Generating Public and Private Keys

Generating RSA Keys

Generate 2048 bit RSA Private Key saved as KEY1.pem

```
openssl genrsa  -out KEY1.pem  2048
```

Generate 4096 bit RSA Private Key, encrypted with AES128

```
openssl genrsa  -out KEY2.pem  -aes128  4096
```

- Key size must be last argument of command
- Omit `-out <FILE>` argument to output to StdOut
- Other encryption algorithms are also supported:
  `-aes128, -aes192, -aes256, -des3, -des`

Generating DSA Keys:

Generate DSA Parameters File

```
openssl dsaparam  -out DSA-PARAM.pem  1024
```

Generate DSA Keys file with Parameters file

```
openssl gendsa  -out DSA-KEY.pem  DSA-PARAM.pem
```

Generate DSA Parameters and Keys in one File

```
openssl dsaparam  -genkey  -out DSA-PARAM-KEY.pem  2048
```

See Inspecting section to view file contents.

Generating Elliptic Curve Keys:

Generate EC Parameters file

```
openssl genpkey  -genparam  -algorithm EC  -pkeyopt ec_paramgen_curve:secp384r1  -out EC-PARAM.pem
```

Generate EC Keys from Parameters file

```
openssl genpkey  -paramfile EC-PARAM.pem  -out EC-KEY.pem
```

Generate EC Keys directly

```
openssl genpkey  -algorithm EC  -pkeyopt ec_paramgen_curve:P-384  -out EC-KEY.pem
```

View supported Elliptic Curves

```
openssl ecparam  -list_curves
```

Recommended Curves: `secp521r1`, `secp384r1`, `secp256k1` *(identical to P-521, P-384, P-256)*

Inspecting RSA, DSA, and Elliptic Curve Keys

Inspecting RSA Key Files

Converting an RSA Private Key into text

```
openssl rsa  -in KEY.pem  -noout -text
```

Removing encryption from an RSA key file

```
openssl rsa  -in ENCRYPTED-KEY.pem  -out KEY.pem
```

Encrypting an RSA Key File

```
openssl rsa  -in KEY.pem  -aes128  -out ENCRYPTED-KEY.pem
```

Inspecting DSA Parameters and Keys

Inspecting DSA Parameters file

```
openssl dsaparam  -in DSA-PARAM.pem  -text -noout
```

Inspecting DSA Private Key file

```
openssl dsa  -in DSA-KEY.pem  -text -noout
```

Inspecting EC Parameters and Keys

Inspecting Elliptic Curve (EC) Parameters file

```
openssl ecparam  -in EC-PARAM.pem  -text -noout
```

Inspecting Elliptic Curve (EC) Private Key file

```
openssl ec  -in EC-KEY.pem  -text -noout
```

Inspecting *any* Key file using **pkey** utility

Converting any Private Key file into text (RSA, DSA, or EC)

```
openssl pkey  -in KEY.pem  -noout -text
```

Extracting only Public Key as text from any Key file

```
openssl pkey  -in KEY.pem  -noout -text_pub
```

Extracting only Public Key in PEM format

```
openssl pkey  -in KEY.pem  -pubout
```

`pkey` expects a *Private* Key file. *Public* Key file can be read with `-pubin`

Check if RSA Key matches a CSR or Cert

Compare Modulus values to see if files match each other

```
openssl req   -in CSR.pem   -noout  -modulus
openssl x509  -in CERT.pem  -noout  -modulus
openssl rsa   -in KEY.pem   -noout  -modulus
```

Check if EC Key matches a CSR or Cert

Compare Public Key values to see if files match each other

```
openssl req   -in EC-CSR.pem  -noout  -pubkey
openssl x509  -in EC-CERT.pem -noout  -pubkey
openssl ec    -in EC-KEY.pem  -pubout
```

OpenSSL Cheat Sheet

Presented by
Practical Networking .net

Latest version of this cheat sheet and training on how to use it are available here:
`pracnet.net/openssl`

Want to *really* understand SSL & TLS?
`pracnet.net/tls`

Generating Certificate Signing Requests (CSRs) and Self-Signed Certificates

Generating CSRs:

Generate CSR with *existing* Private Key file
```
openssl req -new  -key KEY.pem  -out CSR.pem
```

Generate CSR and *new* Private Key file
```
openssl req -new  -newkey <alg:opt>  -nodes  -out CSR.pem
```

