Create the root pair

Acting as a certificate authority (CA) means dealing with cryptographic pairs of private keys and public certificates. The very first cryptographic pair we'll create is the root pair. This consists of the root key (`ca.key.pem`) and root certificate (`ca.cert.pem`). This pair forms the identity of your CA.

Typically, the root CA does not sign server or client certificates directly. The root CA is only ever used to create one or more intermediate CAs, which are trusted by the root CA to sign certificates on their behalf. This is best practice. It allows the root key to be kept offline and unused as much as possible, as any compromise of the root key is disastrous.

> **Note**
>
> It's best practice to create the root pair in a secure environment. Ideally, this should be on a fully encrypted, air gapped computer that is permanently isolated from the Internet. Remove the wireless card and fill the ethernet port with glue.

Prepare the directory

Choose a directory (`/root/ca`) to store all keys and certificates.

```
# mkdir /root/ca
```

Create the directory structure. The `index.txt` and `serial` files act as a flat file database to keep track of signed certificates.

```
# cd /root/ca
# mkdir certs crl newcerts private
# chmod 700 private
# touch index.txt
# echo 1000 > serial
```

Prepare the configuration file

You must create a configuration file for OpenSSL to use. Copy the root CA configuration file from the *Appendix* to `/root/ca/openssl.cnf`.

The `[ ca ]` section is mandatory. Here we tell OpenSSL to use the options from the `[ CA_default ]` section.

```
[ ca ]
# `man ca`
default_ca = CA_default
```

The `[ CA_default ]` section contains a range of defaults. Make sure you declare the directory you chose earlier (`/root/ca`).

```
[ CA_default ]
# Directory and file locations.
dir               = /root/ca
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
```

```
serial            = $dir/serial
RANDFILE          = $dir/private/.rand

# The root key and root certificate.
private_key       = $dir/private/ca.key.pem
certificate       = $dir/certs/ca.cert.pem

# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/ca.crl.pem
crl_extensions    = crl_ext
default_crl_days  = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md        = sha256

name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
preserve          = no
policy            = policy_strict
```

We'll apply `policy_strict` for all root CA signatures, as the root CA is only being used to create intermediate CAs.

```
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

We'll apply `policy_loose` for all intermediate CA signatures, as the intermediate CA is signing server and client certificates that may come from a variety of third-parties.

```
[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Options from the `[ req ]` section are applied when creating certificates or certificate signing requests.

```
[ req ]
# Options for the `req` tool (`man req`).
default_bits        = 2048
distinguished_name  = req_distinguished_name
string_mask         = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md          = sha256

# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca
```

The `[ req_distinguished_name ]` section declares the information normally required in a certificate signing request. You can optionally specify some defaults.

```
[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address

# Optionally, specify some defaults.
countryName_default             = GB
stateOrProvinceName_default     = England
localityName_default            =
0.organizationName_default      = Alice Ltd
#organizationalUnitName_default =
#emailAddress_default           =
```

The next few sections are extensions that can be applied when signing certificates. For example, passing the `-extensions v3_ca` command-line argument will apply the options set in `[ v3_ca ]`.

We'll apply the `v3_ca` extension when we create the root certificate.

```
[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

We'll apply the `v3_ca_intermediate` extension when we create the intermediate certificate. `pathlen:0` ensures that there can be no further certificate authorities below the intermediate CA.

```
[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

We'll apply the `usr_cert` extension when signing client certificates, such as those used for remote user authentication.

```
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
```

We'll apply the `server_cert` extension when signing server certificates, such as those used for web servers.

```
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
```

```
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

The `crl_ext` extension is automatically applied when creating *certificate revocation lists*.

```
[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
```

We'll apply the `ocsp` extension when signing the *Online Certificate Status Protocol (OCSP)* certificate.

```
[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

Create the root key

Create the root key (`ca.key.pem`) and keep it absolutely secure. Anyone in possession of the root key can issue trusted certificates. Encrypt the root key with AES 256-bit encryption and a strong password.

> **Note**
>
> Use 4096 bits for all root and intermediate certificate authority keys. You'll still be able to sign server and client certificates of a shorter length.

```
# cd /root/ca
# openssl genrsa -aes256 -out private/ca.key.pem 4096

Enter pass phrase for ca.key.pem: secretpassword
Verifying - Enter pass phrase for ca.key.pem: secretpassword

# chmod 400 private/ca.key.pem
```

Create the root certificate

Use the root key (`ca.key.pem`) to create a root certificate (`ca.cert.pem`). Give the root certificate a long expiry date, such as twenty years. Once the root certificate expires, all certificates signed by the CA become invalid.

> **Warning**
>
> Whenever you use the `req` tool, you must specify a configuration file to use with the `-config` option, otherwise OpenSSL will default to `/etc/pki/tls/openssl.cnf`.

```
# cd /root/ca
# openssl req -config openssl.cnf \
      -key private/ca.key.pem \
      -new -x509 -days 7300 -sha256 -extensions v3_ca \
      -out certs/ca.cert.pem

Enter pass phrase for ca.key.pem: secretpassword
```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
-----
Country Name (2 letter code) [XX]:GB
State or Province Name []:England
Locality Name []:
Organization Name []:Alice Ltd
Organizational Unit Name []:Alice Ltd Certificate Authority
Common Name []:Alice Ltd Root CA
Email Address []:

# chmod 444 certs/ca.cert.pem
```

Verify the root certificate

```
# openssl x509 -noout -text -in certs/ca.cert.pem
```

The output shows:

- the `Signature Algorithm` used
- the dates of certificate `Validity`
- the `Public-Key` bit length
- the `Issuer`, which is the entity that signed the certificate
- the `Subject`, which refers to the certificate itself

The `Issuer` and `Subject` are identical as the certificate is self-signed. Note that all root certificates are self-signed.

```
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=GB, ST=England,
            O=Alice Ltd, OU=Alice Ltd Certificate Authority,
            CN=Alice Ltd Root CA
    Validity
        Not Before: Apr 11 12:22:58 2015 GMT
        Not After : Apr  6 12:22:58 2035 GMT
    Subject: C=GB, ST=England,
             O=Alice Ltd, OU=Alice Ltd Certificate Authority,
             CN=Alice Ltd Root CA
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
```

The output also shows the **X509v3 extensions**. We applied the `v3_ca` extension, so the options from `[ v3_ca ]` should be reflected in the output.

```
X509v3 extensions:
    X509v3 Subject Key Identifier:
        38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31
    X509v3 Authority Key Identifier:
        keyid:38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31

    X509v3 Basic Constraints: critical
        CA:TRUE
    X509v3 Key Usage: critical
        Digital Signature, Certificate Sign, CRL Sign
```

Comments

← Previous

Next →

Version 1.0.4 — Last updated on 2015-12-09.