

Referencia Rápida de OpenSSL

Certificados, claves, cifrado y depuración

Certificados

Ver Detalles del Certificado

```
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 Formatos

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

Formatos Comunes

PEM	Codificado en Base64, -----BEGIN CERTIFICATE-----
DER	Formato binario, compacto
PFX / P12	Paquete PKCS#12 (certificado + clave + cadena)
CRT / CER	Archivo de certificado (normalmente PEM o DER)

Generación de Claves

Claves RSA

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

Claves EC

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

Claves Ed25519

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

Comparativa de Algoritmos de Clave

RSA 2048/4096	Amplio soporte, claves más grandes
ECDSA (P-256)	Claves más pequeñas, más rápido, TLS moderno
Ed25519	Más rápido, más pequeño, no disponible en todos los sistemas

CSR

Generar CSR

```
openssl req -new -key key.pem \
-out request.csr
# No interactivo
openssl req -new -key key.pem -out req.csr \
-subj "/CN=example.com/O=MyOrg/C=US"
```

Generar Clave + CSR Juntos

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

Inspeccionar CSR

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

Campos Comunes de CSR

CN	Nombre Común (dominio o nombre de host)
O	Nombre de la organización
OU	Unidad organizativa
C	País (código de 2 letras)
ST	Estado o provincia
L	Localidad / ciudad

Autofirmados

Certificado Autofirmado Rápido

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

Con SAN (Nombre Alternativo de Sujeto)

```
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"
```

Desde Clave Existente

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

Verificación

Verificar Certificado

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

Verificar Coincidencia Clave/Certificado

```
# El módulo debe coincidir para clave y certificado
openssl x509 -in cert.pem -modulus -noout
openssl rsa -in key.pem -modulus -noout
openssl req -in req.csr -modulus -noout
```

Verificar Expiración

```
openssl x509 -in cert.pem -checkend 86400
# Retorna 0 si válido por 86400s (24h)
openssl x509 -in cert.pem -enddate -noout
```

Certificado de Servidor Remoto

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

Cifrado

Cifrado Simétrico

```
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
```

Cifrado Asimétrico

```
# Cifrar con clave pública
openssl pkeyutl -encrypt \
-pubin -inkey pub.pem \
-in secret.txt -out secret.enc
# Descifrar con clave privada
openssl pkeyutl -decrypt \
-inkey key.pem \
-in secret.enc -out secret.txt
```

Cifrados Comunes

aes-256-cbc	AES 256 bits, modo CBC (predeterminado común)
aes-256-gcm	AES 256 bits, modo GCM (autenticado)
chacha20-poly1305	Cifrado de flujo moderno (rápido en ARM)

Listar todos: `openssl enc -list`

Hashing

Hashes de Archivos

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

HMAC

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

Algoritmos de Hash

SHA-256	Opción estándar para verificaciones de integridad
SHA-384 / SHA-512	Variantes más fuertes de SHA-2
SHA3-256	Último estándar (basado en Keccak)
MD5	Roto, solo legado — no usar para seguridad
BLAKE2	Alternativa rápida y segura (si está disponible)

S/MIME

Firmar Correo

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

Verificar Correo Firmado

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

Cifrar / Descifrar Correo

```
# Cifrar para destinatario
openssl smime -encrypt -aes256 \
-in msg.txt -out encrypted.msg \
recipient_cert.pem
# Descifrar
openssl smime -decrypt -in encrypted.msg \
-recv cert.pem -inkey key.pem
```

Depuración

Probar Conexión 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 # forzar TLS 1.3
```

Referencia Rápida de OpenSSL

Mostrar Cadena de Certificados

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

Verificar Cifrados TLS

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

Operaciones PKCS#12

```
# Crear paquete PFX  
openssl pkcs12 -export -out bundle.pfx \  
-inkey key.pem -in cert.pem -certfile ca.pem  
# Extraer de PFX  
openssl pkcs12 -in bundle.pfx -nodes \  
-out all.pem
```

Patrones Comunes

Generar Aleatorio Seguro

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

Codificar / Decodificar Base64

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

Hash de Contraseña

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

Atajo: Clave + Certificado + Verificar

```
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
```