OpenSSL Cheat Sheet

by Alberto González (albertx) via cheatography.com/122237/cs/22629/

BASICS

Checking version
openssl version -a

How fast it runs on the system using four CPU cores and testing RSA algorithm
openssl speed -multi 4 rsa

Get basic help
openssl help

Generate 20 random bytes and show them on screen
openssl rand -hex 20

ENCODING / DECODING

Encoding a file using Base64
openssl base64 -in file.data

Encoding some text using Base64
echo -n "some text" | openssl base64

Base64 decode a file with output to another file
openssl base64 -d -in encoded.data -out decoded.data

WORKING WITH HASHES

List digest algorithms available
openssl list -digest-algorithms

Hash a file using SHA256
openssl dgst -sha256 file.data

Hash a file using SHA256 with its output in binary form (no output hex encoding)
No ASCII or encoded characters will be printed out to the console, just pure bytes. You can append ' | xxd'
openssl dgst -binary -sha256 file.data

Hash text using SHA3-512
echo -n "some text" | openssl dgst -sha3-512

Create HMAC - SHA384 of a file using a specific key in bytes
openssl dgst -SHA384 -mac HMAC -macopt hexkey:369bd7d655 file.data

Create HMAC - SHA512 of some text
echo -n "some text" | openssl dgst -mac HMAC -macopt hexkey:36-9bd7d655 -sha512

ASYMMETRIC ENCRYPTION

ASYMMETRIC ENCRYPTION (cont)

Encrypt a file using RSA public key
openssl rsautl -encrypt -inkey pubkey.key -pubin -in cleartext.file -out ciphertext.file

Decrypt a file using RSA private key
openssl rsautl -decrypt -inkey pub_priv.key -in ciphertext.file -out decrypted.file

Create private key using the P-224 elliptic curve
openssl ecparam -name secp224k1 -genkey -out ecpriv.key

Encrypt private key using 3DES algorithm
openssl ec -in ecP384priv.key -des3 -out ecP384priv_enc.key

SYMMETRIC ENCRYPTION

List all supported symmetric encryption ciphers
openssl enc -list

Encrypt a file using an ASCII encoded password provided and AES-128 algorithm
openssl enc -aes-128-ecb -in cleartext.file -out ciphertext.file -pass pass:thisisthepassword

Decrypt a file using AES-256-CBC and a keyfile
openssl enc -d -aes-256-cbc -in ciphertext.file -out cleartext.file -pass file:./key.file

Encrypt a file using a specific encryption key (K) provided as hex digits
openssl enc -aes-128-ecb -in cleartext.file -out ciphertext.file -K 1881807b2d1b3d22f14e9ec52563d981 -nosalt

Encrypt a file using ARIA 256 in CBC block cipher mode using a specifi encryption key (K:256 bits) and initialization vector (iv:128 bits)
openssl enc -aria-256-cbc -in cleartext.file -out ciphertext.file -K f92d2e986b7a2a01683b4c40d0cbcf6feaa669ef2bb5ec3a25ce85d9548 -iv 470bc29762496046882b61ecee68e07c -nosalt

Encrypt a file using Camellia 192 algorithm in COUNTER block cipher r with key and iv provided
openssl enc -camellia-192-ctr -in cleartext.file -out ciphertext.file -K 6c7a1b3487d28d3bf444186d7c529b48d67dd6206c7a1b34 -iv 470bc29762496046882b61ecee68e07c

DIGITAL SIGNATURES

Generate DSA parameters for the private key. 2048 bits length
openssl dsaparam -out dsaparam.pem 2048

Generate DSA public-private key for signing documents and protect it using AES128 algorithm
openssl gendsa -out dsaprivatekey.pem -aes-128-cbc dsaparam.pem

Copy the public key of the DSA public-private key file to another file
openssl dsa -in dsaprivatekey.pem -pubout -out dsapublickey.pem

