OpenSSL Keys, certs, TLS

CONCEPTS

Usage is openssl <subcommand> <options>. Most subcommands read stdin and write stdout, so they pipe into each other. Files are PEM (base64 text) unless you ask for DER.

Private key genpkey / req -newkey
↓ `req -new`
CSR request with public key
↓ `signed by a CA`
Certificate x509 -text to inspect
↓ `pkcs12 -export`
Bundle .p12 / fullchain.pem

Passwords on the command line show up in ps and shell history. Use -passin/-passout with env:VAR, file:path or stdin instead of pass:.

KEYS

`openssl genpkey -algorithm ED25519 -out <key.pem>`
Generate a private key

`openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -out <key.pem>`
RSA key with chosen size

`openssl genpkey -algorithm EC -pkeyopt ec_paramgen_curve:P-256 -out <key.pem>`
Elliptic curve key

`openssl genpkey -aes-256-cbc -algorithm ED25519 -out <key.pem>`
Key protected by a passphrase

`openssl pkey -in <key.pem> -pubout -out <pub.pem>`
Extract the public key

`openssl pkey -in <key.pem> -text -noout`
Show key components

`openssl pkey -in <key.pem> -aes-256-cbc -out <enc.pem>`
Add a passphrase to a key

`openssl pkey -in <enc.pem> -out <plain.pem>`
Remove a passphrase

`openssl rand -base64 <num>`
Random bytes as base64

KEYS (cont.)

`openssl rand -hex <num>`
Random bytes as hex

REQUESTS AND CERTIFICATES

`openssl req -new -key <key.pem> -out <req.csr>`
CSR from an existing key

`openssl req -new -newkey rsa:2048 -noenc -keyout <key.pem> -out <req.csr>`
New key and CSR together

`openssl req -new -key <key.pem> -subj "/CN=<name>" -addext "subjectAltName=DNS:<name>" -out <req.csr>`
CSR with name and SAN, no prompts

`openssl req -in <req.csr> -noout -text`
Read a CSR

`openssl req -x509 -newkey rsa:4096 -noenc -days <num> -keyout <key.pem> -out <cert.pem>`
Self-signed certificate

`openssl req -x509 -key <key.pem> -days <num> -subj "/CN=<name>" -addext "subjectAltName=DNS:<name>" -out <cert.pem>`
Self-signed cert from a key

INSPECT CERTIFICATES

`openssl x509 -in <cert.pem> -noout -text`
Full certificate details

`openssl x509 -in <cert.pem> -noout -subject -issuer`
Subject and issuer

`openssl x509 -in <cert.pem> -noout -dates`
Validity period

`openssl x509 -in <cert.pem> -noout -enddate`
Expiry date only

`openssl x509 -in <cert.pem> -noout -checkend <secs>`
Exit 1 if expiring within N seconds

INSPECT CERTIFICATES (cont.)

`openssl x509 -in <cert.pem> -noout -ext subjectAltName`
Show the SAN list

`openssl x509 -in <cert.pem> -noout -fingerprint -sha256`
SHA-256 fingerprint

`openssl verify -CAfile <ca.pem> <cert.pem>`
Verify against a CA

`openssl verify -CAfile <ca.pem> -untrusted <inter.pem> <cert.pem>`
Verify with intermediates

`openssl pkey -in <key.pem> -pubout | sha256sum`
Hash of key public part

TEST A TLS SERVER

`openssl s_client -connect <host>:443 -servername <host>`
Open a TLS connection

`openssl s_client -connect <host>:443 -brief </dev/null`
Short connection summary

`openssl s_client -connect <host>:443 -showcerts </dev/null`
Show the full chain sent

`openssl s_client -connect <host>:443 </dev/null | openssl x509 -noout -dates`
Check remote cert expiry

`openssl s_client -connect <host>:443 </dev/null | openssl x509 -out <cert.pem>`
Save the server cert

DIGESTS AND ENCRYPTION

`openssl dgst -sha256 <file>`
SHA-256 of a file

CONVERT FORMATS

`openssl pkcs12 -export -inkey <key.pem> -in <cert.pem> -certfile <chain.pem> -out <bundle.p12>`
Bundle key and cert as PKCS#12