csr -modulus -noout
```

만료 확인

```
openssl x509 -in cert.pem -checkend 86400
# Returns 0 if valid for 86400s (24h)
openssl x509 -in cert.pem -enddate -noout
```

원격 서버 인증서

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

암호화

대칭 암호화

```
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
```

비대칭 암호화

```
# Encrypt with public key
openssl pkeyutl -encrypt \
    -pubin -inkey pub.pem \
    -in secret.txt -out secret.enc
# Decrypt with private key
openssl pkeyutl -decrypt \
    -inkey key.pem \
    -in secret.enc -out secret.txt
```

일반 암호

aes-256-cbc	AES 256비트, CBC 모드 (일반적 기본값)
aes-256-gcm	AES 256비트, GCM 모드 (인증됨)
chacha20-poly1305	최신 스트림 암호 (ARM에서 빠름)

전체 목록: `openssl enc -list`

해싱

파일 해시

```
openssl dgst -sha256 file.txt
openssl dgst -sha512 file.txt
openssl dgst -md5 file.txt  # legacy only
```

HMAC

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

해시 알고리즘

SHA-256	무결성 확인을 위한 표준 선택
SHA-384 / SHA-512	더 강력한 SHA-2 변형
SHA3-256	최신 표준 (Keccak 기반)
MD5	취약, 레거시만 — 보안용으로 사용 금지
BLAKE2	빠르고 안전한 대안 (지원 시)

S/MIME

이메일 서명

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

서명된 이메일 검증

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

이메일 암호화 / 복호화

```
# Encrypt for recipient
openssl smime -encrypt -aes256 \
    -in msg.txt -out encrypted.msg \
    recipient_cert.pem
# Decrypt
openssl smime -decrypt -in encrypted.msg \
    -recip cert.pem -inkey key.pem
```

디버깅

TLS 연결 테스트

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

OpenSSL 빠른 참조

인증서 체인 표시

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

TLS 암호 확인

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

PKCS#12 작업

```
# Create PFX bundle
openssl pkcs12 -export -out bundle.pfx \
    -inkey key.pem -in cert.pem -certfile ca.pem
# Extract from PFX
openssl pkcs12 -in bundle.pfx -nodes \
    -out all.pem
```

일반 패턴

보안 난수 생성

```
openssl rand -hex 32    # 32 random bytes, hex
openssl rand -base64 24 # 24 random bytes, b64
```

Base64 인코딩 / 디코딩

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

패스워드 해싱

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

빠른 치트: 키 + 인증서 + 검증

```
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
```