List elliptic curves available

openssl ecparam -list_curves

Create 4096 bits RSA public-private key pair

openssl genrsa -out pub_priv.key 4096

Display detailed private key information

openssl rsa -text -in pub_priv.key -noout

Encrypt public-private key pair using AES-256 algorithm

openssl rsa -in pub_priv.key -out encrypted.key -aes256

Remove keys file encryption and save them to another file

openssl rsa -in encrypted.key -out cleartext.key

Copy the public key of the public-private key pair file to another file

openssl rsa -in pub_priv.key -pubout -out pubkey.key

By **Alberto González** (albertx)
cheatography.com/albertx/
albertx.mx/blog/

Published 25th May, 2020.
Last updated 9th June, 2021.

DIGITAL SIGNATURES (cont)

To print out the contents of a DSA key pair file
openssl dsa -in dsaprivatekey.pem -text -noout

Signing the sha-256 hash of a file using RSA private key
openssl dgst -sha256 -sign rsakey.key -out signature.data document.pdf

Verify a SHA-256 file signature using a public key
openssl dgst -sha256 -verify publickey.pem -signature signature.data original.file

Signing the sha3-512 hash of a file using DSA private key
openssl pkeyutl -sign -pkeyopt digest:sha3-512 -in document.docx -inkey dsaprivatekey.pem -out signature.data

Verify DSA signature
openssl pkeyutl -verify -sigfile dsasignature.data -inkey dsakey.pem -in document.docx

Create a private key using P-384 Elliptic Curve
openssl ecparam -name secp384r1 -genkey -out ecP384priv.key

Encrypt private key using 3DES algorithm
openssl ec -in ecP384priv.key -des3 -out ecP384priv_enc.key

Sign a PDF file using Elliptic Curves with the generated key
openssl pkeyutl -sign -inkey ecP384priv_enc.key -pkeyopt digest:sha3-512 -in document.pdf -out signature.data

Verify the file's signature. If it's ok you must receive "Signature Verified Successfully"
openssl pkeyutl -verify -in document.pdf -sigfile signature.data -inkey ecP384priv_enc.key

DIGITAL CERTIFICATES

Generating a CSR file and a 4096 bits RSA key pair
openssl req -newkey rsa:4096 -keyout private.key -out request.csr

Display Certificate Signing Request (CSR) content
openssl req -text -noout -in request.csr

Display the public key contained in the CSR file
openssl req -pubkey -noout -in request.csr

Creating a Certificate Signing Request (CSR) using an existing private key. *This can be useful when you need to renew the public digital certificate without changing the private key.*
openssl req -new -key private.key -out request.csr

Create EC P384 curve parameters file to generate a CSR using Elliptic Curves in the next step.
openssl genpkey -genparam -algorithm EC -out EC_params.pem -pkeyopt ec_paramgen_curve:secp384r1 -pkeyopt ec_param_enc:named_curve

DIGITAL CERTIFICATES (cont)

Create a CSR file using Elliptic Curve P384 parameters file created in the previous step. *Instead of using RSA keys.*
openssl req -newkey ec:EC_params.pem -keyout EC_P384_priv.key -out EC_request.csr

Create a self-signed certificate, a new 2048 bits RSA key pair with one year of validity
openssl req -newkey rsa:2048 -nodes -keyout priv.key -x509 -days 365 -out cert.crt

Create and sign a new certificate using the CSR file and the private key for signing (you must have a openssl.cnf file prepared)
openssl ca -in request.csr -out certificate.crt -config ./CA/config/openssl.cnf

Display PEM format certificate information
openssl x509 -text -noout -in cert.crt

Display certificate information in Abstract Sintax Notation One (ASN.1)
openssl asn1parse -in cert.crt

Extract the certificate's public key
openssl x509 -pubkey -noout -in cert.crt

Extract the public key's modulus in the certificate
openssl x509 -modulus -noout -in cert.crt

Extract the domain certificate from an HTTPS/TLS connection
openssl s_client -connect domain.com:443 | openssl x509 -out certificate.crt

