

OpenSSL Riferimento Rapido

Certificati, chiavi, cifratura e debug

Certificati

Visualizza Dettagli Certificato

```
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
```

Converti Formati

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

Formati Comuni

PEM	Base64-encoded, -----BEGIN CERTIFICATE-----
DER	Formato binario, compatto
PFX / P12	Bundle PKCS#12 (cert + chiave + chain)
CRT / CER	File certificato (solitamente PEM o DER)

Generazione di Chiavi

Chiavi RSA

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

Chiavi EC

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

Chiavi Ed25519

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

Confronto Algoritmi di Chiave

RSA 2048/4096	Ampiamente supportato, chiavi più grandi
ECDSA (P-256)	Chiavi più piccole, più veloce, TLS moderno
Ed25519	Più veloce e compatto, non disponibile ovunque

CSR

Genera CSR

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

Genera Chiave + CSR Insieme

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

Ispeziona CSR

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

Campi CSR Comuni

CN	Common Name (dominio o hostname)
O	Nome dell'organizzazione
OU	Unità organizzativa
C	Paese (codice a 2 lettere)
ST	Stato o provincia
L	Località / città

Certificati Auto-Firmati

Certificato Auto-Firmato Rapido

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

Con 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"
```

Da Chiave Esistente

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

Verifica

Verifica Certificato

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

Verifica Corrispondenza Chiave/Cert

```
# Il modulus deve coincidere tra chiave e cert
openssl x509 -in cert.pem -modulus -noout
openssl rsa -in key.pem -modulus -noout
openssl req -in req.csr -modulus -noout
```

Controlla Scadenza

```
openssl x509 -in cert.pem -checkend 86400
# Restituisce 0 se valido per 86400s (24h)
openssl x509 -in cert.pem -enddate -noout
```

Certificato Server Remoto

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

Cifratura

Cifratura Simmetrica

```
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
```

Cifratura Asimmetrica

```
# Cifra con chiave pubblica
openssl pkeyutl -encrypt \
-pubin -inkey pub.pem \
-in secret.txt -out secret.enc
# Decifra con chiave privata
openssl pkeyutl -decrypt \
-inkey key.pem \
-in secret.enc -out secret.txt
```

Cifrari Comuni

aes-256-cbc	AES 256-bit, modalità CBC (default comune)
aes-256-gcm	AES 256-bit, modalità GCM (autenticata)
chacha20-poly1305	Cifrario a flusso moderno (veloce su ARM)

Elenca tutti: `openssl enc -list`

Hashing

Hash di File

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

HMAC

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

Algoritmi di Hash

SHA-256	Scelta standard per controlli di integrità
SHA-384 / SHA-512	Varianti SHA-2 più robuste
SHA3-256	Standard più recente (basato su Keccak)
MD5	Compromesso, solo legacy — non usare per sicurezza
BLAKE2	Alternativa veloce e sicura (se supportata)

S/MIME

Firma Email

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

Verifica Email Firmata

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

Cifra / Decifra Email

```
# Cifra per il destinatario
openssl smime -encrypt -aes256 \
-in msg.txt -out encrypted.msg \
recipient_cert.pem
# Decifra
openssl smime -decrypt -in encrypted.msg \
-recv cert.pem -inkey key.pem
```

Debug

Testa la Connessione TLS

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

OpenSSL Riferimento Rapido

Mostra la Chain dei Certificati

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

Controlla i Cipher TLS

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

Operazioni PKCS#12

```
# Crea bundle PFX  
openssl pkcs12 -export -out bundle.pfx \  
-inkey key.pem -in cert.pem -certfile ca.pem  
# Estrai da PFX  
openssl pkcs12 -in bundle.pfx -nodes \  
-out all.pem
```

Pattern Comuni

Genera Casualità Sicura

```
openssl rand -hex 32 # 32 byte casuali, hex  
openssl rand -base64 24 # 24 byte casuali, b64
```

Codifica / Decodifica Base64

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

Hashing della Password

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

Cheat Rapido: Chiave + Cert + Verifica

```
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
```