Generating Self-Signed Certificates

Generate Certificate with *existing* Private Key file
```
openssl req -x509  -key KEY.pem  -out CERT.pem
```

Generate Certificate and *new* Private Key file
```
openssl req -x509  -newkey <alg:opt> -nodes -out CERT.pem
```

Notes / Options

Commands above will prompt you for the Subject Distinguished Name (DN) attributes. Alternatively, you can specify them using `-subj`:

Examples: `-subj "/CN=website.com"` --or-- `-subj "/C=US/ST=Colorado/L=Denver/O=ACME Inc./CN=acme.com"`

`-nodes` - Generate Key File with No DES encryption - Skips prompt for PEM Pass phrase

`-<digest>` - Sign CSR/Cert using `<digest>` hashing algorithm. View supported algorithms: `openssl list --digest-commands`

`-config` - Specify config file with custom options. Default Config file: `openssl.cnf` in directory specified by `openssl version -d`

The argument `-newkey <alg:opt>` lets you create **RSA**, **DSA**, or **EC** Keys:

`-newkey 1024`	- Generate 1024 bit RSA Keys (*legacy*)	`-newkey dsa:DSA-PARAM.pem`	- Generate DSA Keys using DSA Parameters
`-newkey rsa:2048`	- Generate 2048 bit RSA Keys	`-newkey ec:EC-PARAM.pem`	- Generate EC Keys using EC Parameters

If `-key` or `-newkey` is not specified, a private key file will be automatically generated using directives specified in `openssl.cnf`

Inspecting Certificate Signing Requests (CSRs) and Certificates

Viewing contents of Certs and CSRs

Viewing x509 Certificate as human readable Text
```
openssl x509  -in CERT.pem  -noout -text
```

Viewing Certificate Signing Request (CSR) contents as Text:
```
openssl req  -in CSR.pem  -noout -text
```

Extracting Specific Info from Certificates

Extract specific pieces of information from x509 Certificates
```
openssl x509  -in CERT.pem  -noout  -dates
openssl x509  -in CERT.pem  -noout  -issuer -subject
```

Other items you can extract: `-modulus` `-pubkey` `-ocsp_uri` `-ocspid` `-serial` `-startdate` `-enddate`

Extracting x509 Certificate Extensions

Extract specific Extension(s) from a certificate
```
openssl x509  -in CERT.pem  -noout  -ext subjectAltName
openssl x509  -in CERT.pem  -noout  -ext authorityInfoAccess,crlDistributionPoints
```

Other extensions you can extract: `basicConstraints` `nameConstraints` `certificatePolicies` `keyUsage` `extendedKeyUsage` `subjectKeyIdentifier` `authorityKeyIdentifier`

Extract all Extensions from a certificate
```
openssl x509  -in CERT.pem  -noout  -text | sed '/X509v3 extensions/,/Signature Algorithm:/!d'
```

File Formats and Converting between formats (PEM, DER, PFX)

Check if file is PEM, DER, or PFX

To check if file is PEM format
```
openssl  x509  -in FILE
```

To check if file is DER format
```
openssl  x509  -in FILE  -inform DER
```

To check if file is PFX format
```
openssl pkcs12  -in FILE  -nodes
```

To check, or convert, PEM or DER Key Files use `openssl pkey` instead of `openssl x509` and same command arguments.

PEM <==> DER

Convert PEM Certificate file to DER
```
openssl x509  -in CERT.pem  -outform DER  -out CERT.der
```

Convert DER Certificate file to PEM
```
openssl x509  -in CERT.der  -inform der  -out CERT.pem
```

PEM --> PFX

Convert PEM Certificate(s) to PFX
```
openssl pkcs12  -in CERTS.pem  -nokeys  -export  -out CERTS.pfx
```

To include a key in PFX file use `-inkey KEY.pem` instead of `-nokeys`

PFX --> PEM

To extract everything within a PFX file as a PEM file:
```
openssl pkcs12 -in FILE.pfx  -out EVERYTHING.pem  -nodes
```

To extract only the Private Key from a PFX file as PEM:
```
openssl pkcs12 -in FILE.pfx  -out KEY.pem -nodes -nocerts
```

PFX files can contain Certificate(s), or Certificate(s) + *one* matching Key

`-clcerts` - extract only end-entity certificate (client certificate)

`-cacerts` - extract all but end-entity certificate

`-nokeys` - extract only certficiates