Convert a certificate from PEM to DER format
openssl x509 -inform PEM -outform DER -in cert.crt -out cert.der

Checking whether the certificate pubic key matches a private key and request file. One step per file. Must match in the output hashes.
openssl x509 -modulus -in certificate.crt -noout | openssl dgst -sha256
openssl rsa -modulus -in private.key -noout | openssl dgst -sha256
openssl req -modulus -in request.csr -noout | openssl dgst -sha256

WORKING WITH TLS PROTOCOL

List all cipher suites supported
openssl ciphers -V 'ALL'

List all cipher suites supported with AES
openssl ciphers -V 'AES'

List all cipher suites supporting CAMELLIA & SHA256 algorithms.
openssl ciphers -V 'CAMELLIA+SHA256'

TLS connection to a server using port 443 (HTTPS)
openssl s_client -connect domain.com:443

WORKING WITH TLS PROTOCOL (cont)

TLS connection to a server using v1.2
openssl s_client -tls1_2 -connect domain.com:443

TLS connection & disable v1.0
openssl s_client -no_tls1 domain.com:443

TLS connection using a specific cipher suite
openssl s_client -cipher DHE-RSA-AES256-GCM-SHA384 domain.com:443

TLS connection displaying all certificates provided by server
openssl s_client -showcerts domain.com:443

Setting up a listening port to receive TLS connections using a certificate, the private key & supporting only TLS 1.2
openssl s_server -port 443 -cert cert.crt -key priv.key -tls1_2

Extract the domain certificate from an HTTPS/TLS connection
openssl s_client -connect domain.com:443 | openssl x509 -out certificate.crt

nmap command: Display enabled cipher-suites over an HTTPS/TLS Connection
nmap --script ssl-enum-ciphers -p 443 domain.com

nmap command: Display enabled cipher-suites over a TLS (HTTPS) Connection using SNI. *(change it to desired IP & domain name)*
nmap --script ssl-enum-ciphers --script-args=tls.servername=domain.com 172.67.129.11

PERSONAL SECURITY ENVIRONMENTS (PSE)

Convert a certificate from PEM (base64) to DER (binary) format
openssl x509 -in certificate.pem -outform DER -out certificate.der

Insert certificate & private key into PKCS #12 format file. These files can be imported in windows certificate manager or to a Java Key Store (jks) file
openssl pkcs12 -export -out cert_key.p12 -inkey private.key -in certificate.crt

To show the contents of a PKCS #12 file
openssl pkcs12 -in cert_key.p12

Convert the .p12 file into a Java Key Store. *This commnad uses java keytool instead of openssl.*
keytool -importkeystore -destkeystore javakeystore.jks -srckeystore cert_key.p12 -srcstoretype pkcs12

Convert PEM certificate to PKCS #7 format
openssl crl2pkcs7 -nocrl -certfile certificate.crt -out cert.p7b

Convert a PKCS #7 file from PEM to DER
openssl pkcs7 -in cert.p7b -outform DER -out p7.der

SIMPLE CA CONFIGURATION FILE (openssl.cnf)

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir = ./personalCA
database = $dir/index.txt
new_certs_dir = $dir/newcerts

certificate = $dir/cacert.pem
serial = $dir/serial
rand_serial = yes
private_key = $dir/private/cakey.pem
RANDFILE = $dir/private/.rand

default_days = 365
default_crl_days= 30
default_md = SHA256

policy = policy_any
email_in_dn = no

name_opt = ca_default
cert_opt = ca_default
copy_extensions = none

[ policy_any ]
countryName = supplied
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

FINAL NOTES

- All openssl commands were tested using OpenSSL version 1.1.1f
- All nmap commands were tested using nmap version 7.80. nmap is compiled using openssl libraries.
- The default format for almost all operations in openssl is PEM, however you can always specify a DER format using arguments or export to other formats with appropriate commands indicated on the document.